EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE LAUSANNE
POLITECNICO FEDERALE DI LOSANNA
SWISS FEDERAL INSTITUTE OF TECHNOLOGY LAUSANNE

FACULTE SCIENCES ET TECHNIQUES DE L'INGENIEUR

LABORATOIRE DE SYSTEMES AUTONOMES
CH - 1015 LAUSANNE

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Développement d'un module de reconnaissance de mouvement et de contrôle d'imitation pour le robot humanoïde Robota

Marc Kunze
Projet de diplôme

Professeur : Aude Billard
Assistant : Sylvain Calinon

Lausanne, le 20 février 2003

EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE LAUSANNE
POLITECNICO FEDERALE DI LOSANNA
SWISS FEDERAL INSTITUTE OF TECHNOLOGY LAUSANNE

FACULTE SCIENCES ET TECHNIQUES DE L'INGENIEUR

LABORATOIRE DE SYSTEMES AUTONOMES
CH - 1015 LAUSANNE

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

PROJET DE DIPLOME – HIVER 2003/2004

Titre: Développement d'un module de reconnaissance de mouvement et de contrôle d'imitation pour le robot humanoïde Robota.

Candidat: Marc Kunze **Section:** Microtechnique

Professeur: Aude Billard

Assistant 1: Sylvain Calinon **Assistant 2:**

Donnée & travail demandé :

Ce projet visera à développer un module performant de reconnaissance de mouvement humain dans un plan. Il s'agira d'implémenter des routines standard de reconnaissance de flot optique et de couleur de peau pour extraire l'information sur la sortie de deux ou trois caméras (si le temps le permet, on tâchera de faire de la stéréovision sur le mouvement du haut du corps) afin de traquer les mouvements de la tête, bras et jambes de l'utilisateur. Le module de reconnaissance sera utilisé dans le cadre d'une exposition nationale a la Cité de l'Espace à Toulouse dès le 1er Mai 2004.
Ce projet suivra plusieurs phases: 1) dimensionnement et installation du système de caméras et PC de sorte à reproduire les conditions d'utilisation du musée, en collaboration avec les responsables de la Cité de l'Espace (10%) ; 2) implémentation de routines pour l'extraction en continu du flot optique et de la couleur de la peau sur les sorties des deux caméras (20%); 3) extraction de la direction, amplitude et vitesse du mouvement de la tête, mains et jambes à partir de la sortie des routines développées en point 2 (30%); 4) intégration du programme au système de contrôle moteur du robot pour diriger les mouvements de tête, bras et jambes du robot, afin de suivre précisément (avec la même vitesse et même amplitude) le déplacement de l'interlocuteur (10%); 5) implémentation d'un algorithme d'apprentissage couplant entrées visuelles et son, afin de permettre au robot d'apprendre et de reconnaître différentes danses (25%) ; 6) intégration du module de reconnaissance de mouvement au module d'apprentissage et à l'interface utilisateur développés dans le projet de diplôme *Jeu d'apprentissage pour le robot Robota* par Andres Lopez (10%) ;

Remarques :

Un plan de travail sera établi et présenté aux assistants avant le 1er novembre 2003.
Une présentation intermédiaire (environ 10 minutes de présentation et 10 minutes de discussion) de votre travail aura lieu dans le courant du mois de décembre 2003. Elle a pour objectifs de donner un rapide résumé du travail déjà effectué, de proposer un plan précis pour la suite du projet et d'en discuter les options principales.
Un rapport, comprenant en son début l'énoncé du travail (présent document), suivi d'un résumé d'une page (selon canevas), sera remis le vendredi 20 février 2004 en 4 exemplaires au secrétariat de votre section. L'accent sera mis sur les expériences et les résultats obtenus. Le public cible est de type ingénieur EPF sans connaissance pointue du domaine. Une version préliminaire du rapport sera remise à l'assistant le 13 février 2004. Tous les documents en version informatique, y compris le rapport (en version source et en version pdf), le document de la présentation orale et un résumé au format pdf, ainsi que les sources des différents programmes doivent être gravé sur un CD-ROM et remis à l'assistant au plus tard lors de la défense finale.
Une défense de 40 minutes (environ 20 minutes de présentation et démonstration, plus 20 minutes de réponses aux questions) aura lieu dans la période du 8 au 15 mars 2004.

Le professeur responsable: L'assistant responsable:

Signature :	Signature :
Aude Billard	Sylvain Calinon

Lausanne, le 20 octobre 2003

EPFL - ASL Résumé du travail de diplôme – hiver 2003-2004 - mars 2004

Développement d'un module de reconnaissance de mouvement et de contrôle d'imitation pour le robot humanoïde Robota

Marc Kunze, Microtechnique

Assistant : Sylvain Calinon

Professeur : Aude Billard

Le but de ce projet est de développer un module de reconnaissance de mouvement pour le robot humanoïde Robota, afin que celui-ci obtienne une information sur la position des jambes, des bras et de la tête d'un utilisateur se trouvant face à lui.
Pour réaliser cette tâche, nous utilisons deux webcams ; la première filmant le haut du corps et la seconde filmant le bas du corps. De plus, nous utilisons la libraire de vision OpenCV de Intel®.
Le module de détection des mouvements du haut du corps fusionne les résultats de trois algorithmes différents :

- Détection de la couleur de peau.
- Flot optique.
- "Pattern matching".

La détection de la couleur de peau permet de trouver la position des mains et de la tête de l'utilisateur sur l'image. Afin de pouvoir détecter n'importe quel type de peau, nous utilisons le mode de couleur YCrCb séparant la chrominance de la luminosité. En effet, nous savons que les types de peau diffèrent par la luminosité plutôt que par la chrominance. De plus, afin d'augmenter la robustesse de la détection, chaque utilisateur calibre la couleur de sa peau au début de l'utilisation.
La détection du flot optique consiste à extraire l'information de la vitesse à partir d'une séquence d'images, en faisant l'hypothèse que l'intensité des pixels des objets est conservée au cours du déplacement. Cet algorithme nous permet donc de trouver la position des mains, mais en revanche il ne permet pas de détecter la position de la tête, puisque celle-ci n'a pas un mouvement suffisant.

Finalement, nous utilisons le "pattern matching" afin d'affiner la position de la tête. Le "pattern matching" consiste à sélectionner une partie de l'image et de chercher cette partie de l'image dans les images suivantes. Dans notre cas, par calibration, nous sélectionnons la partie de l'image où se trouve le nez. Ainsi nous "trackons" le nez de l'utilisateur.

Module de détection du haut du corps

Le module de détection des mouvements du bas du corps utilise la ségmentation de l'image en fonction de la couleur. En effet, nous connaissons la couleur du "background", ainsi tout les pixels qui ne correspondent pas à cette couleur correspondent aux jambes de l'utilisateur.

Module de détection du bas du corps

Tables des matières

1 Introduction

La Cité de l'espace, parc scientifique, ludique et éducatif, destiné au grand public, inaugure au printemps 2004 une nouvelle exposition temporaire nommée "Mission Biospace I". Cette exposition a pour but de projeter le visiteur à bord d'un vaisseau spatial en route pour un autre système solaire. Par là, on tente donc de démontrer quelle pourrait être la vie à bord d'un tel engin.
Dans ce cadre là, on peut imaginer que Robota est un robot destiné au développement pédagogique des enfants vivants à l'intérieur du vaisseau. Elle est en phase finale de réalisation, c'est pourquoi le visiteur de l'exposition, qui joue le rôle d'habitant du vaisseau spatial, doit lui apprendre à bouger ses membres comme un humain.
L'application vise donc l'apprentissage de mouvements par imitation et utilise les notions de dialogue homme/machine et d'évolution des formes d'intelligence artificielle. L'objectif recherché est de faire expérimenter aux visiteurs le processus d'apprentissage d'un protocole de communication avec un robot, en sensibilisant l'utilisateur à l'apparition d'entités artificiellement intelligentes.

Cette application nécessite donc une interface homme/machine et une intelligence artificielle qui ont toutes deux été développées lors de projets de diplôme.
Dans ce projet de diplôme, nous allons plus particulièrement nous intéresser à la partie interface homme/machine. Cette interface n'est pas une interface conventionnelle utilisant un clavier, une souris ou des buttons. En effet le visiteur doit apprendre à Robota à bouger et à danser par démonstration. Il faut donc que Robota obtienne des informations sur le positionnement des membres du visiteur (jambes, bras et tête). Ces informations sont obtenues à l'aide de webcams filmant le haut et le bas du corps du visiteur. Le but final est de pouvoir identifier la position des jambes, des bras et de la tête à partir des images retournées par les webcams et ensuite transmettre ces informations à la partie intelligence artificielle de l'application. Cette interface homme/machine sera donc réalisée en traitant les informations fournies par les images.

2 Matériel utilisé

2.1 "Hardware"

Caméra :
Les caméras utilisées sont les webcams Logitech® QuickCam® Pro 4000 [1]. Ces caméras se branchent sur le port USB de l'ordinateur et ne nécessitent pas de carte d'acquisition ("frame grabber"). Les caractéristiques techniques sont les suivantes :

- Capture vidéo : jusqu'à 640 x 480 pixels (CCD VGA).
- Capture d'images fixes : jusqu'à 1280 x 960 pixels.
- Débit d'images : jusqu'à 30 images par seconde.
- Capteur CCD VGA haute qualité, micro intégré.

Il faut noter que, dans un premier temps, notre choix s'est porté sur les webcams Philips® ToUcam PRO II. Mais, pour des raisons inconnues, le "driver" de ces caméras ne supporte pas d'avoir deux caméras en fonction simultanément, ce qui est une nécessité dans notre application.

Ordinateur :
Notre application "tourne" sur un PC Dell avec un microprocesseur de 2.8 GHz et 512 MB de RAM. Ceci nous permet d'effectuer un traitement d'images plus conséquent en comparaison aux précédents modules de reconnaissance visuelle [2 et 3] développés au sein du laboratoire, puisque ceux-ci utilisaient un PocketPC Ipaq 3850 de Compaq cadencé à 206 MHz.

2.2 "Software"

Les algorithmes de traitement d'images sont codés en C/C++ et compilés à l'aide de Microsoft Visual C++. De plus nous utilisons la librairie de vision OpenCV fournie par Intel®. Cette librairie est téléchargeable librement sur le site internet de Intel® [4]. Elle contient un ensemble de fonctions utilitaires pour traiter et gérer les images telles que :

- Création, allocation et destruction d'image.
- Recherche, manipulation, simplification et affichage de contour.
- Opérateurs morphologiques.
- Filtres passe-haut, passe-bas, etc.
- Opération sur le "background".
- Opération de seuillage.
- Calcul du flot optique.
- Etc.

Pour de plus amples informations, une documentation (incomplète et parfois incorrect) est également disponible sur le site internet de Intel®. En revanche, un forum de discussion [5] contient très souvent les informations utiles, ainsi que les dernières versions téléchargeables.

3 Module de détection de la tête et des mains

3.1 Introduction

Relativement peu de recherches ont eu pour but de localiser une tête et deux mains dans une même image. En revanche, il existe déjà une grande quantité de travaux traitant de l'extraction de visages dans les images [6]. C'est pourquoi nous allons baser notre approche sur celles déjà adoptées pour l'extraction de visages et essayer de les généraliser pour notre cas particulier.

Il existe une grande quantité de méthodes différentes pour détecter un visage dans une image, cependant nous pouvons citer les principales méthodes qui sont :

- "Pattern matching" : cette méthode consiste à corréler un masque connu et l'image. Le masque est généralement constitué d'éléments clés du visage, tels que la bouche, le nez, les yeux ou les sourcils.
- "Eigenface" : cette méthode est basée sur la décomposition des visages en vecteurs propres, puis vérifie la similitude avec les vecteurs propres des visages tests.
- Couleur de peau : cette méthode nécessite une image couleur et est basée sur un modèle de couleur de la peau.

Il existe encore de nombreuses autres méthodes se basant sur la texture, sur les réseaux de neurones, sur le modèle de Markov caché, etc.
Le challenge de la détection de visages consiste à implémenter une technique suffisamment robuste pour ne pas être sensible aux changements de luminosité, de taille, d'orientation, de position, etc.

Puisque nous disposons d'images couleurs, notre application utilise la méthode se basant sur la couleur de peau. Celle-ci peut s'appliquer non seulement pour détecter un visage, mais peut très bien s'appliquer pour détecter des mains. Parallèlement à cette méthode nous utilisons la méthode du flot optique qui nous permet de détecter le mouvement dans une image (cette méthode n'est pas spécifique à la détection de visages). En fusionnant l'information provenant de ces deux méthodes totalement indépendantes l'une de l'autre, cela permet de réaliser une application plus robuste. Finalement, nous utilisons également la méthode de "pattern matching" afin d'affiner le "tracking" du visage.
Il faut toutefois noter que notre tâche est simplifiée, puisque nous faisons l'hypothèse qu'une seule personne se trouve face à Robota, ce qui signifie que nous devons chercher uniquement une tête et deux mains dans l'image.

3.2 Couleur de peau

3.2.1 Introduction

Dans de nombreuses applications, la couleur de la peau a prouvé être un moyen efficace pour localiser une tête ou des mains dans une image. Bien que certaines personnes ont une peau de couleur différente (type caucasien, mongoloïde, négroïde ou métis), plusieurs études ont montré que la différence majeure réside dans l'intensité plutôt que dans la chrominance [7]. Ainsi, théoriquement, en se basant sur la chrominance uniquement, nous serions capables de détecter tous les types de peau dans une image. Pour ce faire, il suffit de coder l'image dans un système de couleur séparant l'intensité de la chrominance. Plusieurs types de représentations (YCrCb, YIQ, YES, etc.) séparent ces deux composantes de la couleur. Dans cette application, nous avons choisi d'utiliser le mode YCrCb (Luminance-Chrominance) qui, d'après l'étude de D. Chai et A. Bouzerdoum [8], se prête particulièrement bien à la détection de la peau.

3.2.2 Mode YCrCb

Pour transformer les valeurs RGB (Red-Green-Blue) retournée par la webcam en YCrCb, il suffit d'appliquer la transformation suivante [8] :

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 65.481 & 128.553 & 24.966 \\ -37.797 & -74.203 & 112 \\ 112 & -93.786 & -18.214 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \qquad (3.2.1)$$

Si les valeurs de RGB sont dans l'intervalle [0,1], les valeurs de Y sont dans l'intervalle [16,235] et celles de Cr et Cb dans l'intervalle [16,240].

Avec ce mode de couleur, les valeurs de chrominance Cr et Cb devraient être totalement indépendantes de la luminosité, vu que celle-ci est représentée par Y, et partiellement indépendantes du type de peau.
Une première expérience nous permet de vérifier l'indépendance de Cr et Cb par rapport à la luminosité. En effet nous avons illuminé une main avec deux intensités différentes (néon éteint et allumé) et nous avons enregistré les valeurs YCrCb correspondantes (voir figure 3-1).

Nous remarquons immédiatement que la valeur moyenne de Y varie fortement en fonction de l'illumination et, dans une moindre mesure, également celle de Cb et de Cr, ce qui théoriquement ne devraient pas être le cas.

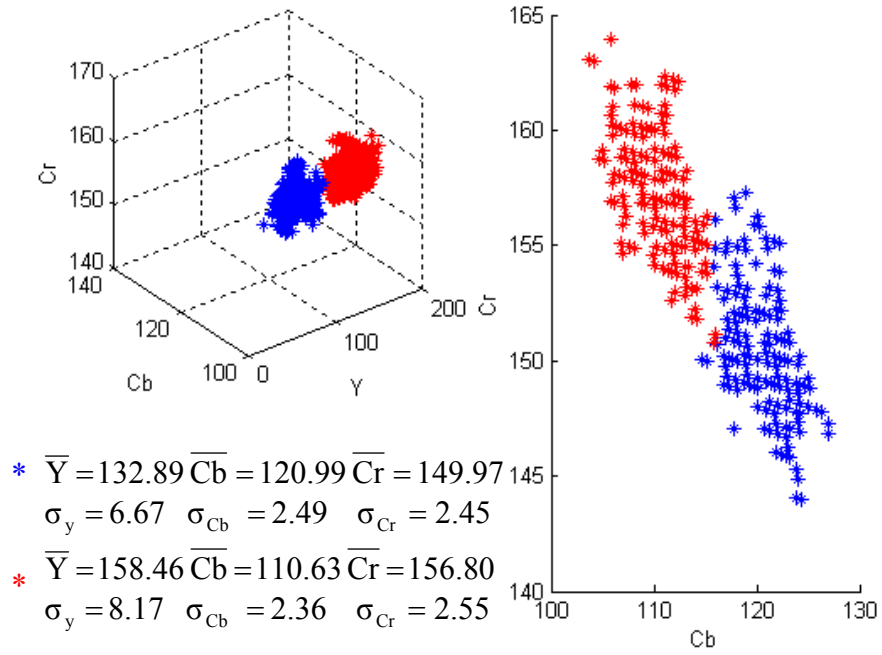

Figure 3-1: Couleur de peau sous deux illuminations différentes (néon éteint et allumé)

Une deuxième expérience nous permet de vérifier l'indépendance de Cr et Cb par rapport aux types de peau. En effet nous avons enregistré les valeurs de YCrCb correspondantes à deux visages de type différent (un visage de type caucasien et un de type négroïde) tous deux illuminés avec une intensité identique (voir figure 3-2).

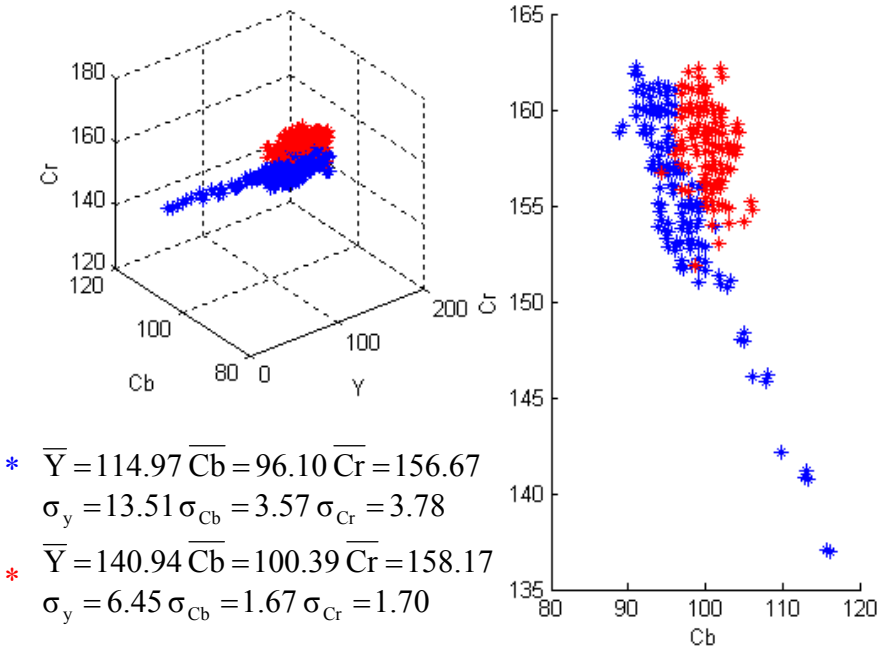

Figure 3-2: Type de peau caucasien et négroïde sous une illumination identique

A la vue de ce graphique, nous observons que les valeurs de Y varient fortement d'un type de peau à l'autre, alors que les valeurs de Cb et Cr restent relativement proche.

Finalement, nous avons fait une dernière expérience afin de vérifier si les différentes couleurs sont séparées dans l'espace YCrCb. Nous n'avons évidemment pas testé toutes les couleurs, mais nous pouvons tout de même constater que les couleurs de base, ainsi que le noir et le blanc sont suffisamment distincts de la couleur de peau.

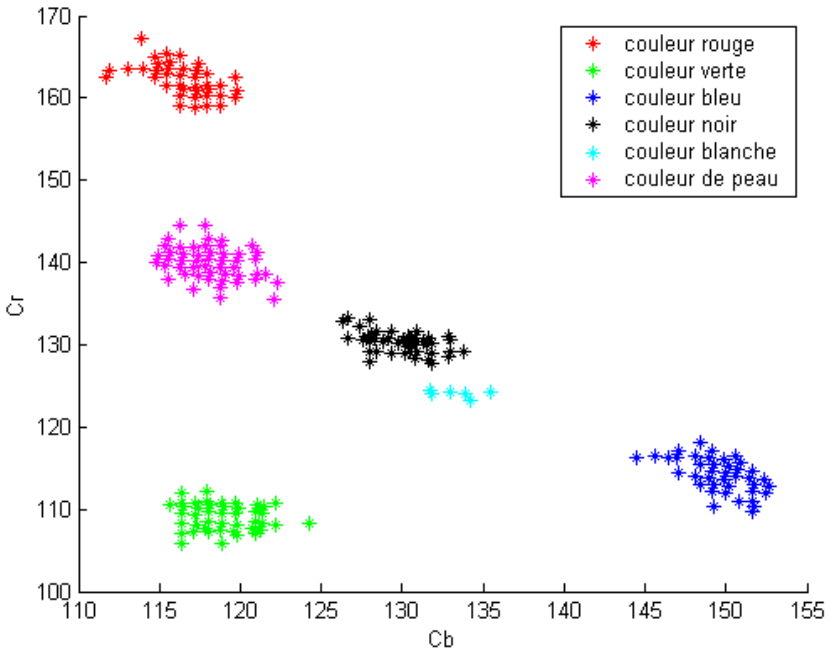

Figure 3-3 : Divers échantillons de couleur dans l'espace CbCr

Ces trois expériences nous permettent de conclure qu'en utilisant un seuil minimum et maximum constant sur les valeurs de Cr et Cb, nous serions capable de ségmenter l'image et de détecter ainsi les mains et la tête de tous les types de peau, à condition que la luminosité varie relativement peu et que la caméra utilisée soit toujours la même ; en effet les valeurs de Cr et Cb obtenues varient fortement en fonction de la caméra utilisée.

Etant donné que notre programme est destiné à une exposition, la détection de la couleur de peau doit être extrêmement robuste. Celle-ci ne doit ainsi pas être sensible aux :

- Variations de luminosité.
- Changement de caméra.
- Détérioration du capteur optique de la caméra.

C'est pourquoi nous avons décidé de calibrer la couleur de peau de chaque utilisateur afin de pouvoir adapter la ségmentation de l'image lors de chaque utilisation.

3.2.3 Calibration de la couleur de peau

Le principe de la calibration consiste à prélever les données de la couleur de peau de l'utilisateur. Pour ce faire, nous demandons à ce dernier de placer ces deux mains ainsi que sa tête dans trois carrés verts qui apparaissent sur l'image (voir figure 3-4).

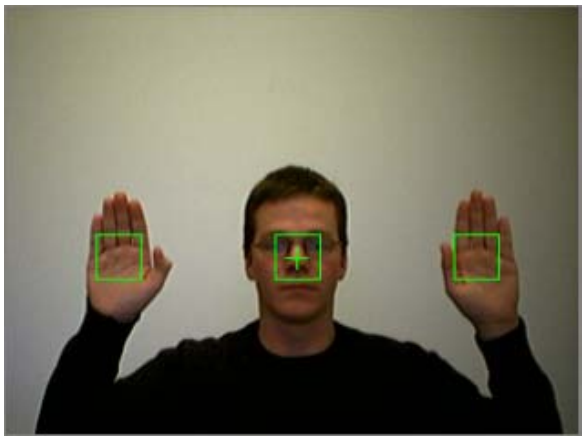

Figure 3-4 : Phase de calibration

Une fois les données prélevées, il suffit d'adapter la ségmentation de l'image en fonction de celles-ci. Une façon simple de le faire consiste à utiliser 2 seuils minimums et maximums de valeurs égales aux valeurs minimales et maximales de Cb et Cr relevées. Ainsi, pratiquement tous les pixels de couleurs de peau seront inclus dans cette zone. Mais, comme nous pouvons le voir sur la figure 3-5, de nombreux pixels n'étant pas de la couleur de peau seront également inclus dans cette zone et serons ainsi considérés comme étant de la peau.

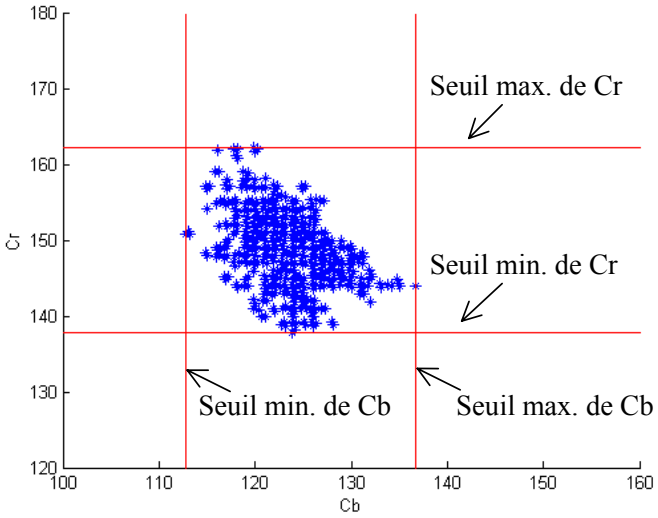

Figure 3-5: Couleur de peau d'une personne et seuils maximums et minimums de Cb et Cr

Lors des différentes calibrations, nous avons pu remarquer que la forme des données était très souvent semblable et pouvait facilement s'inscrire dans une ellipse. Ainsi, à la place de considérer tous les pixels ayant des valeurs de Cb et Cr à l'intérieur du rectangle de la figure 3-5 comme étant de la peau, nous allons construire une ellipse autour de ces données et considérer tous les pixels ayant des valeurs de Cb et Cr à l'intérieur de cette ellipse comme étant de la peau. Afin de construire cette ellipse nous procédons à une analyse en composantes principales (ACP) [9].
La première et la deuxième composantes principales de ces données nous donnent la direction des axes de l'ellipse, alors que les variances par rapport à ces composantes principales nous donnent la longueur des axes.

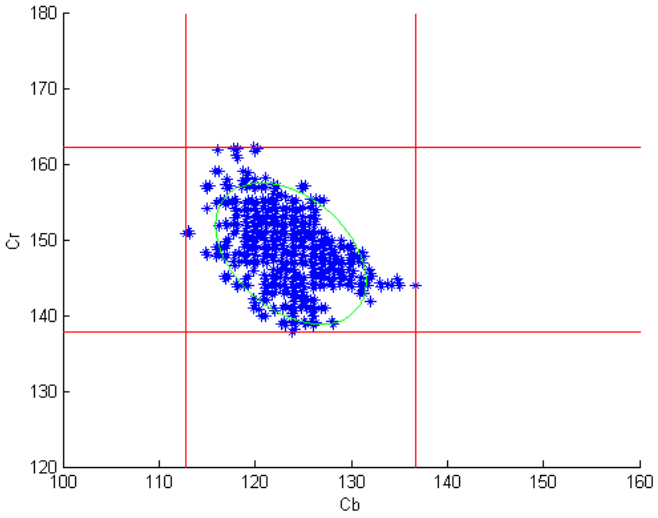

Figure 3-6: Couleur de peau d'une personne et ségmentation à l'aide d'une ellipse

Sur la figure 3-6, nous pouvons observer graphiquement le résultat de l'analyse en composantes principales. La longueur des axes de l'ellipse a été choisie expérimentalement comme étant égale à 4 fois la variance des composantes principales des données. Si les données ont une distribution gaussienne et ne sont pas corrélées selon les composantes principales, 91% des pixels représentant de la peau seront alors inclus dans cet intervalle, pour autant que les données prélevées soient réellement représentatives de la couleur de la peau. Ceci permet de diminuer drastiquement le nombre de pixels qui sont classés comme étant de la peau, alors que ce n'est pas le cas.

Remarque : Le succès de cette calibration dépend de l'utilisateur. En effet, si celui-ci n'a pas placé ses mains et sa tête à l'intérieur des carrés, cela engendre une mauvaise calibration. Pour éviter une "fausse" calibration, nous calculons les valeurs moyennes de Cb et Cr prélevées et nous validons la calibration uniquement si ces valeurs sont comprises entre 77 et 127 pour Cb et 133 et 173 pour Cr. Ces valeurs sont fixées d'après l'étude de D. Chai et K. N. Ngan [10]. Si, après plusieurs essais, la calibration n'est toujours pas réussie, nous construisons une ellipse centrée en (102,153), ayant des axes parallèles aux axes de Cb et Cr et de longueur égale à l'intervalle donné ci-dessus pour Cb et Cr. Ainsi cette ellipse couvre une région suffisamment large pour assurer la détection de la couleur de peau de n'importe quel utilisateur. En revanche, le nombre de pixels considérés comme de la peau, mais n'en étant pas, est plus important.

3.2.4 "Tracking" de la tête et des mains

Une fois l'étape de calibration terminée, nous pouvons procéder au "tracking" de la tête et des mains. Cet algorithme se décompose de la manière suivante :

- Ségmentation de l'image.
- Application d'opérateurs morphologiques.
- Détection et attribution des "blobs".

Ségmentation de l'image :
La ségmentation de l'image consiste à transformer l'image rendue par la caméra (image couleur) en une image binaire (noir-blanc). En utilisant l'ellipse construite lors de la calibration, nous considérons que tous les pixels ayant des valeurs de Cr et Cb à l'intérieur de celle-ci sont de la peau et nous les transformons en pixels de couleur blanche, alors que tous les pixels ayant des valeurs de Cr et Cb à l'extérieur de celle-ci ne sont pas de la peau et nous les transformons en pixels de couleur noire.
La figure 3-7 montre le résultat de la ségmentation. Nous remarquons immédiatement que cette image est fortement bruitée. Si nous voulions détecter des "blobs" à partir de cette image, il y en aurait beaucoup trop, il serait alors difficile de trouver trois "blobs" correspondant aux mains et à la tête.

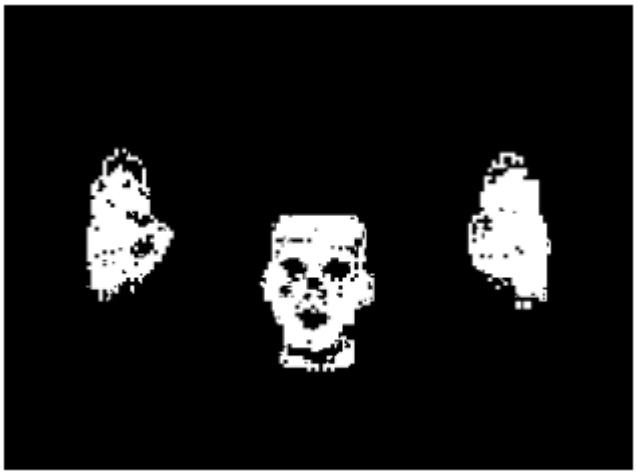

Figure 3-7: Image binarisée en fonction de la calibration

Opérateurs morphologiques :
Afin de diminuer le bruit et ainsi rendre l'image plus homogène, nous appliquons successivement deux opérateurs morphologiques qui sont la fermeture et l'ouverture.
L'opérateur morphologique de fermeture a pour but de supprimer les "petites tâches noires" à l'intérieur des objets en effectuant successivement une dilatation et une érosion, alors que l'opérateur d'ouverture est utilisé afin de supprimer les "petites tâches blanches" en effectuant successivement une érosion et une dilatation. L'image obtenue (voir figure 3-8) est beaucoup plus "propre".

Figure 3-8: Image après application des opérateurs morphologiques

Détection et attribution des "blobs" :

Le traitement précédent a séparé l'image en différents "blobs" qu'il faut maintenant trier et attribuer à la tête et aux mains. Pour ce faire, nous utilisons les fonctions suivantes de openCV:

- *cvFindContours* : cette fonction retourne les contours de chaque "blob".
- *cvMoments* : cette fonction permet de calculer le centre de gravité, ainsi que l'aire de chaque "blob" en fonction de son contour.

En connaissant le centre de gravité, ainsi que l'aire de chaque "blob", il ne reste alors plus qu'à définir quels "blobs" représentent la tête et les mains. Pour ce faire, nous définissons des zones dans l'image (voir figure 3-9). Ainsi, le "blob" le plus haut de la zone 1 est attribué à la main gauche, le "blob" le plus haut de la zone 2 est attribué à la tête et le "blob" le plus haut de la zone 3 est attribué à la main droite. Il faut remarquer que nous choisissons le "blob" le plus haut de chaque zone, car nous supposons que l'utilisateur est seul sur l'image et qu'il tient ses mains verticalement. Ainsi le "blob" supérieur est automatiquement la main ou la tête, alors que le "blob" inférieur peut être le pullover qui est semblable à la couleur de peau ou le bras de l'utlisateur lorsqu'il porte un t-shirt.
Lorsque nous attribuons pour la première fois les "blobs", nous définissons les trois zones comme sur l'image de gauche de la figure 3-9, alors que pour les attributions suivantes nous définissons les trois zones comme sur l'image de droite de la figure 3-9. Nous diminuons la taille de la zone 2 (zone dans laquelle se trouve la tête), puisque nous supposons que l'utilisateur va relativement peu bouger la tête de manière verticale (nous supposons qu'il ne va pas sauter).

Nous pouvons encore signaler que les "blobs" dont l'aire est inférieure à un seuil fixé sont considérés comme du bruit et ne sont donc pas pris en compte lors de l'attribution. Il en est de même pour les "blobs" dont l'aire est supérieure à un seuil fixé ; ceux-ci sont ne sont pas uniquement constitués de peau et ne sont donc pas pris en compte lors de l'attribution. Le seuil inférieur vaut 300 pixels, alors que le seuil supérieur vaut 8000 pixels. Ces seuils ont été fixés expérimentalement.

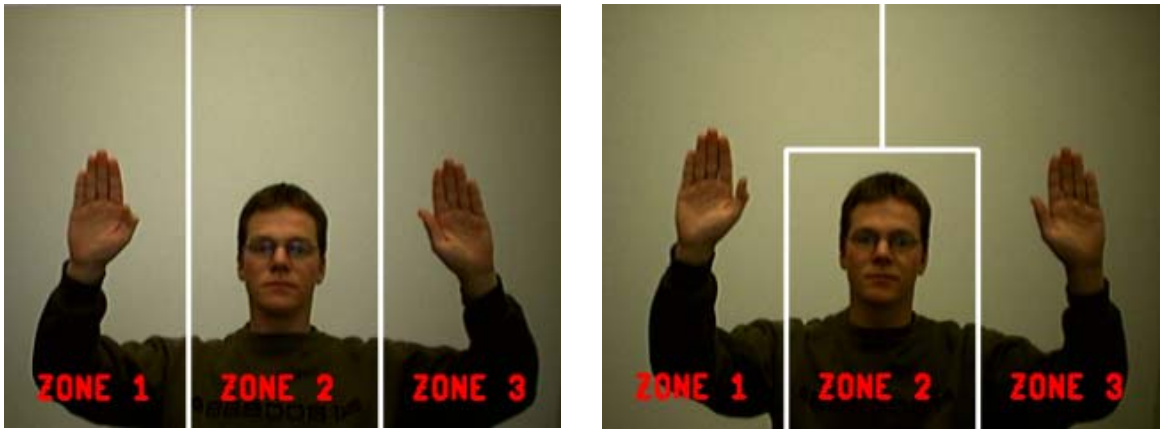

Figure 3-9 : Définition des zones

Sur la figure 3-10 nous pouvons observer le résultat final obtenu.

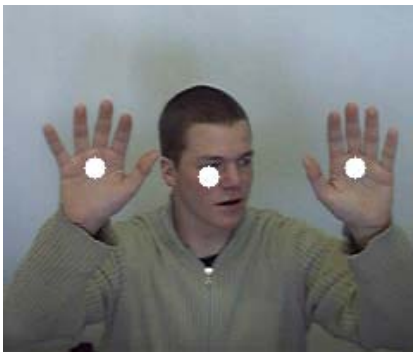

Figure 3-10 : Image du résultat obtenu

En guise de conclusion de ce chapitre, nous pouvons affirmer que le "tracking" de la tête et des mains utilisant la couleur de peau comme critère est :

- Facile à implémenter.
- Insensible à la position et la rotation de la tête et des mains.
- Sensible à l'illumination. Si après la calibration l'illumination varie fortement, les valeurs de Cr et Cb correspondant à la peau vont changer et nous ne serons donc plus capables de détecter celle-ci correctement.

3.3 Flot optique

3.3.1 Introduction

La détection du flot optique consiste à extraire l'information de la vitesse à partir d'une séquence d'images en faisant l'hypothèse que l'intensité (ou la couleur) des pixels des objets est conservée au cours du déplacement.
L'obtention de cette information est importante dans notre cas, puisque nous savons qu'une personne bouge devant un fond immobile. Ainsi, si l'on repère du mouvement dans une partie de l'image nous savons que cette partie de l'image représente la personne. Il faut tout de même veiller à ce que l'utilisateur ne crée pas d'ombres sur le fond, car si tel est le cas nous allons également détecter le mouvement de ces ombres.

Soit $I(x, y, t)$ l'intensité du pixel à la position (x, y) et au temps t. Si u(x, y) et v(x, y) sont les composantes en x et y du vecteur du flot optique en ce point, nous devons chercher un point dans l'image suivante où l'intensité sera la même au temps t + dt et à la position $(x + dx, y + dy)$:

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \qquad (3.3.1)$$

Comme cette équation n'a pas une solution unique, nous devons faire l'hypothèse suivante :

- L'intensité varie lentement avec x, y et t.

Si cette hypothèse est respectée, nous pouvons développer la partie de droite de l'équation (3.3.1) en série de Taylor :

$$I(x, y, t) = I(x, y, t) + \frac{\partial I}{\partial x} \cdot dx + \frac{\partial I}{\partial y} \cdot dy + \frac{\partial I}{\partial t} \cdot dt + e \qquad (3.3.2)$$

e contenant les termes d'ordres plus élevés. Si dt tend vers 0 et que l'on néglige e on obtient :

$$\frac{\partial I}{\partial x} \cdot \frac{dx}{dt} + \frac{\partial I}{\partial y} \cdot \frac{dy}{dt} + \frac{\partial I}{\partial t} = 0 \qquad (3.3.3)$$

Comme :

$$\frac{dx}{dt} = u \text{ et } \frac{dy}{dt} = v \qquad (3.3.4)$$

nous obtenons finalement :

$$\frac{\partial I}{\partial x} \cdot u + \frac{\partial I}{\partial y} \cdot v + \frac{\partial I}{\partial t} = 0 \qquad (3.3.5)$$

Cette équation est connue sous le nom d'**équation de contrainte du flot optique**. Afin de trouver une solution unique pour u et v, nous avons besoin de deux équations indépendantes, ce qui n'est pas le cas.

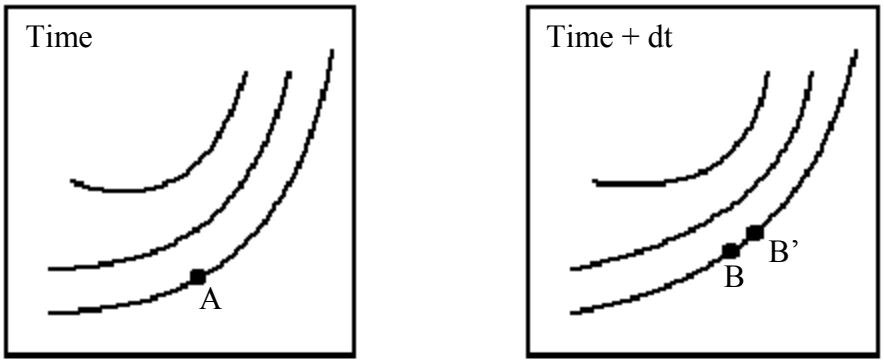

Figure 3-11 : Ambiguïté du flot optique

La figure 3-11 illustre l'ambiguïté qui subsiste en utilisant l'**équation de contrainte du flot optique**. Chaque courbe représente un contour d'intensité égale sur deux images consécutives. Ainsi, avec l'équation dont nous disposons, nous savons uniquement que le point A dans la première image s'est déplacé au point B, B' ou même un autre point sur cette même courbe dans la deuxième image.
Afin de trouver une solution unique nous avons besoin de contraintes supplémentaires. Différentes méthodes s'offrent à nous. Nous pouvons citer :

- L'algorithme de Lucas et Kanade [13].
  Cet algorithme utilise l'équation de contrainte du flot optique pour un groupe de pixels adjacents et fait l'hypothèse que ceux-ci ont une vitesse identique. Cette hypothèse nous mène à résoudre par la méthode des moindres carrés le système linéaire suivant :

$$\begin{aligned} \sum_{x\,y} W(x,y)\cdot\frac{\partial I}{\partial x}\cdot\frac{\partial I}{\partial y}\cdot u + \sum_{x\,y} W(x,y)\cdot\left(\frac{\partial I}{\partial y}\right)^2\cdot v &= -\sum_{x\,y}\frac{\partial I}{\partial y}\cdot\frac{\partial I}{\partial t} \\ \sum_{x\,y} W(x,y)\cdot\left(\frac{\partial I}{\partial x}\right)^2\cdot u + \sum_{x\,y} W(x,y)\cdot\frac{\partial I}{\partial x}\cdot\frac{\partial I}{\partial y}\cdot v &= -\sum_{x\,y}\frac{\partial I}{\partial x}\cdot\frac{\partial I}{\partial t} \end{aligned} \qquad (3.3.6)$$

  W étant une fenêtre gaussienne.
- L'algorithme de Horn et Schunck [14].
  Cet algorithme fait l'hypothèse que des pixels adjacents vont suivre une trajectoire similaire (à l'exception des pixels proches du bord des objets en mouvement), ou exprimé autrement, que le flot optique est "lisse". Mathématiquement, nous pouvons calculer l'erreur par rapport à cette hypohtèse :

$$e_s = \iint_{image} (u^2 + v^2)\,dxdy \qquad (3.3.7)$$

  De même, nous pouvons calculer l'erreur de l'équation de contrainte du flot optique :

$$e_c = \iint_{image} \left(\frac{\partial I}{\partial x}\cdot u + \frac{\partial I}{\partial y}\cdot v + \frac{\partial I}{\partial t}\right)^2 dxdy \qquad (3.3.8)$$

  Le but est finalement de trouver une solution pour u et v qui minimise $e_s + \lambda e_c$, λ étant un paramètre nommé le multiplieur de Lagrange.

Avec ces nouvelles contraintes, il est alors possible de trouver une solution unique.

3.3.2 Fonction de OpenCV

La librairie de vision que nous utilisons (OpenCV) fournit quatre fonctions calculant le flot optique :

- *cvCalcOpticalFlowHS*:
  Cette fonction calcule le flot optique pour chaque pixel entre deux images en utilisant l'algorithme de Horn et Schunck.
- *cvCalcOpticalFlowLK*:
  Cette fonction calcule le flot optique pour chaque pixel entre deux images en utilisant l'algorithme de Lucas et Kanade.
- *cvCalcOpticalFlowBM*:
  Cette fonction calcule le flot optique par bloc. Elle essaie, pour chaque bloc de la première image, de trouver un bloc correspondant dans la deuxième image.
- *cvCalcOpticalFlowPyrLK*:
  Cette fonction calcule le flot optique entre deux images pour quelques points sélectionnés au préalable. Pour ce faire, elle utilise l'algorithme itératif de Lucas et Kanade de manière pyramidale [15]. Cela signifie que cette fonction calcule, tout d'abord grossièrement, le flot optique sur une image de taille réduite, puis, en passant par plusieurs niveaux, revient progressivement à la taille de l'image de base et ainsi affine le calcul du flot optique. Les images de taille réduite sont obtenues à partir de l'image de base en leur appliquant un filtre passe-bas, puis en les sous-échantillonant. Cette méthode permet de calculer le flot optique pour des déplacements importants, tout en étant précis.

Après avoir effectué de nombreux tests qualitatifs, il s'est avéré que la fonction utilisant l'algorithme itératif de Lucas et Kanade de manière pyramidale donne les meilleurs résultats (*cvCalcOpticalFlowPyrLK*).
En effet, celle-ci fournit le flot optique pour certains points présélectionnés avec une précision inférieure au pixel ; pour cela elle utilise l'interpolation bilinéaire.

3.3.3 Application du flot optique

"Good features to track":
Puisque nous utilisons *cvCalcOpticalFlowPyrLK*, la première opération consiste à sélectionner des pixels sur l'image pour lesquels nous désirons calculer le flot optique. Afin que le résultat du flot optique soit le meilleur possible, il faut choisir des pixels "qui sont facilement retrouvables" sur l'image suivante. Pour sélectionner ces pixels nous appliquons une fonction de OpenCV : *cvGoodFeaturesToTrack*. Cette fonction applique tout d'abord sur l'image un filtre de Sobel : ce filtre est utilisé pour obtenir la dérivée de l'image selon x et y. Puis, à partir de la dérivée de cette image, elle calcule la matrice de covariance d'un groupe de pixel. Finalement, elle calcule les valeurs propres de cette matrice. Les pixels sélectionnés sont ceux ayant des valeurs propres élevées (s'ils ont des valeurs propres élevées, cela signifie qu'ils appartiennent très certainement à un contour) et ayant une distance minimale avec les pixels sélectionnés les plus proches [15 et 16].

Nous pouvons observer le résultat de cette fonction sur la figure 3-12.

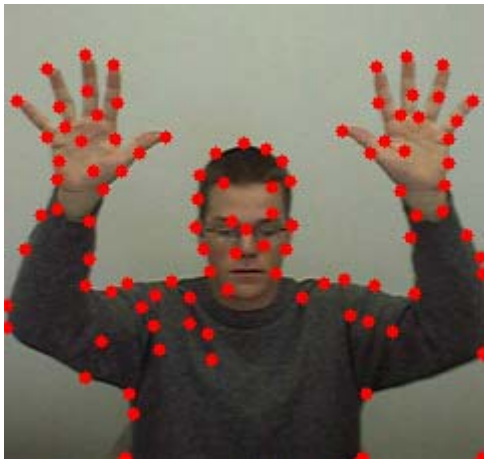

Figure 3-12 : Sélection des pixels pour le flot optique

"CvCalcOpticalFlowPyrLK" :
Une fois l'image suivante disponible, il s'agit de retrouver les pixels sélectionnés de l'image précédente dans l'image actuelle. Nous appliquons alors la fonction calculant le flot optique. Ainsi, avec deux images, nous connaissons la position, mais surtout la vitesse des pixels sélectionnés. A ce stade, nous n'avons en revanche encore aucune indication sur la position des mains et de la tête.

"Clustering" :
Pour trouver la position des mains et de la tête, nous allons opérer un "clustering" sur les pixels sélectionnés. En effet, si nous regroupons les pixels sélectionnés en fonction de la vitesse nous devrions séparer les mains et la tête, puisque tous les pixels sélectionnés d'une main bougent à la même vitesse, mais probablement à une vitesse différente de ceux de la tête et de l'autre main. A noter que pour effectuer ce "clustering" nous ne prenons pas seulement en compte la vitesse, mais également, dans une moindre mesure, la position. Ceci nous permet, par exemple, de séparer les pixels des deux mains si celles-ci bougent à la même vitesse. Ce "clustering" est réalisé en appliquant l'algorithme K-means [17].

Cet algorithme fonctionne de la manière suivante :

Initialisation :

1. Choix du nombre de "clusters" à créer. Dans notre cas, nous choisissons d'en créer trois (deux mains et une tête).
2. Choix aléatoire d'un centre pour chaque "cluster". Ce centre doit être un des pixels sélectionnés.
3. Calcul de la distance euclidienne entre un pixel et chaque centre de "cluster". Le pixel est assigné au "cluster" le plus proche (pour lequel la distance euclidienne calculée est la plus petite).

Cette distance se calcule de la manière suivante :

$$L_2 = (p_{vx} - c_{vx})^2 + (p_{vy} - c_{vy})^2 + w \cdot (p_x - c_x)^2 + w \cdot (p_y - c_y)^2 \quad (3.3.9)$$

p_{vx} = vitesse selon x du pixel.
p_{vy} = vitesse selon y du pixel.
p_x = position selon x du pixel.
p_y = position selon y du pixel.
c_{vx} = vitesse selon x du centre d'un "cluster".
c_{vy} = vitesse selon y du centre d'un "cluster".
c_x = position selon x du centre d'un "cluster".
c_y = position selon y du centre d'un "cluster".
w = pondération de la position.

Dans notre cas, nous avons choisi de donner plus d'importance à la vitesse plutôt qu'à la position, puisque nous désirons favoriser le regroupement des points en fonction de la vitesse. Les valeurs de la position s'étendent de 0 à 320 (taille max. de l'image), alors que les valeurs de la vitesse s'étendent de 0 à 120 (valeur max. mesurée). Dans notre cas, w vaut 0.01, ainsi la vitesse à environ 14 fois plus de poids que la position ((120 * 120)/(320 * 320 * 0.01)).
Ce point (3) est répété pour tous les pixels sélectionnés.

4. En fonction de l'assignement des pixels au point 3, calcule des nouveaux centres des "clusters".

Itération :

5. Pour chaque pixel sélectionné on cacule la distance euclidienne avec chaque centre de "cluster". Si la distance minimale est calculée pour un autre "cluster" que celui auquel il appartient, on réassigne le pixel à ce nouveau "cluster" et on recalcule les nouveaux centres des "clusters". Cette opération est répétée jusqu'à ce que plus aucun pixel ne soit réassigné ou jusqu'à ce que cette opération ait été répétée un nombre limite de fois. Dans notre cas, après avoir répété 20 fois le point 5, on passe à l'étape suivante, qui est l'attribution des "clusters".

Comme nous pouvons le voir sur la figure ci-dessous (voir figure 3-13), le "clustering" a permis de séparer les mains et la tête.

Figure 3-13 : Résultat du "clustering"

Attribution des "clusters"

Une fois le "clustering" terminé, nous connaissons la position des trois "clusters" qu'il s'agit d'attribuer à la main gauche, à la main droite et à la tête. Pour le faire, nous procédons de la même manière que lors de l'attribution des "blobs" pour la détection de la couleur de peau (voir chapitre 3.2.4). Ainsi, pour chaque zone, nous attribuons le "cluster" ayant la position la plus haute. Toutefois nous attribuons les "clusters" uniquement s'ils remplissent les conditions suivantes :

- Chaque "cluster" doit être composé de plus de huit points. Si un des "clusters" à moins de huit points, cela peut signifier qu'il est constitué de bruit et il peut ainsi fausser complètement la création des autres "clusters".
- Les "clusters" doivent avoir une vitesse moyenne supérieure à 1 pixel par image. Si ce n'est pas le cas nous considérons que le "clustering" s'est principalement basé sur la position des pixels et n'est donc pas représentatif du mouvement.

Si ces conditions ne sont pas remplies, les positions données par le flot optique ne sont pas mises à jour. Cet algorithme ne sera ainsi pas pris en compte lors de la fusion des informations (voir chapitre 3.5).

Puisque nous demandons à cet algorithme de créer uniquement trois "clusters", très souvent deux "clusters" regroupent les bras, alors que le troisième "cluster" regroupe la tête et le haut du corps. Ainsi la position donnée pour les bras est généralement correcte, alros que celle pour la tête est très souvent fausse. Cet algorithme n'est donc pas utilisable pour détecter la tête de l'utilisateur.

3.4 Recherche d'un modèle

3.4.1 Introduction

La détection de la tête, que ce soit par la méthode de la couleur de peau ou par la méthode du flot optique, ne donne pas des résultats suffisants. En effet, la détection de la tête par la couleur de peau ne fonctionne pas lorsque l'utilisateur effectue une rotation. Le centre de gravité du "blob" attribué à la tête ne se déplace pas suffisamment par rapport à l'amplitude de la rotation. Alors que dans le cas de la détection par la méthode du flot optique, le "cluster" de la tête regroupe la tête ainsi que le haut du corps. En utilisant ces 2 méthodes, Robota serait incapable de reproduire le mouvement de rotation réalisé par l'utilisateur. C'est pourquoi nous utilisons une troisième méthode pour suivre les mouvements de la tête. Cette méthode consiste à prendre un modèle d'une partie de la tête et de chercher ce modèle dans l'image suivante. Le problème de cette méthode, pour avoir un suivi de la tête robuste, consiste à choisir un modèle qui reste suffisamment invariant lors de déplacements et de rotations du visage. D'après [18], le nez est la partie du visage qui s'y prête le mieux. Dû à sa symétrie et à sa forme convexe, le nez est toujours visible par la caméra et reste pratiquement identique lors de rotations ou lorsque celui-ci avance et recule par rapport à la caméra. En effet, si nous choisissions un œil, celui-ci serait, lors de rotation importante, invisible sur l'image.

3.4.2 Calibration

La première étape consiste à construire le modèle du nez. Ce modèle est construit pendant la phase de calibration, c'est à dire au même instant que la calibration de la couleur de la peau. Pour ce faire, on demande simplement à l'utilisateur de placer le bout de son nez au centre de la croix se trouvant dans le carré central (voir figure 3-4), puis on copie dans une image de taille 9x9 pixels les 9x9 pixels ayant pour centre la croix. Cette image constitue notre image de référence.

3.4.3 "Tracking"

Une fois la calibration effectuée, il s'agit de suivre le nez au fur et à mesure de ses déplacements en fonction de l'image de référence. Pour ce faire, nous balayons l'image actuelle dans une zone de taille 81x31 centrée en l'ancienne position du nez avec l'image de référence, et pour chaque position de l'image de référence nous calculons la corrélation entre l'image de référence et la zone de l'image actuelle couverte par cette image de référence (voir figure 3-14).

La corrélation utilisée est la suivante :

$$R(x,y) = \frac{\sum_{x'=0}^{8}\left(\sum_{y'=0}^{8}\left(T(x',y')\cdot I(x+x',y+y')\right)\right)}{\sqrt{\left(\sum_{x'=0}^{8}\left(\sum_{y'=0}^{8}T(x',y')^2\right)\right)\cdot\left(\sum_{x'=0}^{8}\left(\sum_{y'=0}^{8}I(x+x',y+y')^2\right)\right)}} \qquad (3.4.1)$$

R = image résultat (73x23).
T = image de référence (9x9).
I = image dans laquelle on recherche le nez (81x31).

La position à laquelle la corrélation est la plus élevée sera considérée comme la nouvelle position du nez.

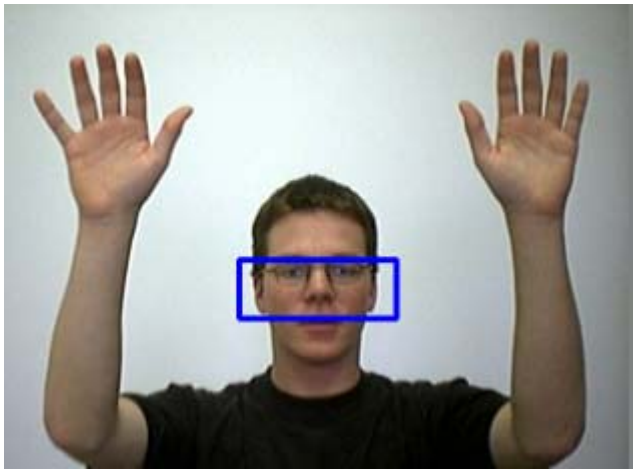

Figure 3-14 : Recherche du nez à l'intérieur de la fenêtre bleu

Remarque : Lors de l'acquisition de la première image, nous ne pouvons pas rechercher le nez en fonction de l'ancienne position. Dans ce cas, soit nous cherchons le nez dans une fenêtre ayant comme centre la position de la tête détectée à l'aide de la couleur de peau, soit nous cherchons le nez dans toute l'image dans le cas où la détection de la tête à l'aide de la couleur de peau ne fonctionne pas.

Sur la figure 3-15, nous pouvons observer les résultats obtenus avec les différentes méthodes, lorsque l'utilisateur effectue un mouvement de ± 90° selon la direction horizontale.
En comparant les trois méthodes, nous observons que :

- La méthode du flot optique ne fonctionne pas pour la détection de la tête, comme cela était prévu.
- La méthode de détection par la couleur de peau fonctionne, mais donne une amplitude de mouvement atténuée par rapport à la réalité.
- La méthode de recherche du nez fonctionne relativement bien, à l'exception des positions extrêmes. En effet lorsque la rotation de la tête approche ± 90°, le nez varie fortement par rapport à l'image de référence et ainsi la corrélation est plus élevée pour d'autre partie du visage, ce qui est logique puisqu'en position extrême le bout du nez n'est plus entouré de peau, mais du "background".

Pour remédier à ce problème, nous multiplions les résultats de la corrélation par un facteur qui est fonction de la distance. Ce facteur varie de 1 pour l'ancienne position du nez, à 0.95 pour les positions les plus éloignées de l'ancienne position du nez. Nous donnons ainsi plus de poids aux résultats proche de l'ancienne position.
La figure 3-16 montre les résultats ainsi obtenus. On remarque immédiatement que la méthode de recherche du nez fonctionne mieux.

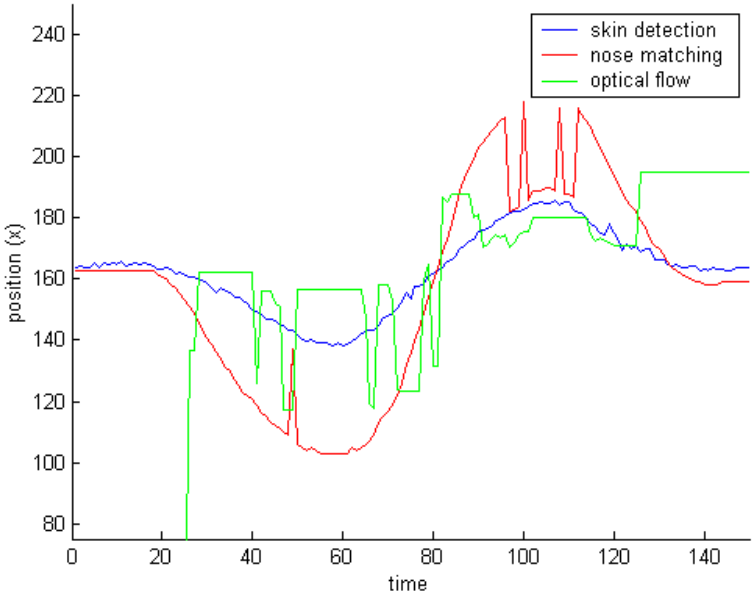

Figure 3-15 : Détection du nez sans pondération en fonction de la distance

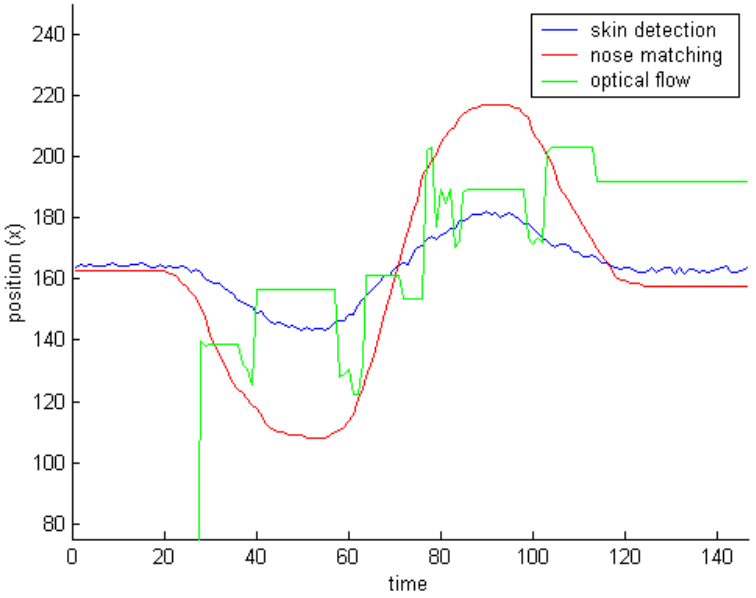

Figure 3-16 : Détection du nez avec pondération en fonction de la distance

Il faut encore noter qu'avant la mise à jour de la position du nez, nous calculons la distance entre cette dernière et la position de la tête donnée par l'algorithme de détection de la couleur de la peau. Si l'écart est trop important nous savons que l'une des deux positions est fausse, c'est pourquoi nous réinitialisons la recherche du nez.

La qualité du résultat de cet algorithme dépend fortement de la calibration. En effet, si l'utilisateur a réellement placé son nez au centre de la croix lors de la calibration, l'image de référence obtenue permet de "tracker" le nez dans les positions extrêmes. En revanche si l'image de référence n'est pas idéale, le nez peut être rapidement perdu.

3.5 Fusion des informations

Puisque nous avons trois algorithmes qui "trackent" la position des membres, il s'agit de fusionner les différentes informations, afin d'avoir un résultat final s'approchant le plus possible de la position réelle des membres.

Main gauche et main droite :
La position de la main gauche et de la main droite est fournie par l'algorithme de détection de la couleur de la peau et l'algorithme de flot optique. La fusion de ces deux algorithmes se fait de la manière suivante :

- Si les deux algorithmes fournissent une position pour l'image actuelle, le résultat final est alors simplement une moyenne entre les résultats des deux algorithmes.
- Si seul l'algorithme de la détection de la couleur de la peau fournit une position pour l'image actuelle, le résultat est alors celui de cet algorithme uniquement.
- Si seul l'algorithme du flot optique fournit une position pour l'image actuelle, le résultat est alors celui de cet algorithme uniquement.
- Si aucun des algorithmes ne fournit un résultat, la position obtenue de l'image précédente est conservée.

Sur la figure 3-17, nous pouvons observer le résultat de cette fusion pour la position selon y de la main gauche.

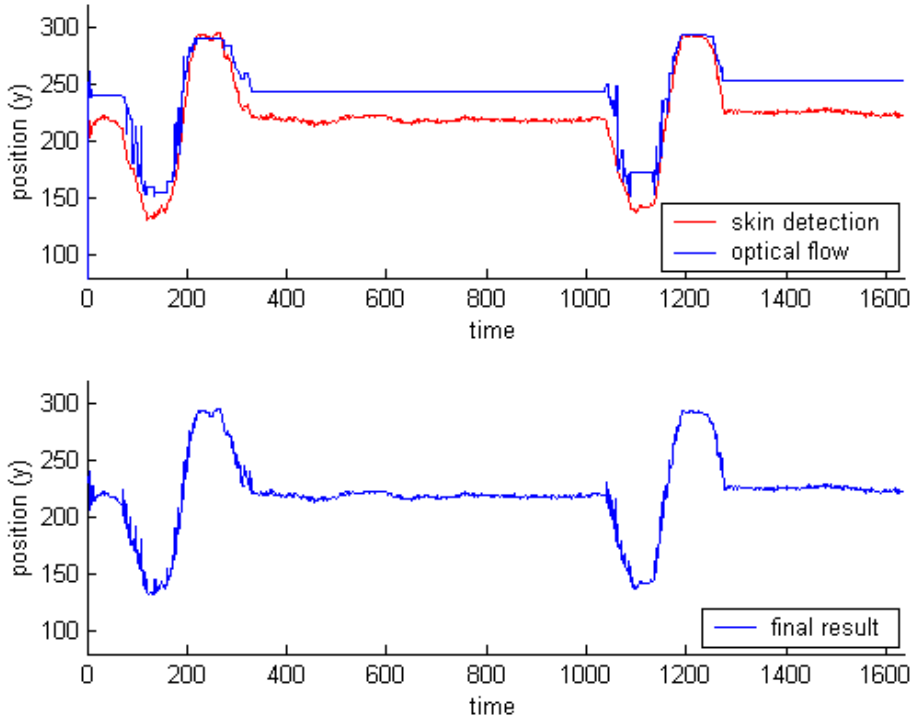

Figure 3-17 : Résultat de la détection de la couleur de peau, du flot optique et final pour la main gauche selon y

Entre le temps 400 et le temps 1000 environ, l'utilisateur a gardé sa main gauche statique. Ainsi, l'algorithme du flot optique n'a pas rafraîchit la position de ce membre, ce qui est correct vu que la vitesse minimale d'un pixel par image n'était pas atteinte (voir chapitre 3.3.3). C'est pourquoi, pendant ce temps, le résultat final a été composé uniquement par le résultat de l'algorithme de détection de la couleur de la peau.

Tête :

La position de la tête est fournie par les trois alogrithmes. Mais comme nous l'avons vu, l'algorithme de flot optique fournit de mauvais résultats pour cette partie du corps. Ainsi, nous utilisons uniquement l'algorithme de détection de la couleur de la peau et l'algorithme de recherche de modèle du nez pour obtenir le résultat final. La fusion se fait de la manière suivante :

- Si l'algorithme de recherche du modèle du nez fournit un résultat pour l'image actuelle, nous prenons ce résultat comme résultat final.
- Si seul l'algorithme de détection de la couleur de la peau fournit un résultat pour l'image actuelle, nous prenons ce résultat comme résultat final.
- Si aucun de deux algorithmes ne fournit une solution pour l'image actuelle, nous conservons le résultat fournit par l'image précédente.

Sur la figure 3-18, nous pouvons observer le résultat de cette fusion pour la position selon x de la tête.

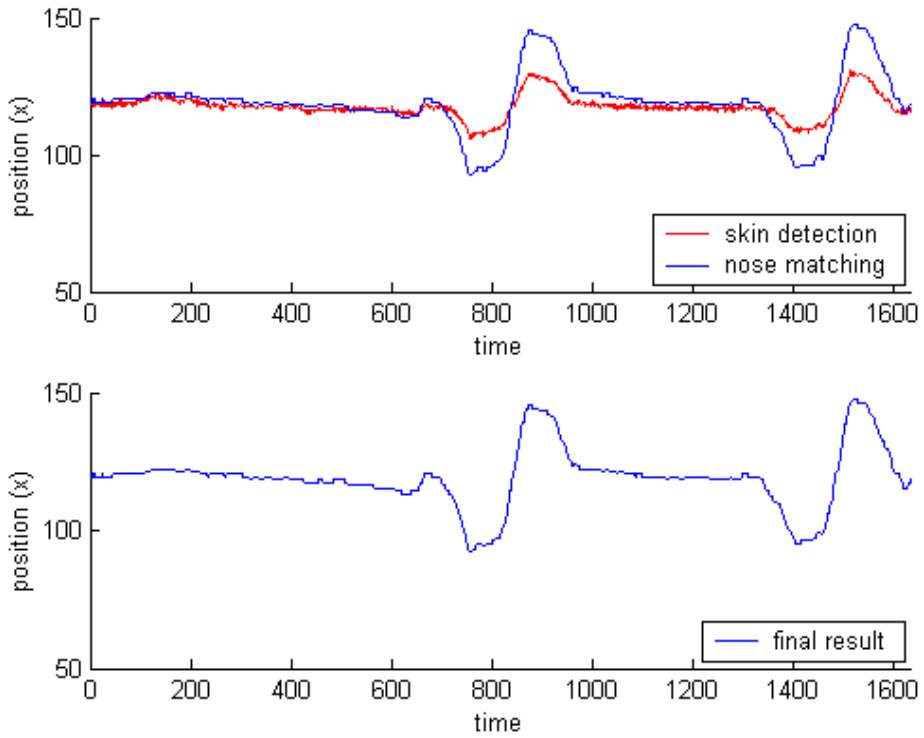

Figure 3-18 : Résultat de la détection de la couleur de la peau, de la recherche du modèle et final pour la tête selon x

Dans ce cas, l'algorithme de recherche de modèle du nez a fourni un résultat pour toutes les images. C'est pourquoi le résultat final est uniquement composé par cet algorithme.

3.6 Problèmes non résolus

Comme nous le verrons dans le chapitre traitant des résultats, le module de détection du haut du corps fonctionne relativement bien. Cependant, il faut tout de même être conscient de ses limites. En effet, nous pouvons répertorier une série d'actions qui ne permettent plus de "tracker" les mains et la tête de manière correcte :

- Passage d'une main ou de la tête dans une autre zone. En effet, si un des membres change de zone, il sera identifié comme étant le membre de la zone correspondante. Pour remédier à ce problème, il faudrait supprimer les zones et identifier les différents membres en fonction de la distance les séparant de leur ancienne position. Ceci est, dans notre cas, difficile à réaliser, car notre application est trop dynamique par rapport à la fréquence de rafraîchissement de l'image. En effet, la position d'une main peut varier de 120 pixels entre deux images consécutives, soit plus d'un tiers de l'image.
- Lors de fortes rotations de la tête, la recherche du modèle du nez n'est parfois plus capable de retrouver celui-ci sur l'image. Elle choisit une autre partie du visage comme étant le nez. En effet, dans les positions extrêmes le nez n'est plus entouré de peau, mais est entouré du "background", il est ainsi plus difficile à détecter. Nous avons tout de même réussi à limiter ce problème en évitant les sauts importants en donnant plus de poids aux points les plus proches de l'ancienne position (voir chapitre 3.4.3). Une fois que le nez est perdu, il est malheureusement impossible de s'en rendre compte. Le seul instant auquel on réinitialise la recherche est quand le nez est éloigné d'une distance suffisante de la position de la tête détectée à l'aide de la couleur de peau. A cet instant seulement, nous pouvons être certain que la recherche du nez est fausse.

4 Module de détection des jambes

4.1 Introduction

Le module de détection des jambes a pour but de donner à Robota l'information sur la position des jambes de l'utilisateur. Pour ce faire, nous pouvons utiliser différentes méthodes :

- Flot optique.
- Ségmentation de l'image en fonction des couleurs.

Notre premier choix s'est porté sur l'application du flot optique. Le flot optique permet de détecter les jambes de l'utilisateur, puisque nous savons que celui-ci est seul devant la caméra. Ainsi, dès que nous détectons un mouvement nous savons qu'il s'agit soit de la jambe gauche, soit de la jambe droite. En revanche cette méthode nécessite une quantité de calculs importante, ce qui diminue considérablement la fréquence de rafraîchissement de notre système, puisque qu'une caméra est déjà utilisée pour "tracker" le haut du corps. C'est pourquoi nous avons finalement opté pour la deuxième solution, la ségmentation de l'image en fonction de la couleur.

4.2 Ségmentation de l'image en fonction de la couleur

L'utilisateur évolue devant un "background" de couleur uniforme, qui, dans le cadre de l'exposition de la Cité de l'espace, est blanc. Ainsi, nous pouvons affirmer que tout ce qui est différent de la couleur blanche constitue l'utilisateur. Il est alors relativement facile de ségmenter l'image en fonction de la couleur du "background".

4.2.1 Calibration de la couleur du "background"

Comme nous connaissons la couleur du "background", il n'est pas nécessaire d'effectuer une calibration de cette dernière. Mais, pour une question de robustesse, nous effectuons tout de même une calibration. Ainsi le système est indépendant :

- Du changement de luminosité.
- Du changement de couleur du "background".
- De la détérioration du capteur optique de la caméra.

Comme nous pouvons le voir sur la figure 4-1, l'utilisateur doit placer ses 2 jambes à l'intérieur des carrés verts et ne pas les introduire à l'intérieur des carrés rouges, puisque ce sont les pixels à l'intérieur de ceux-ci qui sont utilisés pour la calibration. Finalement nous calculons les valeurs R, G et B moyennes des pixels. Il faut noter que, dans ce cas, nous n'utilisons pas le mode de couleur YCrCb. En effet, cela n'est pas utile, puisque nous devons uniquement différencier l'utilisateur d'un "background" blanc. Nous n'avons donc pas besoin de mettre en évidence la couleur de peau.

Figure 4-1 : Calibration des jambes

4.2.2 "Tracking" des jambes

Une fois l'étape de calibration terminée, nous pouvons procéder au "tracking" des jambes. Cet algorithme se décompose de la manière suivante :

- Ségmentation de l'image.
- Application d'opérateurs morphologiques.
- Détection et attribution des "blobs".

Ségmentation de l'image

Lors de l'étape de calibration nous avons calculé les valeurs R, G et B moyennes du "background". En prenant des marges supérieures et inférieures par rapport à ces valeurs, nous pouvons représenter un cube dans l'espace RGB : les marges ont été fixées expérimentalement et valent 30. La ségmentation de l'image s'effectue alors de la manière suivante :

- Si les valeurs R, G et B d'un pixel sont à l'intérieur de ce cube, nous considérons que ce pixel fait partie du "background" et nous le colorions en noir.
- Si les valeurs R, G et B d'un pixel sont à l'extérieur de ce cube, nous considérons que ce pixel est l'utilisateur et nous le colorions en blanc.

La figure 4-2 illustre le résultat obtenu.

Figure 4-2 : Image binarisée en fonction de la calibration

Opérateurs morphologiques

Comme dans le cas de la détection de la couleur de peau, nous appliquons successivement les opérateurs morphologiques de fermeture et d'ouverture, afin de réduire le bruit et de diminuer le nombre de "blobs". La figure 4-3 illustre le résultat obtenu.

Figure 4-3 : Image après utilisation des opérateurs morphologiques

Détection et attribution des "blobs"

De manière identique à ce que nous avons déjà fait pour détecter les "blobs" de la couleur de peau, nous détectons les "blobs" représentant les jambes. Par la suite, il s'agit d'attribuer un "blob" à la jambe gauche et un "blob" à la jambe droite. Pour ce faire, nous séparons l'image de manière verticale en deux parties. Dans la partie gauche de l'image nous attribuons le "blob" inférieur à la jambe gauche et dans la partie droite à la jambe droite, à condition que celui-ci ait une aire supérieure à un seuil fixé : ce seuil a été fixé expérimentalement et vaut 200. Si ce n'est pas le cas, nous considérons ce "blob" comme étant du bruit et nous choisissons le "blob" supérieur. Sur la figure 4-4, nous pouvons observer le résultat obtenu. Le point vert indique l'extrémité inférieure du "blob" représentant la jambe gauche et le point bleu indique l'extrémité inférieure du "blob" représentant la jambe droite. Ces deux points indiquent la position des jambes.

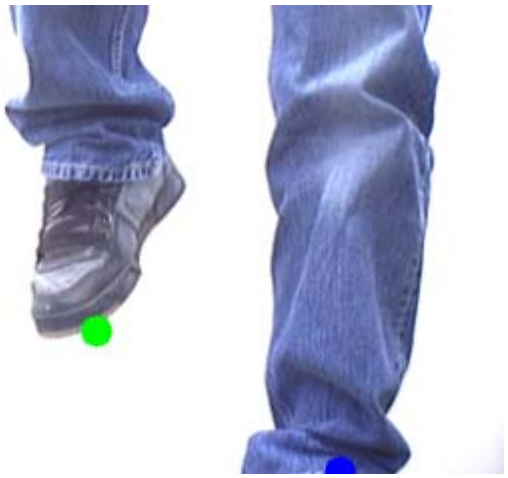

Figure 4-4 : Résultat final

Il faut noter que si l'utilisateur positionne mal ses jambes, il se peut que la jambe droite touche la jambe gauche. Dans ce cas, il est fort probable que l'on ne détecte qu'un seul "blob" sur l'image. Nous appliquons alors l'opérateur morphologique d'érosion, jusqu'à ce que nous obtenions deux "blobs" distincts. Si, malgré l'application de l'opérateur d'érosion, nous n'obtenons pas deux "blobs" distincts, nous abandonnons et nous ne rafraîchissons pas la position des jambes pour cette image. Ci-dessous nous pouvons observer un cas où, en appliquant l'opérateur d'érosion, nous avons réussi à constituer deux "blobs".

Figure 4-5 : Images avant et après l'érosion

4.3 Problèmes non résolus

Le "tracking" des pieds fonctionnent bien, cependant il subsiste tout de même quelques situations avec lesquelles il y a des problèmes :

- Si l'utilisateur a des pantalons, ainsi que des chaussures de la même couleur ou d'une couleur relativement proche de celle du "background", il sera impossible d'identifier la position de ses jambes.
- Si l'utilisateur porte une jupe, nous aurons, après la ségmentation, qu'un seul "blob" sur l'image. Malgré les érosions supplémentaires que nous appliquons, il est impossible de séparer ce "blob" en deux parties.
- Si l'utilisateur place ses deux pieds dans la même zone, l'algorithme ne va détecter qu'un seul pied.

5 Emplacement des caméras

5.1 Caméra pour le haut du corps

Angles d'ouverture :
Afin de pouvoir calculer l'emplacement adéquat des caméras, il s'agit dans un premier temps de calculer leurs angles d'ouverture.
Pour ce faire, nous avons placé la caméra de manière perpendiculaire à une distance connue d'un plan (z = 0.365 m) et nous avons mesuré son champ de vision selon x et y.

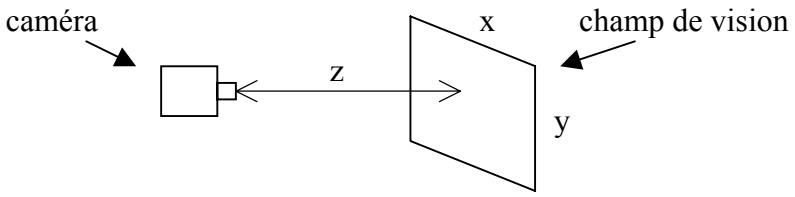

Figure 5-1 : Schéma du montage pour calculer les angles d'ouverture

Nous avons obtenus :
x = 28.5 cm
y = 22 cm

Nous trouvons alors immédiatement les demi-angles d'ouverture de la caméra :

$$\alpha_x = \arctan(\frac{x/2}{z}) = 21.33° \quad (5.1.1)$$

$$\alpha_y = \arctan(\frac{y/2}{z}) = 16.77° \quad (5.1.2)$$

Position et champ de vision :
Lors de l'exposition, l'utilisateur se trouve à 1m de Robota et est positionné sur un podium de 5 cm de haut. C'est pourquoi, en accord avec la Cité de l'espace, il a été décidé d'avoir un champ de vision selon y de 0.8 m allant de 1.25 m à 2.05 m. Ainsi les utilisateurs ayant une taille comprise entre 1.5 m et 1.9 m pourront utiliser notre système sans problème.
Comme le demi-angle d'ouverture de la caméra est plus important selon x (α_x = 21.33°), nous plaçons la caméra avec un angle de 90° par rapport à sa position standard. Ainsi nous obtenons un plus grand champ de vision selon y, ce qui se justifie pour notre système, puisque l'utilisateur va principalement bouger ses bras selon cette direction.

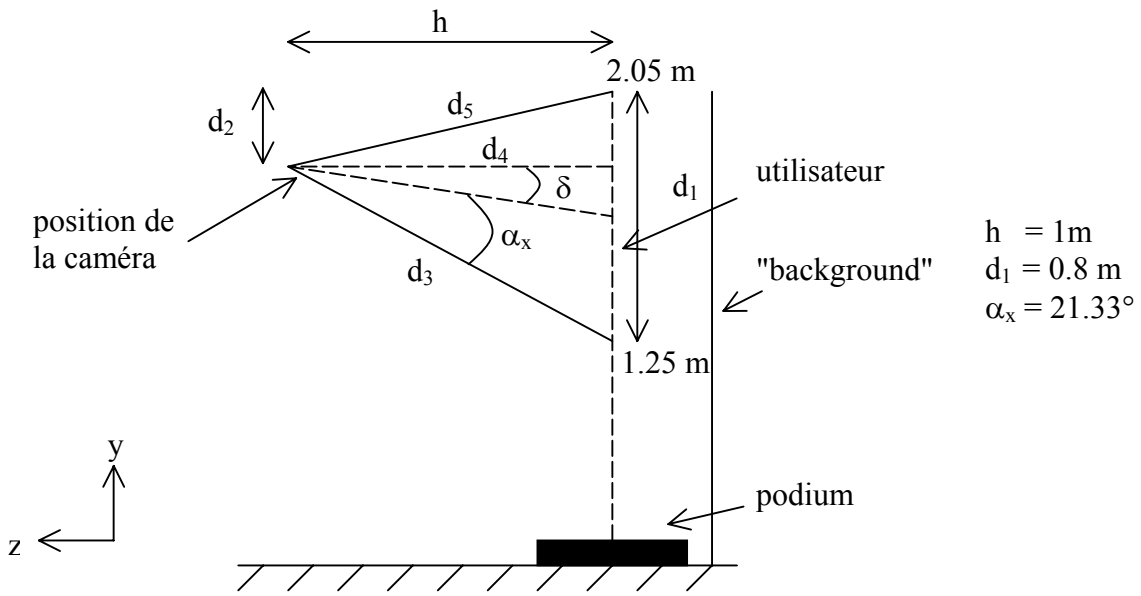

Figure 5-2 : Schéma du positionnement de la caméra pour le haut du corps

Afin de trouver d_2 et δ, il suffit de résoudre ces deux équations :

$$\tan(\alpha_x - \delta) = \frac{d_2}{h} \qquad (5.1.3)$$

$$\tan(\alpha_x + \delta) = \frac{d_1 - d_2}{h} \qquad (5.1.4)$$

Après résolution, nous trouvons :

$$\tan\delta = \pm\sqrt{\frac{\frac{d_1}{h} - 2\cdot\tan\alpha_x}{2\cdot\tan\alpha_x + \frac{d_1}{h}\cdot\tan^2\alpha_x}} \qquad (5.1.5)$$

En choisissant tan δ positif et en utilisant les valeurs numériques, nous trouvons :

$$\delta = 8.26°$$

$$d_2 = 0.23\,\text{m}$$

Ainsi la caméra doit se trouver à une hauteur de 1.82 m et faire un angle de 8.26° par rapport à la verticale, pour remplir les spécificités de l'exposition.

Nous avons imposé le champ de vision selon y (0.8 m). A titre indicatif, nous calculons également le champ de vision que nous obtenons selon x. Pour ce faire, nous devons connaître d_3, d_4 et d_5 :

$$d_3 = \frac{h}{\cos(\alpha_x + \delta)} = 1.15\,\text{m} \qquad (5.1.6)$$

$$d_4 = 1\,\text{m} \qquad (5.1.7)$$

$$d_5 = \frac{h}{\cos(\alpha_x - \delta)} = 1.03\,\text{m} \qquad (5.1.8)$$

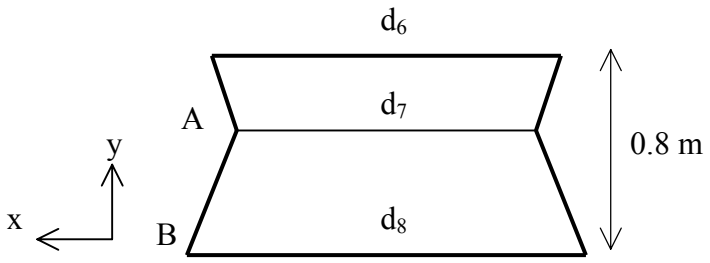

Figure 5-3 : Schéma du champ de vision

Ainsi nous pouvons calculer le champ de vision selon x. Celui-ci dépend évidemment de la hauteur à laquelle on le calcule. C'est pourquoi nous allons calculer le minimum (d_7), ainsi que les deux extrêmes (d_6 et d_8) :

$$d_6 = 2 \cdot \tan\alpha_y \cdot d_5 = 0.62\,\text{m} \qquad (5.1.9)$$

$$d_7 = 2 \cdot \tan\alpha_y \cdot d_4 = 0.60\,\text{m} \qquad (5.1.10)$$

$$d_8 = 2 \cdot \tan\alpha_y \cdot d_3 = 0.69\,\text{m} \qquad (5.1.11)$$

Résolution :

La résolution n'est pas identique en tout point de l'image. Ainsi, la résolution selon x (Rx) varie en fonction de y. En effet, comme nous venons de le calculer, le champ de vision selon x varie en fonction de y, alors que le nombre de pixels ne varie pas. Nous pouvons à titre indicatif calculer les résolutions maximale et minimale :

$$\text{Rx max} = \frac{d_7}{240} = 2.5\,\text{mm/pixel} \qquad (5.1.12)$$

$$\text{Rx min} = \frac{d_8}{240} = 2.9\,\text{mm/pixel} \qquad (5.1.13)$$

La résolution selon y (Ry) varie également en fonction de y. Nous pouvons calculer à titre indicatif les résolutions minimale et maximale. Puisque nous connaissons l'angle d'ouverture α_x ainsi que le nombre de pixel selon y (320), nous pouvons calculer l'angle d'ouverture d'un pixel :

$$\alpha_{px} = \frac{2 \cdot \alpha_x}{320} = 0.13° \qquad (5.1.14)$$

Ainsi la résolution d'un pixel à la hauteur de A (voir figure 5-3) est de :

$$\text{Ry max} \cong \frac{\alpha_{px} \cdot \pi}{180} \cdot d_4 = 2.3\,\text{mm/pixel} \qquad (5.1.15)$$

alors que la résolution d'un pixel à la hauteur de B (voir figure 5-3) est de :

$$\text{Ry min} \cong \frac{\alpha_{px} \cdot \pi}{180} \cdot d_3 = 2.6\,\text{mm/pixel} \qquad (5.1.16)$$

Pour calculer Ry min et Ry max, nous avons approximer le plan de projection du pixel par un arc de cercle.

5.2 Caméra pour le bas du corps

Angles d'ouverture :
Puisque nous utilisons une caméra identique à celle utilisée pour le haut du corps, nous avons les mêmes caractéristiques techniques, donc les mêmes angles d'ouverture.

Position et champ de vision :
Le visiteur est positionné à 1m de Robota et se trouve sur un podium de 5cm de haut. Pour le module de détection du bas du corps il est important de n'avoir que le "background" et le visiteur dans le champ de vision de la caméra. C'est pourquoi nous avons décidé de placer la caméra de la manière suivante :

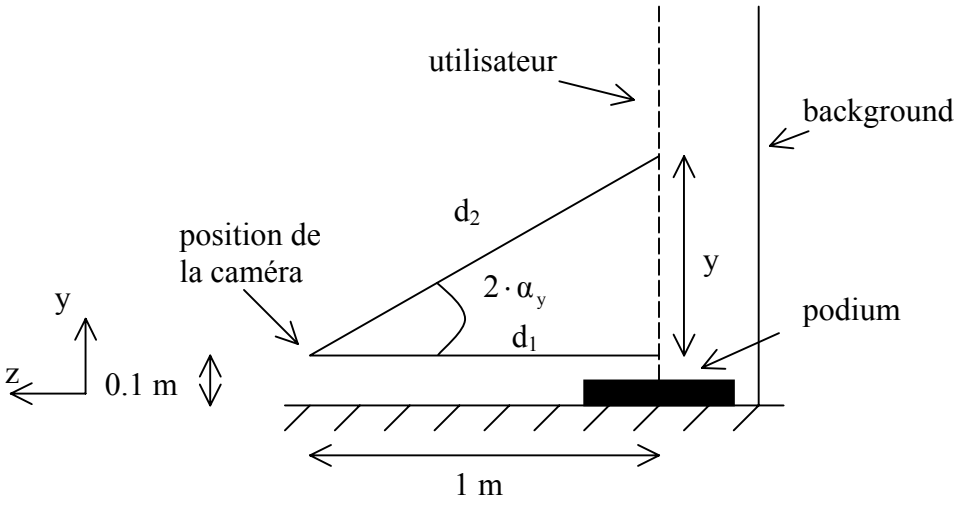

Figure 5-4 : Schéma du positionnement de la caméra pour le bas du corps

Pour s'assurer de ne pas avoir le podium dans le champ de vision, il faut incliner la caméra verticalement avec un angle égal au demi-angle d'ouverture de la caméra (16.77°).
Calculons le champ de vision que nous obtenons selon y :

$$y = \tan(2\alpha_y) = 0.66\,\text{m} \qquad (5.2.1)$$

Nous pouvons également calculer à titre indicatif le champ de vision que nous obtenons selon x. Pour ce faire, nous devons calculé d_2 :

$$d_2 = \frac{y}{\sin(2 \cdot \alpha_y)} = 1.19\,\text{m} \qquad (5.2.2.)$$

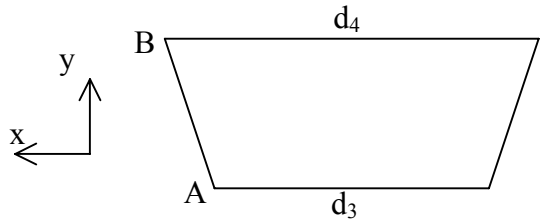

Figure 5-5 : Schéma du champ de vision

Ainsi nous pouvons calculer le champ de vision minimal (d_3) et le champ de vision maximal (d_4) :

$$d_3 = 2 \cdot \tan \alpha_x = 0.78\,\text{m} \qquad (5.2.3)$$

$$d_4 = 2 \cdot \tan \alpha_x \cdot d_3 = 0.93\,\text{m} \qquad (5.2.4)$$

Résolution :

Ayant calculé le champ de vision, nous pouvons maintenant calculer la résolution selon x et y. La résolution selon x (Rx) varie en fonction de y. Nous pouvons ainsi calculer, à titre indicatif, les résolutions minimale et maximale :

$$\text{Rx min} = \frac{d_4}{320} = 2.9\,\text{mm/pixel} \qquad (5.2.5)$$

$$\text{Rx max} = \frac{d_3}{320} = 2.4\,\text{mm/pixel} \qquad (5.2.6)$$

La résolution selon y (Ry) varie également en fonction de y. Nous pouvons, à titre indicatif, calculer les résolutions minimale et maximale. Puisque nous connaissons l'angle d'ouverture α_y ainsi que le nombre de pixel selon y (240), nous pouvons calculer l'angle d'ouverture d'un pixel :

$$\alpha_{py} = \frac{2 \cdot \alpha_y}{240} = 0.14° \qquad (5.2.7)$$

Ainsi, la résolution d'un pixel à la hauteur de A (voir figure 5-5) est de :

$$\text{Ry max} \cong \frac{\alpha_{py} \cdot \pi}{180} \cdot d_1 = 2.4\,\text{mm/pixel} \qquad (5.2.8)$$

alors que la résolution d'un pixel à la hauteur de B (voir figure 5-5) est de :

$$\text{Ry min} \cong \frac{\alpha_{py} \cdot \pi}{180} \cdot d_2 = 2.9\,\text{mm/pixel} \qquad (5.2.9)$$

Pour calculer Ry min et Ry max, nous avons approximer le plan de projection du pixel par un arc de cercle.

6 Tests et résultats

6.1 Fréquence de rafraîchissement

La fréquence de rafraîchissement maximale dont nous disposons avec les webcams Logitech® QuickCam® Pro 4000 est de 30 Hz. Celle-ci est suffisante, puisqu'il suffit d'avoir une fréquence de rafraîchissement de 25 Hz pour que l'homme ait l'impression d'avoir un flux d'image continu.
Il est alors intéressant de mesurer la fréquence de rafraîchissement obtenue lorsque les différents algorithmes sont utilisés séparément, puis lorsqu'ils sont utilisés simultanément.

	Fréquence de rafraîchissement [Hz]
Couleur de peau	25
Flot optique	18
Recherche du nez	30
Détection des jambes	30
TOTAL	14

Nous constatons que l'algorithme de recherche du nez et l'algorithme de détection des jambes ne diminuent pas la fréquence de rafraîchissement. En revanche, l'algorithme détectant la couleur de peau et l'algorithme de flot optique ralentissent le système. Finalement, lorsque tous ces algorithmes tournent simultanément, nous obtenons une fréquence de rafraîchissement de 14 Hz.
Il faut bien entendu remarquer que plus la fréquence de rafraîchissement est basse, plus la performance du "tracking" diminue. Par exemple, pour l'algorithme de recherche du nez, si le temps écoulé entre deux images est grand, cela signifie que le nez peut s'être déplacé d'une plus grande distance. Ainsi, la fenêtre de recherche du nez doit être plus grande, ce qui rend la recherche plus difficile.

La fréquence de rafraîchissement a été mesurée lorsque les deux modules de détection uniquement sont utilisés. Celle-ci va encore diminuer lorsque ces deux modules seront intégrés aux modules de l'intelligence artificielle, de la reconnaissance vocale et de la parole développés par Andres Lopez.

6.2 "Tracking"

Afin de tester les performances des différents algorithmes de "tracking" du haut du corps (détection de la couleur de peau, flot optique, recherche du modèle du nez), nous demandons à un utilisateur de suivre avec ses mains et sa tête une trajectoire prédéfinie et affichée à l'écran à l'aide de points rouges (un point pour chaque membre). Cette trajectoire se décompose de la manière suivante :

- Mouvement vertical du bras gauche vers le haut.
- Mouvement vertical du bras gauche vers le bas.
- Mouvement vertical du bras gauche vers le haut (retour en position médiane).
- Pause.
- Mouvement vertical du bras droit vers le haut.
- Mouvement vertical du bras droit vers le bas.
- Mouvement vertical du bras droit vers le haut (retour en position médiane).
- Pause.
- Mouvement horizontal de la tête vers la gauche.
- Mouvement horizontal de la tête vers la droite.
- Mouvement horizontal de la tête vers la gauche (retour en position médiane).
- Pause.
- Mouvement vertical des deux bras vers le haut.
- Mouvement vertical des deux bras vers le bas.
- Mouvement vertical des deux bras vers le haut (retour en position médiane).
- Pause.
- Mouvement horizontal de la tête vers la gauche.
- Mouvement horizontal de la tête vers la droite.
- Mouvement horizontal de la tête vers la gauche (retour en position médiane).

Le profil de vitesse de chacune des trajectoires est de type cycloïdal. Ce type de profil est bien adapté au mouvement que peut reproduire l'être humain, puisque l'accélération ne comporte aucun saut.

Lors du test, nous enregistrons les positions des mains et de la tête retournées par les différents algorithmes, ainsi que la position désirée (trajectoire).
Finalement, nous calculons la moyenne de l'erreur au carré (mean square error) selon x et y entre la trajectoire demandée et la trajectoire détectée par l'algorithme:

$$MSEx = \frac{\sum_{i=1}^{total}((Xr_i - Xd_i)^2)}{total} \qquad (6.2.1)$$

$$MSEy = \frac{\sum_{i=1}^{total}((Yr_i - Yd_i)^2)}{total} \qquad (6.2.2)$$

(Xd_i, Yd_i) = position demandée à l'instant i.
(Xr_i, Yr_i) = position retournée par l'algorithme à l'instant i.
total = nombre total de positions.

La figure 6-1 montre la trajectoire que les utilisateurs doivent reproduire avec la main gauche.

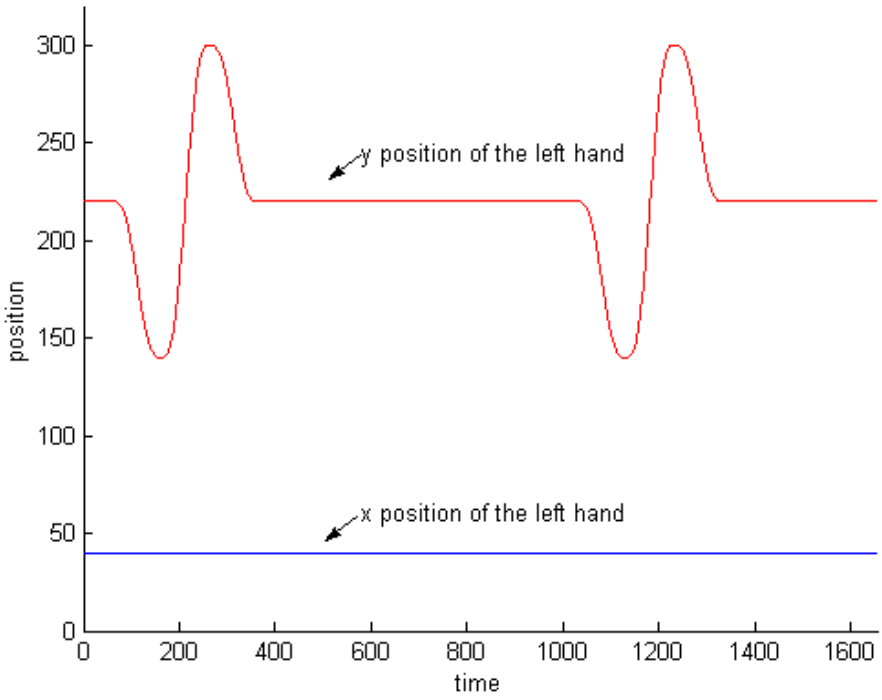

Figure 6-1 : Trajectoire de la main gauche

Remarque :

- Lors des tests, nous avons remarqué que l'utilisateur néglige le positionnement du bras gauche lorsqu'il doit suivre une trajectoire avec son bras droit et vice-versa. C'est pourquoi nous calculons le MSE des bras et de la tête uniquement lorsqu'ils doivent être en mouvement.
- Chaque utilisateur a effectué le test deux fois. Une première fois afin d'assimiler les mouvements à reproduire et une deuxième fois pour enregistrer les données. Il est réellement nécessaire que l'utilisateur soit entraîné pour faire ce test, afin que l'on puisse mesurer les performances des algorithmes et non pas la performance de l'utilisateur à suivre les points affichés. Malgré ceci, il est parfois difficile de séparer les erreurs dues aux algorithmes de "tracking" et les erreurs dues aux performances de l'utilisateur.

MSE pour la main gauche pour les différents algorithmes :

MSE	couleur de peau		flot optique		résultat final	
	x	y	x	y	x	y
Testeur 1	10.25	27.48	202.04	212.87	25.84	25.01
Testeur 2	19.19	115.12	140.90	373.94	25.43	111.15
Testeur 3	36.95	87.76	198.14	470.31	49.75	118.63
Testeur 4	36.48	213.90	273.99	406.40	49.43	209.71
Testeur 5	15.32	144.79	203.47	859.90	19.21	135.93
Testeur 6	13.09	454.58	249.13	3079.76	20.57	462.86
Testeur 7	13.37	75.68	167.32	2592.83	19.83	70.23
Testeur 8	38.36	243.47	219.00	466.68	52.85	255.08
Testeur 9	18.35	228.37	169.62	654.01	19.00	191.87
Testeur 10	6.96	15.74	51.96	125.84	12.11	21.40
Testeur 11	65.03	148.60	506.52	981.82	67.92	201.42
Testeur 12	13.80	96.59	196.55	300.15	19.69	99.19
Testeur 13	20.48	68.72	235.35	214.51	37.46	78.67
Testeur 14	41.58	449.83	472.02	2603.69	78.07	348.68
Moyenne	**24.94**	**169.33**	**234.72**	**953.05**	**35.51**	**166.42**

MSE pour la main droite pour les différents algorithmes :

MSE	couleur de peau		flot optique		résultat final	
	x	y	x	y	x	y
Testeur 1	27.79	57.59	126.26	268.28	36.46	52.26
Testeur 2	30.55	115.16	121.28	559.33	38.53	161.65
Testeur 3	34.51	98.85	180.50	352.28	41.55	113.57
Testeur 4	19.07	98.73	123.58	342.92	22.81	117.04
Testeur 5	20.63	148.73	73.45	510.84	23.23	146.91
Testeur 6	6.91	70.87	90.82	313.94	11.66	86.01
Testeur 7	12.01	36.57	59.79	273.11	15.90	43.16
Testeur 8	54.32	123.43	132.22	366.61	59.74	126.64
Testeur 9	14.84	294.18	114.40	714.16	19.08	282.37
Testeur 10	3.81	17.82	33.86	182.88	6.32	30.41
Testeur 11	54.25	111.12	107.81	570.39	46.28	128.90
Testeur 12	25.56	76.12	134.72	508.39	27.92	91.64
Testeur 13	19.19	86.44	266.83	393.85	25.77	94.40
Testeur 14	22.57	170.45	425.95	558.29	44.50	199.84
Moyenne	**24.72**	**107.58**	**142.25**	**422.52**	**29.98**	**119.63**

Les résultats pour les mains nous permettent de tirer les conclusions suivantes :

- L'algorithme de détection de la couleur de peau fonctionne bien. Cependant, nous pouvons remarquer une différence entre les résultats de la main gauche et de la main droite. Nous avons de la peine à expliquer cette différence. Ceci provient peut-être d'un éclairage différent à gauche et à droite du setup ou alors du manque d'habileté du bras gauche de l'utilisateur, puisque celui-ci est, dans la plupart des cas, droitier.

- L'algorithme du flot optique donne des résultats relativement mauvais. Cependant, il faut tenir compte du fait que, dans de nombreux, cas la position donnée par cet algorithme n'est pas rafraîchie dû à une vitesse insuffisante. Ceci fausse donc les résultats de ce test.
- Finalement, les résultats finaux sont bons. Nous remarquons que la fusion entre ces deux algorithmes n'apporte pas une réelle amélioration, puisque les résultats finaux sont très proches de ceux donnés par l'algorithme de détection de la couleur de la peau. Par exemple, une valeur de 119.63 pour le MSE du bras droit selon y indique que l'erreur moyenne entre la position détectée et la trajectoire est d'environ 10 pixels, ce qui est tout à fait acceptable.

MSE pour la tête pour les différents algorithmes :

	couleur de peau		flot optique		recherche du nez		résultat final	
MSE	x	y	x	y	x	y	x	y
Testeur 1	238.88	21.84	15057.52	50235.30	777.69	40.02	777.69	40.02
Testeur 2	72.34	362.14	321.03	117.41	11.24	3.86	11.24	3.86
Testeur 3	134.73	101.67	389.11	352.17	18.96	2.75	18.96	2.75
Testeur 4	194.52	77.76	566.78	535.86	62.00	2.95	62.00	2.95
Testeur 5	324.34	30.37	1008.27	335.87	53.59	3.72	53.59	3.72
Testeur 6	203.72	239.55	1792.14	18.43	31.52	43.15	31.52	43.15
Testeur 7	175.85	90.08	608.35	3982.01	18.95	1.80	18.95	1.80
Testeur 8	233.79	83.28	1000.95	1884.02	34.46	14.87	34.46	14.87
Testeur 9	211.77	94.90	8238.32	29134.20	14.25	88.27	14.25	88.27
Testeur 10	178.57	31.09	627.65	618.64	10.32	1.37	10.32	1.37
Testeur 11	158.70	1114.64	378.71	3357.34	24.27	117.45	24.27	117.45
Testeur 12	120.26	749.41	377.06	1435.22	20.63	2.93	20.63	2.93
Testeur 13	165.89	29.89	1642.55	869.11	12.32	1.49	12.32	1.49
Testeur 14	198.49	114.51	1600.09	2436.78	46.81	41.96	46.81	41.96
Moyenne	**186.56**	**224.37**	**2400.61**	**6808.03**	**81.22**	**26.19**	**81.22**	**26.19**

Les résultats pour la tête nous permettent de tirer les conclusions suivantes :

- L'algorithme du flot optique, comme nous l'avons déjà dit, ne fonctionne pas. C'est pourquoi nous n'en tenons pas compte lorsque nous fusionnons les informations pour obtenir le résultat final.
- L'algorithme de détection de la couleur de peau fonctionne bien. Lors du test, nous demandons à l'utilisateur de tourner la tête à gauche et à droite. Comme nous l'avons déjà dit, cet algorithme ne permet pas de suivre précisement les rotations de la tête, mais en revanche peut très bien suivre les déplacements en translation. Ceci explique donc les valeurs relativement élevées obtenues pour le MSE.
- L'algorithme de recherche du nez est l'algorithme fonctionnant le mieux. Si on regarde les différents testeurs, celui-ci a toujours fonctionné (valeur de MSE obtenue relativement basse) à l'exception du testeur 1. C'est pourquoi cet algorithme est privilégié lors de la fusion pour obtenir le résultat final. Cependant, nous remarquons que pour le testeur 1, cet algorithme n'a pas fonctionné, mais a

tout de même été utilisé pour calculer le résultat final. Ceci vient du fait que, lors du test, le nez a été perdu, mais que ceci n'a pas été détecté. Pour le détecter il faut que la position du nez, donnée par l'algorithme de recherche du nez, soit suffisamment éloignée de la position de la tête donnée par la détection de la couleur de peau.

Un test semblable aurait dû être effectué pour le module de détection du bas du corps. Cependant, faute de temps, nous n'avons pas pu le réaliser.

7 Améliorations futures

Module de détection du haut du corps :
Le module de détection du haut du corps fonctionne bien. Cependant, il a tout de même certaines limites. Afin de le rendre plus robuste nous proposons les améliorations suivantes :

- L'utilisation d'une deuxième caméra pourrait être une bonne solution. Grâce à celle-ci, il serait possible de réaliser de la stéréovision et ainsi n'avoir pas uniquement l'information en deux dimensions, mais en trois. A l'aide de cette dimension supplémentaire, les problèmes d'occlusion pourraient être résolus. Ainsi nous serions capables de "tracker" une main dans toute l'image. Nous pourrions alors supprimer les zones. Cependant l'ajout d'une caméra supplémentaire rendrait l'application certainement trop lente. Il faudrait alors diminuer le nombre d'algorithmes utilisés pour le haut du corps. Au vu des résultats, nous pourrions supprimer l'algorithme du flot optique.

Module de détection du bas du corps :
Le module de détection du bas du corps fonctionne également bien. Cependant, nous proposons les améliorations suivantes afin de le rendre plus robuste :

- Lors de la calibration, aucun contrôle n'est effectué pour vérifier les valeurs prélevées. Ainsi une fausse calibration peut se produire (si par exemple l'utilisateur place ses jambes dans les carrés rouges). C'est pourquoi une vérification de la calibration, semblable à celle réalisée pour le haut du corps, serait nécessaire.
- Le champ de vision selon y est relativement important. Ainsi une personne de petite taille n'a pas seulement ses deux jambes filmées, mais également le bas de son corps. Lors du traitement d'images, nous détectons alors qu'un seul blob, ce qui rend impossible la détection de la position des jambes. La solution consisterait à réaliser la ségmentation sur une partie de l'image uniquement (la partie de l'image contenant uniquement les jambes). Plus la personne est petite, plus la partie de l'image traitée devrait être petite.
- Pour les mêmes raisons que pour le haut du corps, une deuxième caméra serait un atout. Elle résoudrait également les problèmes d'occlusion qui peuvent se produire lorsqu'un utilisateur croise les jambes.

8 Conclusion

Le but du projet était de développer un module de détection et de reconnaissance de mouvement pour les mains, la tête et les jambes. Ce but a été atteint, puisque nous avons deux modules indépendants ; l'un pour détecter la tête et les mains, l'autre pour détecter les pieds ; tous deux capables de réaliser cette tâche. La difficulté réside généralement dans la robustesse d'une telle application. Celle-ci est relativement robuste, même si l'utilisateur ne doit jamais croiser ses mains, ni ses pieds, afin d'éviter les occlusions. Comme nous l'avons dit dans le chapitre relatant des améliorations futures, cette faiblesse pourrait être éliminée en utlisant la stéréovision.

D'un point de vue pédagogique, ce projet de diplôme m'a permis de mettre en pratique mes connaissances acquises au cours des quatres années passées à l'EPFL. Cela m'a plus particulièrement permis d'utiliser et d'approfondir mes connaissances acquises au cours de traitement d'images. De plus j'ai également pu consolider mes bases en programmation. Puisque c'est la première fois que j'ai developpé un programme qui devait être intégré par une autre personne, j'ai réalisé l'importance de documenter le code et de le rendre facilement intégrable.

9 Remerciements

Je tiens à remercier toutes les personnes au sein du laboratoire pour l'aide qu'ils m'ont apporté tout au long du projet. Plus particulièrement, je remercie Andres Lopez, avec qui j'ai collaboré pour développer l'application destinée à la Cité de l'espace. Je remercie également Aude Billard et Sylvain Calinon pour le temps qu'ils ont consacré à me guider et à me donner de précieux conseils.

Marc Kunze, Lausanne, le 20 février 2004

10 Références bibliographiques

[1] Logitech, www.logitech.com.

[2] S. Calinon, *Pocket-PC Interface for Robots*, projet de diplôme, ASL – EPFL, Lausanne, 2003

[3] Y. Linder, *Développement d'un système de reconnaissance visuelle pour Robota*, projet de semestre, ASL – EPFL, Lausanne, 2003

[4] Intel, www.intel.com/research/mrl/research/opencv/

[5] Forum de discussion sur OpenCV, http://groups.yahoo.com/group/OpenCV/

[6] M. H. Yang, D. J. Kriegman, N. Ahuja, "Detecting Faces in Images: A survey", *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, Jan., 2002.

[7] J. Yang et A. Weibel, "A Real-Time Face Tracker", *Proc. Third Workshop Applications of Computer Vision*, pp.142-147, 1996

[8] D. Chai et A. Bouzerdoum, "A Bayesian Approach to Skin Color Classification in YCbCr Color Space", *IEEE Region Ten Conference (TENCON'2000)*, vol. II, pp.421-424, Sep., 2000.

[9] S. Morgenthaler, *Introduction à la statistique*. Lausanne : Presses polytechniques et universitaires romandes, 2001.

[10] D. Chai et K. N. Ngan, "Locating Facial Region of a Head-and-Shoulders Color Image", *Third IEEE International Conference on Automatic Face and Gesture Recognition*, pp 124-129, avril, 1998.

[11] M. Unser, *Traitement d'images.* Polycopié, Volume 1, EPFL, Lausanne, 2002

[12] M. Unser, *Traitement d'images.* Polycopié, Volume 2, EPFL, Lausanne, 2002

[13] B. D. Lucas et T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision", *Proceedings of Imaging Understanding Workshop*, pp. 121-130, 1981.

[14] B. K. P. Horn et B. G. Schunck, "Determining Optical Flow", *Artificial Intelligence*, vol. 17, pp 185-203, août, 1981.

[15] J. Y. Bouguet, *Pyramidal Implementation of the Lucas Kanade Feature Tracker. Description of the algorithm.* Intel Corporation, Microprocessor Research Labs.

[16] J. Shi et C. Tomasi, "Good Features to Track", *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pp. 593 – 600, 1994.

[17] P. Berkhin, *Survey of Clustering Data Mining Techniques*, Technical Report, Accrue Software, 2002

[18] D. O. Gorodnichy, "On Importance of Nose for Face Tracking", *Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, pp 181-186, mai, 2002.

11 Annexes

I Plan de travail

I Plan de travail