

Projet sur l'image

I - Filtres et autres transformations

Télécharger sur Internet une image libre de droit (pour modification). Dans la partie 2 de l'activité, on devra utiliser une image carrée, donc soit vous en trouvez une, soit vous utilisez GIMP en sélectionnant une zone carrée de l'image choisie, puis *Image/Rogner suivant sélection* - un petit ajustement peut être nécessaire à l'aide de *Image/Échelle et taille de l'image*.









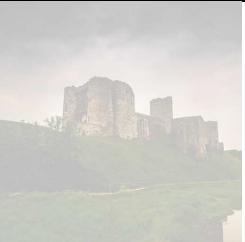

Nous allons repartir du travail effectué dans le chapitre 3 (création d'un filtre rouge) que j'ai adapté sous la forme d'une fonction.

```
1 from PIL import Image
2
3 def filtre(image1, couleur):
4     """cette fonction modifie l'image initiale en conservant des informations sur les
5     pixels de départ"""
6     larg,haut=image1.size
7     newImage=Image.new("RGB",(larg,haut))
8     for x in range(larg):
9         for y in range(haut):
10             rouge,vert,bleu=image1.getpixel((x,y))
11             if couleur == 'rouge' :
12                 newImage.putpixel((x,y),(rouge,0,0))
13     return newImage
14
15 im1 = Image.open("Chateau.jpg")
16 image=filtre(im1, 'rouge')
17 image.save("filtreRouge.jpg")
```

Travail :

1. Modifier la fonction filtre (ou plutôt la compléter) pour donner la possibilité d'en créer d'autres :
 - Filtre vert : seules les composantes vertes de l'image sont conservées.
 - Filtre bleu : seules les composantes bleues de l'image sont conservées.
 - Filtre gris : chaque composante sera la moyenne des trois composantes rouge, vert et bleu.
 - Noir et blanc : on se fixe un seuil, si le niveau de gris est inférieur à ce seuil, alors on met du noir (0), sinon du blanc (255). Chaque composante aura alors cette valeur noir ou blanc.
 - On conserve le même seuil que précédemment. Si le niveau de gris est supérieur à ce seuil, alors on aura du rouge, sinon ce sera du cyan (vert et bleu).
 - Filtre inversion négative : chaque composante sera son complément pour arriver à 255.
2. On peut aussi proposer différentes transformations de l'image de départ. Pour cela, il est peut-être plus judicieux de créer une autre fonction, mais au départ, elle ressemblera à la fonction filtre, seule la partie qui suit `getpixel` sera différente.
 - Symétrie verticale : on conserve la couleur de chaque pixel, mais au lieu de la positionner en (x,y), on la met en (larg-x-1,y). La soustraction par 1 est due au fait que les abscisses des pixels vont de 0 à larg-1.
 - Symétrie horizontale : même principe que le précédent, juste une petite chose à changer...
 - Effet de transparence : seul le format initial png permettra d'utiliser cet effet. Par ailleurs, à la création de l'image, il faut déjà remplacer RGB par RGBA.
Le deuxième élément de la méthode `putpixel` prend alors un paramètre de plus : la transparence. Il pourra alors être de la forme (rouge,vert,bleu,100). L'image obtenue devra être enregistrée au format png.

- Flou : on remplace chaque composante de couleur par la moyenne de celles des pixels environnants. On peut par exemple prendre les 8 pixels environnants (ou plutôt 9 en comptant le pixel sur lequel on travaille).
Pour les pixels sur les bords, il faut adapter car il n'y en a pas 8 autour, le plus simple est sans doute de conserver le pixel s'il est sur un bord et de faire la moyenne proposée pour tous les autres.

				
Filtre vert	Filtre bleu	Filtre gris	Noir et blanc	Filtre rouge-cyan
				
Filtre inversion négative	Symétrie verticale	symétrie horizontale	Effet de transparence	Flou

II - Principe du photomaton

Dans cette partie, il est nécessaire de disposer d'une image carrée dont la longueur est une puissance de 2. Dans la partie 1, vous avez travaillé avec une image carrée, l'ouvrir avec GIMP, puis dans *Image/Échelle et taille de l'image*, modifier la largeur et la hauteur pour avoir une dimension souhaitée.

La méthode du photomatin consiste à construire 4 petites images de l'image de départ :

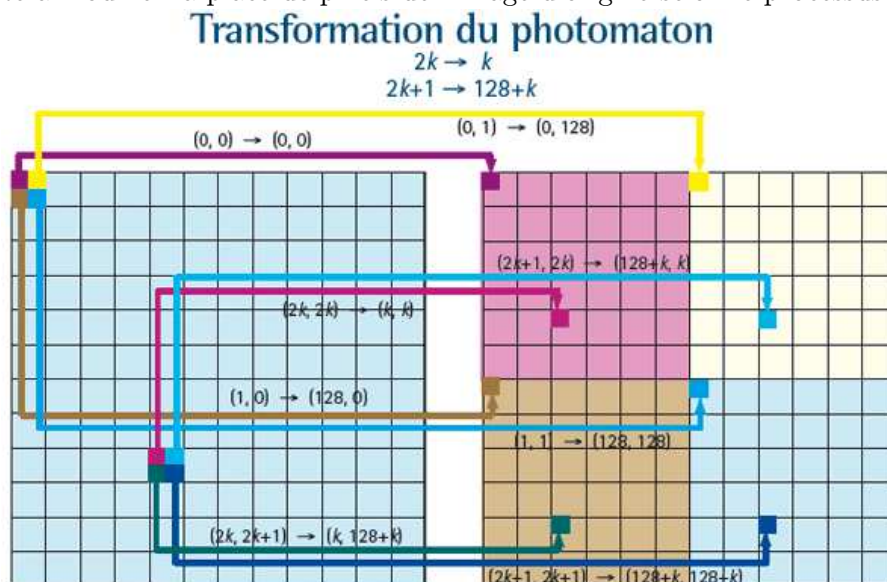


Image de départ



Image obtenue

Le principe consiste à modifier la place de pixels de l'image d'origine selon le processus ci-dessous :



Dans cette image, il y a une erreur d'écriture : en abscisse et en ordonnée, on utilise la variable k . Or il faut utiliser k en abscisse et k' en ordonnée. De plus dans cet exemple, on part d'une image de taille 256×256 . Adaptez cette valeur à la taille de votre image ou, ce qui est mieux, utiliser l'attribut `size` de python pour connaître sa taille (qui restera de toute façon une puissance de 2).

Comme dans la partie précédente, il faudra parcourir tous les pixels de l'image de départ pour les placer au bon endroit dans la nouvelle image. Cependant, ce parcours se fera par bloc de 4 (comme sur l'image), donc dans la double boucle imbriquée, il ne faudra pas utiliser `larg` et `haut`, mais ... (à vous de compléter).

Pour une image de taille $2^k \times 2^k$, faire successivement k fois l'opération (en reprenant à chaque fois l'image obtenue). Que constate-t-on ?