TP1 : Erosion et Dilatation

1 Coder une érosion (et une dilatation) tout seul comme un grand

1. Vous devez écrire la fonction **my_first_erode** qui prend en paramètre un élément structurant E (sous forme d'une liste de points) et une image I, et renvoie l'érosion $I \ominus E$. Afin de simplifier votre travail, vous pourrez ignorer les pixels x de I qui sont trop proches du bord et tels que E_x pourrait "sortir" du cadre de I.

   Pour réaliser cette érosion, vous pouvez suivre l'algorithme suivant :

Algorithme 1 : my_first_erode(I, E)

Données : $I : A \to B, E \subset \mathbb{Z}^n$
Résultat : $I \ominus E$
pour tous $p \in I$ **faire**
  si $E_p \subseteq A$ **alors**
    $S(p) = \min_{t \in E_p} I(t)$
  sinon
    $S(p) = I(p)$
  fin
fin

2. Testez votre fonction sur l'image *chien.png* jointe avec ce TP. Le résultat de l'érosion de cette image par un disque de rayon 3 est donné dans l'image *test_erode_chien.png* : trouvez-vous le même résultat ?

3. A l'aide de votre fonction d'érosion codée précédemment, codez une fonction de dilatation d'une image par un élément structurant. Le résultat de la dilatation de *chien.png* par un diamant de taille 5 est donné dans l'image *test_dilate_chien.png* : trouvez-vous le même résultat ?

2 Utiliser OpenCV pour réaliser des érosions et des dilatations

1. En lisant la documentation d'OpenCV, écrivez une fonction **myerode** permettant de faire une érosion d'une image I par un élément structurant E. En érodant l'image *chien.png* par un disque de rayon 3, trouvez-vous le même résultat que l'image *test_erode_chien.png* ?

2. Ecrivez une fonction **mydilate** à l'aide d'OpenCV, et testez-la.

3. Qui est le plus rapide pour faire une érosion : la fonction d'OpenCV, ou bien votre fonction codée en première partie ?

4. Codez une fonction de gradient morphologique **mygrad**, qui calcule le gradient morphologique d'une image I par rapport à un élément structurant E, et testez-la sur l'image *voiture.jpg*.

5. Rassemblez vos fonctions de base (**mydilate**, **myerode**, et **mygrad**) dans un module **morpho.py**.

3 Application : trouver l'orientation d'une feuille de papier

On souhaite maintenant calculer automatiquement l'orientation, sur une photographie, d'une feuille de papier. Vous possédez trois exemples de feuilles avec différentes orientations : *papier_60.png* (orientation de 60°), *papier_35.png* (orientation de 35°) et *papier_15.png* (orientation de 15°).

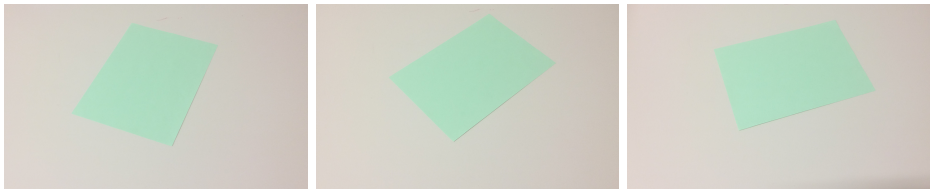

FIGURE 1 – De gauche à droite, la feuille de papier avec une orientation respective de 60, 35 et 15 degrés

1. Le premier problème à résoudre est la couleur. Bien qu'ils puissent être appliqués à des images couleur sans problème, les outils que vous avez développés sont pensés pour des images en niveaux de gris : il va donc falloir transformer les images couleurs en noir et blanc.
   La feuille de papier est un vert turquoise clair (mélange de vert et de bleu), et le fond est blanc crème... Quel canal couleur conserver afin de faire ressortir au mieux la feuille sur le fond clair ? Testez votre hypothèse.

2. Considérons l'image *papier_60.png* que vous aurez transformée en niveaux de gris. Affichez les différents gradients obtenus sur l'image à l'aide de lignes de 40 pixels de long, allant de -90 à 90°, par pas de 5°.

3. Déduisez-en un algorithme permettant de calculer automatiquement l'orientation (à quelques degrés près) de la feuille de papier, et testez-le sur les trois images.