# Práctica 03
# Cuantización Vectorial

Visión Computacional Aplicada a la Robótica

UNAM, 2021-2

**Resumen**

El alumno aprenderá a aplicar técnicas de cuantización vectorial para obtener un patrón a partir de imágenes de entrenamiento. Posteriormente, el alumno obtendrá un cuantizador para las imágenes de prueba y determinará el patrón al que corresponde.

## Duración

2 semanas

## 1. Desarrollo

### 1.1. Algoritmo LBG

Programe en Python o C++ el siguiente algoritmo:

1. Inicialización: Obtenga un *codebook* inicial $D_1$, con solo un centroide $C_1$, el cual se obtiene promediando todos los pixeles $p_j$ de la imagen. Sea $m = 1$ la iteración actual y $L_m = 1$, el número actual de centroides $C_i$ en el *codebook* $D_m$.

2. Perturbación: Para cada centroide $C_i$, $i = 1, ..., L_m$ en el *codebook* $D_m$, obtenga dos nuevos centroides sumando una perturbación $\pm\psi$ de una magnitud pequeña $\epsilon$. Esto es, el nuevo *codebook* $D_{m+1}$ contendrá $L_{m+1} = 2L_m$ nuevos centroides.

3. Dado el codebook $D_m = C_1, ..., C_{L_m}$, assigne cada pixel $p_j$ al cluster $R_k$, cuyo centroide correspondiente $C_k$ es el más cercano a $p_j$.

4. Para cada cluster $R_k$, recalcule el centroide $C_k$ promediando todos los pixeles $p_j \in R_k$.

5. Calcule la diferencia entre la distancia media $\overline{d_t} = \frac{1}{N_v} \sum d\left(p_j, C_k\right)$, en la iteración $t$, y la distancia media en la iteración anterior $\overline{d_{t-1}}$. Si esta diferencia es mayor que una tolerancia, i.e. $\left|\overline{d_t} - \overline{d_{t-1}}\right| > tol$, entonces regrese al paso 3, de lo contrario, vaya al siguiente paso.

6. Si $L_m < L_d$, vaya al paso 2.

Donde $L_d$ es el tamaño deseado del *codebook*. Elija este tamaño considerando un equilibrio entre costo computacional y la precisión deseada. Pruebe el algoritmo con las imágenes para entrenamiento del dataset.

### 1.2. Generación de patrones

- Aplique el algoritmo LGB para generar un *codebook* para cada uno de los objetos de entrenamiento. Considere que el conjunto de vectores (pixeles) $p_j$ es la unión de todos los pixeles de todas las imágenes de entrenamiento correspondientes a un objeto.
- Guarde el conjunto de centroides de cada objeto en un archivo en disco. Se recomienda usar los formatos `yaml` o `xml`.

### 1.3. Reconocimiento de objetos

Realice un programa que ejecute lo siguiente:

1. Elegir aleatoriamente alguna imagen del conjunto de imágenes de prueba.
2. Obtener un `codebook` para la imagen seleccionada.
3. Comparar el `codebook` observado con cada uno de los `codebooks` entrenados.
4. Anunciar el objeto reconocido correspondiente al *codebook* entrenado más parecido al *codebook* observado.

## 2. Código fuente de referencia

```
import cv2
import os
import sys
import numpy as np

def disturb_centroids(centroids, epsilon):
    new_centroids = []
    for c in centroids:
        v = np.random.rand(3)
        v = epsilon * v / np.linalg.norm(v)
        new_centroids.append(c + v)
        new_centroids.append(c - v)
    return new_centroids

def get_centroids(m, img, epsilon, tol):
    img = np.reshape(img, (-1,3))
    centroids = [np.mean(img, axis=0)]
    print(centroids)
    while len(centroids) < m:
        centroids = disturb_centroids(centroids, epsilon)
        print("Calculating " + str(len(centroids)) + " centroids")
        delta = tol + 1
        while delta > tol:
            clusters = [[] for c in centroids]
            for p in img:
                idx = np.argmin([np.linalg.norm(p-c) for c in centroids])
                clusters[idx].append(p)
            new_centroids = [np.mean(c, axis=0) for c in clusters]
            delta = np.sum([np.linalg.norm(new_centroids[i] - centroids[i
    ]) for i in range(len(centroids))])
            centroids = new_centroids
```

```
31              print("Current delta: " + str(delta))
32      print centroids
33
34  def main():
35      img = cv2.imread(sys.argv[1])
36      get_centroids(8, img, 1.0, 5.0)
37
38
39  if __name__ == '__main__':
40      main()
```

## 3. Entregables:

Los siguientes entregables se refieren a los incisos 1.2 y 1.3 del desarrollo.

- Código fuente para la obtención de los *codebooks* de cada objeto.
- Archivos `yaml`, `xml` o extensión similar correspondientes a los patrones entrenados.
- Código fuente para el reconocimiento de objetos.