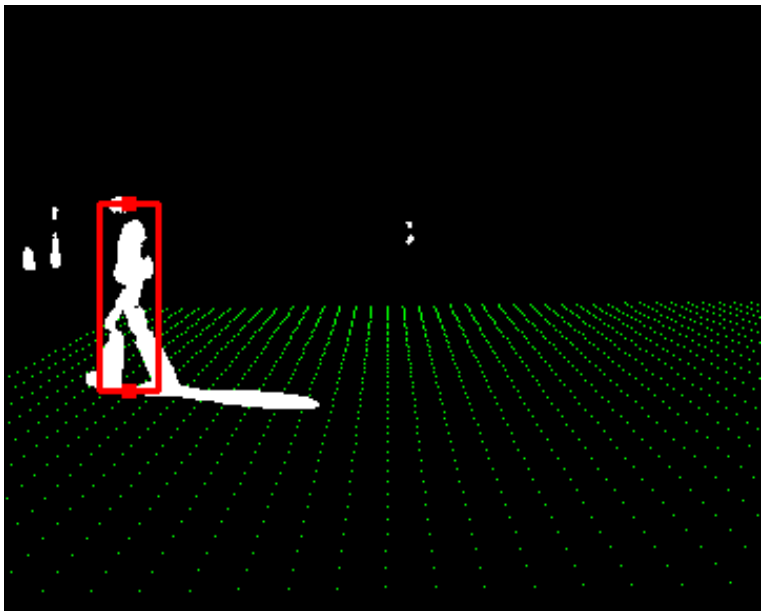
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Calibration automatique d'une caméra



Fanny Gilliéron

Projet supervisé par
François Fleuret et Jérôme Berclaz

10 juillet 2007

Table des matières

1

Introduction

Une méthode utilisée pour faire du tracking de personne[1] consiste à évaluer les points d'une carte du sol ayant la plus grande probabilité de contenir quelqu'un. Etant donné une position sur la carte, il s'agit de représenter une personne comme un rectangle dans l'image provenant d'une caméra, et d'estimer à quel point l'image binaire dérivée de l'image réelle ressemble à cette image synthétique, la comparaison étant faite par l'algorithme "FPPF". Cette opération est faite sur plusieurs caméras placées à différents endroits, et filmant la même scène sous un angle différent. Il est alors possible de localiser la personne par la position la plus probable correspondant à toutes les vues caméra.

Pour utiliser cette méthode, il faut connaître les transformations donnant la position de la tête, respectivement des pieds de la personne dans une vue caméra, étant donné un point de la carte. Ces transformations peuvent être déterminées de manière exacte si tous les paramètres de la caméra sont connus (position exacte, focale, etc..). Si toutes ces informations ne sont pas disponibles de manière précise, il faut alors calibrer la caméra par des méthodes annexes, par exemple en posant des marques au sol, et en les retrouvant manuellement sur les images des caméras, mais ce travail est relativement long.

Durant ce projet, nous avons testé une méthode de calibration automatique des caméras permettant d'obtenir la transformation voulue sans avoir besoin de repères posés à la main. Cette calibration se base sur la détection de la position d'une personne en mouvement dans toute la zone couverte par les caméras ; cette position est ensuite utilisée à la place des repères au sol pour estimer la transformation.

Nous commencerons par donner quelques définitions et concepts relatifs aux

[1] Voir [3]

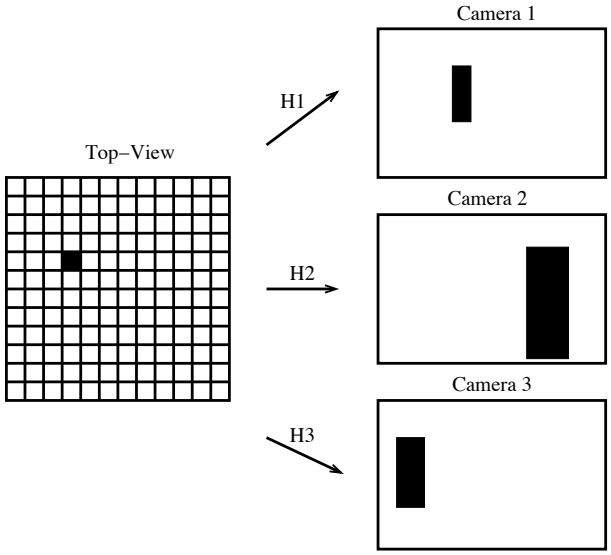


FIG. 1.1 – Correspondance entre la top-view et les images synthétiques dans les vues caméra

caméras, puis, dans le chapitre 4, nous détaillerons la méthode utilisée pour trouver la position des pieds et de la tête d'une personne dans une image. Nous expliquerons ensuite comment estimer les homographies entre les différents plans du sol dans les vues caméras (chapitre 5), et proposerons, dans le chapitre 6, une méthode pour estimer l'homographie donnant la top-view. Dans chacune des parties, des exemples de résultats obtenus seront également donnés.

2

Géométrie de la caméra et paramètres de calibration

Avant de présenter notre méthode de calibration automatique, nous allons définir quelques notions relatives aux caméras et aux images qui en résultent.

2.1 Quelques notions de géométrie[1]

2.1.1 Espace projectif

L'espace projectif de dimension n, noté $\mathcal{P}^n$, est l'ensemble des classes d'équivalences de $\mathbb{R}^{n+1}\backslash(0,...,0)$ définies par la relation d'équivalence suivante :

$$(x_1,...,\ x_{n+1}) \sim (x'_1,...,\ x'_{n+1})$$

$$\Longleftrightarrow$$

$$\exists\lambda \neq 0 \text{ tel que } (x_1,...,\ x_{n+1}) = \lambda(x'_1,...,\ x'_{n+1})$$

Les représentants de la classe sont appelés les coordonnées projectives, ou coordonnées homogènes.

2.1.2 Cas particulier : le plan projectif $\mathcal{P}^2$

Un point de $\mathcal{P}^2$ est défini par trois coordonnées. De même, une droite de $\mathcal{P}^2$ est donnée par une équation du type $a_1x_1 + a_2x_2 + a_3x_3 = 0$. Ceci s'explique par le fait que $\mathcal{P}^2$ est une projection de $\mathbb{R}^3$ sur un plan, et qu'une droite peut donc être représentée par l'intersection entre un plan de $\mathbb{R}^3$ et ce plan projectif. Ainsi, il n'y a pas de différences entre un point de l'espace projectif

[1] Voir [5]

et une droite. C'est le principe de dualité.

Pour déterminer l'équation d'une droite passant par deux points p_1 et p_2, il suffit de remarquer que $p_1(p_1 \wedge p_2) = p_2(p_1 \wedge p_2) = 0$. Ainsi, le point $p_1 \wedge p_2$ représente aussi la droite passant par p_1 et p_2. Par le principe de dualité, nous pouvons aussi voir p_1 et p_2 comme deux droites ; dans ce cas, leur intersection est donnée par $p_1 \wedge p_2$.

2.1.3 Homographie

Une homographie est une matrice régulière $(n+1) \times (n+1)$ définissant une transformation linéaire de $\mathcal{P}^n$ dans $\mathcal{P}^n$.

Une homographie est définie à un facteur multiplicatif près :

$$y = Hx \Leftrightarrow y = \lambda Hx$$

car x et y sont des représentants des classes d'équivalences définies ci-dessus, et sont donc définis à un facteur multiplicatif près.

Pour déterminer une homographie satisfaisant $y = Hx$, il faudra donc avoir $n+2$ correspondances entre des x et des y, et les représenter avec leurs coordonnées homogènes.

2.2 Application aux caméras[2]

La caméra effectue une transformation d'un sous-ensemble de $\mathbb{R}^3 \cup \{\infty\}$ (la zone de l'espace couverte par la caméra) dans un sous-ensemble de $\mathbb{R}^2$, l'image. Pour définir entièrement cette projection, il faut connaître certains paramètres relatifs à la caméra.

Dans la pratique, il peut être intéressant de chercher ces paramètres pour pouvoir obtenir les homographies entre la carte (top-view) et les vues caméra. Cependant, dans notre méthode, nous n'allons pas estimer explicitement ces paramètres, mais directement l'homographie ; les informations qui suivent sont donc données à titre informatif, mais n'interviendront pas dans la méthode de calibration proposée.

[2] Voir [1] et [6]

2.2.1 Matrice de projection dans un cas simple

Supposons pour simplifier les choses que la caméra se trouve à l'origine, que l'image est orthogonale à l'axe z et que le point principal soit en $(0, 0)$. L'image d'un point de l'espace est obtenue en prenant l'intersection entre la droite définie par ce point et la caméra, et le plan image, comme illustré sur la figure 2.2.1. Les points de $\mathbb{R}^3$ représentés sur l'image sont donc tous les points dans le cône défini par la caméra et l'image.

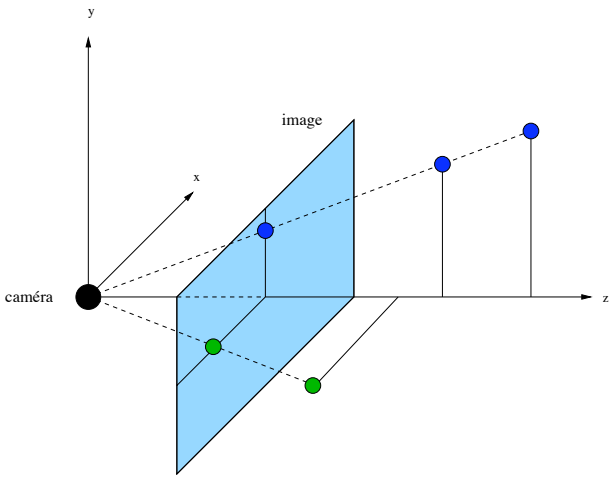


FIG. 2.1 – Principe d'une caméra

Tous les points de l'espace situés sur une même droite passant par la caméra et coupant l'image seront représentés par un même point sur l'image, c'est pourquoi nous allons pouvoir travailler avec les coordonnées homogènes, en considérant les points de l'image comme des éléments de l'espace projectif de dimension 2.

L'application envoyant les points de l'espace sur l'image n'est pas linéaire ; elle ne peut donc pas être écrite de manière simple. Remplaçons les points de $\mathbb{R}^3$ par des éléments de $\mathcal{P}^4$ d'après la règle suivante :

$(x, y, z) \longrightarrow (x, y, z, 1)$ si (x, y, z) est fini,
$(x, y, z) \longrightarrow (x, y, z, 0)$ si (x, y, z) est à l'infini.

Avec cette transformation, nous assimilons le point $(x, y, z, u) \in \mathcal{P}^4$ au point $(\frac{x}{u}, \frac{y}{u}, \frac{z}{u}) \in \mathbb{R}^3 \cup \{\infty\}$.

De même, nous allons considérer des éléments de $\mathcal{P}^3$ au lieu de $\mathbb{R}^2$ pour décrire l'image.

Avec ces transformations, la projection devient linéaire, et peut être décrite par :

$$\begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_e \\ y_e \\ z_e \\ u_e \end{pmatrix}$$
$$= \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_e \\ y_e \\ z_e \\ u_e \end{pmatrix}$$

où f est la distance entre la caméra et le plan image, $(\frac{x_e}{u_e}, \frac{y_e}{u_e}, \frac{z_e}{u_e})$ est un point de l'espace, et $(u,\ v) = (\frac{x_i}{z_i},\ \frac{y_i}{z_i})$ est son image.

2.2.2 Paramètres intrinsèques et extrinsèques de la caméra

Dans la partie précédente, nous avons supposé que la caméra était centrée à l'origine et pointait dans la direction de l'axe z ; dans la réalité, ce n'est pas toujours le cas, et nous devons tenir compte de la position de la caméra pour obtenir la matrice de projection.

Pour exprimer la position de la caméra dans l'espace, nous allons utiliser des coordonnées sphériques. Nous décomposons le déplacement de la caméra de l'origine à sa position réelle par une translation selon $(0,\ 0,\ z)$, une rotation autour de l'axe x d'angle θ et une rotation autour de l'axe z d'angle ρ. Nous pouvons modifier la matrice de projection obtenue précédemment en appliquant la transformation inverse aux points de l'espace. A nouveau, l'utilisation des coordonnées homogènes nous permet d'obtenir une application linéaire, bien que nous effectuions une translation.

$$\begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} R_\rho^z \cdot R_\theta^x \cdot T_z \begin{pmatrix} x_e \\ y_e \\ z_e \\ u_e \end{pmatrix}$$

avec

$$R_\rho^z = \begin{pmatrix} \cos(\rho) & -\sin(\rho) & 0 \\ \sin(\rho) & \cos(\rho) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R_\theta^x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{pmatrix}$$

$$T_z = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -z \end{pmatrix}$$

(ρ, θ, z) est le vecteur des paramètres extrinsèques.

Si le point principal (image de la caméra) n'est pas $(0, 0)$ mais (p_u, p_v), nous ajoutons une translation de vecteur (p_u, p_v) aux coordonnées de l'image obtenue. Ceci correspond à la modification suivante :

$$\begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \longrightarrow \begin{pmatrix} f & 0 & p_u \\ 0 & f & p_v \\ 0 & 0 & 1 \end{pmatrix}$$

Si les pixels de l'image ne sont pas carrés (échelles différentes sur les deux axes de l'image), c'est le facteur d'échelle, initialement f, qui doit être modifié.

$$\begin{pmatrix} f & 0 & p_u \\ 0 & f & p_v \\ 0 & 0 & 1 \end{pmatrix} \longrightarrow \begin{pmatrix} \alpha_u & 0 & p_u \\ 0 & \alpha_v & p_v \\ 0 & 0 & 1 \end{pmatrix}$$

Enfin, si les axes de l'image ne sont pas orthogonaux, nous ajoutons un facteur de distorsion de l'image :

$$\begin{pmatrix} \alpha_u & 0 & p_u \\ 0 & \alpha_v & p_v \\ 0 & 0 & 1 \end{pmatrix} \longrightarrow \begin{pmatrix} \alpha_u & s & p_u \\ 0 & \alpha_v & p_v \\ 0 & 0 & 1 \end{pmatrix} = K$$

Les paramètres p_u, p_v, α_u, α_v et s sont les paramètres intrinsèques de la caméra.

2.2.3 Forme générale de la matrice de projection

La matrice de projection de la caméra est finalement donnée par :

$$X_i = K \cdot R_\rho^z \cdot R_\theta^x \cdot T_z \cdot X_e$$

avec

$$K = \begin{pmatrix} \alpha_u & s & p_u \\ 0 & \alpha_v & p_v \\ 0 & 0 & 1 \end{pmatrix}$$

$$R_\rho^z = \begin{pmatrix} \cos(\rho) & -\sin(\rho) & 0 \\ \sin(\rho) & \cos(\rho) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R_\theta^x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{pmatrix}$$

$$T_z = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -z \end{pmatrix}$$

et X_e et X_i dans les espaces projectifs.

3

Contexte de travail

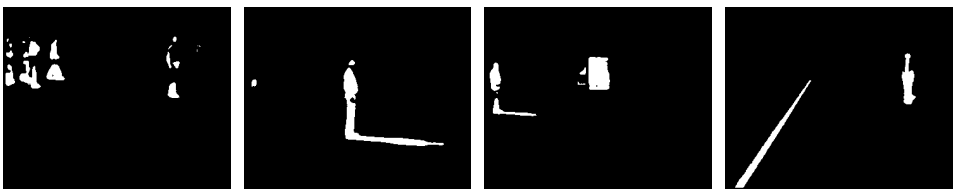
3.1 Background subtraction

Nous avons à disposition des images provenant de plusieurs caméras, sur lesquelles un algorithme de background subtraction (détection de mouvement) a été appliqué. Nous considérons donc des images binaires différentiant les zones statiques (en noir) et dynamiques (en blanc) de l'image de départ.

Idéalement, la partie dynamique correspond à une personne se déplaçant dans la zone couverte par les caméras ; en réalité, des éléments extérieurs sont sélectionnés par l'algorithme (autres personnes, portes, ...), mais devraient être ignorés lors du processus de calibration. Ces valeurs aberrantes sont plus ou moins importantes selon les prises de vues, et nécessiteront une implémentation robuste à chaque étape du processus.

3.2 Quelques exemples

Voici quelques images sur lesquelles nous avons travaillé, montrant différents facteurs de bruit : personnes en arrière plan, ombres, éléments extérieurs.



(a) (b) (c) (d)

FIG. 3.1 – Exemples d'images issues du background subtraction

4

Détermination de la tête et des pieds

Pour calculer des homographies entre les caméras, nous avons besoin de points en correpondance. Nous avons choisi de déterminer la position de la tête et des pieds de la personne sur chacune des images, et de les mettre en relation. La méthode utilisée pour trouver la position de la tête et des pieds est décrite ci-dessous.

4.1 Analyse en composantes principales[1]

L'analyse en composantes principales (ACP) est une méthode mathématique permettant d'obtenir des informations sur un jeu de données. Le but de cette méthode est de réduire la dimension des données en recherchant les directions contenant le plus d'information sur la variance de l'échantillon.

4.1.1 Description de la méthode

Géométriquement, la recherche des composantes principales revient à ajuster une ellipse autour des données, et à considérer ses axes principaux. La première composante correspondra au plus grand axe de l'ellipse, et représentera donc la direction expliquant la plus grande partie de la corrélation reliant les observations.

Le calcul des composantes principales s'obtient à l'aide de la matrice de covariance de l'échantillon, notée Σ. Cette matrice est donnée par

[1] Voir [8]

$$\Sigma = \frac{1}{n} X^T X$$

où X est la matrice des observations, avec chaque ligne correspondant à une observation, et n représente le nombre total d'observations. L'élément $\Sigma_{i,j}$ est donc l'estimateur de $Cov(X_i, X_j)$.

Les directions des composantes principales sont données par les vecteurs propres de Σ ayant les valeurs propres associées les plus grandes. L'ellipse dans laquelle sont contenues les observations aura donc un volume proportionnel à la racine du déterminant de la matrice Σ.

4.1.2 Application au problème

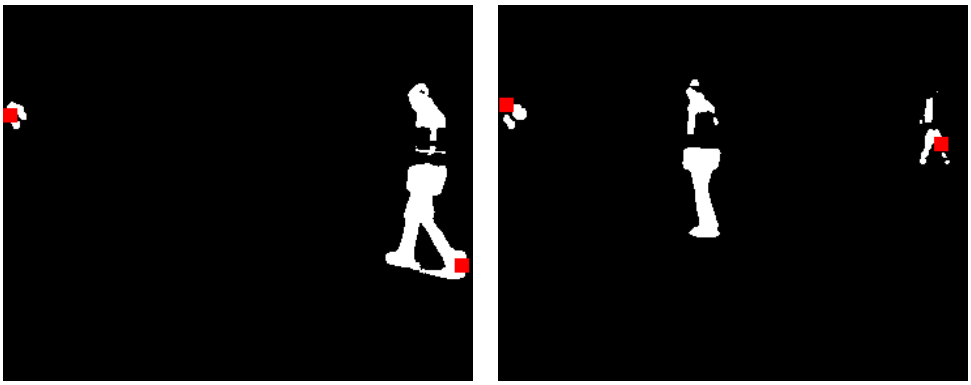
Pour déterminer les pieds et la tête d'une personne, nous allons rechercher la direction de la composante principale des pixels blancs (représentant la partie en mouvement dans l'image), puis considérer la loi empirique que suivent ces pixels une fois projetés sur cette direction. La détermination de certains quantiles (par exemple 1% et 99%) de cette loi donnera un estimateur de la position de la tête et des pieds.

Nous considérons ici les pixels blancs de l'image binaire comme un nuage de points de $\mathbb{R}^2$, et nous supposons que l'image d'une personne peut être représentée par une ellipse, dont l'axe principal passe par la tête et les pieds. La détermination des composantes principales nous permet de déterminer cet axe, sur lequel devraient se trouver les points qui nous intéressent.

4.2 Robustesse

Il est facile de se convaincre que si le jeu de données contient des valeurs aberrantes, les composantes principales seront fortement affectées ; il suffit d'imaginer la déformation de l'ellipse lorsqu'on introduit un point extérieur. La figure 4.1 montre la position des quantiles que nous obtenons en appliquant la méthode proposée ci-dessus à une image contenant des données parasites.

Dans notre cas, les images traitées sont bruitées, notamment à cause du passage d'autres personnes en arrière-plan ; il est donc nécessaires de modifier l'algorithme pour le rendre robuste.



(a) (b)

FIG. 4.1 – ACP classique

4.2.1 Algorithme MCD

L'algorithme MCD (Minimum Covariance Determinant) propose une alternative robuste à l'ACP. Il s'agit de rechercher dans les données le sous-ensemble de taille $\alpha \cdot n$ contenu dans l'ellipsoide de taille minimale, où $(1-\alpha)$ est le taux supposé de valeurs aberrantes, toujours inférieur à 50%. Ce sous-ensemble de données aura une matrice de covariance à déterminant minimal.

Pour déterminer de manière exacte le sous-ensemble optimal, il est nécessaire de tester un grand nombre de sous-ensembles, ce qui donne à l'algorithme une complexité exponentielle qui n'est pas acceptable avec des jeux de données de grande taille.

4.2.2 Algorithme MCD-MIT[2]

L'algorithme MCD-MIT (Multistart Iterative Trimming) est une variante de l'algorithme MCD permettant de trouver un sous-ensemble relativement proche de l'optimum en un temps acceptable.

Partant d'une sélection aléatoire des données, l'algorithme va réordonner toutes les observations selon leur distance de Mahalanobis par rapport à μ et Σ. La distance de Mahalanobis est définie par

$$d_M(x \mid \mu, \Sigma) = (x - \mu)^T \Sigma (x - \mu)$$

Il s'agit donc de trier les observations selon leur distance au centre de l'ellipse contenant la sélection aléatoire. Lorsqu'aucun réarrangement n'est plus nécessaire, cela signifie que le sous-ensemble sélectionné est contenu dans

[2]Voir [7]

une ellipse de taille minimale. En d'autres termes, l'algorithme va tour à tour ajuster un modèle gaussien aux données sélectionnées, puis sélectionner les données les plus vraisemblables par rapport à ce modèle.

Cette procédure est répétée k fois (Multistart) pour éviter de boucler si plusieurs points ont la même distance de Mahalanobis. Dans notre cas, il n'est pas nécessaire de choisir une valeur élevée pour k, mais il est conseillé de prendre $k > 1$ pour diminuer le risque de non-convergence de l'algorithme.

Algorithme MCD-MIT

Données : $x_1, .. \, x_n \in \mathbb{R}^d, k \in \{d+1, .. \, n\}, \; s \in \mathbb{N}$
Résultat : $\{y_1, ... \, y_k\} \in \{x_1, ... \, x_n\}$ avec matrice de covariance à déterminant minimal

Initialisation : $DetMin \leftarrow \infty$

pour $i = 1$ **à** s **faire**
    Permuter $x_1, ... \, x_n$ aléatoirement;

    répéter
        $\mu \leftarrow$ moyenne$(x_1, ..., \; x_k)$;
        $\Sigma \leftarrow$ covariance$(x_1, ..., \; x_k)$;
        Réordonner $x_1, ..., \; x_n$ pour que $x_1, ..., \; x_k$ aient la plus petite distance de Mahalanobis par rapport à μ et Σ ;

    jusqu'à *ce qu'aucun ordonnancement ne soit nécessaire* ;

    $d \leftarrow$ det(Σ);

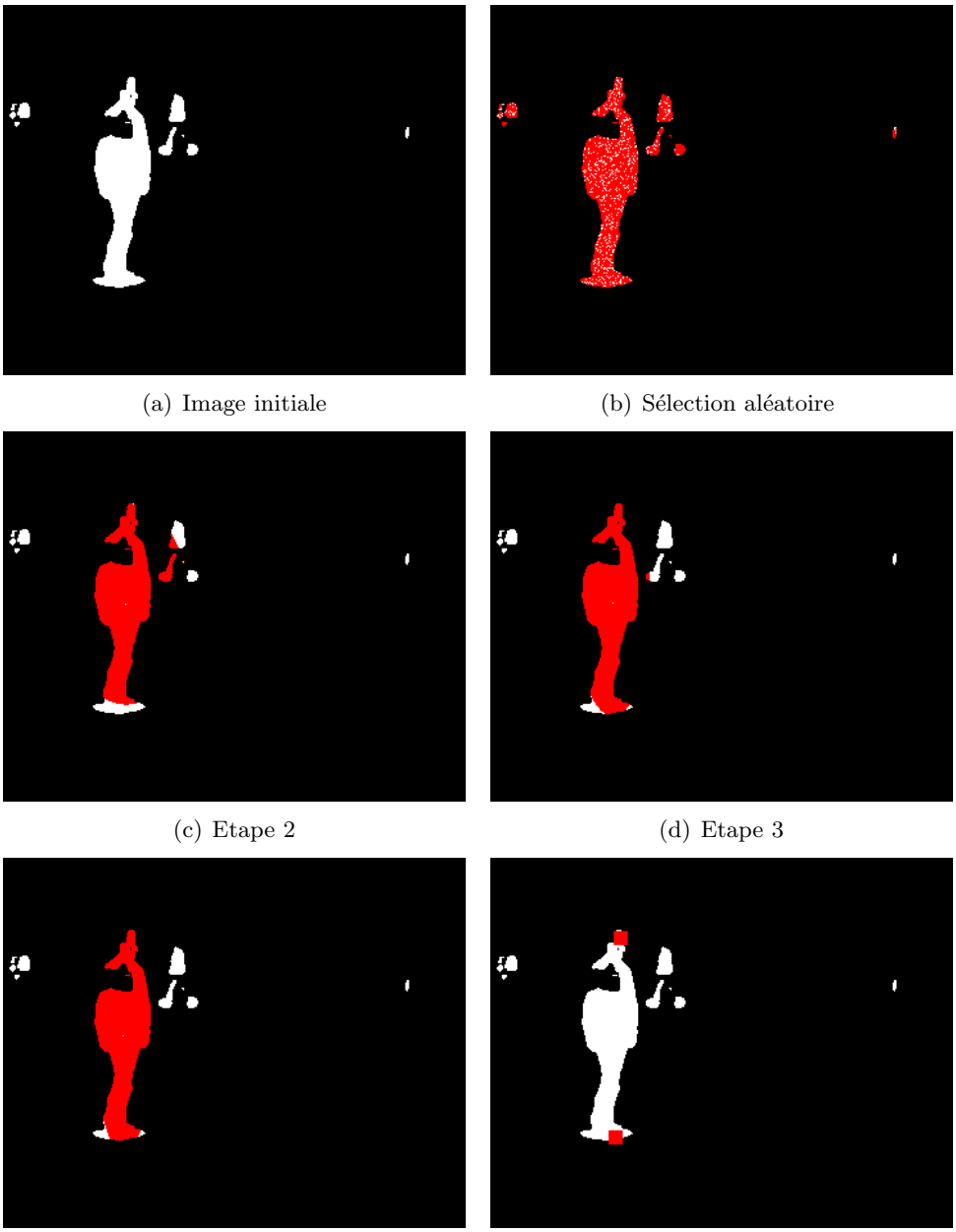
    si $d < DetMin$ **alors**
        $DetMin \leftarrow d$;

    fin
fin

Retourner $x_1, ..., \; x_k$

La figure 4.2 montre l'évolution du sous-ensemble des observations lorsqu'on effectue l'algorithme sur une image bruitée, et en choisissant de considérer un sous-ensemble contenant 85% des pixels blancs. L'algorithme converge très rapidement vers la zone voulue (convergence en 7 étapes, les 3 dernières étapes apportant des modifications minimes au sous-ensemble) et il en découle une

position satisfaisante pour la tête et les pieds.



(a) Image initiale (b) Sélection aléatoire

(c) Etape 2 (d) Etape 3

(e) Etape 4 (f) Estimation des pieds et de la tête

FIG. 4.2 – Algorithme MCD-MIT

4.2.3 Gestion du taux de valeurs aberrantes

Un problème qui se pose avec l'algorithme MCD est qu'il faut connaître approximativement le taux de valeurs aberrantes. Si nous voulons un algorithme qui fonctionne correctement pour n'importe quelle caméra, nous devons le modifier pour gérer la différence de valeurs aberrantes entre les images.

Nous avons choisi de sélectionner les pixels correspondant à la tête et aux pieds avec l'algorithme MCD-MIT en fixant le taux de valeurs aberrantes à 50% (taux maximal). Ceci nous évite d'obtenir de trop mauvais résultats sur des images très bruitées, mais à l'inconvénient de positionner la tête trop basse et/ou les pieds trop hauts sur des images dont le taux de valeurs aberrantes est petit, comme illustré sur l'image 4.3.

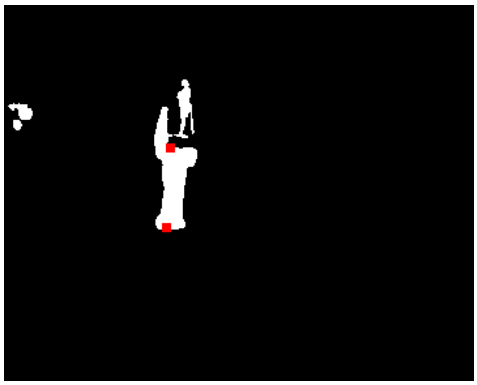
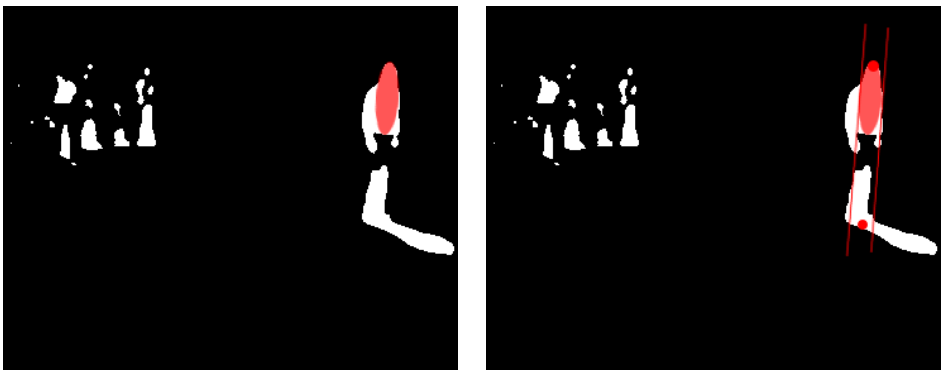


FIG. 4.3 – MCD-MIT avec 50%

Pour contrer cet effet, nous proposons de considérer une bande de la même largeur que l'ellipse obtenue par l'algorithme (la racine de la plus petite valeur propre) autour de la droite donnant la composante principale, et de sélectionner tous les pixels blancs contenus à l'intérieur de cette bande, même s'ils n'appartiennent pas au jeu de donnée robuste sélectionné par l'algorithme. Nous détemineront alors la position de la tête et des pieds comme étant les quantiles extrêmes de la projection de ce nouveau jeu de donnée sur la droite principale (voir figures 4.4(a) et 4.4(b)). Des exemples de résultats obtenus sont proposés dans les figures 4.5 et 4.6. Nous pouvons voir que certaines estimations sont très bonnes, mais qu'il arrive également que l'algorithme se trompe totalement, notamment lorsque l'image comporte une zone de bruit très concentrée.



(a) Exemple d'ellipse pouvant être obtenue par l'algorithme

(b) Bande considérée pour trouver la tête et les pieds

FIG. 4.4 – Heuristique pour gérer le taux de valeurs aberrantes

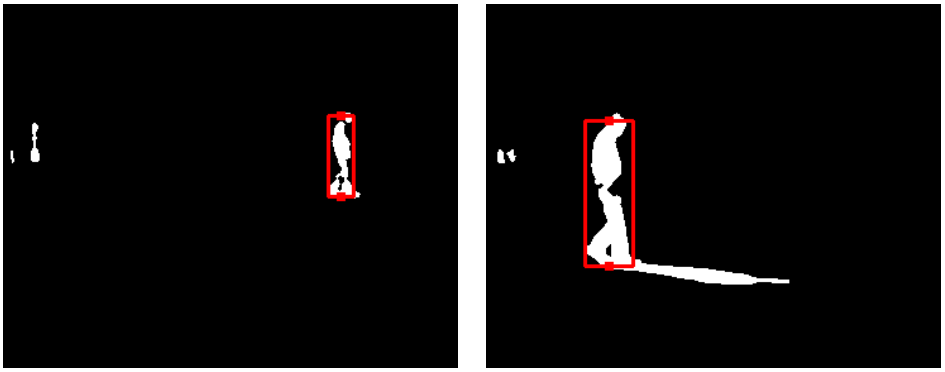


FIG. 4.5 – Détection correcte des pieds et de la tête

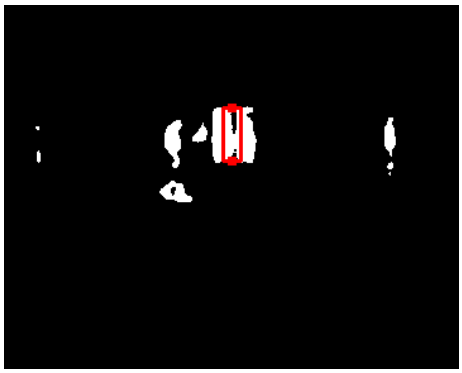


FIG. 4.6 – Détection erronée de la personne

5

Estimation des homographies relatives aux vues caméras

Nous allons maintenant expliquer comment nous avons pu estimer les homographies entre les différentes vues caméras à l'aide de l'estimation des pieds et de la tête. Nous verrons au chapitre suivant comment ces homographies peuvent être utilisées pour obtenir une top-view de la zone couverte par les caméras.

5.1 Détermination d'une homographie[1]

Pour déterminer une homographie dans les plans projectifs (les images provenant des caméras), nous avons besoin de 4 points en correspondance, chacun formé de deux coordonnées. Nous aurons ainsi suffisamment d'information pour déterminer de manière "unique" l'homographie. Comme cette dernière est définie à un facteur près, les couples de points en correspondances vérifient $y = \lambda Hx$; la condition peut se réécrire comme $y \wedge Hx = 0$.

Cette condition correspond au système d'équations suivant :

$$\begin{bmatrix} y_1^i \\ y_2^i \\ y_3^i \end{bmatrix} \wedge \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1^i \\ x_2^i \\ x_3^i \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \qquad i = 1, ..., 4$$

$$\begin{bmatrix} y_1^i \\ y_2^i \\ y_3^i \end{bmatrix} \wedge \begin{bmatrix} h_{11}x_1^i + h_{12}x_2^i + h_{13}x_3^i \\ h_{21}x_1^i + h_{22}x_2^i + h_{23}x_3^i \\ h_{31}x_1^i + h_{32}x_2^i + h_{33}x_3^i \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \qquad i = 1, ..., 4$$

[1] Voir [4]

En choisissant de fixer $x_3^i = y_3^i = 1$ (toujours possible lors de l'utilisation des coordonnées homogènes), nous obtenons finalement le système d'équations linéaires en h_{ij} suivant :

$$\begin{array}{lcll} h_{31}x_1^i y_2^i + h_{32}x_2^i y_2^i + h_{33}y_2^i - h_{21}x_1^i - h_{22}x_2^i - h_{23} & = & 0 & \\ h_{11}x_1 + h_{12}x_2^i + h_{13} - h_{31}x_1^i y_1^i - h_{32}x_2^i y_1^i - h_{33}y_1^i & = & 0 & i = 1, ..., 4 \\ h_{21}x_1^i y_1^i + h_{22}x_2^i y_1^i + h_{23}y_1^i - h_{11}x_1^i y_2^i - h_{12}x_2^i y_2^i - h_{13}y_2^i & = & 0 & \end{array}$$

La troisième équation est combinaison linéaire des deux premières. Nous avons donc deux équations linéaires pour chacun des couples de points en correspondance, qui peuvent se réécrire comme suit :

$$\underbrace{\begin{bmatrix} 0 & 0 & 0 & -x_1^1 & -x_2^1 & -1 & x_1^1 y_2^1 & x_2^1 y_2^1 & x_3^1 y_2^1 \\ x_1^1 & x_2^1 & 1 & 0 & 0 & 0 & -x_1^1 y_1^1 & -x_2^1 y_1^1 & -x_3^1 y_1^1 \\ & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & -x_1^4 & -x_2^4 & -1 & x_1^4 y_2^4 & x_2^4 y_2^4 & x_3^4 y_2^4 \\ x_1^4 & x_2^4 & 1 & 0 & 0 & 0 & -x_1^4 y_1^4 & -x_2^4 y_1^4 & -x_3^4 y_1^4 \end{bmatrix}}_{A} \begin{pmatrix} h_1 \\ h_2 \\ \vdots \\ h_8 \\ h_9 \end{pmatrix} = 0$$

La matrice A étant entièrement déterminée par les quatre correspondances, il est possible d'obtenir les coefficients de H en utilisant l'algorithme DLT donné dans la section suivante. Remarquons toutefois que si 3 points sont colinéaires parmi les 4 points choisis, il ne sera pas possible de déterminer l'homographie.

5.1.1 Algorithme DLT

L'algorithme DLT (Direct Linear Transformation) permet d'obtenir une solution non triviale pour l'équation $Ah = 0$.

Algorithme DLT

Données : 4 correspondances entre points en 2D $\{x^i \leftrightarrow y^i\}$
Résultat : Homographie H vérifiant $y^i = Hx^i$

1. Calculer la matrice A comme proposé au paragraphe 5.1.
2. Calculer la décomposition en valeurs singulières de la matrice A. $A = UDV^T$ avec D matrice diagonale contenant les valeurs propres de A^TA et V contenant les vecteurs propres correspondants.
3. Déterminer h, donné par le vecteur propre correspondant à la plus petite valeur propre positive.
4. Calculer $H = \begin{pmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{pmatrix}$

Retourner H

5.1.2 Homographie surdéterminée

L'algorithme DLT permet aussi d'ajuster une homographie a un jeu de correspondances ; il ne s'agit donc plus de déterminer de manière exacte (à un facteur près) une homographie, mais de trouver l'homographie minimisant les erreurs par rapport au jeu de donnée.

L'algorithme reste le même ; il suffit de construire la matrice A à partir de toutes les correspondances, et donc de considérer une matrice A de taille $2n \times 9$ si n est le nombre de correspondances à disposition.

5.2 Robustesse

L'algorithme DLT est relativement sensibles aux valeurs aberrantes, puisqu'il cherche à minimiser une somme d'erreurs sur chaque point, accordant donc autant d'importance à chaque donnée. Dans le cas de données bruitées, il est nécessaire d'effectuer un tri préalable afin de déterminer quelles sont les correspondances fiables qui peuvent être utilisées lors de l'estimation. L'algorithme RANSAC, proposé ci-dessous, permet de sélectionner des inliers parmi les données avant d'ajuster l'homographie.

5.2.1 Algorithme RANSAC[2]

L'algorithme proposé ci-dessous permet de déterminer des correspondances aberrantes. Il est basé sur l'observation suivante :

Les inliers se comportent pratiquement comme l'homographie, à une petite erreur près qui ne devrait faire varier la position du point que de quelques pixels, alors que les outliers prennent des valeurs complètement différentes de celles données par l'homographie. La méthode utilisée ici consiste à ajuster des homographies sur quatres correspondances choisies au hasard de manière itérative, puis à compter les correspondances qui ne diffèrent que de quelques pixels (seuil fixé) des positions théoriques. Si une homographie engendre un nombre important d'inliers, elle doit raisonnablement être proche de l'homographie réelle, car les erreurs des outliers ne peuvent pas toutes se comporter de la même manière.

Algorithme RANSAC

Données : Ensemble de correspondances entre points en 2D $\{x^i \leftrightarrow y^i\}$ contenant des valeurs aberrantes, seuil de précision s, nombre d'itérations n

Résultat : Homographie H vérfiant au mieux $y^i = Hx^i$ pour un sous-ensemble de correspondances considérées comme inliers

pour $i = 1$ **à** n **faire**

1. Choisir 4 correspondances au hasard.
2. Calculer l'homographie H par l'algorithme DLT.
3. Pour chaque correspondance, si $d(x^i, H^{-1}y^i)^2 + d(y^i, Hx^i)^2 < s$, la correpondance $\{x^i \leftrightarrow y^i\}$ est un inlier pour l'homographie H.
4. Garder H si le nombre d'inliers est maximal parmi toutes les homographies testées jusqu'ici.

fin

Réestimer H par l'algorithme DLT sur le sous-ensemble de correpondances considérées comme inliers.

Retourner H

[2] Voir [2]

5.2.2 Exemples d'estimations

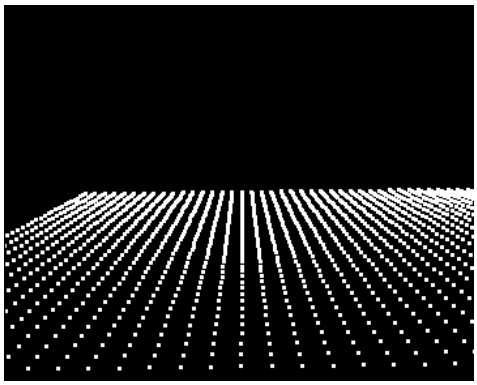
Dans notre situation, nous avons a disposition des estimations de la tête et des pieds d'une même personne sur trois vues caméras différentes. Le but est d'estimer certaines homographies pour pouvoir déduire des informations sur la position des caméras et faire le lien entre les images et la top-view.

Les homographies qui nous intéressent sont celles reliant les pieds dans une vue aux pieds d'une autre vue, donc faisant le lien entre l'image du sol sur chacune des images provenant de deux caméras différentes. Nous pourrions également estimer les homographies reliant les têtes entre elles ou celles permettant de passer des pieds à la tête sur une vue particulière.

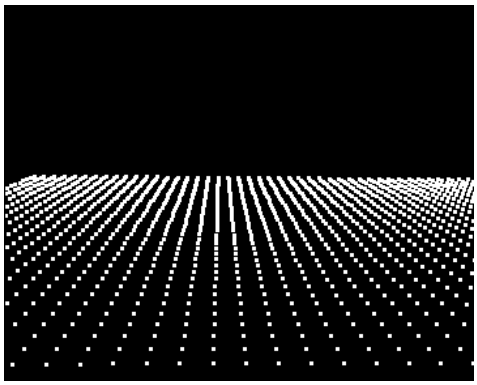
Dans tous les cas, la méthode d'estimation est la même. Il suffit de considérer les correspondances entre points concernés par l'homographie, et d'appliquer l'algorithme RANSAC.

Pour contrôler la qualité de ces estimations, nous avons considéré une grille au sol dans chacune des vues caméras qui a été générée "à la main" en cliquant sur des repères au sol dans des images provenant de chacune des caméras. Nous allons regarder a quel point l'image de la première grille envoyée sur une autre vue caméra par l'homographie estimée automatiquement diffère de la grille réelle, que nous considérerons comme exacte.

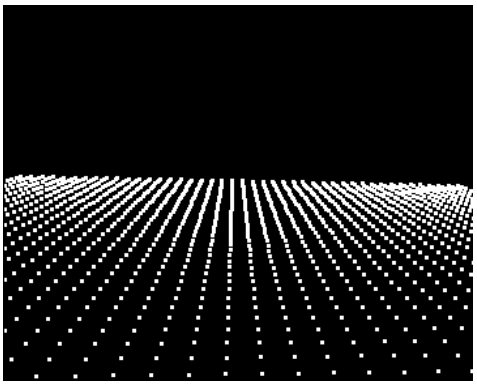
Les figures 5.1(a) à 5.1(e) montrent deux estimations d'homographies à partir de la position des pieds estimée automatiquement par la méthode proposée précédemment. Nous pouvons voir que la qualité est assez bonne, mais que des erreurs systématiques peuvent modifier l'inclinaison du sol. Nous pouvons notamment remarquer que la grille de la deuxième caméra est légèrement décalée sur la gauche, et que la grille de la troisième caméra est un peu trop en avant. Ces décalages sont généralement dûs à des erreurs systématiques dans la position des pieds : par exemple, les images de la troisième caméra avaient une grande ombre portée sur la droite, ce qui avait tendance à positionner les pieds un peu trop à droite, et induit cette erreur sur la grille.



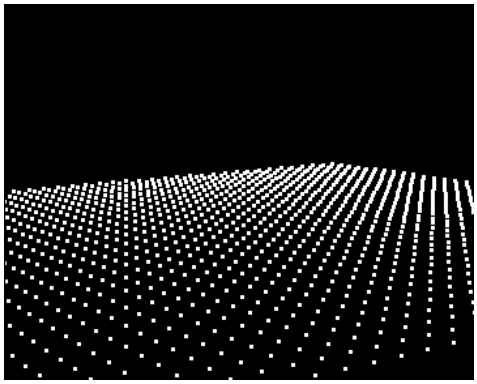
(a) Grille réelle dans la vue caméra 1



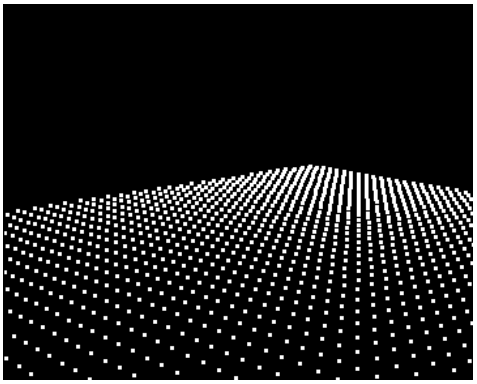
(b) Grille réelle dans la vue caméra 2



(c) Grille estimée dans la vue caméra 2



(d) Grille réelle dans la vue caméra 3



(e) Grille estimée dans la vue caméra 3

FIG. 5.1 – Estimation des homographies entre les plans du sol

6

Estimation de la top-view

Pour positionner les personnes présentes sur les caméras, nous avons besoin d'une vue du ciel (top-view) de la zone couverte par les caméras. Ce chapitre donne une méthode pour estimer l'homographie d'une vue caméra à la top-view, utilisant plusieurs vues caméras d'une même scène.

6.1 Détermination de vecteurs orthogonaux

Pour estimer la top-view, nous allons nous baser sur la mesure des angles entre des paires de vecteurs bien choisis. La méthode est issue de l'observation suivante :

S'il n'y a pas de rotation de la caméra par rapport à la ligne d'horizon, nous pouvons faire l'hypothèse qu'un vecteur vertical au milieu de l'image et un vecteur horizontal en ce même point forment dans la réalité un angle droit s'ils sont contenus dans le plan du sol (figure 6.1). Les images de ces vecteurs par l'homographie menant à la top-view devraient donc former un angle droit.

Dans les chapitre précédent, nous avons expliqué comment estimer automatiquement une homographie entre le plan du sol de deux vues caméra. Cette homographie peut être utilisée pour placer les vecteurs définis ci-dessus dans le plan horizontal d'une autre vue caméra. Ces vecteurs ne formeront plus un angle droit dans la vue caméra, mais devront tout de même être orthogonaux dans la top-view.

Ceci nous permet de définir des paires de vecteurs pour chaque caméra filmant une même scène, et d'obtenir un ensemble de contraintes pour l'estimation de la top-view.

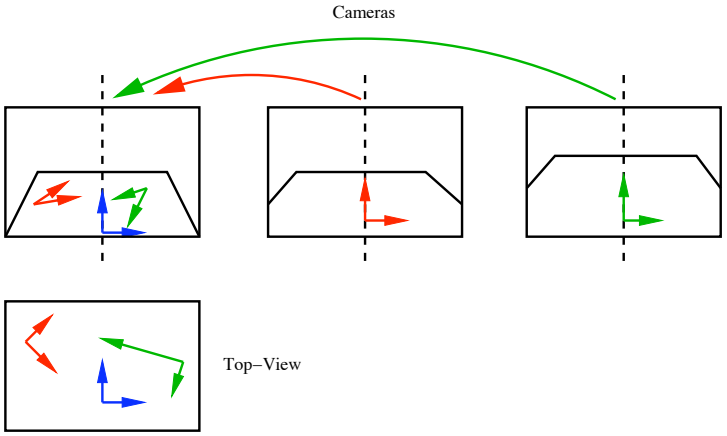
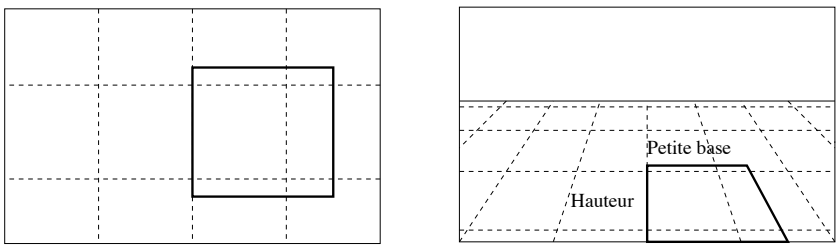


FIG. 6.1 – Vecteurs orthogonaux dans le plan du sol

6.2 Estimation de l'homographie

La top-view est définie à un facteur d'échelle et une translation près. Nous pouvons donc fixer certaines correspondances à notre convenance. Une possibilité est de considérer un rectangle dans la top-view, qui correspondra à un quadrilatère quelconque dans la vue caméra. Il est alors possible de forcer ce quadrilatère à avoir deux angles droits en le plaçant au centre de la vue caméra, par le même principe que les paires de vecteurs orthogonaux décrits ci-dessus. Le rectangle (figure 6.2(a)) correspondra donc à un trapèze rectangle dont il faudra déterminer la hauteur et la petite base (figure 6.2(b)).



(a) Carré fixé dans la top-view

(b) trapèze correspondant dans la vue caméra

FIG. 6.2 – Méthode d'estimation de la top-view

La hauteur et la petite base du trapèze doivent être choisis de manière à ce que l'homographie définie par les quatre correspondances donne des angles droits entre les paires de vecteurs provenant de chacune des vues caméra. Nous cherchons donc à minimiser la somme des carrés des cosinus entre les

paires de vecteurs. Remarquons que la paire de vecteurs correspondant à la première caméra est automatiquement redressée à angle droit par notre définition de l'homographie (choix du trapèze rectangle).

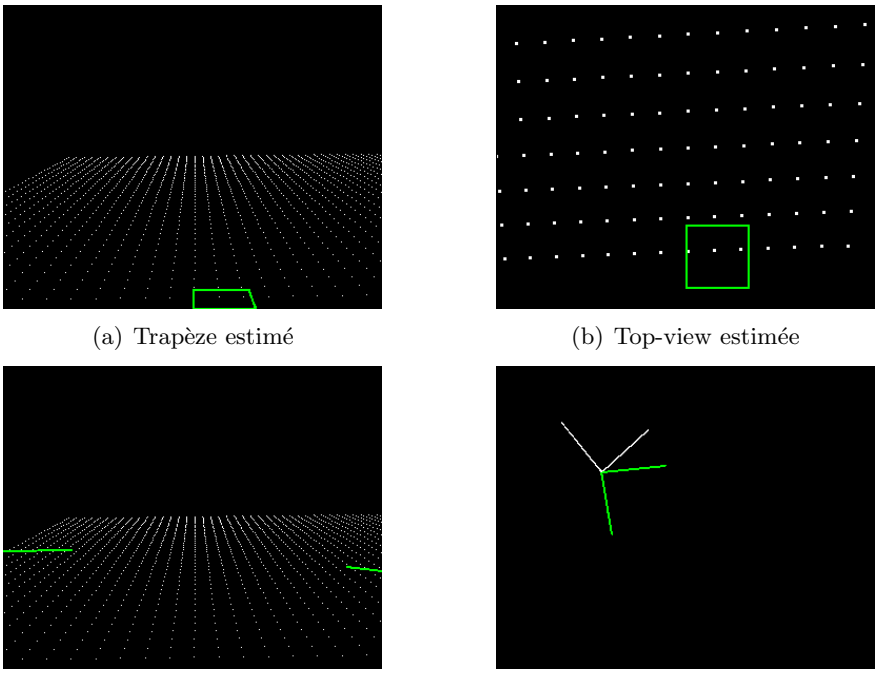
Pour trouver la hauteur et la petite base, nous avons généré aléatoirement la position du coin supérieur droit du trapèze, uniformément dans la moitié droite de l'image et le tiers inférieur, afin de s'assurer de rester dans le plan horizontal. Nous avons simplement pris le point minimisant la somme des carrés des cosinus parmi tous les tirs ; l'optimisation est très basique, mais vu la taille des images, elle permet d'obtenir un résultat de qualité en un temps raisonnable.

6.3 Qualité des estimations obtenues

Nous avons utilisé les homographies estimées par la méthode proposée dans les chapitres 4 et 5 pour ramener les vecteurs orthogonaux dans la vue de la première caméra. Nous avons ensuite estimé l'homographie donnant la top-view, et dessiné l'image de la grille contenue dans le plan du sol de la première caméra pour voir si l'estimation redresse le plan correctement. Les résultats obtenus sont donnés dans les figures 6.3(a) et 6.3(b).

Les figures 6.3(c) et 6.3(d) montrent le redressement des vecteurs orthogonaux. Nous voyons qu'effectivement l'homographie trouvée redresse ces paires de vecteurs jusqu'à un angle proche de 90°.

Cette méthode d'estimation de la top-view donne donc des résultats très satisfaisant : une fois que l'homographie a été trouvée, il ne restera plus qu'à centrer la top-view et à la redimensionner si besoin est, puis à considérer les transformations inverses pour obtenir le passage de la top-view aux vues caméras.



(a) Trapèze estimé

(b) Top-view estimée

(c) couples de vecteurs orthogonaux ramenés dans une vue caméra

(d) Couples de vecteurs redressés (normalisés et repositionnés)

FIG. 6.3 – Qualité des estimations obtenues pour la top-view

Conclusion

Durant ce travail, nous avons testé une méthode de calibration automatique de caméras en trois étapes : détection des pieds et de la tête, estimation des homographies entre les vues caméras et estimation de l'homographie donnant la top-view. La qualité de la calibration finale est relativement bonne, ce qui permet de penser que notre méthode est efficace.

Il est toutefois important de préciser que si l'une des étape donne de mauvais résultats, l'estimation finale sera fortement affectée. Ceci arrive tout particulièrement si la position estimée des pieds et de la tête n'est pas suffisamment précise. Il est donc très important de travailler sur des images de qualité ; ces images peuvent être bruitées, mais ne doivent pas contenir de bruit systématique comme une très grande ombre portée, un reflet au sol ou encore des éléments externes n'ayant pas été éliminés par l'algorithme de background substraction et formant une zone de taille supérieure à la taille de la personne. En effet, dans ce cas, l'algorithme de détection des pieds et de la tête convergera automatiquement vers une zone erronée, et il en résultera une mauvaise orientation du plan du sol. Notons toutefois que si un facteur de bruit se retrouve sur chacune des vues caméra de manière suffisamment importante pour que l'algorithme converge vers lui, la correspondance engendrée peut être considérée comme valide si l'objet est sur le sol. Ceci arrive notamment si une autre personne passe dans la zone couverte par la vue caméra. Toutefois, il est rare qu'un élément extérieur indésirable soit important dans les trois vues caméras en même temps ; il arrivera donc plus fréquemment que l'algorithme converge vers des objets différents selon les caméras.

D'autre part, avant d'utiliser cette méthode de calibration pour le tracking de personnes, il faudra encore estimer la position de la tête des personnes sur une vue caméra, étant donné la position des pieds dans la top-view ou dans

une vue caméra. Cette estimation ne peut pas être faite de manière similaire à celle donnant les homographies liées aux pieds, car le plan contenant les têtes est en règle générale pratiquement réduit à une droite, par le fait que la caméra se situe à peu près à hauteur de la tête. Les petites erreurs qui arrivent forcément lors du positionnement automatique de la tête auront alors un effet très important sur les homographies estimées, ce qui n'est pas acceptable. Il est donc nécessaire de déterminer la position de la tête par une méthode différente de celles développées dans ce travail.

Remarquons finalement que le modèle utilisé pour la détection des pieds et de la tête n'est pas forcément réaliste, puisque nous supposons que les personnes forment des ellipses. Il pourrait être intéressant de modéliser la position des personnes comme des rectangles, idée qui est sous-jacente à l'heuristique utilisée pour gérer le taux de valeurs aberrantes : en effet, lorsque nous considérons une bande de la même largeur que l'ellipse et dans la direction de la composante principale, nous nous rapprochons plus d'un modèle avec un rectangle qu'une ellipse. Le problème avec cette modélisation est la difficulté à estimer la position du rectangle ; alors que l'estimation d'une ellipse est donnée par une formule exacte, l'estimation d'un modèle avec une loi plus complexe comme un rectangle uniforme peut s'avérer relativement compliquée d'un point de vue calculatoire.

Bibliographie

[1] Ondrej Chum and Jiri Matas. Matching with prosac - progressive sample consensus. *CVPR*, 2005.

[2] Martin A. Fischler and Robert C. Bolles. Random sample consensus : A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 1981.

[3] François Fleuret, Jérôme Berclaz, Richard Lengagne, and Pascal Fua. Multi-camera people tracking with a probabilistic occupancy map. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2007, to be published.

[4] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN : 0521540518, second edition, 2004.

[5] Radu Horaud. Vision 3-d projective, affine et euclidienne. *Institut National de Recherche en Informatique et en Automatique*, 2000.

[6] Nils Krahnstoever and Paulo R. S. Mendonça. Bayesian autocalibration for surveillance.

[7] Christophe Pesch. Fast computation of the minimum covariance determinant estimator. *Universitãt Passau*, 1998.

[8] Lindsay I Smith. A tutorial on principal components analysis, 2002.