

ASSERVISSEMENT VISUEL.

1• OBJECTIF DE CE TRAVAIL.

Il s'agit pour vous de réaliser un asservissement visuel simple à l'aide de la caméra pan-tilt EVI-D100P de Sony. Cette caméra comporte trois motorisations permettant de réaliser deux rotations autour de deux axes perpendiculaires (a priori vertical et horizontal) et une motorisation du zoom.



Figure 1 : caméra EVI-D100P.

Pour estimer le déplacement du motif-cible à poursuivre, nous allons utiliser la technique du flot optique. Il s'agira ici simplement de maintenir le motif au centre de l'image tandis que la cible est déplacée.

2• RAPPELS CONCERNANT LE FLOT OPTIQUE.

L'objet du calcul du flot optique consiste à relier un déplacement projectif aux variations de niveau de gris d'une image.

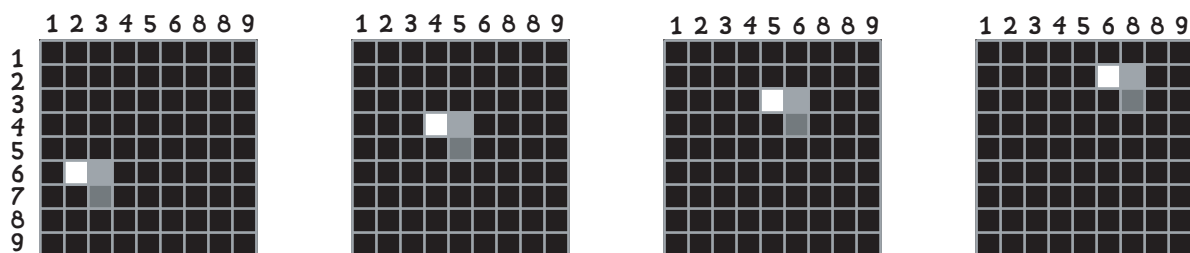


Figure 2 : quatre étapes du déplacement d'un motif.

Sur la figure 2 sont représentées quatre images issues d'une séquence au cours de laquelle un motif se déplace sur une image. La luminance du pixel de coordonnées (x,y) au temps t est notée $I(x,y,t)$. Contrairement à ce qui est représenté ici, la luminance est supposée continue.

Si, entre l'instant t et l'instant $t+dt$, le pixel de coordonnées (x,y) s'est déplacé de (dx,dy) alors on doit avoir :

$$I(x + dx, y + dy, t + dt) = I(x, y, t). \quad (1)$$

Or, si on fait une décomposition au premier ordre de $I(x + dx, y + dy, t + dt)$:

$$I(x + dx, y + dy, t + dt) = I(x, y, t) + \frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt + \varepsilon, \quad (2)$$

où ε est un terme du second ordre négligeable (ou supposé comme tel). Si on met en relation les expressions (1) et (2) alors il vient :

$$\frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt = 0 \quad (3)$$

qui est l'équation du flot optique. $\frac{\partial I}{\partial x}$ et $\frac{\partial I}{\partial y}$ sont les composantes du gradient de luminance et $\frac{\partial I}{\partial t}$ est la variation temporelle de la luminance au point considéré.

On note généralement $I_x(x, y)$, $I_y(x, y)$ et $I_t(x, y)$ les dérivées spatiales et temporelles de la luminance au point (x, y) . L'équation (3) peut donc être réécrite sous :

$$I_x(x, y)dx + I_y(x, y)dy = -I_t(x, y)dt \quad (4)$$

Si on veut poursuivre un motif, on suppose connue la première image, et le but est alors de connaître le mouvement global du motif (de façon à le compenser par la motorisation). Une première vision naïve de ce problème serait de dire qu'on essaye d'estimer le déplacement probable de chaque pixel du motif, c'est à dire de déduire (dx, dy) des variations de luminance de chaque pixel du motif.

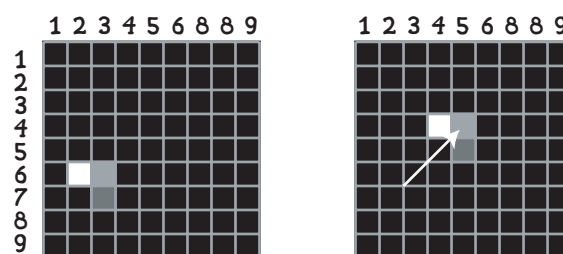


Figure 3 : déplacement d'un pixel du motif.

Or, comme nous l'avons vu en cours, l'équation (3) n'est pas inversible. Il faut donc voir chaque équation (3) associée à chaque pixel du motif comme une contrainte permettant de déterminer la meilleure commande à envoyer aux moteurs pour maintenir le motif sélectionné à la même position sur l'image.

3• RAPPELS CONCERNANT LE CODAGE DES IMAGES EN C.

Contrairement à Matlab, il est d'usage en C de coder une image, comme n'importe quelle matrice d'ailleurs et ce quelle que soit sa dimension, en un seul tableau à une dimension. Les différents pixels sont numérotés en fonction des colonnes puis des lignes selon le principe illustré sur la figure 4.

0	1	2	3	4	5
6	7	8	9	...	
				...	35

Figure 4 : codage des images et des tableaux en C.

Vous pouvez donc indexer la valeur (lin,col) tableau bidimensionnel nommé Tableau de Nlin lignes et Ncol colonnes par : `Tableau[(lin*Ncol)+col]`, ou utiliser des pointeurs. Vous pouvez vérifier que celà marche sur la figure 4.

4• RAPPELS CONCERNANT L'ASSERVISSEMENT VISUEL.

Le principe de l'asservissement visuel, dans le cadre de cette application, est de tenter de compenser les mouvements réels du motif par les mouvements de la caméra. Pour ce faire, il suffit d'estimer les paramètres du mouvement qu'aurait dû faire la caméra pour placer le motif dans sa nouvelle position et d'envoyer en consigne l'opposé de ces paramètres.

Pour ce faire il vous faut écrire de façon explicite la relation existant entre la commande ($d\theta_x, d\theta_y$) et la variation de position de la projection des points du motif sur l'image (dx, dy) sous la forme. En supposant qu'un point de coordonnée (X,Y,Z) du motif donné dans un repère associé à la position initiale de la cible se projette en (x,y) sur l'image initiale, on a :

$$\begin{bmatrix} sX \\ sY \\ s \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{12} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \quad (5)$$

où $\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{12} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$ est la matrice de rotation et $\begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$ le vecteur de translation associés au

changement de repère. $\alpha_u, \alpha_v, u_0, v_0$ sont les paramètres extrinsèques associés à la caméra.

Comme le repère utilisé est arbitraire, nous pouvons supposer qu'il est en simple translation suivant l'axe optique par rapport au repère caméra.

Question 1 : transformez la relation (5) pour faire intervenir cette simplification.

Si maintenant on provoque une faible rotation ($d\theta_x, d\theta_y$), les coordonnées de chaque point vont être modifiées et donc les coordonnées de leur projection.

En faisant l'approximation des petits angles ($\sin(d\theta) \approx d\theta$, $\cos(d\theta) \approx 1$), et donc des approximations au premier ordre, re-écrivez la relation (5) en faisant apparaître $(x+dx, y+dy, s+ds, d\theta_x, d\theta_y)$.

4.1• Asservissement Pan-Tilt à distance constante.

Si on ne réalise qu'un asservissement pan-tilt, on ne peut pas prendre en compte les variations de profondeur de la cible dans l'asservissement.

Nous allons, dans un premier temps, supposer que cette distance est constante (Z est identique pour tous les points et ne varie pas au cours du temps) et qu'on ne peut pas compenser les variations de taille de la projection ($ds=0$).

Question 2 : montrez que dans ce cas là, la relation entre $(d\theta_x, d\theta_y)$ d'une part et (dx, dy) d'autre part est linéaire et ne dépend pas de la position du point (X,Y,Z) ou (x,y) .

On a donc une relation de type $\begin{bmatrix} dx \\ dy \end{bmatrix} \approx A \begin{bmatrix} d\theta_x \\ d\theta_y \end{bmatrix}$. Le déplacement étant donc le même pour

tous les n points du motif $(x_1, y_1) \dots (x_n, y_n)$ on peut écrire :

$$\begin{bmatrix} I_x(x_1, y_1) & I_y(x_1, y_1) \\ \dots & \dots \\ I_x(x_n, y_n) & I_y(x_n, y_n) \end{bmatrix} \begin{bmatrix} dx \\ dy \end{bmatrix} = - \begin{bmatrix} I_t(x_1, y_1) \\ \dots \\ I_t(x_n, y_n) \end{bmatrix} : \mathcal{M} \begin{bmatrix} dx \\ dy \end{bmatrix} = - \begin{bmatrix} I_t(x_1, y_1) \\ \dots \\ I_t(x_n, y_n) \end{bmatrix}$$

Cette équation peut être inversée au sens des moindres carrés en utilisant la pseudo-inverse :

$$\begin{bmatrix} dx \\ dy \end{bmatrix} = -\mathcal{M}^\dagger \begin{bmatrix} I_t(x_1, y_1) \\ \dots \\ I_t(x_n, y_n) \end{bmatrix}, \text{ où } \mathcal{M}^\dagger \text{ est la pseudo-inverse de } \mathcal{M}.$$

Les valeurs $I_t(x_1, y_1) \dots I_t(x_n, y_n)$ sont les variations temporelles de la luminance aux points $(x_1, y_1) \dots (x_n, y_n)$. Elles sont supposées dues au déplacement du motif. Les valeurs $I_x(x_1, y_1) \dots I_x(x_n, y_n)$ et $I_y(x_1, y_1) \dots I_y(x_n, y_n)$ sont les valeurs de la dérivée de la luminance en $(x_1, y_1) \dots (x_n, y_n)$. Elles peuvent être obtenues par l'application d'un opérateur de dérivation sur le motif.

Remarquez que la matrice \mathcal{M} ne varie pas au cours du temps, elle peut donc être pseudo-inversée une bonne fois pour toute. Le calcul de la commande des moteurs de la caméra se résume donc à une multiplication de la matrice \mathcal{M} par la soustraction des niveaux de gris des pixels du motif aux niveaux de gris de l'image courante au même emplacement.

Attention les valeurs des β_u et β_v sont arbitraires, mais leur signe ne le sont pas.

4.2• Asservissement Pan-Tilt à distance variable en commandant de zoom.

En équation projective, en supposant que la projection réalisée par la caméra peut être

modélisée par un modèle sténopée, la modélisation du zoom fait simplement intervenir un facteur multiplicatif ζ modélisant le grandissement. Tout se passe comme si on modifiait la focale du système. L'équation (5) devient alors :

$$\begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} \alpha_u/\zeta & 0 & u_0 & 0 \\ 0 & \alpha_v/\zeta & v_0 & 0 \\ 0 & 0 & 1/\zeta & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{12} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \quad (6)$$

en supposant que la position d'équilibre est $\zeta = 1$.

Question 3 : il vous faut donc modifier vos équations de façon à prendre en compte le zoom dans vos équations et relier $d\theta_x$, $d\theta_y$ et $d\zeta$ aux variations $I_t(x_1, y_1) \dots I_t(x_n, y_n)$. Il sera plus simple de le faire en remarquant que s et $1/\zeta$ ont quasiment le même rôle dans ces équations et que des variations de s peuvent modéliser les variations de ζ . Le but est de maintenir le système dans un état tel que les variations de la commande compensent les variations de la projection sur l'image.

5. AIDE POUR LA PROGRAMMATION.

5.1. Le programme de base.

Vous disposez d'un projet de base comportant quatre fichiers : `ProgrammePrincipal.c`, `CommandeCamera.c`, `Gauss.c` et `Castan.c`. Copiez le dossier `TP_Asservissement` dans un dossier portant le nom de votre binôme, ouvrez-le (en cliquant sur `TP_Asservissement.slh`), compilez-le (menu Générer) et exécutez-le (menu Débugger).

Dans sa version actuelle, ce projet réalise toutes les initialisations nécessaires par rapport à la carte d'acquisition (MeteorII-Matrox), l'allocation et la libération d'un certain nombre de buffers utiles à l'application, l'acquisition d'un flot d'image et son transfert dans un buffer au format byte, la visualisation dans deux fenêtres indépendantes de l'image complète d'une part et de la zone d'intérêt d'autre part ainsi que deux boucles, l'une assurant l'acquisition de la zone d'intérêt, et l'autre permettant de visualiser la différence entre le motif de la zone d'intérêt et le motif courant de l'image à la même position (c'est cette information qui permet de commander la caméra).

Au cours de ce TP vous ne devez modifier que `ProgrammePrincipal.c`. Vous pouvez ajouter un autre fichier si vous le désirez mais en aucun cas ne modifiez les fichiers `CommandeCamera.c`, `Gauss.c` et `Castan.c`.

5.2. Les fonction à votre disposition.

Dans le fichier `CommandeCamera.c` vous avez à votre disposition les fonctions `Pan_Tilt` qui permet de commander en vitesse les moteurs de la caméra et `Zoom` qui permet de commander le zoom de la caméra. Le prototype de ces fonctions sont :

```
void Pan_Tilt(double vitesse_pan, double vitesse_tilt) ;
void Zoom(double vitesse_zoom) ;
```

sachant que pan est une rotation selon l'axe vertical et tilt une rotation selon l'axe horizontal, et que la vitesse du zoom varie de -7 (zoom out) à +7 (zoom in).

Dans le fichier Gauss.c vous avez à votre disposition la fonction PseudoInverse qui réalise la pseudo-inversion d'une matrice. Son prototype est :

```
int PseudoInverse(double *Matrice, int NombreLin, int NombreCol) ;
```

Cette fonction renvoie à la place de la matrice d'appel sa pseudo-inverse qui comportera donc NombreLin **colonnes** et NombreCol **lignes**. La valeur de retour vaut 1 si la matrice n'est pas singulière, 0 sinon.

Dans le fichier Castan.c vous trouverez la fonction castan qui réalise la dérivée d'une image au sens de l'algorithme de Shen-Castan. Son prototype est :

```
int castan(double *I, double *Ix, double *Iy, int NLin, int NCol, alpha) ;
```

Cette fonction admet en entrée un pointeur sur un tableau de doubles représentant l'image à dériver (*I), et deux pointeurs sur les images résultantes respectivement (*Ix) la dérivée suivant l'axe horizontal et (*Iy) la dérivée suivant l'axe vertical. alpha est le facteur d'extension du noyau utilisé pour la dérivation. Sa valeur typique est 0.4.

Vous pourrez essayer de voir son influence sur l'asservissement.

Le reste des fonctions utilisées sont issues de la librairie Matrox (Mil-lite). L'appel à ces fonctions est commenté dans le programme. Vous n'avez à priori pas besoin de modifier leur appel.

6• ASSERVISSEMENT D'UN SYSTÈME PAN-TILT SUR UNE CIBLE EN MOUVEMENT DE TRANSLATION.

C'est donc l'objet de votre travail. Le minimum que vous devrez réaliser et le suivi pan-tilt d'une cible (donnée en annexe) par ce système de vision. Pour les plus rapides d'entre vous, vous pouvez rajouter l'utilisation du zoom.

6.1• Asservissement pan-tilt.

Modifiez le fichier ProgrammePrincipal.c de façon à mettre en application la méthode de suivi par flot optique proposée ci-dessus. Lorsque cet asservissement semble fonctionner, testez sa robustesse à :

- changement d'aspect (d'orientation) de l'objet à suivre
- changement de profondeur
- changement de luminosité
- modification de paramètres (alpha par exemple)
- changement de cible, ...

Vous remarquerez d'autre part que l'asservissement n'est pas très stable.

Question 4 : expliquez cette instabilité et proposez et réalisez une solution.

6.2• Asservissement pan-tilt-zoom.

Modifiez votre programme de façon à asservir le zoom. Répondez aux mêmes questions que dans la partie 6.2.