Suivi du regard non intrusif en tête libre à partir d'images vidéo

Ba Linh Nguyen, Youssef Chahir, Michèle Molina, François Jouen

Suivi du regard non intrusif en tête libre à partir d'images vidéo

Ba Linh NGUYEN[*], Youssef CHAHIR[**], Michèle MOLINA [***], François JOUEN [*]

[*] *CHArt EA 4004-UMS CNRS 2809*
Ecole Pratique des Hautes Etudes
41 rue Gay Lussac
F -75005 Paris
[**] *GREYC - UMR CNRS 6072*
Université de Caen
Campus Côte de Nacre
F-14032 Caen Cedex
[***] *Laboratoire PALM JE 2528*
Université de Caen
F-14032 Caen Cedex

RÉSUMÉ. *Le problème de suivi du regard a été étudié depuis longtemps. Le problème le plus difficile dans un système de suivi du regard non intrusif est la prise en compte des mouvements de la tête. Certaines méthodes utilisent deux caméras et une lumière d'infrarouge pour résoudre ce problème. Avec une seule caméra, l'utilisateur doit maintenir la tête fixe lors du suivi du regard. Si la tête de l'utilisateur s'éloigne de la position initiale, la précision du système baisse de façon spectaculaire. Dans cet article, nous proposons une solution non intrusive avec une seule caméra permettant à l'utilisateur de bouger la tête et utilisant un processus gaussien pour le suivi du regard .*

ABSTRACT. *The problem of eye gaze tracking has been researched and developed for a long time. The most difficult problem in the non-intrusive system of eye gaze tracking is the problem of head movements. Some of existing methods have to use two cameras and an active infrared (IR) illumination to solve this problem. Otherwise, with a single camera, the user has to hold the head uncomfortably still when performing a session of eye gaze tracking. If the head of the user moves away from original position, the accuracy of these eye gaze-tracking systems drops dramatically. In this paper, we propose a solution using a gaussian process for eye gaze tracking that allows free head movements with a single camera.*

Studia Informatica Universalis.

MOTS-CLÉS : *Détection du regard, suivi du regard, processus gaussiens.*
KEYWORDS: *Gaze detection, eye tracking, gaussian processes.*

1. Introduction

Au cours des dernières années, de nombreux systèmes non-intrusifs de suivi du regard ont été étudiés et développés. Ces systèmes n'ont pas besoin d'imposer une contrainte à l'utilisateur et la technologie de la vidéo ouvre une voie des plus prometteuses pour la construction d'un système non-intrusif de suivi du regard. Fondées sur l'analyse des images de l'œil captées par une caméra vidéo, différentes techniques ont été proposées pour calculer une estimation du regard. La plupart d'entre elles utilisent l'infrarouge (IR) pour illuminer la pupille de l'œil puis extraire la position de l'œil par l'exploitation des propriétés géométriques de la lumière IR réfléchie par l'oeil[1], [2], [3], [4], [5], [6], [7]. De la même façon la plupart des méthodes a recours à deux caméras pour la calibration [1], [7]. Pourtant, ces méthodes non-intrusives de suivi du regard en temps réel n'ont pas encore résolu le problème du mouvement de la tête. L'utilisateur doit maintenir sa tête la plus immobile possible durant le recueil des données. Si la tête de l'utilisateur s'éloigne de la position de référence, fixée au moment de la calibration, la précision du système de suivi du regard baisse de façon spectaculaire.

Pour résoudre le problème du mouvement de la tête, Zhiwei Zhu et Qiang Ji ont utilisé deux caméras pour la calibration, et une technique infrarouge pour obtenir la location des yeux par réflexion [8]. Ce système permet à l'utilisateur de se bouger librement la tête pendant la session. Bien que peu onéreuse, cette technique reste lourde à mettre en oeuvre et reste peu utilisée dans la pratique.

Dans cet article, nous proposons une solution non-intrusive qui permet, avec une simple webcam, de suivre le regard indépendamment des mouvements de la tête. Pour ce faire, nous utilisons une méthode de détection fondée sur descripteurs de Haar [9], [10] pour détecter les yeux dans le visage. Cette méthode permet d'entraîner des classificateurs avec quelques milliers d'échantillons d'image puis de construire

une cascade de classificateurs afin de détecter rapidement les yeux dans l'image vidéo. Ensuite, le suivi des mouvements de tête est réalisé par le suivi de traits dans le visage (les yeux, le nez et la bouche) à l'aide de l'algorithme de Lucas Kanade [11]. La méthode de détection Outliers [12] permet de détecter et de corriger les traits distinctifs qui sont perdus par les mouvements de la tête. Il est ensuite possible de procéder au suivi du regard sur l'écran. Nous devons pour cela trouver les liens fonctionnels entre l'œil de l'utilisateur et le point fixé sur l'écran par l'utilisateur. Nous n'utilisons pas de réseaux de neurones pour effectuer cette fonction, mais estimons la distribution de cette fonction. Les Processus Gaussiens permettent d'en calculer la moyenne et la covariance et de faire ainsi des prédictions des nouvelles données. Avec cette méthode, l'utilisateur doit effectuer une procédure de calibration pour obtenir la fonction de suivi du regard sous les différents angles de la tête. L'utilisateur peut ensuite se déplacer librement la tête devant la caméra. Cette technique fonctionne avec n'importe quelle caméra vidéo y compris avec les caméras USB 640X480 qui équipent la majorité des ordinateurs portables.

Cet article est organisé de la façon suivante. La section 2 détaille le processus de détection de l'œil par les descripteurs de Haar. La section 3 expose la méthode pour la détection et le suivi des traits distinctifs du visage. La section 4 décrit l'utilisation des Processus Gaussiens pour prédire la position du regard. La section 5 présente les résultats expérimentaux et la section 6 présente nos conclusions et des perspectives pour de futures recherches.

2. Détection des yeux

Pour faire la prédiction du regard de l'utilisateur, nous devons détecter la position d'œil de l'utilisateur à partir de l'image fournie la caméra. Nous utilisons le système de détection rapide de l'objet basé sur une cascade de classificateurs utilisant des descripteurs simples de Haar pour détecter les yeux. Cette méthode a été initialement proposée par Paul Viola [9] et améliorée par Rainer Lienhart [10]. Tout d'abord, les classificateurs sont construits avec des milliers d'images positives et négatives (des images d'objets recherchés et des images de non-

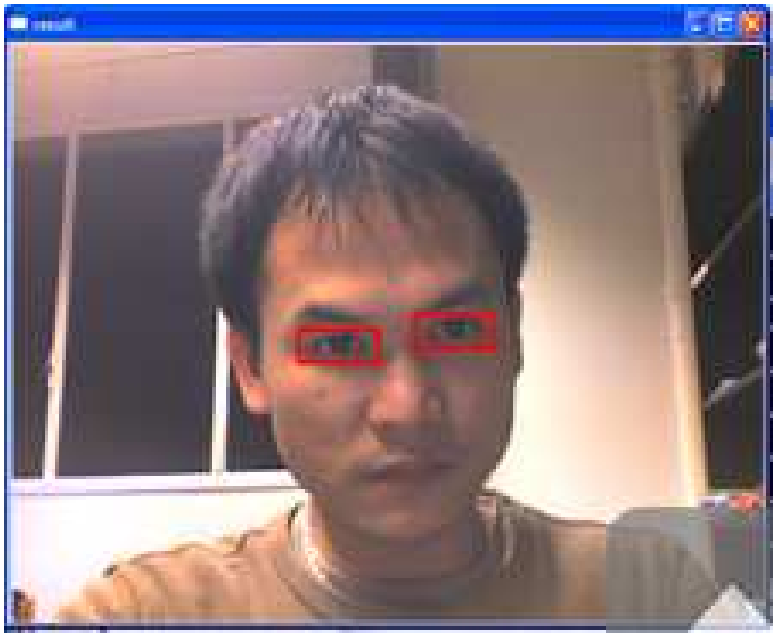

Figure 1 – Détection des yeux.

objets) en utilisant des descripteurs simples (appelés descripteurs de Haar [9]). Il existe un grand nombre de descripteurs dans une sous-fenêtre de l'image 24x24 pixels (117,941 descripteurs) [10], c'est-à-dire un nombre beaucoup plus grand que le nombre de pixels. L'algorithme de la construction retenu est AdaBoost [9] : il sélectionne un petit ensemble de descripteurs permettant de séparer au mieux les exemples positifs et négatifs et crée des classificateurs fondés sur des descripteurs sélectionnés. Après la formation des classificateurs simples, une cascade de classificateurs est élaborée pour accroître la performance de la détection tout en réduisant radicalement les temps de calcul. La cascade de classificateurs est construite de manière à rejeter un grand nombre de sous fenêtres négatives et à détecter la plupart des sous fenêtres positives. Les classificateurs simples sont utilisés pour rejeter la majorité des sous fenêtres avant d'utiliser des classificateurs plus complexes afin réduire de temps de calcul et d'obtenir un faible taux de faux positifs.

Dans notre application, nous utilisons la cascade de classificateurs qui est proposée par OpenCV Swiki et qui a été élaborée avec 7000

échantillons positifs de l'œil [1]. Le résultat de ce classificateur est performant pour la détection, en temps réel, des yeux à partir des images fournies par la caméra. Le résultat de la détection des yeux est illustré par la figure 1.

En utilisant ce classificateur pour suivre l'œil en temps réel, le traitement consistant à rechercher l'œil dans toutes les images successives fournies par la caméra conduirait à des temps de calcul trop importants. Nous avons donc choisi d'utiliser cette cascade de classificateurs pour détecter l'œil dans la *première image* et d'utiliser ensuite une autre méthode pour suivre l'œil dans toutes les images suivantes. Nous présentons cette méthode dans la section suivante.

3. Suivi des yeux

Pour suivre le déplacement de l'œil en temps réel, il s'agit de suivre les epicanthi externes et internes des deux yeux (les coins des yeux) puis de récupérer leur position dans chaque image. Pour avoir un bon suivi, nous devons suivre d'autres traits distinctifs du visage comme le nez et de la bouche afin de savoir si ces traits sont toujours en cours de suivi ou en perte de suivi. Nous utilisons la méthode de détection Outliers pour détecter les traits perdus et les corriger.

3.1. *Chercher les traits distinctifs*

Nous allons maintenant détecter les les traits distinctifs des yeux, du nez et la bouche. Pour cela, nous utilisons la fonction CVGoodFeatureToTrack fournie par la bibliothèque OpenCV pour trouver deux points les plus forts pour chaque chaque œil, dans la région de chacun des deux yeux que nous avons détectée précédemment. Cette fonction cherche les points qui ont les grandes "eigenvalues" dans l'image en utilisant l'opérateur Harris [13]. Pour trouver des les traits distinctifs du nez et des lèvres, nous utilisons à nouveau la méthode CvGoodFeatureToTrack en l'appliquent à la la région au-dessous des yeux. Le résultat est illustré dans la figure 2.

1. sur le site Web http ://alereimondo.no-ip.org/OpenCV

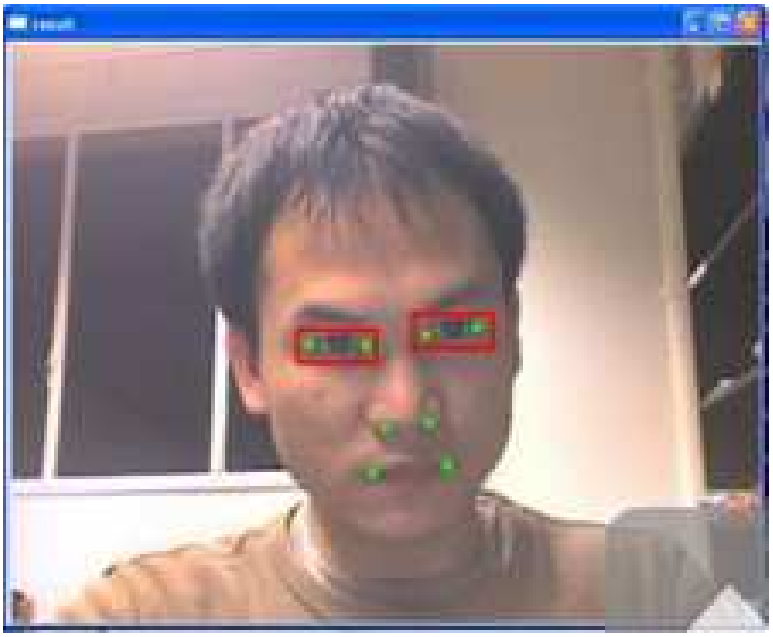

Figure 2 – Chercher les traits distinctifs.

3.2. *Suivi des traits*

Pour effectuer le suivi de l'œil, nous utilisons l'algorithme de suivi Lucas-Kanade [11] très efficace pour le suivi d'objets en temps réel. Après la détection des coins des yeux présentée dans la section précédente, nous utilisons la méthode CvCalcOpticalflowPyrLK proposée par OpenCV pour le suivi. Cette méthode calcule un flux optique en utilisant l'algorithme itératif pyramidal de Lucas-Kanade. Le détail de cet algorithme est décrit dans [11]. Le résultat de cette méthode de suivi est excellent en temps réel. Toutefois, lorsque l'utilisateur bouge la tête très vite, des erreurs peuvent se produire et nous devons donc détecter ces erreurs et les récupérer.

3.3. *Détecter et corriger les erreurs de suivi*

Nous utilisons la méthode de détection Outliers pour détecter le point suivi qui a été perdu. Un Outlier peut être défini comme une observa-

tion qui s'écarte beaucoup des autres observations laissant ainsi penser qu'il a été généré par un mécanisme différent [14]. C'est donc est un objet dans des données qui n'est pas conforme au comportement des autres données. Il peut être considéré comme un bruit ou une exception, ce qui est très utile dans l'analyse d'événements rares. Nous utilisons l'approche fondée sur la distance pour détecter ces événements Outlier [12]. Une méthode assez utilisée d'identification des Outliers consiste à prendre les K voisins les plus proches d'une mesure. Parmi les K voisins, si le point est relativement proche, alors la mesure est considérée comme normal ; si le point est loin, la mesure est alors considérée comme un fait inhabituel. Un des avantages de cette méthode fondée sur la distance est qu'aucune distribution des valeurs ne doit être définie *a priori*. Par ailleurs cette méthode peut être appliquée à tout espace de caractéristiques que nous pouvons définir par une mesure de distance.

Dans notre problème, les données sont les transitions de tous les traits distinctifs d'une image à une autre image. L'Outlier est donc la transition du trait qui présente des mouvements différents des autres traits. Cela signifie que, si la distance de transition d'un point est supérieure à une distance précise, cette transition est l'Outlier et le suivi du point est donc perdu.

Pour corriger le point perdu, la nouvelle valeur du point est égale à la somme de l'ancienne position du point et de la moyenne de la transition de tous les autres points. La tâche de détection des points perdus et de leur correction est réalisée dans chaque image, de sorte que tous les points distinctifs sont parfaitement suivis à chaque instant.

Maintenant, pour savoir quelle sel la position de l'œil de l'utilisateur sur l'écran, nous allons présenter en section 4 l'utilisation des processus gaussiens pour détecter le regard.

4. Prédiction du regard par Processus Gaussien

Nous disposons d'une base de données (les données de formation), qui comprend en entrée $\mathbf{x}$ l'image de l'œil, de sa taille de (32×16), et en sortie y la position du point sur l'écran que l'utilisateur regarde. y comprend deux valeurs (t_x, t_y) que nous allons utiliser indépendamment

dans la prédiction du regard. Nous appelons cette base de données $\mathcal{D}$: $\mathcal{D} = \{(\mathbf{x}_i, y_i)|i = 1, ..., n\}$ où $\mathbf{x}$ désigne un vecteur d'entrée (covariables) de dimension D et y désigne une sortie scalaire ou une cible (variable dépendante), n est le nombre d'observations. Les vecteurs colonne entrée pour tous les n cas sont regroupés dans une matrice de X de dimension $D \times n$. Les cibles sont rassemblées dans le vecteur $\mathbf{y}$ et on peut écrire $\mathcal{D} = (X, \mathbf{y})$. Dans le calcul de la régression, les cibles sont des valeurs réelles. Nous nous sommes intéressés aux inférences que l'on peut faire sur les relations entre les entrées et les cibles (Figure 3).

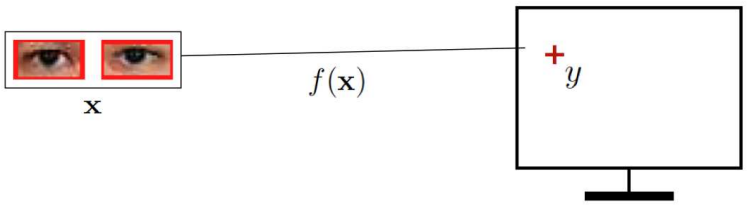

Figure 3 – Relation entre les entrées et les cibles.

Une grande variété de méthodes a été proposée pour faire face à ce problème d'apprentissage supervisé. Ces méthodes utilisent le plus souvent un grand ensemble de données de formation et des réseaux neuronaux pour entraîner la fonction. Comme il est difficile d'avoir une bonne base de données de formation avec toutes les entrées possibles, nous avons choisi d'utiliser les processus gaussiens pour cette raison. Les propriétés de la fonction à un nombre fini de point d'inférence dans un processus gaussien nous donnent la même réponse que si on prend un nombre infini de points. Nous devons donc faire des prédictions pour la nouvelle entrée $\mathbf{x}_*$. Comme nous ne disposons pas de données de formation dans la base, nous avons besoin de transformer, à partir d'un nombre fini de données de formation $\mathcal{D}$, une fonction f qui peut faire des prédictions pour toutes les valeurs entrées possibles.

L'idée principale de cette méthode est de calculer la distribution prédictive pour $f_* \equiv f(\mathbf{x}_*)$ à $\mathbf{x}_*$: $p(f_*|\mathbf{x}_*, \mathcal{D})$ et nous utilisons un processus gaussien pour calculer cette distribution prédictive. Un processus gaussien est entièrement spécifié par sa fonction de moyenne et sa fonc-

tion de covariance. Nous allons donc estimer la fonction de moyenne et la fonction de covariance de f_*.

Avec toutes les valeurs de données de formation $\mathbf{x}_1, ..., \mathbf{x}_n$ nous pouvons calculer des échantillons $f(\mathbf{x}_1), ..., f(\mathbf{x}_n)$ à partir d'une probabilité a priori sur les fonctions spécifiées par un processus gaussien avec une moyenne égale à zéro et une fonction de covariance K.

$$f(\mathbf{x}_1), ..., f(\mathbf{x}_n) \sim \mathcal{N}(\mathbf{0}, K) \tag{1}$$

où

$$\begin{aligned} K_{p,q} &= \text{cov}(f(\mathbf{x}_p)f(\mathbf{x}_q)) - k(\mathbf{x}_p, \mathbf{x}_q) \\ &= \exp(-\frac{1}{2}|\mathbf{x}_p - \mathbf{x}_q|^2) \end{aligned} \tag{2}$$

La spécification de la fonction de covariance implique une distribution sur les fonctions. Nous pouvons calculer des échantillons à partir de la distribution des fonctions évaluées à tout nombre de points. En détail, nous choisissons un nombre de points d'entrée, X_*, écrivons la matrice de covariance correspondante en utilisant (2), et ensuite générons un vecteur gaussien aléatoire avec cette matrice de covariance

$$\mathbf{f}_* \sim \mathcal{N}(\mathbf{0}, K(X_*, X_*)) \tag{3}$$

La génération des échantillons gaussiens est décrite dans [15]. La définition du processus gaussien : *un processus gaussien est une collection de variables aléatoires ; tout sous ensemble fini de variables de cette collection a une distribution gaussienne conjointe.*

D'abord, nous considérons que dans le cas particulier simple où les observations n'ont pas de bruit, c'est que nous connaissons $\{(\mathbf{x}_i, f_i)|i = 1, ..., n\}$. À partir de la définition du processus gaussien, on peut écrire une distribution conjointe entre des sorties de données de formation $\mathbf{f}$, et des sorties de tests $\mathbf{f}_*$, dont la distribution est *a priori*

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right). \tag{4}$$

S'il y a n points de formation et n_* points de test, alors $K(X, X_*)$ désigne la $n \times n_*$ matrice de covariance évaluée pour toutes les paires

de points de formation et de points de test. On procède de même pour les autres $K(X,X), K(X_*,X_*)$ et $K(X_*,X)$. Pour obtenir la distribution *a posteriori* sur les fonctions, nous avons besoin de limiter *a priori* cette distribution conjointe pour ne conserver que les fonctions qui sont en accord avec les points des données observées. Correspondant à la condition de distribution conjointe gaussienne *a priori* sur les observations (voir [15] section A.2) on obtient

$$\begin{aligned}\mathbf{f}_*|X_*,X,\mathbf{f} \ \sim \mathcal{N}\big(&K(X_*,X)K(X,X)^{-1}\mathbf{f},\\ &K(X_*,X_*)-K(X_*,X)K(X,X)^{-1}K(X,X_*)\big).\end{aligned} \tag{5}$$

Les valeurs de la fonction $\mathbf{f}_*$ (ce qui correspond aux entrées de test X_*) peuvent être échantillonnées à partir de la distribution conjointe *a posteriori* par une évaluation de la moyenne et de la matrice de covariance de l'équation (5). La méthode de génération des échantillons est décrite dans ([15] section A.2).

Pour notre problème, nous avons supposé que les valeurs observées y différent des valeurs de fonction $f(\mathbf{x})$ par le bruit additif ε, $y = f(\mathbf{x})+\varepsilon$. Nous avons assumé que ce bruit possède une distribution indépendante de moyenne zéro et de variance σ_n^2

$$\varepsilon \sim \mathcal{N}(0,\sigma_n^2) \tag{6}$$

Et l'a priori sur les bruit observés devient

$$\begin{aligned}\mathrm{cov}(y_p,y_q) &= k(\mathbf{x}_p,\mathbf{y}_q)+\sigma_n^2\delta_{pq}\\ \text{ou}\quad \mathrm{cov}(\mathbf{y}) &= K(X,X)+\sigma_n^2 I,\end{aligned} \tag{7}$$

où δ_{pq} est un delta de Kronecker qui est 1 si $p=q$ et 0 sinon. Il résulte de l'hypothèse d'indépendance du bruit, qu'une matrice diagonale est ajoutée, en comparaison avec le cas sans bruit, l'éq. (2). En introduisant le bruit dans l'éq. (4), on peut écrire la distribution conjointe entre des valeurs cibles observées et les valeurs de la fonctions aux positions de test sous l'a priori comme

$$\begin{bmatrix}\mathbf{y}\\ \mathbf{f}_*\end{bmatrix} \sim \mathcal{N}\left(\mathbf{0},\begin{bmatrix}K(X,X)+\sigma_n^2 I & K(X,X_*)\\ K(X_*,X) & K(X_*,X_*)\end{bmatrix}\right). \tag{8}$$

Découlant de la distribution conditionnelle correspondant à (5), nous arrivons à la clé des équations de prédiction de la régression du processus gaussien

$$\mathbf{f}_*|X,\mathbf{y},X_* \sim \mathcal{N}\big(\bar{\mathbf{f}}_*, \mathrm{cov}(\mathbf{f}_*)\big), \quad \text{où} \tag{9}$$

$$\bar{\mathbf{f}}_* \triangleq \mathbb{E}[\mathbf{f}_*|X,\mathbf{y},X_*] = K(X_*,X)[K(X,X)+\sigma_n^2 I]^{-1}\mathbf{y}, \tag{10}$$

$$\mathrm{cov}(\mathbf{f}_*) = K(X_*,X_*) - K(X_*,X)[K(X,X)+\sigma_n^2 I]^{-1}K(X,X_*). \tag{11}$$

Nous proposons d'introduire une forme compacte de la notation. On note $K = K(X,X)$ et $K_* = K(X,X_*)$. Dans le cas où il n'existe qu'un seul point de test $\mathbf{x}_*$, on écrit $\mathbf{k}(\mathbf{x}_*) = \mathbf{k}_*$ pour désigner le vecteur des covariances entre le point de test et les points de formation n. En utilisant cette notation compacte pour un point de test $\mathbf{x}_*$, les équations (10) et (11) s'écrivent :

$$\bar{f}_* = \mathbf{k}_*^\top (K+\sigma_n^2 I)^{-1}\mathbf{y}, \tag{12}$$

$$\mathbb{V}[f_*] = k(\mathbf{x}_*,\mathbf{x}_*) - \mathbf{k}_*^\top (K+\sigma_n^2 I)^{-1}\mathbf{k}_*. \tag{13}$$

Nous avons donc la mise en œuvre de la régression du processus gaussien pour la prédiction de la moyenne et la variance dans la table 1. L'algorithme utilise la décomposition de Cholesky, au lieu d'inverser la matrice directement, car il est plus rapide et plus stable.

5. Résultat expérimental

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

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

$L := \text{cholesky}(K + \sigma_n^2 I)$
$\boldsymbol{\alpha} := L^\top \backslash (L \backslash \mathbf{y})$
$\bar{f}_* := \mathbf{k}_*^\top \boldsymbol{\alpha}$ } prédiction de moyenne l'éq. (12)
$\mathbf{v} := L \backslash \mathbf{k}_*$
$\mathbb{V}[f_*] := k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{v}^\top \mathbf{v}$ } prédiction de variance l'éq. (13)
return : $\bar{f}_*(\text{mean}), \mathbb{V}[f_*](\text{variance})$

Tableau 1 – Prédiction de la régression du processus gaussien.

5.1. *Prédiction du regard avec la tête stable*

Nous faisons d'abord un test de suivi du regard avec la tête stable. L'utilisateur maintient la tête stable lors de la procédure de calibration, puis réalise une tâche de détection en tête fixe (la tête reste immobile après la calibration). Nous réalisons le test de suivi du regard avec 16 autres points. Le résultat est illustré dans la figure 4 : les points triangles sont des points cibles du test, les autres points sont les points de la prédiction du regard.

Les résultats présentés dans la figure 4 sont excellents. Cependant, ils exigent que l'utilisateur maintienne la tête immobile. Si l'utilisateur bouge la tête après la période de calibration, la précision de ce système de suivi du regard baisse de façon spectaculaire. La raison est que, lorsque l'utilisateur bouge la tête, l'image de l'œil change et l'image de l'œil d'entrée est différente de l'image de l'œil dans la base de données de formation. Le résultat de la prédiction du regard sera donc incorrect. Nous proposons une solution à ce problème dans la prochaine section.

5.2. *Prédiction du regard en bougeant la tête*

Pour résoudre le problème de suivi du regard avec la tête mobile, nous proposons de répéter la procédure de calibration avec les différents angles de vue de la tête de l'utilisateur. Avec cette méthode, nous ajou-

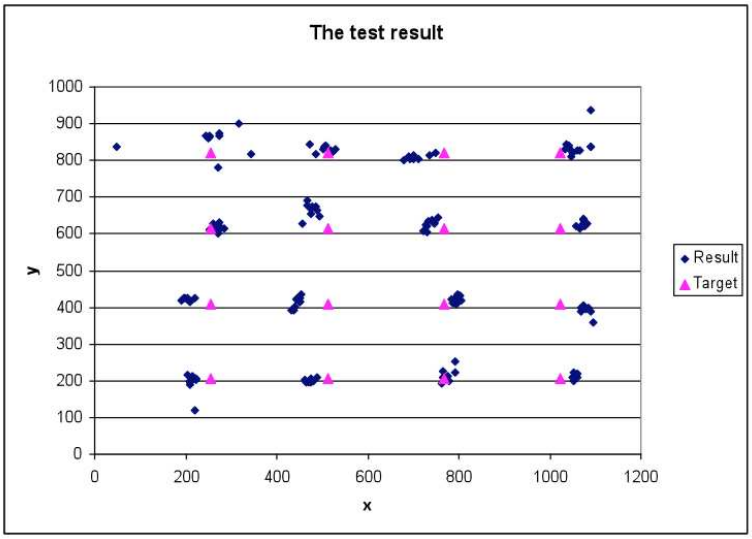

Figure 4 – Le résultat de test avec la tête stable.

tons plus de données à la base de données de formation. Cela signifie que l'on ajoute différentes images de l'œil de l'utilisateur lorsque celui ci regarde un même point sur l'écran à partir de différentes positions. Après la procédure de calibration, l'utilisateur peut librement déplacer sa tête pendant la session de suivi du regard.

Pour cela, la procédure de calibration est répétée quatre fois sous quatre angles de vue différents : avec la tête de l'utilisateur tournée à gauche et à droite, puis avec la tête de l'utilisateur orientée vers le haut et vers le bas. Le degré maximum de la rotation de chaque direction est la position où tous les points distinctifs ne sont pas perdus lors du suivi. La figure 5 montre les quatre positions de la tête de l'utilisateur pour faire la calibration.

Après la calibration, le même test de prédiction du regard avec 16 points est effectué (comme indiqué dans la section précédente) mais à la différence que l'utilisateur peut déplacer librement la tête. L'utilisateur fait tourner sa tête de gauche à droite ou du bas vers le haut pour regarder chaque point. Le résultat est illustré dans la figure 6.

Figure 5 – Positions de la tête de l'utilisateur sous différents angles de vue.

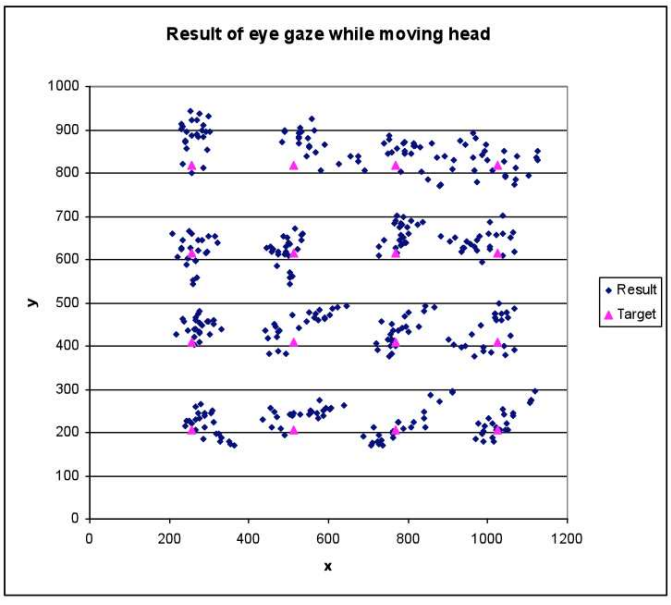

Figure 6 – Le résultat du test de prédiction du regard avec la tête libre après cinq calibrations.

Nous pouvons voir que le résultat de la figure 6 n'est pas très bon, les erreurs de suivi du regard sont importantes. La raison est qu'il y a encore des positions de la tête de l'utilisateur qui rendent l'image de l'œil d'entrée différente de celle de l'œil contenue dans la base de données de formation. Ainsi, nous continuons d'effectuer cinq calibrations supplémentaires pour enrichir à la base de données de formation. Et nous faisons le nouveau test. Le résultat est présenté la figure 7.

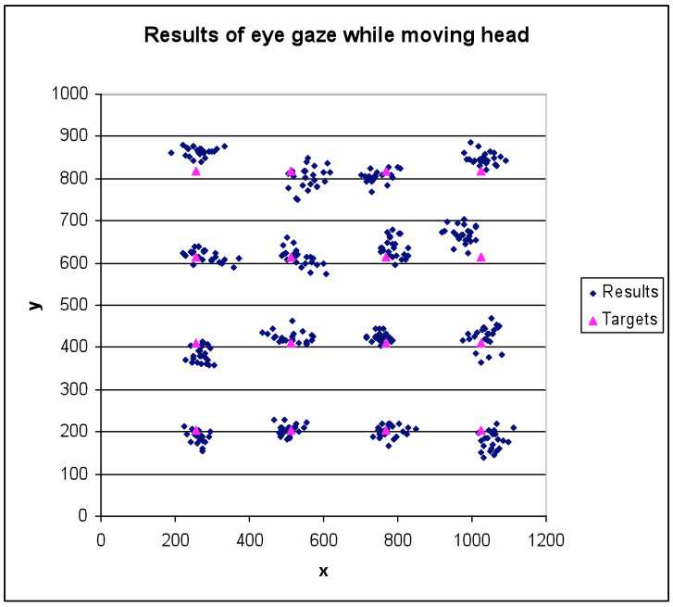

Figure 7 – Le résultat de test de prédiction du regard avec la tête libre après dix calibrations.

Après avoir effectué 10 fois de calibration, le résultat du regard bougeant la tête est correct. Nous pouvons améliorer encore ce résultat en effectuant plus d'essais de calibration.

5.3. *Discussion*

Comme le montre la figure 7 le résultat est très correct avec 10 calibrations comportant différents angles de vue. Cependant, le résultat ne sera pas bon si l'utilisateur déplace latéralement la tête ou si l'utilisateur

s'approche ou s'éloigne de la caméra. Ainsi, au cours de la session de suivi du regard en tête libre, s'il y a une position de la tête qui perturbe la prédiction du regard, l'utilisateur devra faire des nouvelles calibrations pour améliorer résultat. En d'autres termes, pour que la détection et le suivi du regard fonctionnent, la tête doit rester dans le champ de la caméra. On peut penser que ce système de suivi du regard est difficile et compliqué à utiliser. En pratique, lors de la première utilisation du système, l'utilisateur prend environ dix minutes pour faire la calibration de toutes les positions possibles de la tête qu'il peut réutiliser ensuite rendant ainsi le dispositif facile à utiliser. Par ailleurs, il faut rappeler que ce système fonctionne avec une simple webcam usb intégrée dans l'écran de l'ordinateur. De façon générale, l'utilisateur placé à un dispositif de ce type ne bouge pratiquement pas et rend donc le système de détection et de suivi du regard très simple à utiliser.

6. Conclusions et perspectives

Nous avons présenté une approche non-intrusive pour suivre le regard en temps réel qui permet d'avoir la tête libre et qui utilise une simple caméra. Tout d'abord, nous avons utilisé les descripteurs Haar-like pour détecter les yeux dans l'image fournie par la caméra. Ensuite, nous avons utilisé l'algorithme de Lucas-Kanade pour suivre l'œil en temps réel, et la méthode de détection Outliers pour détecter les erreurs de suivi et pour les corriger. Enfin, nous avons utilisé un processus gaussien pour faire la prédiction du regard en temps réel. Notre système a les avantages suivants :

– Pour utiliser ce système de suivi du regard, l'utilisateur n'a besoin que d'une caméra simple ou de la caméra intégrée dans la plupart des ordinateurs portables aujourd'hui.

– Pour prédire le regard, l'utilisateur ne consacre que dix minutes environ pour réaliser la procédure de calibration dans les différentes positions de la tête. Cette calibration n'est nécessaire que pour la première utilisation. C'est la propre base de données de l'utilisateur qui est utilisée pour faire la prédiction de son regard. Dès la deuxième session, l'utilisateur peut réutiliser sa base de données et n'a pas besoin d'effectuer de nouvelle calibration.

– L'utilisateur peut bouger sa tête librement lors de la session : il n'y a pas de contrainte de stabilité de la tête.

Comme tous les systèmes qui utilisent une entrée vidéo, le problème de ce système reste celui de sa sensibilité à la lumière. L'utilisateur doit travailler avec la même luminosité que celle de la phase de calibration. Si la lumière est modifiée, le résultat sera moins bon, et l'utilisateur pourra être amené à refaire la calibration. La raison est que, lorsque la lumière est changée, l'image de l'œil change, de sorte que l'image d'entrée est différente de l'image de l'œil contenue dans la base de données. Cet écart induit alors une prédiction incorrecte. Pour résoudre ce problème, nous devons ajuster en temps réel toutes les images de l'œil en entrée à la même luminosité. De futures recherches devraient permettre de résoudre ce problème.

Références

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

[2] **C. Hennessey, B. Noureddin, and P. Lawrence**, *A single camera eyegaze tracking system with free head motion*, pp. 87–94, 2006.

[3] **D. Yoo and M. Chung**, *A novel non-intrusive eye gaze estimation using cross-ratio under large head motion*, Computer Vision and Image Understanding, vol. 98, no. 1, pp. 25–51, 2005.

[4] **A. Perez, M. Cordoba, A. Garcia, R. Mendez, M. Munoz, J. Pedraza, and F. Sanchez**, *A precise eye-gaze detection and tracking system*, 2003.

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

[6] **C. Morimoto, D. Koons, A. Amir, and M. Flickner**, *Frame-rate pupil detector and gaze tracker*, vol. 99, 1999.

[7] **D. Beymer, M. Flickner, I. Center, and C. San Jose**, *Eye gaze tracking using an active stereo head*, vol. 2, 2003.

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

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

[10] **R. Lienhart and J. Maydt**, *An Extended Set of Haar-like Features for Rapid Object Detection*, vol. 2, no. 1, pp. 900–903, 2002.

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

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

[13] **C. Harris and M. Stephens**, *A combined corner and edge detector*, vol. 15, p. 50,

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

[15] **C. Rasmussen, C. Williams**, *Gaussian processes for machine learning*, Books24x7, Springer, 2006.

ANNEXE POUR LA FABRICATION
A FOURNIR PAR LES AUTEURS AVEC UN EXEMPLAIRE
PAPIER
DE LEUR ARTICLE

1. ARTICLE POUR LA REVUE :
   Studia Informatica Universalis.

2. AUTEURS :
   Ba Linh NGUYEN[*]*, Youssef CHAHIR*[**]*,*
   Michèle MOLINA [***]*, François JOUEN* [*]

3. TITRE DE L'ARTICLE :
   Suivi du regard non intrusif en tête libre à partir d'images vidéo

4. TITRE ABRG POUR LE HAUT DE PAGE MOINS DE 40 SIGNES :
   Suivi du regard

5. DATE DE CETTE VERSION :
   10 octobre 2010

6. COORDONNES DES AUTEURS :

– adresse postale :

[*] CHArt EA 4004-UMS CNRS 2809
Ecole Pratique des Hautes Etudes
41 rue Gay Lussac
F -75005 Paris
[**] GREYC - UMR CNRS 6072
Université de Caen
Campus Côte de Nacre
F-14032 Caen Cedex
[***] Laboratoire PALM JE 2528
Université de Caen
F-14032 Caen Cedex

– tlphone : 01 44 10 84

– tlcopie : 00 00 00 00

– e-mail : ivan.lavallee@gmail.com

7. LOGICIEL UTILIS POUR LA PRPARATION DE CET ARTICLE :
   LaTeX, avec le fichier de style `studia-Hermann.cls`,
   version 1.2 du 03/12/2007.

SERVICE ÉDITORIAL – STUDIA
UNIVERSALIS
41 rue Gay Lussac, 75005 Paris
Tl : 01-44-10-84/83
mel : contact@complexica.net
Serveur web :
http://studia.complexica.net