Projet de Physique Numérique : Mesure de distances par stéréoscopie

Lucas Verney

2012/2013

1 Objectif

Le but de ce projet était de déterminer des distances séparant un observateur d'un objet à partir de deux photos de cet objet prises en déplaçant horizontalement l'appareil entre les prises de vue. En utilisant la stéréoscopie, on peut en effet déterminer la distance nous séparant de l'objet, tout comme notre cerveau est capable d'estimer les distances nous séparant des objets à partir des "images" captées par nos yeux.

Exemple :

Image de gauche

Image de droite

2 Méthode choisie

Il faut, dans un premier temps, déterminer le déplacement en pixels de l'objet entre les deux images. Dans un deuxième temps, il faut traduire ce déplacement en distance réelle nous séparant de l'objet.

2.1 Distance euclidienne dans l'espace RGB

Pour déterminer le déplacement en pixels, on considère une grille de $n \times n$ pixels, qu'on superpose à notre image de gauche (par défaut, $n = 50$ dans le programme). Pour chaque carrés de $n \times n$ pixels, on cherche le carré qui lui correspond le plus dans l'image de droite. Pour ce faire, on va parcourir chaque pixel de l'image de droite et comparer le carré de l'image de gauche et le carré dont le coin supérieur gauche est le pixel courant dans l'image de droite.

Les images étant chargées en couleur (RGB), on est dans un espace à trois dimensions (R, G et B) dont les coordonnées des pixels sont entre 0 et 255 (profondeur de couleur de 8 bits). On peut donc estimer la "distance" séparant les deux carrés par la moyenne sur le carré des distances pixel à pixel où la distance pixel à pixel est la distance euclidienne standard ($= \sqrt{(R_2 - R_1)^2 + (G_2 - G_1)^2 + (B_2 - B_1)^2}$).

2.2 Cross-correlation *via* FFT

Une autre possibilité est d'utiliser la cross-correlation (ou la phase correlation) afin de déterminer la position de notre objet.

On commence par charger les images en niveaux de gris et on récupère le carré qu'on considère dans une nouvelle image, de la même taille que l'image de droite. On complète cette image par du noir (des 0) qui n'interviendront donc pas dans la transformation de Fourier.

On effectue ensuite les transformations de fourier de chaque image et on multiplie le spectre de l'image de droite par le conjugué du spectre du carré dont on cherche la position. On normalise ensuite la transformation de Fourier (en divisant chaque la valeur de chaque point par son module) et on repasse en espace réel (*via* une transformation de Fourier inverse.

On cherche ensuite la valeur de la plus grande amplitude sur cette image, qui correspond à la position du carré qu'on cherche.

Une fois cette distance déterminée, on peut en déduire la distance nous séparant de l'objet en utilisant les formules explicitées dans [4]. Connaissant les caractéristiques de l'appareil, il est facile de traduire le déplacement en pixels précédent en distance réelle.

3 Résultats

Le programme obtenu a été testée sur le jeu d'images suivant (les tests ont été effectués en considérant le coin inférieur gauche du carré noir) :

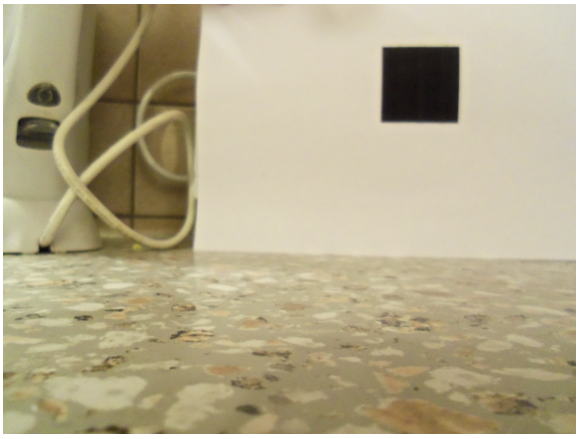

Image de gauche

Image de droite

Le programme décrit précédemment permet de déterminer très précisément le déplacement du coin inférieur gauche et on ne peut obtenir une meilleure précision manuellement. Ce résultat a été obtenu en une vingtaine de secondes en utilisant les 4 cœurs du processeur, ce qui est relativement long. En effet, pour parcourir toute l'image avec la méthode précédente et des carrés de 50 pixels de large, il faudrait environ $30h$.

Après conversion du déplacement en pixels en distance réelle, on trouve une distance réelle de $15cm$ contre $35cm$ en réalité (en utilisant des caractéristiques "moyennes" pour l'appareil photo). Cependant, le déplacement étant convenablement détecté, cet écart provient d'une méconnaissance des caractéristiques précises de l'appareil photo utilisé (notamment sa distance focale).

En conclusion, cette technique nous permet de déterminer des distances très précisément mais est très coûteuse en ressources et il est difficilement envisageable de couvrir une photo entière avec cette méthode.

La technique utilisant la transformée de fourier rapide est nettement plus rapide (une vingtaine de secondes par image) et donne des résultats légèrement moins précis mais largement comparables.

Remarque : Il faut également noter qu'on ne détermine pas la distance nous séparant d'un objet (ou d'un point) précis mais une moyenne de cette distance sur un carré de n pixels de large. Il faut donc choisir la taille de ce carré petite devant l'échelle caractéristique de variation des distances (liée à la taille transversale des objets).

Remarque : On peut également utiliser un algorithme de détection de contours (type filtre de Sobel qui calcule le "gradient" de l'image) pour travailler sur une image en niveau de gris et déterminer les distances nous séparant des contours des objets, ceci afin d'obtenir une meilleure précision sur des images chargées.

4 Optimisations possibles

Comme on l'a vu précédemment, la méthode employée est très coûteuse en ressources. On peut envisager diverses optimisations :

- On peut envisager d'appliquer un filtre de détection de contours (type filtre de Sobel) avant de travailler sur l'image. On peut alors se restreindre aux zones d'intérêt, c'est-à-dire aux contours représentés par des pixels blancs.
- Pour accélérer le calcul des distances, on peut envisager de paralléliser au maximum les opérations. Dans le code présenté, on utilise jusqu'à 4 cœurs du processeur mais l'utilisation de la puce graphique pourrait accélérer le calcul, celle-ci travaillant naturellement avec des calculs parallèles.
- Jusqu'ici, on n'a envisagé qu'un déplacement horizontal (ou vertical en travaillant sur une image retournée). On pourrait envisager le cas d'un déplacement quelconque dans l'espace (translation) en adaptant les formules.

Références

[1] Opencv wiki page about template matching. http://docs.opencv.org/doc/tutorials/imgproc/histograms/template_matching/template_matching.html, dernière consultation en janvier 2013.

[2] Wikipedia page about phase correlation. http://en.wikipedia.org/wiki/Phase_correlation, dernière consultation en janvier 2013.

[3] Jernej Mrovlje1 et Damir Vranči : Distance measuring based on stereoscopic pictures. http://photon07.pd.infn.it:5210/users/dazzi/Thesis_doctorate/Info/Chapter_6/Stereoscopy_(Mrovlje).pdf, article daté de 2008, dernière consultation en décembre 2012.

[4] Edwin TjandranegaraB : Distance estimation algorithm for stereo pair images. http://docs.lib.purdue.edu/ecetr/64/, article daté de 2005, dernière consultation en décembre 2012.