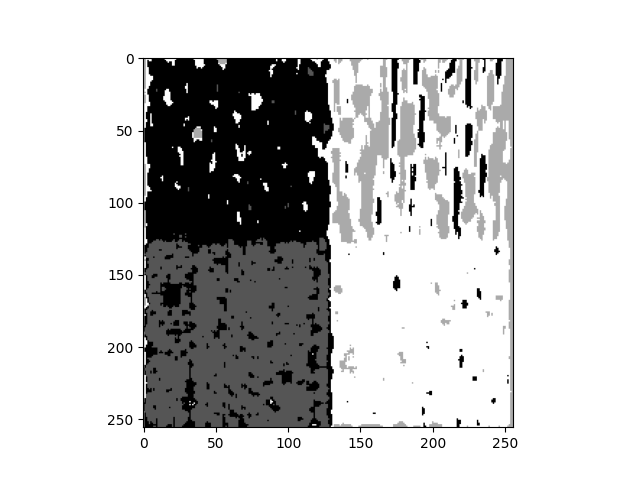
## 1.Design

### 1.1导入sklearn库里面的kmean算法和em算法：

from sklearn.cluster import KMeans

from sklearn.mixture import GaussianMixture

### 1.2用K-means算法进行聚类查看效果，k=4，误差控制0.0001，得到kmean.png：



### 1.3 Initialization of the em algorithm using the cluster class center of the kmean algorithm to train a Gaussian mixture model

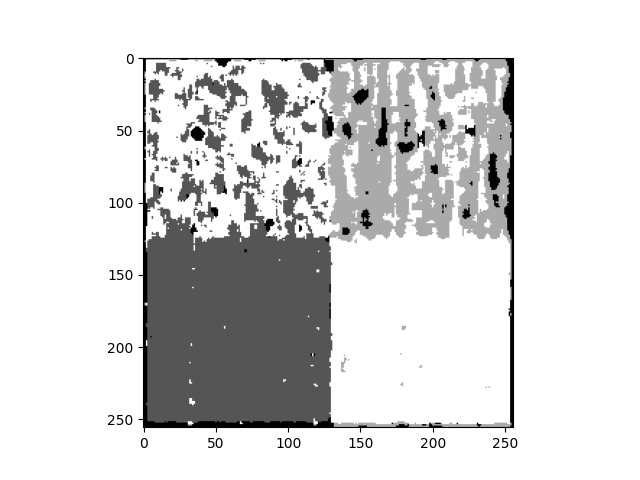
centers = kmeans.cluster\_centers\_

means\_init = centers

# 构建GMM模型  
gmm = GaussianMixture(n\_components=4, means\_init=means\_init)

# 训练模型  
gmm.fit(data)

### 1.4final output em.png:



## 2.Numerical comparison

### **2.1 Calculate the eigenvalues of the original mapB.bmp by defining gaborcls function**



fea：[-0.17984725 -0.17984725 -0.17984725 ... -0.22221344 -0.22221344 -0.21162189]

### 2.2cosine

filename = 'kmean.png'  
newfea = gaborcls(filename)  
print(newfea.shape)  
tmp=cosine(newfea,fea)  
print(newfea)  
print(tmp)

[ 1.27901367 -0.46500328 -0.91844252 ... -0.94187435 0.5104541 2.13034495]

0.9456748530729379

filename = 'em.png'  
newfea = gaborcls(filename)  
tmp=cosine(newfea,fea)  
print(newfea)  
print(tmp)

[ 1.00008654 2.41939006 1.61951262 ... -0.2713849 -0.29896437 1.06233238]

0.9209753832021864

### 2.3 Reflection and Summary

By calculating the difference between cosine similarity comparison and mapA.bmp effect, by comparison, in general, combining kmean and em algorithm does have better effect, and the result obtained is obviously better than the kmean algorithm alone.