Atelier Traitement d'Images
réalisation d'une plate-forme de stabilisation visuelle

M2 SIC Pro
Pierre Andry, Ghiles Mostfaoui

1 Introduction

Nous souhaitons réaliser une plateforme de stabilisation visuelle. Cette plate-forme sera articulée autour d'un capteur CCD classique permettant de réaliser l'acquisition d'un flux d'images continu. Le capteur sera monté sur une unité de stabilisation composée de deux servo moteurs, pilotés via une carte de contrôle recevant les ordres de stabilisation d'un ordinateur classique (liaison RS 232).

1.1 Vue d'ensemble

Le but de l'atelier consiste à produire :

- l'assemblage du dispositif de stabilisation (montage des servomoteurs, montage de la caméra, alimentation, liaison avec l'ordinateur)
- assurer l'acquisition du flux d'image (gestion de l'acquisition, de l'affichage, et traitements de base sur les images)
- assurer la communication entre le l'ordinateur et la carte de contrôle des servo moteurs
- réaliser un algorithme de suivi de points d'intérêt dans un flux vidéo pour compenser les mouvements de la caméra par action des servomoteurs
- réaliser une documentation du code conforme aux standard `doxygen`.

1.2 Constitution des équipes

Nous disposons de 6 dispositifs de plateformes pour effectuer vos tests. Vous devrez vous regrouper par équipes de 3 ou 4 personnes. Chaque plate-forme sera reliée à un ordinateur *fixe* (un fois montées, les plate-formes ne devrons pas être déplacées d'une machine à l'autre). S'il y a plus d'équipes que de plate-formes, **à vous de vous entendre pour le partage du temps de test sur les plateformes**.

1.3 Déroulement et notation

L'atelier se déroule sur deux journées pleines pendant lesquelles les équipent développent à temps plein leurs dispositifs de stabilisation. Bien que le sujet soit présenté sous la forme séquentielle de deux parties distinctes (partie I et Partie II) il est recommandé aux équipes de veiller à un bon découpage des lots de travail de manière à atteindre leurs objectifs. Un encadrant de la partie I sera présent le premier jour, et un encadrant de la partie II le deuxième jour.

La notation se fera sur la base d'une démonstration de 20mn par équipe le 3ème jour, ainsi que d'un rapport de 5 pages maximum décrivant le dispositif, ses fonctionnalités et l'organisation de l'équipe.

2 Partie I - Pierre Andry

2.1 Installation Matérielle

2.1.1 Liaison vidéo

Relier les caméras aux ordinateurs munis d'une carte d'acquisition BTTV. Vérifiez que les modules du noyau Linux `bttv.o` et VideoForLinux (`v4l2.o`) sont bien actifs. Vérifiez l'acquisition d'image à l'aide du programme `Xawtv` (source : composite 1, acquisition : Grabdisplay, Image : PAL).

2.1.2 Connexion des servos moteurs

Montez le support pan-tilt, connectez les servos moteurs à la carte mini-SSCII. Reliez la carte à l'ordinateur. Testez la communication (configuration : minicom) en redirigeant les informations appropriées sur le port série à l'aide de la commande `cat`.

2.2 Acquisition vidéo

Pour effectuer l'acquisition vidéo, nous utiliserons les librairies Video For Linux (V4L). Nous pourrons ainsi travailler directement en C sur les images stockées en mémoire. Vous pouvez, si vous le souhaitez, utiliser l'exemple de code `mvideo.c` utilisé pour interfacer la carte d'acquisition BTTV et afficher les images avec les librairies graphiques `gtk2+` . Si le code fournit peut servir de base au développement de votre application, il est vivement conseillé de le modifier afin de satisfaire aux contraintes de l'atelier selon les recommandations suivantes :

- identifiez la structure ou sont mises en mémoire les images.
- prévoyez de pouvoir exécuter l'application en mode graphique ou non graphique (pour plus de rapidité)
- prévoyez de modifier la structure de mise en mémoire des images pour pouvoir stocker plusieurs images successives
- prévoyez une option pour pouvoir connaître la fréquence de récupération des images (par consultation du temps système)

Les options importantes, (graphique *vs* non-graphique; affichage du temps vs pas d'affichage, etc...) devront être des options de compilation permettant de produire un exécutable le plus proche possible des besoins (et pas des options d'execution). Vous utiliserez la conventions des options `-D` pour passer les options de compilations qui seront interprétées par le pré-processeur (par exemple l'option `-DHAVE_GRAPHICS` pour signifier que l'on souhaite avoir l'interface `gtk2+`).

2.3 Premiers traitements video

Testez un seuillage dans l'espace RGB afin de pouvoir détecter une couleur vive dont vous vous servirez de base pour effectuer le tracking (par exemple une teinte rose, rouge, ou orange très prononcée). En mode graphique, vous veillerez a pouvoir afficher à l'écran les pixels satisfaisant les conditions du seuillage (c'est a dire les pixel correspondant à la teinte détectée). Essayez, autant que faire ce peut, d'obtenir un compromis entre un seuillage précis et robuste à la fois selon des conditions de luminosité les plus variées possibles.

A chaque test de nouveau traitement, vous veillerez à bien enregistrer les performances temporelles de votre système.

2.4 Communication série

Ecrivez un petit programme en C permettant de communiquer avec le carte mini-SSCII via le port RS232. Dans un premier temps, nous considérerons que les paramètres de communications sont fixés par `minicom`. Le programme n'aura plus alors qu'a écrire sur le *buffer* d'échange dédié au port série à l'aide de l'appel système C `write`.

2.5 Première fusion : Réalisation d'un suivi de couleur

Pour tester une première boucle de contrôle de la plate-forme, intégrez le code de commande des servo-moteurs à votre application de détection d'une couleur. La commande de la plate-forme sera découpée en une commande pour le servomoteur Pan (horizontal) et une commande pour le servomoteur TILT (vertical). Les commandes devront permettre à la caméra de rester centrée sur la zone de couleur détectée. Vous ajusterez donc les commandes motrices pour que le centre du champ de vision converge sur le centre de l'objet détecté. Vous veillerez à ce que la caméra reste stable quand elle atteint son objectif (pas d'oscillations parasites).

2.6 Réalisation d'un produit de convolution

L'objectif de cette partie est de mettre en place les outils nécessaire à la construction des détecteurs qui vont permettre d'identifier les points intéressants (indépendamment de la couleur cette fois) qui permettrons à notre système de savoir si l'image a bougé, et donc d'effectuer la compensation mécanique.

2.7 Préalable

Dans un premier temps, notre produit de convolution sera calculé à partir d'images en niveau de gris. Il faudra donc prévoir dans notre structure une conversion en niveaux de gris. La valeur du pixel i en niveaux de gris sera la moyenne des valeurs Rouge, Bleue, et Verte de i :

$$V_i^{NG} = (R_i + B_i + V_i)/3 \ .$$

avec $V_i^{NG}, R_i, B_i, V_i \in [0, 255]$

2.8 Produit de convolution

Le résultat de notre traitement (filtrage) sera une nouvelle image $g(x, y)$ égale au produit de convolution entre l'image d'origine $f(x, y)$ et la réponse impultionnelle du filtre $h(x, y)$:

$$g(x, y) = f(x, y) * h(x, y) \ .$$

.

pour rappel :

$$g(x, y) = \int_{-\infty}^{\infty} \int_{\infty}^{\infty} f(x - u, y - v) \cdot h(u, v) du dv$$

Vous testerez le produit de convolutions en filtrant l'image afin d'obtenir le gradient vertical en utilisant le masque (représentant la réponse impultionnelle) $3x3$ de Sobel :

-1	-2	-1
0	0	0
+1	+2	+1

.

Rappel : Attention, dans notre cas, notre signal 2D (image) n'est pas infini, il y aura donc un effet de bord à gérer, le produit de convolution de pourra être calculé en dehors des bornes de notre signal (image)

3 Partie II - Ghiles Mostafaoui

L'objectif est ici de vous initier au traitement d'images temps réel par une approche applicative. Cet atelier vous permettra, en premier lieu, de mettre en pratique les algorithmes "classiques" vus en cours de remise à niveau (filtrage, calcul du gradient etc.). La réalisation du produit final de stabilisation d'images nécessitera, dans un second temps, de pousser un peu plus loin vos connaissances et votre réflexion en abordant des aspects de niveau de traitement plus élevés tels que : l'estimation de mouvement, le suivi de zones d'intérêt etc.

3.1 Calcul de dérivée (gradient)

En utilisant les masques de Sobel (horizontal et vertical), programmez une fonction permettant le calcul est l'affichage de la valeur absolue du gradient des images acquises par la caméra.

3.2 Recherche des points d'intérêt

En utilisant la mesure proposée par Harris ou celle basée sur le gradient (voir cours joint), programmez un détecteur de point d'intérêt (pensez à afficher le résultat de la détection).

3.3 Estimation des vecteurs déplacement des points d'intérêt

Afin d'estimer les déplacement des points d'intérêt nous procéderons comme suit :

- Détecter les points d'intérêt de l'image I_t au temps t
- Détecter les points d'intérêt de l'image I_{t+1}au temps $t+1$
- Faire correspondre les points d'intérêt de l'image I_{t+1} avec ceux de l'image I_t. Le point $p(x, y, t+1)$ qui correspond à $p(x, y, t)$ est le point le plus proche (distance spatiale) dont le voisinage est ressemblant à celui de $p(x, y, t)$ (corrélation)
- Calculer les vecteurs déplacement des points d'intérêt (différence entre les coordonnées spatiales de $p(x, y, t+1)$ et $p(x, y, t)$)

3.4 Recherche du vecteur déplacement de la caméra et compensation par servo-moteurs

Afin d'estimer le mouvement global de la caméra nous partirons du principe que le fond immobile représente la majeure partie de l'image acquise par la caméra. En d'autres termes, les pixels représentant les objets "susceptibles" de bouger sont minoritaires. Cette hypothèse est évidemment loin d'être toujours valable mais elle bien pratique dans le cadre de cet atelier.

Si on arrive alors à calculer les vecteurs vitesse des pixels d'intérêt dans l'image (section 3.3) et qu'on quantifie l'occurrence de chaque vecteur déplacement, celui qui est, statistiquement, le plus représenté correspondra au mouvement de la caméra.

Nous procéderont alors comme suit :

- Créer une table d'accumulation $accu[][]$ dont les cases sont toutes mises à zéro dans un premier temps
- Pour chaque vecteur déplacement (dx, dy) calculé précédemment, incrémenter la cellule (case !) $accu[dx][dy]$ qui lui correspond dans la table d'accumulation
- Rechercher le maximum global de la table d'accumulation. Il représentera le vecteur déplacement (Cx, Cy) de votre caméra
- A partir du vecteur calculé (Cx, Cy), compenser le mouvement de votre caméra grâce aux servo-moteurs