Détection de cercles avec la transformée de Hough

Sébastien Klasa - Polytech Paris-Sud
klasa.sebastien@gmail.com

3 janvier 2019

1 Exercice 1

Question 1 Pour $\delta r = 2$ nous aurons $\left\lfloor \frac{r_{max} - r_{min}}{\delta r} \right\rfloor = 49$ valeurs discrètes. Pour $\delta r = 0.5$ nous aurons 198 valeurs.

Question 2 Nous pouvons décrire $\left\lfloor \frac{r_{max} - r_{min}}{\delta r} \right\rfloor \times \left\lfloor \frac{c_{max} - c_{min}}{\delta c} \right\rfloor \times \left\lfloor \frac{rad_{max} - rad_{min}}{\delta rad} \right\rfloor = 1332936$ cercles avec ces trois variables.

Question 3 $acc(1, 1, 1)$ correspond au cercle de rayon 1 centré en $(1, 1)$. $acc(10, 7, 30)$ correspond au cercle de rayon 30 centré en $(10, 7)$.

Question 4 La case associée est $acc(40, 40, 13)$.

2 Exercice 2

Avant tout traitement, j'applique un léger filtre gaussien pour atténuer le bruit. Ensuite, j'applique un filtre de Sobel pour détecter les contours. Cette image des contours me permet d'incrémenter l'accumulateur par la magnitude des pixels des cercles.

Pour éviter de privilégier les cercles plus grands, je normalise les valeurs de l'accumulateur par le périmètre du cercle associé. Je cherche ensuite les maxima locaux pour finalement sélectionner les n plus grands.

Les figures 1, 2 et 3 sont quelques exemples de détection de cercles.

FIGURE 1 – Exemple de détection de cercle.

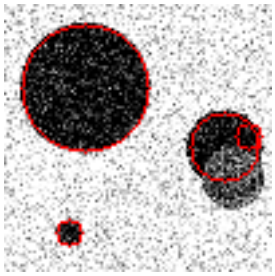

FIGURE 2 – Exemple de détection de cercle.

FIGURE 3 – Exemple de détection de cercle.

3 Exercice 3

Question 1 Le temps de calcul nécessaire pour l'image four.png est 2150 ms. La complexité en $O(n^4)$ peut être expliquée par le fait que nous parcourons tous les pixels de l'image ($O(n^2)$) pour chaque ligne et chaque colonne de l'accumulateur. Le temps estimé pour une image de 600 px est 2150 ms $\times 100^{-4} \times 600^4 =$ 2786400 ms $= 46.44$ minutes .

Question 2 - Sans du gradient La droite passant par le point P dans le sens du gradient d'angle θ peut être représentée comme suit :

$$y = \tan(\theta)x + b$$
$$b = P_y - \tan(\theta)P_x$$

on peut ensuite définir deux cônes avec deux autres droites :

$$y = d_1(x) = \tan(\theta + \delta\theta)x + b_1$$
$$b_1 = P_y - \tan(\theta + \delta\theta)P_x$$

$$y = d_2(x) = \tan(\theta - \delta\theta)x + b_2$$
$$b_2 = P_y - \tan(\theta - \delta\theta)P_x$$

et nous allons parcourir uniquement les pixels dans le cône. Un pixel P est dans un des deux cônes si :

$$d_1(P_x) \leq P_y \leq d_2(P_x) \text{ ou } d_1(P_x) \geq P_y \geq d_2(P_x)$$

Nous allons donc parcourir, pour chaque ligne x, les pixels entre $d_1(x)$ et $d_2(x)$. Cependant, je n'ai pas eu le temps d'implémenter cette amélioration.