**TP 4**

**Exercice 1 :**

À partir d’une image noir et blanc, ou couleur, on veut créer un floutage sur tout ou une partie d’une photo. Pour cela, une idée simple est de remplacer chaque composante couleur de chaque pixel par la moyenne de cette composante sur tous les pixels voisins ou proche à une certaine distance.

Pour simplifier, on calcule la moyenne de chaque composante sur des carrés de côté 2*a* +1 sur les pixels qui ont un indice de ligne et un indice de colonne distant d’au plus *a* de l’indice du pixel courant.

Programmer une fonction floutage sur une partie centrale de la photo et afficher le résultat pour différentes valeurs de *a*.

Programmer une fonction floutage sur toute la photo en tenant compte de la particularité des bords de photo, et afficher le résultat.

**Exercice 2 :**

À partir d’un filtre différentiel de type :

Une image contenant horloge

Description générée automatiquement

On détecte les contours sur une photo en noir et blanc.

Pour cela, on applique ce filtre au calcul et tous les pixels de la photo en évitant les problèmes de bord. On obtiendra ainsi une photo éventuellement plus petite.

Chaque nouvelle valeur du pixel est alors soit un nombre réel entre -2 et 2 soit un nombre entier entre -510 et 510 selon le format de la photo.

Il faut bien évidemment remettre ce format au format d’une image en tenant compte de ce qui nous intéresse ici : la détection de contour.

On dira alors qu’il y a un contour si la valeur du calcul donne un nombre de valeur absolu supérieure à un seuil que l’on fixera en paramètre dans la fonction **contour** :

* Si la valeur absolue du calcul est supérieure à ce seuil, on mettre le pixel à la couleur noire (*i.e.* valeur nulle) ;
* Si la valeur absolue du calcul est inférieure à ce seuil, on mettre le pixel à la couleur blanche (*i.e.* valeur 1 ou 255 selon le format de l’image) ;

Adapter le seuil pour obtenir au mieux les contours de la photo.

Faire des tests avec plusieurs filtres de différentes tailles.

**Exercice 3 :**

Dans la photo "sirenedansgaronne" il a été caché une image ou photo selon le procédé suivant : Cette photo est un tableau numpy de dimension *n*×*p* ×3 telle que le coefficient *i* , *j* ,*k* est un nombre entier compris entre 0 et 255 représentant selon l’entier *k* l’intensité une des trois couleurs rouge verte ou bleue du pixel de la *i* -ième ligne et *j* -ième colonne de la photo.

Ce nombre entier est donc un octet dont

* Les 4 premiers bits sont les quatre premiers bits du nombre correspondant de la photo originale
* Les 4 derniers bits sont les quatre premiers bits du nombre correspondant de la photo cachée

Retrouver la photo cachée.

À partir de deux photos en noir et blanc, cacher la plus petite des deux photos dans la seconde selon ce même principe.

**Exercice 4 :**

Créez un script qui affiche une nouvelle image correspondant au symétrique de l'image.

**Exercice 5:**

Créez un script qui affiche une nouvelle image correspondant à l'image quatre fois plus petite.

**Exercice 6:**

Créez un script qui affiche une image correspondant à cette image tournée de 30° et superposée à la même image tournée de -30°.

**Exercice 7:**

Récupérez les trois différents canaux de l'image téléchargée, affichez les trois images correspondantes l'une à côté de l'autre et enregistrez l'image obtenue.