Challenge 7

Student: Xabier Oyanguren Asua
NIU: 1456628 Date: 01/05/2022

1 Objectives

Using Python and some libraries like numpy and scipy (listed in the fifth section), we implement a K means clustering algorithm and test it employing colour palette reduction for a sample image in different representation spaces for the points.

2 Data Acquisition

The test image was acquired using the camera of a Xiaomi Redmi 5 Plus, and downscaled using local averaging to a third of the original size (to a 240x427 shape). The test image can be seen in Figure 1.

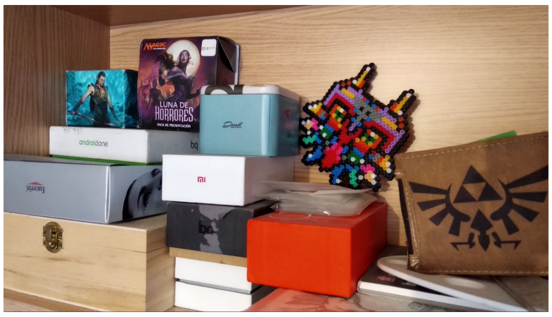

Figure 1: Captured image for the exercise.

3 Procedure

Following the recipie for the k means clustering algorithm seen in the theory lecture, the Python code attached in the next section was implemented in order to find k clusters of a given array with the different points as rows and representation features as columns.

Employing this, we could then take the test image, get an array with the RGB, HSV, RGBxy or HSVxy features of each pixel with the specified shape (where x,y represent the pixel indices in the image), and pass them to the k means clustering algorithm to obtain the cluster centers in those representation spaces and each pixel's cluster label.

Finally, we can plot the found clusters by assigning to each cluster the colour of the converging cluster center.

4 Results

We see in Figure 2 the clustering result with four different k values (see the subplot titles) for the RGB space, in Figure 3 for the RGBxy space, in Figure 4 the HSV space and in Figure 5 the HSVxy space.

References

[1] Class notes.

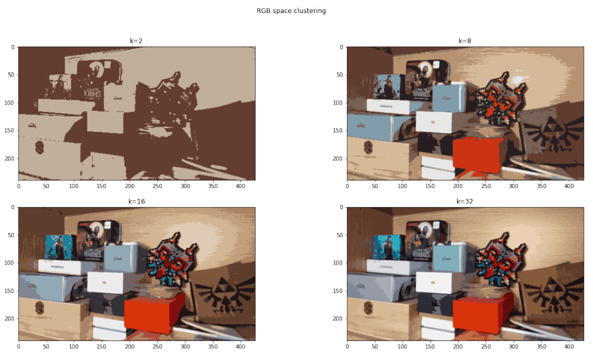

Figure 2: Clustering in the RGB space for the test image for different k values.

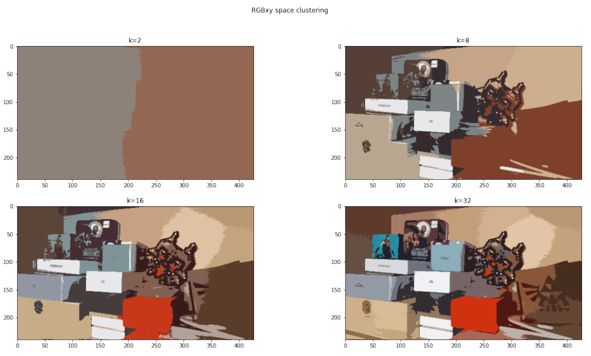

Figure 3: Clustering in the RGBxy space for the test image for different k values.

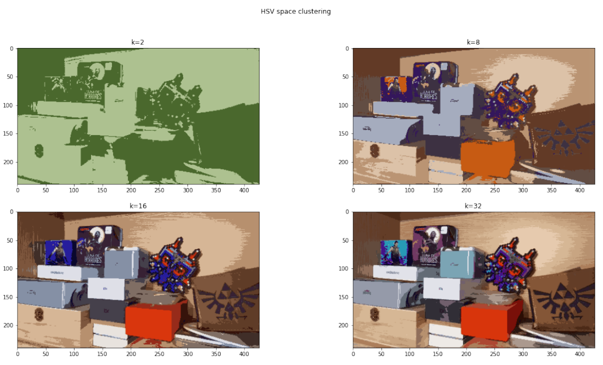

Figure 4: Clustering in the HSV space for the test image for different k values.

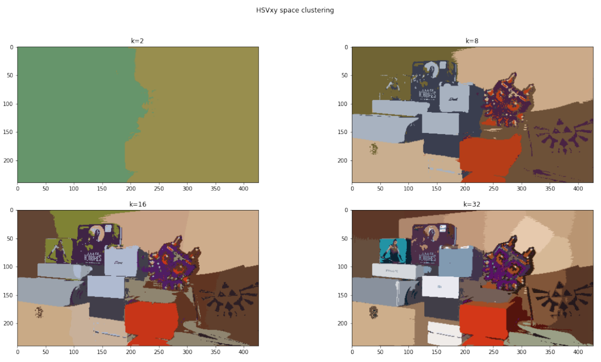

Figure 5: Clustering in the HSVxy space for the test image for different k values.

5 Packages

Employed packages and external functions were imported as:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import scipy.spatial
from skimage.transform import downscale_local_mean
```

6 Code

The k means algorithm was implemented as a function:

```
def k_means(features_per_point, k, max_its=50000, rtol=0): # expected input of dims [N_points, N_features_per_point]
    # we can select the initial cluster centers as k random points among the inputed ones
    clusters = features_per_point[ np.random.choice(np.arange(features_per_point.shape[0]), k) ] #[k, N_features_per_point]
    for it in range(max_its):
        # distance to each cluster
        distances_to_each_cluster = scipy.spatial.distance_matrix( features_per_point, clusters, p=2) #[N_points, k]
        # relabel identities of each point
        cluster_id_belongings = distances_to_each_cluster.argmin(axis=1)
        # recompute cluster-centers as the baricenters of the clusters
        clusters_new = np.array([np.mean(features_per_point[cluster_id_belongings==k_id], axis=0) for k_id in range(k)])
        if np.allclose(clusters, clusters_new, rtol=rtol):
            break
        clusters = clusters_new
    return cluster_id_belongings, clusters_new
```

As an example, the RGB images were generated using:

```
im_pixel_RGB = im_RGB.reshape(-1, 3)
ks=[2, 8, 16, 32]

images = []
for k in ks:
    cluster_ids, cluster_centers = k_means(im_pixel_RGB, k)
    images.append( cluster_centers[cluster_ids].reshape(im_RGB.shape).astype(np.uint8) )
```

While the HSVxy images were generated with:

```
im_pixel_HSV = cv2.cvtColor(im_RGB, cv2.COLOR_RGB2HSV).reshape(-1, 3)
ks=[2, 8, 16, 32]

images = []
for k in ks:
    cluster_ids, cluster_centers = k_means(im_pixel_HSV, k)
    images.append( cv2.cvtColor(
            cluster_centers[cluster_ids].reshape(im_RGB.shape).astype(np.uint8), cv2.COLOR_HSV2RGB ))
```