TP 2

Exercice 1: La transformée de Hough pour les cercles

Q1) ***Pour l'image four.png fournie, de taille 100 100 pixels, considérons que rmin = 1, rmax = 100, deltar =2. Combien de valeurs discrètes aura-t-on pour la coordonnée r des cercles? Et si deltar = 0.5?***

Solution :deltar

La discrétisation de r dépend de la formule suivante :

Nr=((rmax-rmin)/deltar)+1

1. **Pour deltar=2**:
   Nr=((100-1)/2)+1=50
   Donc, il y aura **50 valeurs discrètes** pour r.
2. **Pour deltar=0.5**:
   Nr=((100-1)/0.5)+1=199
   Donc, il y aura **199 valeurs discrètes** pour r.

Q2) ***Pour la même images, en supposant que rmin = 1, rmax = 100, deltar= 1, cmin = 1, cmax = 100, deltac = 1, radmin = 5, radmax = 100√ 2 deltarad = 1, quel est le nombre total de cercles qu'on peut décrire avec ces trois variables?***

Le nombre total de combinaisons possibles dans l'accumulateur dépend des valeurs discrètes pour r, c, et rad. Utilisons la formule :

$$NCercles=Nr\times Nc\times Nrad$$

1. **Pour r** :
   Nr=((100-1)/1)+1=100.
2. **Pour c** :
   Nc=((100-1)/1)+1=100.
3. **Pour rad** :
   radmax=((100√ 2 -5)/1)+1=137.
4. **Nombre total de cercles** :
   NCercles=100×100×137=1,370,000.
   Donc, **1,370,000 cercles** peuvent être décrits.

Q3) ***Quel cercle est associé à acc(1,1,1) et acc(10,7,30)?***

Pour chaque case (i,j,k)(i, j, k)(i,j,k) dans l'accumulateur, les paramètres (r,c,rad)(r, c, \text{rad})(r,c,rad) sont calculés comme suit :

r=rmin+(i−1)*deltar

c=cmin+(j−1)*deltac

rad=radmin+(k−1)*deltarad

1. **Pour acc(1,1,1)**:
    - $r=1+(1-1)\cdot 1=1$
    - $c=1+(1-1)\cdot 1=1$
    - $rad=5+(1-1)\cdot 1=5$
    - Le cercle associé est (r=1,c=1,rad=5)
2. **Pour acc(10,7,30)** :
    - $r=1+(10-1)\cdot 1=10$
    - $c=1+(7-1)\cdot 1=7$
    - $rad=5+(30-1)\cdot 1=34$
    - Le cercle associé est (r=10,c=7,rad=34)

<u>Q4</u>)*"Quelle est la case acc(i,j,k) associée au cercle centré en (40,40) avec rad=13 ?"*

Inversons les formules pour calculer i,j,k à partir de r,c,rad:

i=((r-rmin)/deltar)+1

j=((c-cmin)/deltac)+1

k=((rad-radmin)/deltarad)+1

1. **Pour r=40** :
   i=((40−1)/1)+1=40
2. **Pour c=40** :
   j=((40−1)/1)+1=40
3. **Pour rad=13** :
   k=((13−5)/1)+1=9

Case associée :acc(40,40,9)

Exercice 2: Implémentation du détecteur

Cet exercice vise à implémenter un détecteur de cercles basé sur la transformée de Hough, une méthode cumulative qui permet de détecter des formes géométriques dans des images. À travers ce TP, l'objectif est de discrétiser les paramètres des cercles et d'utiliser un accumulateur tridimensionnel pour identifier les cercles présents dans une image donnée.

Ce rapport présente la démarche suivie, les résultats obtenus, et les ajustements réalisés pour corriger les problèmes rencontrés, notamment l'apparition de cercles parasites dans les résultats initiaux.

Méthodologie

Étape 1 : Prétraitement de l'image

- Pour l'image "four.png" ,un filtre de Sobel a été appliqué pour calculer les gradients en x et y. La magnitude du gradient a ensuite été dérivée et normalisée pour obtenir des valeurs comprises entre 0 et 255.
- Un seuillage simple a permis d'extraire les pixels correspondant aux contours, essentiels pour la phase de vote dans l'accumulateur. Ces contours ont été visualisés pour s'assurer qu'ils incluent bien les objets recherchés.

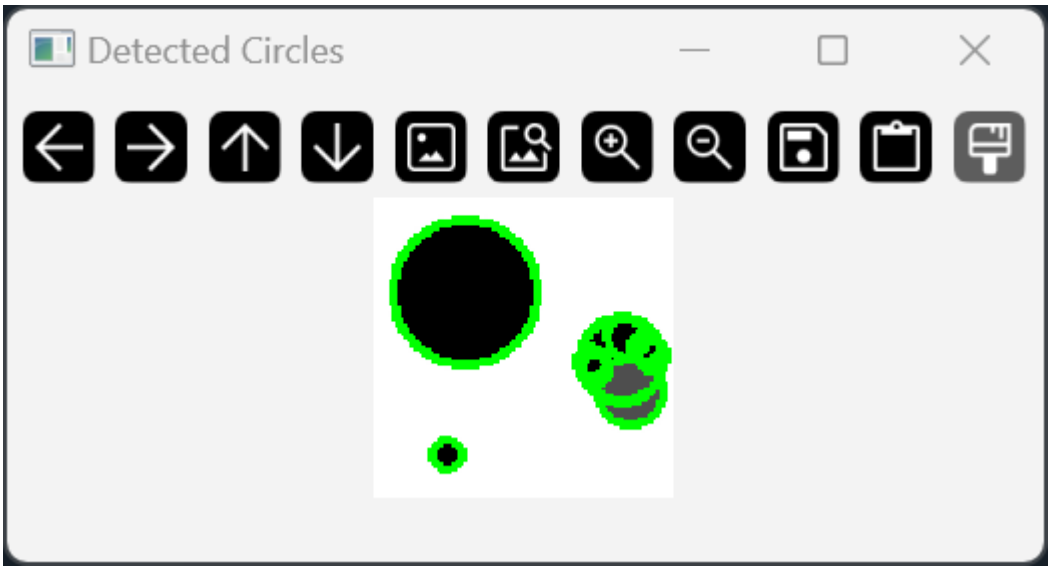
Étape 2 : Mise en place de l'accumulateur tridimensionnel

- Les paramètres de discrétisation des cercles ont été définis comme suit :
  - **Centre :** $r \in [1,100]$, $c \in [1,100]$, pas = 1.
  - **Rayon :** $rad \in [5,50]$, pas = 1.
- L'accumulateur a été initié avec des zéros, puis chaque pixel de contour a contribué aux votes pour les cercles qu'il peut constituer. Une normalisation des votes par le rayon a été intégrée pour éviter de favoriser les grands cercles.

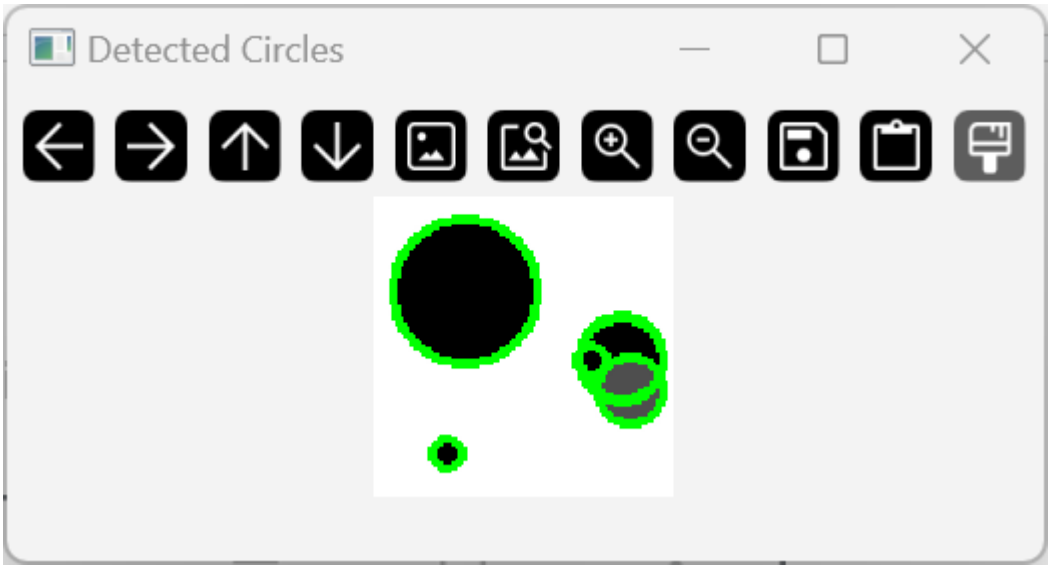
Étape 3 : Identification des maxima locaux

- Un filtre de maximum local a été appliqué pour identifier les points correspondant aux cercles détectés. Initialement, le seuil pour retenir les maxima a été fixé à 0.5×acc.max() et puis d'autres valeurs de test, mais cela a conduit à l'apparition de cercles

parasites, notamment un petit cercle détecté à l'intérieur d'un grand cercle.



Avec un seuil de 0.4×acc.max()



Avec un seuil de 0.4×acc.max()

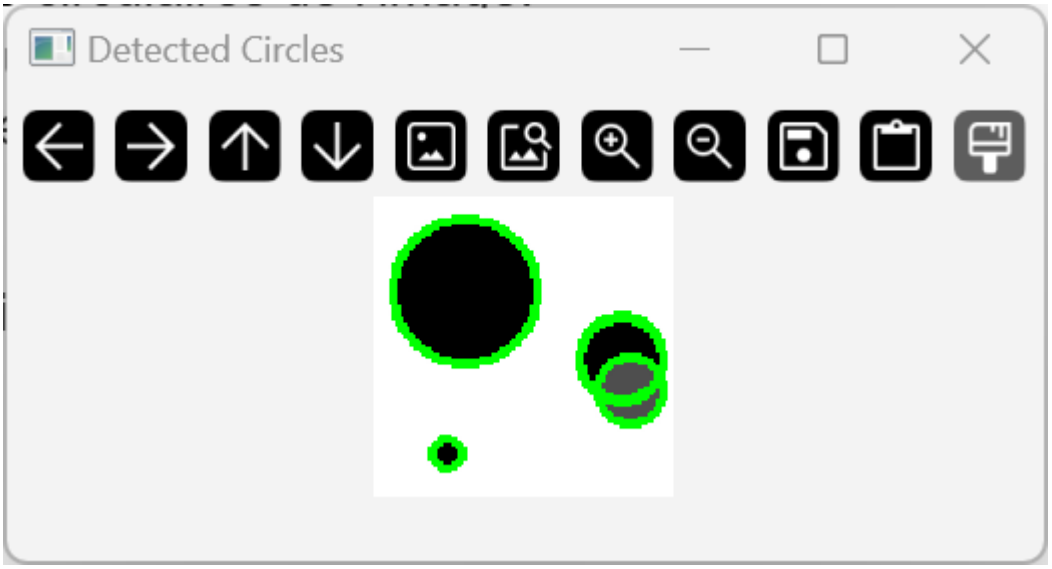
- Pour corriger ce problème, le seuil a été ajusté à 0.55×acc.max(), ce qui a permis d'éliminer les cercles indésirables. Ce simple ajustement a considérablement amélioré la qualité des résultats, sans nécessiter d'autres modifications du code, pourtant ce n'était pas facile pour le détecter.

<u>Étape 4 :</u> Visualisation des cercles détectés

- Les cercles détectés ont été superposés à l'image en utilisant OpenCV. Le résultat final est satisfaisant : tous les cercles pertinents ont été correctement identifiés sans apparition de formes parasites.

  Résultats obtenus

  - **Contours détectés :** Les contours identifiés par le filtre Sobel incluent efficacement les limites des objets circulaires de l'image.
  - **Détection des cercles :** Après ajustement du seuil, tous les cercles présents dans l'image ont été détectés avec précision, sans cercle parasite.
- Le résultat final est illustré par l'image ci-dessous, où les cercles détectés sont affichés en vert.



- Pour tester la robustesse de l'algorithme, une version bruitée de l'image "fourn.png" a été utilisée. Cette étape vise à évaluer la capacité de l'algorithme à détecter les cercles en présence d'un bruit significatif dans l'image.

Étape 1 : Élimination du bruit

- Un filtre Gaussien a été appliqué à l'image pour atténuer le bruit. Ce prétraitement a permis de conserver les contours principaux tout en réduisant le bruit visuel susceptible d'interférer avec les étapes suivantes.

Étape 2 : Détection des contours

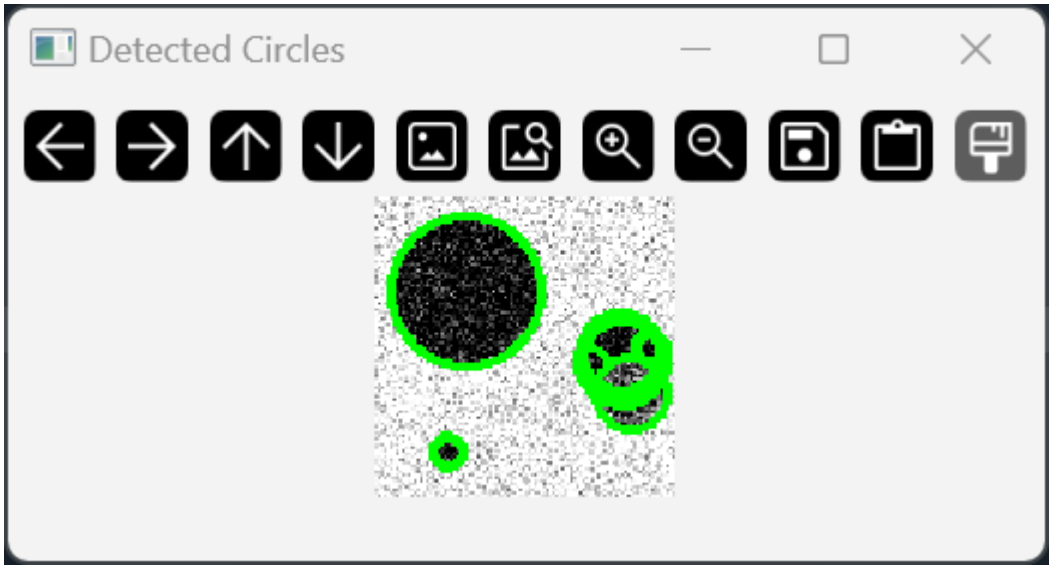
- Après filtrage, les contours ont été déterminés à l'aide du filtre Sobel et d'un seuil ajusté. Les contours obtenus incluent les éléments circulaires principaux et éliminent la majorité des perturbations dues au bruit. Cette étape a confirmé que le prétraitement réduit efficacement le bruit.

Étape 3 : Détection des cercles

- L'application de l'algorithme sur cette image bruitée a révélé plusieurs défis :
  - En réglant le seuil pour les maxima locaux sur différentes valeurs, plusieurs cercles indésirables ont été détectés. Ces cercles supplémentaires varient en nombre et en position selon le seuil utilisé.
  - Ces détections indésirables sont probablement dues à des zones de bruit résiduel interprétées comme des contours mais cela n'apparaît pas lors de l'affichage de contours.

Résultats:

Le résultat final, illustré dans l'image ci-dessous, montre les cercles principaux correctement détectés, mais également la présence de cercles supplémentaires. Ces derniers sont principalement dus à des pixels bruités qui, bien que atténués, continuent d'affecter les votes dans l'accumulateur.

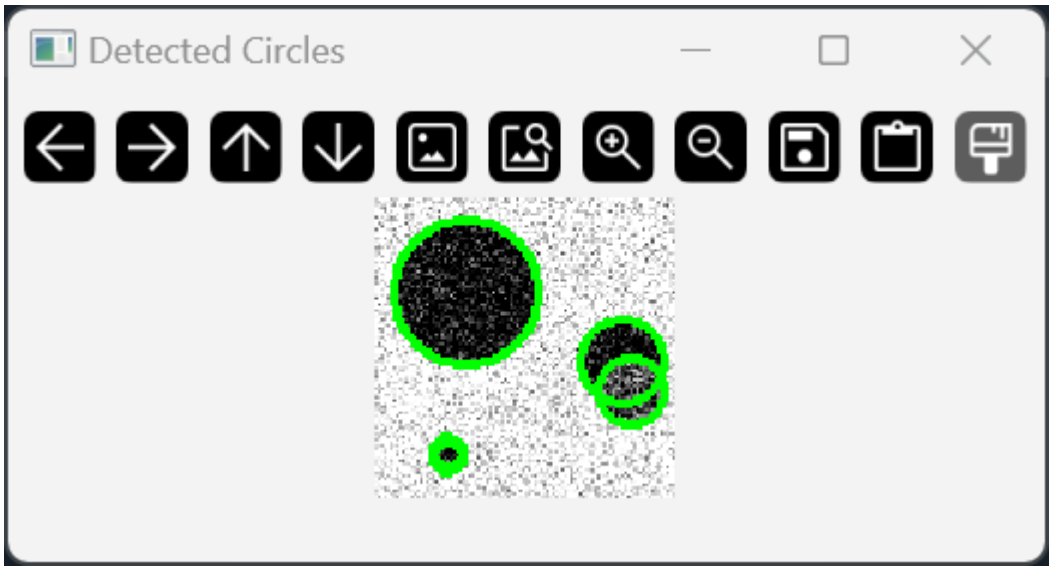


Pistes d’amélioration tentées:

1. **Filtrage plus agressif :**
    - Augmenter la force du filtre Gaussien pour réduire davantage le bruit, mais avec le risque de perdre des détails fins dans l’image.
    - Essayer d'autres méthodes de filtrage, comme les filtres médians ou bilatéraux, pour un meilleur compromis entre réduction du bruit et préservation des contours.
2. **Seuil dynamique :**
    - Ajuster dynamiquement le seuil en fonction des caractéristiques globales de l’image, comme le niveau de bruit estimé.

L’algorithme fonctionne correctement en présence d’une image bruitée, mais le bruit affecte encore les résultats de manière significative. La prochaine étape consiste à affiner les étapes de prétraitement pour améliorer la robustesse et réduire les détections indésirables. Pour cela on a essayé de changer la manière avec laquelle on détecte nos contours, on a utilisé un filtre de détection de contours externe appelé Canny, et le résultat est très bien puisqu’on détecte bien et nos contours et nos cercles comme voulu, vous trouverez le code associé à cette partie dans le fichier ”exo2_fourn_canny.py”.

Voici le résultat:



Exercice 3 : Temps de calcul

Q1)

Pour l'image four.png:

- Temps de calcul : 9.44 secondes

Pour fourn.png:

- Temps de calcul avec gradient: 16.12 secondes
- Temps de calcul avec Canny : 4.77 secondes

Q2) Amélioration par l'ajout de la direction du gradient

Une optimisation a été implémentée pour réduire le temps de calcul tout en augmentant la précision des détections. Cette amélioration repose sur l'utilisation de la direction des gradients.

Principe de l'optimisation

Plutôt que d'effectuer des votes pour tous les cercles possibles à partir des pixels de contour, seuls les cercles dont le centre se trouve dans la direction du gradient local du pixel sont pris en compte. Une tolérance angulaire de 5° (π/36) a été appliquée pour permettre une certaine flexibilité dans les directions estimées.

Résultats et observations

1. Sur l'image sans bruit

- Tous les cercles présents dans l'image originale ont été détectés avec une précision parfaite.
- L'utilisation de la direction des gradients a considérablement réduit les votes dans l'accumulateur, diminuant le temps de calcul et améliorant la détection en éliminant les cercles parasites.

2. Sur l'image avec bruit

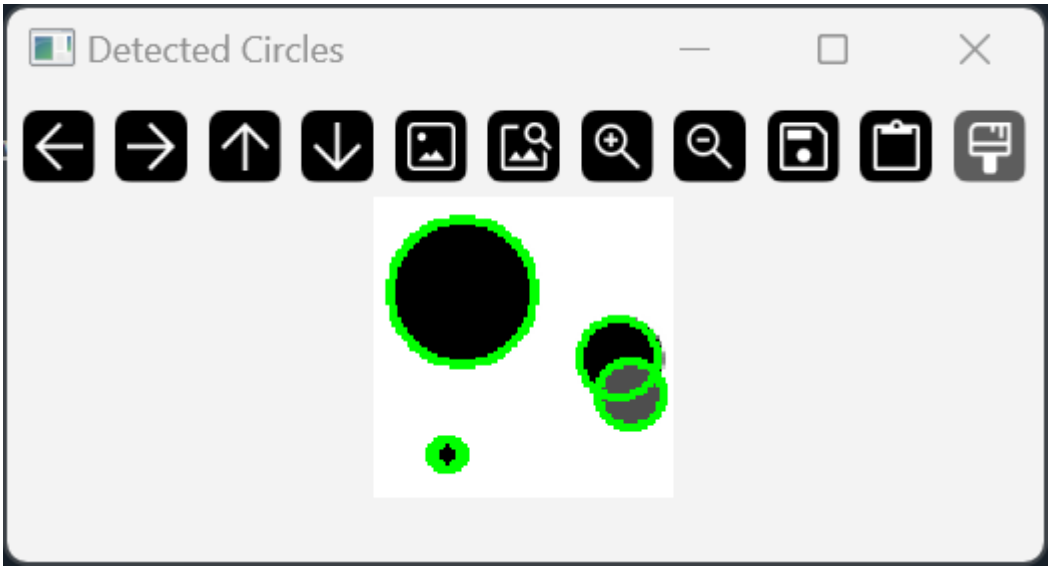
- Malgré la présence de bruit, les cercles d'intérêt ont été détectés sans détections parasites.
- L'approche utilisant la direction des gradients s'est révélée particulièrement efficace pour éliminer les contributions dues aux

pixels bruités, réduisant ainsi les fausses détections observées auparavant.

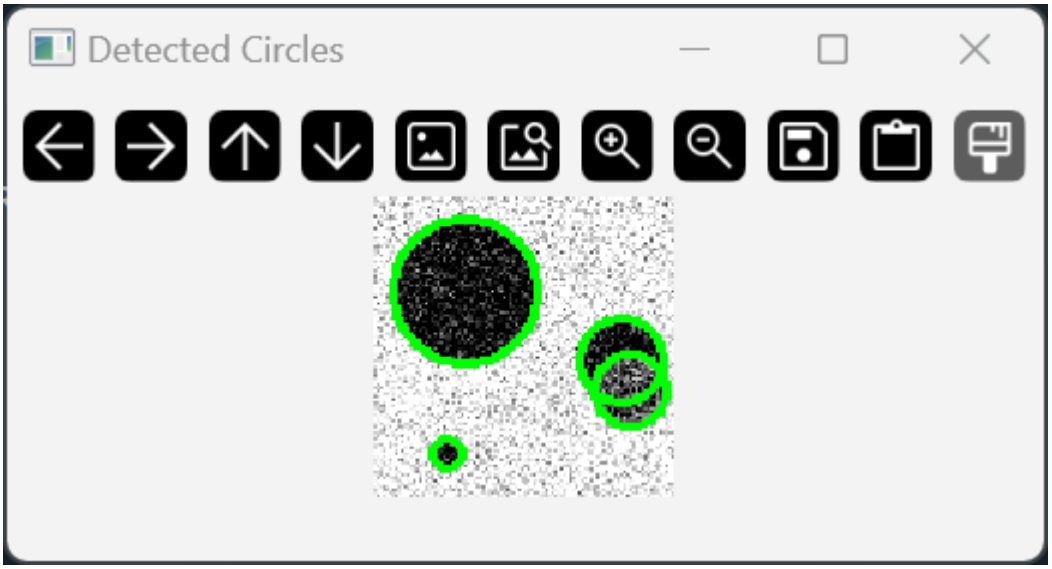
3. Temps de calcul :

- Grâce à cette méthode, le temps de calcul a été réduit, car seuls les pixels alignés avec les gradients ont contribué aux votes dans l'accumulateur. Cela a permis une exécution plus rapide et une meilleure gestion des ressources.

Voici les résultats finaux avec les durées d'exécution:



Temps de calcul : 3.57 secondes



Temps de calcul : 16.27 secondes

En conclusion, le détecteur fonctionne désormais parfaitement, même sur des images bruitées, avec des temps de calcul optimisés et des détections précises. Ces améliorations démontrent l'efficacité des optimisations basées sur des approches directionnelles et des traitements adaptés au bruit.

Réalisé par: Achali Ikram && Racim Ziani