TP – Transformée de Hough

Durée allouée pour ce TP : 2 séances de 2h
Pour les plus rapides d'entre vous : un bonus (voir fin du TP)

Question 1 : préliminaires

Créez une première fenêtre GLUT ; cette fenêtre n° 1 va nous servir à éditer des points à la souris : à chaque fois que l'on clique avec le bouton gauche de celle-ci, cela entraîne :

- la mémorisation dans un tableau nommé PTS des coordonnées du point cliqué sur la fenêtre; aidez-vous de la structure suivante pour représenter un point :

```
typedef struct
    {
        int abs;
        int ord;
    } point;
```

- la représentation graphique de ce point dans la fenêtre n° 1.

Prévoyez de plus les évènements « clavier » suivants :

- touche 'a' : affichage dans la console des coordonnées de tous les points contenus dans le tableau PTS.
- touche 'e' : effacement de tous les points contenus dans le tableau PTS.

Question 2 : le vif du sujet

a - Créez deux autres fenêtres, qui vont nous servir à représenter deux espaces de Hough (ou encore espaces des paramètres).

b - Créez une première fonction « onetomany » qui utilise le tableau PTS et qui affiche dans la fenêtre n° 2 les points calculés par une transformée de Hough en version « one to many ».

c - Créez une deuxième fonction « manytoone» qui utilise le tableau PTS et qui affiche dans la fenêtre n° 3 les points calculés par une transformée de Hough en version « many to one».

Petit rappel pour vous aider : on représente une droite par l'équation $x*\cos\theta + y*\sin\theta = \rho$.

- En version « one to many » : pour tout point i du tableau PTS, il suffit de tracer dans la fenêtre n° 2 les points de coordonnées (θ,ρ) pour $\theta \in]-90°, 90°]$, avec ρ=PTS[i].abs*cos θ + PTS[i].ord*sin θ.

- En version « many to one » : pour tout point i et pour tout point j du tableau PTS (i>j), il suffit de tracer dans la fenêtre n° 3 le point de coordonnées (θ,ρ), qui est la solution du système suivant :

$$\begin{cases} PTS[i].abs * \cos\theta + PTS[i].ord * \sin\theta = \rho \\ PTS[j].abs * \cos\theta + PTS[j].ord * \sin\theta = \rho \end{cases}$$

Solution :

$$\begin{cases} \theta = -\arctan\dfrac{x1 - x2}{y1 - y2} \\ \rho = \dfrac{-(x1 * y2 - x2 * y1)}{\sqrt{(x1 - x2)^2 + (y1 - y2)^2}} \end{cases}$$

Aidez-vous de la structure suivante pour représenter un point de l'espace de Hough :

```
typedef struct
    {
      float theta;
      float rho;
    } ptHough;
```

Question 3 : premiers tests

Réalisez le petit scénario suivant :

- Dans la fenêtre n°1, éditez une quinzaine de points quasiment alignés et observez ce qu'il se passe dans les espaces de Hough version « one to many » et version « many to one » (fenêtre n°2 et n° 3). Que remarquez-vous ?

- Ajoutez une autre quinzaine de points quasiment alignés eux aussi, mais selon une droite différente de la première. Que remarquez-vous ?

- Effacez tout (en tapant sur la touche 'e' du clavier). Dans la fenêtre n°1, éditez une quinzaine de points quasiment alignés, et ajoutez ensuite d'autres points répartis au hasard. Que remarquez-vous ?

- Effacez tout (en tapant sur la touche 'e' du clavier). Dans la fenêtre n°1, commencez à éditer une première série de points quasiment alignés, puis un peu plus loin, terminez votre droite imaginaire avec une seconde série de points. Que remarquez-vous ?

Question 4 : Détection de zones d'accumulation dans un espace de Hough.

a - par l'utilisateur

Ajoutez la fonctionnalité suivante à votre programme : lorsque l'on clique un point dans la fenêtre n° 3 (« many to one »), on récupère un point de l'espace de Hough de coordonnées (θ,ρ). Ce point correspond à une droite dans l'espace de la fenêtre n° 1, que vous allez tracer tout en maintenant l'affichage des points contenus dans le tableau PTS.

Que se passe-t-il si :

- vous cliquez n'importe où dans l'espace de Hough ?
- vous cliquez sur une zone où il existe une faible accumulation ?
- vous cliquez sur une zone où il existe une forte accumulation ?

b - détection automatique

Vous avez compris avec la question précédente que grâce à la transformée de Hough, le problème de détecter des droites dans l'espace d'une image revient à trouver des agglomérations de points dans l'espace des paramètres.

Nous allons essayer de détecter automatiquement l'accumulation dans l'espace de Hough la plus évidente. Pour cela, l'idée la plus simple consiste à diviser l'espace de Hough en cellules et à compter les points qui « tombent » dans chaque cellule. La cellule qui contient le plus de points correspond à la zone de forte accumulation que nous recherchons.

Un peu de technique pour vous aider :

1) Déclarez un tableau d'entiers 2D, appelé Accu (pour accumulateur). Chaque case de ce tableau correspond à une cellule de l'espace de Hough (voir figure 1). N'oubliez pas d'initialiser toutes les cases à zéro !

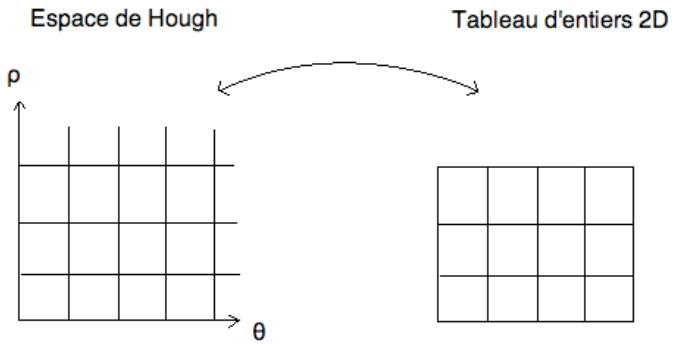

Fig. 1 : Le concept de discrétisation de l'espace de Hough se concrétise dans votre programme par un tableau d'entiers 2D.

2) Modifier votre fonction « manytoone » (créée à la question 2-c), de sorte que non seulement cette fonction trace les points de l'espace de Hough, mais qu'elle incrémente aussi les cases du tableau Accu : chaque fois qu'un point (θ,ρ) « tombe » dans la case (i,j), l'entier contenu dans cette case est augmenté de 1. Vous allez devoir vous aider d'une petite fonction qui fait correspondre un point (θ,ρ) à une case (i,j) du tableau Accu.

3) Ecrivez une courte fonction qui recherche la case du tableau Accu ayant le plus grand « score », et qui retourne en sortie un point (θ,ρ) qui correspond à cette case. Vous allez devoir imaginer un moyen de passer d'une case (i,j) à un point (θ,ρ).

4) En tapant la touche ‘s’ du clavier (‘s’ pour solution), tracez dans la fenêtre n° 1 la droite que votre programme a détectée automatiquement.

Le nombre de cases du tableau Accu est lié à la taille des cellules de l’espace de Hough : plus il y a de cases dans Accu et plus les cellules sont petites (et inversement). Jouez sur la taille du tableau Accu pour modifier la taille des cellules.

Que se passe-t-il si les cellules sont trop petites ? Et si elles sont trop grandes ?

BONUS :

La transformée de Hough est un algorithme de vote, tandis que la méthode des moindres carrés est une méthode variationnelle.

Munissez-vous de l’équation cartésienne d’une droite, $y = ax+b$, et tentez d’estimer les paramètres a et b par la méthode des moindres carrés. Notez que la représentation cartésienne d’une droite est acceptable pour la méthode des moindres carrés, contrairement au cas de la transformée de Hough.

L’appui sur la touche ‘m’ du clavier provoque le tracé dans la fenêtre n° 1 de la droite estimée par les moindres carrés. Reprenez le scénario décrit dans la question 3 ; dans quels cas la transformée de Hough et la méthode des moindres carrés sont-elles en accord ? Pourquoi la transformée de Hough fonctionne-t-elle systématiquement ?