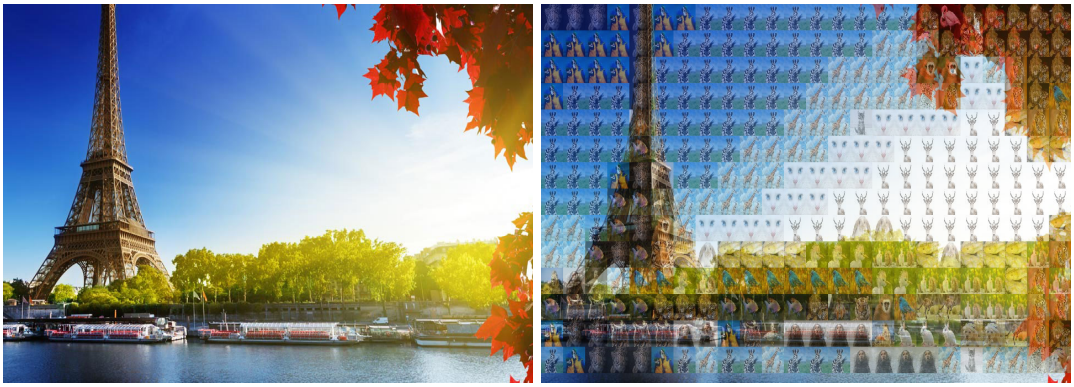
TD 11 : décomposition mosaïque d'une image

Arnaud de Myttenaere

28 novembre 2014

1 Introduction



(a) Image d'origine (b) Résultat

Figure 1 – Avant/après la décomposition d'une image sous forme de mosaïque

1.1 Lecture et écriture d'une image

```
import Image
im = Image.open('image.jpg') # Lecture
im.save('output.png') # Sauvegarde
```

1.2 Manipulation d'une image

La taille d'une image est donnée par la fonction *size*. Ainsi, *im.size[0]* et *im.size[1]* correspondent à la largeur et à la hauteur de l'image *im*.

Exercice 1 A l'aide de la fonction *resize* : lire, redimensionner une image à la taille 100×100, et sauvegarder le résultat.

Le format RGB Une image est représentée par un tableau de pixel, ou un pixel correspond à un triplet (R,G,B) indiquant la teinte du pixel selon 3 couleurs : Red, Green, Blue.

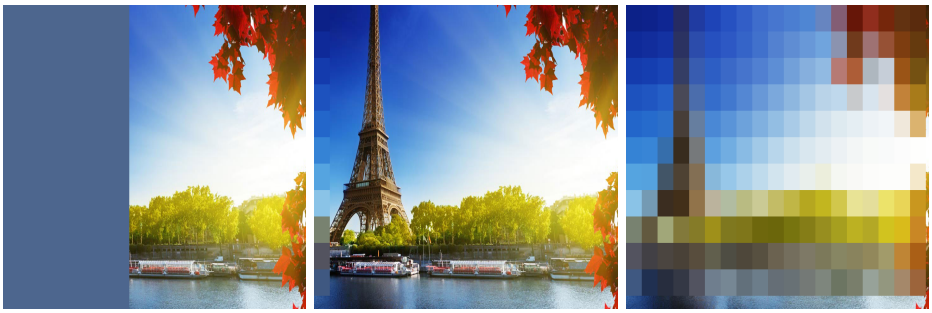
Une image peut donc est représentée par 3 matrices, pour les teintes rouge, verte et bleu respectivement.

```
# Pour accéder aux matrices R, G, B de l'image im
import numpy as np
r, g, b = np.array(im).T
print(r)

# Construire une image à partir des matrices R, G, B
im = Image.fromarray(np.dstack([item.T for item in (r,g,b)]))
```

Exercice 2

1. Modifier une image pour faire apparaître une bande de largeur L à gauche, dont la couleur correspond à la couleur moyenne sur cette même bande sur l'image d'origine.
2. Adapter le code de la question précédente pour construire colonne composée de plusieurs carrés moyennés
3. Adapter le code de la question précédente pour construire une mosaïque d'image dont chaque carré est de la couleur moyenne du même carré sur l'image d'origine.



(a) bande verticale (b) une ligne en mosaïque (c) mosaïque complète

Figure 2 – Avant/après la décomposition d'une image sous forme de mosaïque

2 Application

2.1 Distance entre deux images

Pour calculer la distance entre deux images, on calculera d'abord la couleur moyenne de ces images selon les teintes R, G et B, ce qui fournira un triplet pour chaque image. Ensuite, la distance entre deux triplets sera la distance euclidienne donnée par :

$$dist(RGB0, RGB1) = (RGB0[0] - RGB1[0])^2 + (RGB0[1] - RGB1[1])^2 + (RGB0[2] - RGB1[2])^2$$

```
def getMeanColor(im):
    r, g, b = np.array(im).T
    return (r.mean(), g.mean(), b.mean())

def getDist(RGB0, RGB1):
    return((RGB0[0] - RGB1[0])**2 + (RGB0[1] - RGB1[1])**2 + (RGB0[2] - RGB1[2])**2)
```

Exercice 3

1. Fixer une variable L, qui sera la longueur de chaque carré de la masaïque (L = 50 pour des carrés de côté 50 pixel)
2. Redimensionner l'ensemble d'images disponibles à la taille $L \times L$ et les enregistrer dans une nouveau dossier.
3. Charger toutes les petites images dans une liste *list_im*.
4. Construire une liste où le i-ieme élément correspond au triplet RGB moyen de l'image *list_im[i]*
5. Construire la mosaïque en s'inspirant du code de la question 2.3. Pour chaque bloc :
   — Calculer son triplet RGB moyen,
   — Trouver l'image dans *list_im* la plus proche de ce bloc
   — Remplacer ce bloc par l'image

Exercice 4 Pour un meilleur rendu, reprendre le code de l'exo 3 et superposer l'image obtenue avec celle d'origine selon la pondération $a * (mosaique) + (1 - a) * image$. Tester plusieurs valeurs de a et comparer les résultats ($a = 0.5$ à la figure 1).