Khelassi Nour El Imene

VMI .

1. État d'avancement

les travaux réalisés sont les suivants :

- **La Dilatation et l'Érosion :** Ce sont les deux fonctions piliers. J'ai utilisé des boucles pour parcourir l'image et appliquer l'élément structurant. Le plus complexe a été de bien gérer les bords de l'image (le padding).
- **L'Ouverture et la Fermeture :** Une fois que la dilatation et l'érosion fonctionnaient, ces deux-là ont été plus simples à mettre en place puisqu'il s'agit juste de combiner les deux premières dans le bon ordre.

Note : exercice 5 n'est pas dans l'avancement

2. Méthodologie et Implémentation

Le code a été structuré autour de deux fonctions piliers :

1. dilation(image, SE) : Utilise np.max sur le voisinage défini par le SE.
2. erosion(image, SE) : Utilise np.min sur le voisinage.

Pour optimiser le calcul, j'ai utilisé l'indexation booléenne de NumPy (neighborhood[SE == 1]), ce qui permet de ne prendre en compte que les pixels actifs de l'élément structurant, quelle que soit sa forme (croix, carré et horizontal).

3. Analyse des résultats illustratifs

Les tests ont été effectués sur des images binaires ainsi que sur l'image classique "Cameraman" à niveaux de gris.

- **Image Binaire :** L'ouverture a démontré son efficacité pour la suppression du bruit de type "poivre et sel" (petits points blancs isolés) tout en respectant la géométrie des objets plus larges.
- **Niveaux de Gris :** Les résultats confirment la dualité des opérateurs : la dilatation étend les zones de haute luminance (tons clairs) tandis que l'érosion favorise les zones de basse luminance (tons sombres).

4. Conclusion

L'implémentation manuelle de ces opérateurs permet de bien distinguer la morphologie mathématique des filtres linéaires.. Le code source complet, accompagné des scripts de test, est disponible sur le dépôt github