TP5 : RECOLORATION D'IMAGES ET EVALUATION

1 Recoloration des images avec les kmeans

Pour recolorer les images, nous allons utiliser l'algo des kmeans fait précédemment afin d'assigner à chaque pixel une classe.

J'ai dans un premier temps créer une fonction pour transformer l'image à recolorer en fichier csv :

```
def im_to_csv(data):

    with open('miro.csv', 'w', newline='') as csvfile:
        writer = csv.writer(csvfile, delimiter=',')
        #parcours de tout les pixels
        for i in range(data.shape[0]):
            for j in range(data.shape[1]):
                px = data[i, j]
                writer.writerow([float(px[0]),float(px[1]),float(px[2]),0])
```

Une fois le csv obtenu on peut appliquer l'algo des kmeans et récupérer la liste des classes pour chaque pixels :

```
#chargement des données csv
rv, labels = load_dataset('joconde.csv')

# initialisation de l'objet KMeans
kmeans = KMeans(n_clusters=6,
                max_iter=100,
                early_stopping=True,
                tol=1e-6,
                display=True)

# calcule les clusters
classes = kmeans.fit(rv)
```

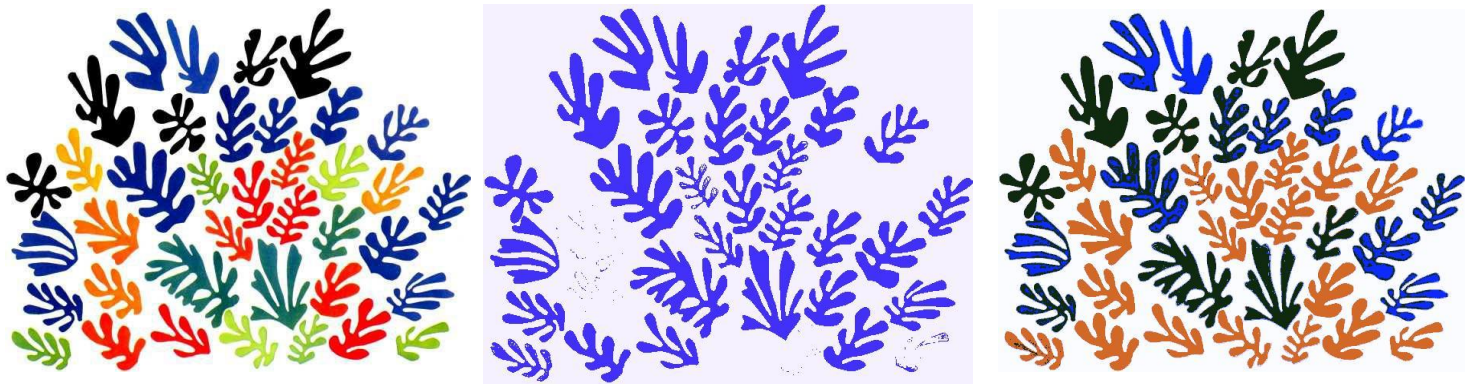
Ensuite une fois la liste des classes on peut lancer la fonction de recoloration :

```
def recolorisation(name_image,data, y,clusters):

    im = Image.open('./images/'+name_image+'.jpg')
    im_reco = im

    #choix des couleurs pour la recoloration
    colors = []
    for i in clusters:
        c = (int(i[0]), int(i[1]), int(i[2]))
        colors.append(c)

    long = data.shape[0]
    larg = data.shape[1]

    compt = 0
    #On parcours tout les pixels
    for i in range(0,long):
        for j in range(0,larg):
            # recolorisation
            im_reco.putpixel((j,i),colors[int(y[compt])])
            compt += 1


    im_reco.save('./images/' + name_image + '_recolorized_2clusters.jpg', 'JPEG')

    return 0
```

Dans un premier temps on récupère les couleurs des clusters, qui seront par la suite les couleurs des classes.

On parcourt chaque pixel de l'image et en fonction de la classe on donne une couleur définie dans le tableau colors.

2 Résultats



De gauche à droite : image originale, image recolorée 2 clusters, image recolorée 6 clusters

3 Evaluation de la qualité de la recoloration

Pour évaluer la qualité de la recoloration j'effectue une comparaison pixel par pixel entre l'image originale et l'image recolorée.

```
def calculDiff(image1, image2):
    im1 = img.imread('./images/' + image1 + '.jpg')
    im2 = img.imread('./images/' + image2 + '.jpg')

    diff = 0

    # On parcours tout les pixels
    for i in range(0, im1.shape[0]):
        for j in range(0, im1.shape[1]):
            px1 = im1[i, j]
            px2 = im2[i, j]

            diff += (abs(int(px1[0])-int(px2[0])) + (abs(int(px1[1])-int(px2[1])) + (abs(int(px1[2])-int(px2[2])))

    diff = (diff/(im1.shape[0]*im1.shape[1]))/765

    return diff
```

Je fais la moyenne des différences entre les 3 couleurs de chaque pixel. Je divise par 765 pour me ramener à un pourcentage, car 765 (255*3) est la différence maximale que l'on peut obtenir entre 2 pixels (un blanc 255,255,255 et un noir 0,0,0).

Donc par exemple si on compare le tableau miro original et le recoloré à 2 cluster :

```
print("diff =", calculDiff("miro", "miro_recolorized_2clusters"))
```

diff = 0.18683218616846933

Avec 6 clusters la différence est logiquement plus faible :

```
print("diff =", calculDiff("miro", "miro_recolorized_6clusters"))
```

diff = 0.07411095044376322

On a le même schéma avec le tableau de picasso :

```
print("diff =", calculDiff("picasso", "picasso_recolorized_2clusters"))
```

diff = 0.16489172048158243

```
print("diff =", calculDiff("picasso", "picasso_recolorized_6clusters"))
```

diff = 0.07797362662055354

4 Bonus

On fait le test de recoloré un tableau dans un style bien différents des 3 premiers, la Joconde :

De gauche à droite : image originale, image recolorée 2 clusters, image recolorée 6 clusters

```
print("diff =", calculDiff("joconde", "joconde_recolorized_2clusters"))
```
diff = 0.08434813957385777

```
print("diff =", calculDiff("joconde", "joconde_recolorized_6clusters"))
```
diff = 0.04347334110216007

On peut constater que les résultats étonnamment tout aussi satisfaisant. La Joconde possède beaucoup plus de nuances de couleurs que les autres tableaux. Mais toutes ces couleurs ne sont pas si éloignées donc la recoloration est plutôt réussie.