Traitement d'images (2019-2020)

Travaux pratiques - séance 2

- A la fin du TP, envoyez une archive avec ce que vous avez réussi à faire (sources + fichier readme.txt très concis) à emanuel.aldea@u-psud.fr, avec le sujet [TI5A]"nom"-"prenom"-travail-TP2 (par exemple [TI5A]Dupont-Pierre-travail-TP2). Pas de fichiers image dans l'archive.
- Avant 22 dec. 23 :59, envoyez à la même adresse la version finale des sources + un compte-rendu où vous détaillez bien les problèmes rencontrés, en justifiant vos solutions. Le sujet du message sera [TI5A]nom-prenom-final2-CR. De même, évitez de joindre les images de test que j'ai mises à votre disposition au début.

L'objectif du TP est d'implémenter un détecteur d'objets circulaires par une méthode cumulative de type Hough.

Exercice 1 — La transformée de Hough pour les cercles

On considère une paramétrisation de type (r, c, rad) pour définir un cercle de rayon rad pixels dans l'image, dont le centre est situé sur la ligne r et la colonne c. Comme nous sommes obligés de considérer un ensemble discret de paramètres, on retient toutes les positions $r \in [r_{min}, r_{max}]$ avec un pas de discrétisation de δr, et ainsi de suite pour $c \in [c_{min}, c_{max}]$ avec un pas de discrétisation de δc et pour $rad \in [rad_{min}, rad_{max}]$ avec un pas de discrétisation de δrad.

1. Pour l'image four.png fournie, de taille 100 × 100 pixels, considérons que $r_{min} = 1$, $r_{max} = 100$, $\delta r = 2$. Combien de valeurs discrètes aura-t-on pour la coordonnée r des cercles ? Et si $\delta r = 0.5$?
2. Pour la même images, en supposant que $r_{min} = 1$, $r_{max} = 100$, $\delta r = 1$, $c_{min} = 1$, $c_{max} = 100$, $\delta c = 1$, $rad_{min} = 5, rad_{max} = 100\sqrt{2}, \delta rad = 1$, quel est le nombre total de cercles qu'on peut décrire avec ces trois variables ?
3. Le tableau tridimensionnel acc associe à la case $acc(i, j, k)$ le cercle situé à la i-ème valeur discrète de r, la j-ème valeur discrète de c, et la k-ème valeur discrète de rad. Quel est le cercle associé au $acc(1, 1, 1)$? Au $acc(10, 7, 30)$?
4. Inversement, quelle est la case de l'accumulateur associée au cercle centré dans le pixel (40,40) et de rayon $rad = 13$? Attention : les indices i,j,k doivent être entiers.

Exercice 2 — Implementation du détecteur

1. On rappelle le fonctionnement de l'algorithme de détection de cercles :
   1. (optionnel) Filtrage Gaussien (en cas de bruit ou de détails très fins)
   2. Filtrage de Sobel, calcul de la magnitude de gradient I_{mag} dans chaque pixel
   3. Tous les pixels dont la magnitude est au dessus d'une fraction t de la valeur maximale dans I_{mag} sont considérés comme des pixels du contour. Note : visualisez l'image des contours pour être sûrs que vous avez dedans les contours des objets recherchés.
   4. Initialisez toutes les valeurs de l'accumulateur acc à 0.
   5. Pour chaque pixel de contour, considérez toutes les (r,c) possibles, calculez le rayon rad pour que le cercle situé en (r,c) passe par le pixel respectif, et incrémentez dans l'accumulateur la case qui correspond à (r, c, rad).

FIGURE 1 – Détection de cercles

6. Identifiez dans l'accumulateur les maximas locaux - les cases avec des valeurs supérieures aux 26 cases voisines (car l'accumulateur est tridimensionnel).
7. Selectionnez les N valeurs les plus grandes, et à partir des indices (i,j,k) récupérez les (r,c,rad) correspondants et visualisez les cercles en passant par OpenCV.

Note 1 : les cercles plus grands reçoivent plus de votes, donc il faudrait normaliser les valeurs de l'accumulateur pour ne pas privilégier les cercles grands.
Note 2 : pour mettre un vote, on peut incrémenter soit par 1, soit par la magnitude du gradient dans le pixel respectif etc.
Essayez de trouver une solution qui fonctionne pour des images variées (voir par exemple Figure 1 pour des images avec ou sans bruit).

Exercice 3 — Temps de calcul

1. Avec une fonction adaptée (par exemple `getTickCount()` de OpenCV ou `gettimeofday`), mesurez le temps de calcul pour `four.png`. En regardant le code, on peut se rendre compte que la complexité de l'algorithme est de l'ordre de N^4 (pour quoi ?), ou N est la taille en pixels de l'image (100 px). Quel sera le temps de calcul envisagé pour traiter une image avec N proche de 600 px, comme `coins2.jpg` ?
2. Pour résoudre ce problème de complexité et arriver à traiter des images de taille importante, vous pouvez choisir une des deux solutions suivantes (bonus si vous implémentez les deux) :
   1. Calculez à l'aide du filtre de Sobel la direction du gradient, et n'incrémentez dans l'accumulateur les cases que pour les cercles dont le centre se trouve dans la direction du gradient (modulo un petit angle β pour l'incertitude). Dans ce cas, il faudrait essayer d'identifier ces pixels qui se trouvent dans l'image dans la direction du gradient (à l'intérieur d'un cône) de manière très efficace, sans parcourir tous les pixels de l'image.
   2. Utilisez une représentation pyramidale de l'image. En supposant que vous réduisez la taille de votre image par 2, vous pouvez faire une détection pour des cercles de rayon $[rad_1 \;\; rad_2]$ et garder les N meilleurs détections. Après vous revenez à la résolution initiale, mais vous n'effectuez une recherche que pour les rayons en $[rad_{min} \;\; 2 * rad_1]$ parce que le reste a été déjà analysé à la résolution plus faible. Vous pouvez néanmoins affiner la position des N cercles détectés auparavant. Vous pouvez faire cette opération en cascade, à partir d'une résolution très réduite de l'image initiale (4-8 fois).