Assignment 5 - Image Segmentation

Federica Bruni (fbruni@student.ethz.ch) 16 November 2023

1 Code Overview:

The objective of this report is to analyze and discuss the implementation and results of the mean-shift algorithm for image segmentation.

1. Distance Calculation
The Euclidean distance between a point and a set of points is calculated using the np.linalg.norm function.

```
def distance(x, X):
    return np.linalg.norm(X - x, axis=1)
```

2. Gaussian Kernel
This function defines a Gaussian kernel given a distance and bandwidth. It computes the weights for the mean-shift update.

```
def gaussian(dist, bandwidth):
    return  np.exp(-0.5 * (dist / bandwidth)**2) / (bandwidth * np.
    sqrt(2 * np.pi))
```

3. Update Point
This function updates a point based on the calculated weights.

```
def update_point(weight, X):
    return np.sum(weight[:, np.newaxis] * X, axis=0) / np.sum(weight)
```

4. Mean-Shift Step
This function performs a single iteration of the mean-shift algorithm. For each point in the input, it calculates distances, computes weights using the Gaussian kernel, and updates the point.

```
def meanshift_step(X, bandwidth=2.5):
    y = np.copy(X)
    for i in range(X.shape[0]):
        dist = distance(X[i], X)
        weight = gaussian(dist, bandwidth)
        y[i] = update_point(weight, X)
    return y
```

5. Mean-Shift Algorithm
This function iteratively applies the mean-shift step, repeating the process 20 times.

```
def meanshift(X):
    for i in range(20):
        X = meanshift_step(X)
    return X
```

2 Results

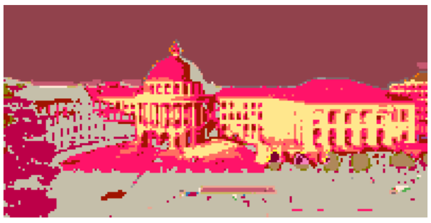
The last request was to experiment with different bandwidth values, specifically [1, 3, 5, 7].

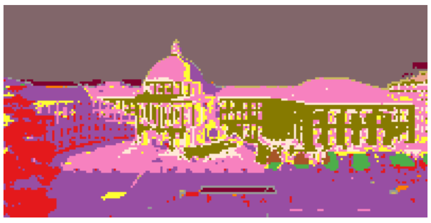
- Default bandwidth = 2.5
  This bandwidth yields a satisfactory outcome. The image is clear and highly detailed. The algorithm successfully identifies windows and trees. Additionally, the algorithm effectively distinguishes the road from the square resulting in a good segmentation.

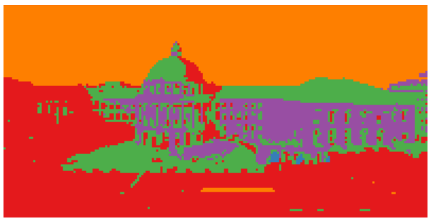
- Bandwidth = 3
  When the bandwidth is 3, the result is very similar to the default bandwidth.

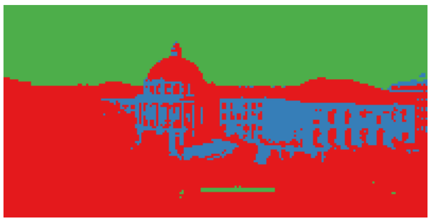
- Bandwidth = 5
  A larger bandwidth of 5 suggests a broader spatial influence for each data point during the clustering process. Which is exactly what we observe from the image: the picture is less detailed, especially in the red cluster that includes the road, buildings, and trees as one thing.

- Bandwidth = 7
  When the bandwidth is set to 7 the blurriness is even more evident. At this point the classifier can't recognize the main features of the image.


(a) bandwidth = 2.5


(b) bandwidth = 3

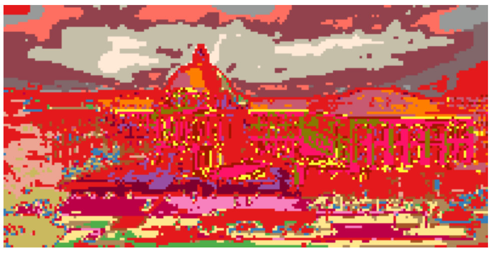

(c) bandwidth = 5


(d) bandwidth = 7

- Bandwidth = 1
  When the bandwidth is set to 1, the code is not runnable. In fact this parameter determines the width of the Gaussian kernel and with this value. When bandwidth is set to 1, the kernel becomes very narrow. This means that only very close points will contribute significantly to the update of each point. This leads to the issue of overfitting that happens in this case as the algorithm identifies more clusters than the number of colors.
  In order to solve this issue I filtered from the result only the 24 most frequent clusters and mapped the others to the nearest most frequent cluster. The results show still a lot of overfitting and noise but the code is runnable and the building profile is identified. The code implemented to overcome this issue is reported below:

```
def reduce_labels(labels):
    unique_labels, counts = np.unique(labels, return_counts=True)
    most_frequent_labels = unique_labels[np.argsort(counts)[-24:]]
    label_mapping = {}
    for i, label in enumerate(most_frequent_labels):
        label_mapping[label] = i
    reduced_labels = np.array([label_mapping.get(label, len(most_frequent_labels))
    for label in labels])
    reduced_labels = reduced_labels % 24
    return reduced_labels
```

Using this code the result I obtain is the following:


Figuur 2: bandwidth = 7