Lab5

Mean-shift

Implementation

For every step, do the operations below to get a set of new data points, each of them is the shifted center from the corresponding center of last step. Finally there should be N data points, represented the N centers finally. Some of them should be overlap a lot(first divide with 4 then round, and use **np.unique** function) which can be seen as the same cluster.

```
centroids, labels = np.unique((X / 4).round(), return_inverse=True, axis=0)
```

At last, I get the cluster(centroids) and every data points belong to which cluster(labels), then draw the corresponding color to each point, and change the data points back to original image form. Finally, I get the segmented image.

Distance Calculation

Because the radius is infinity, so I just need to calculate the distance between current point and other points, for broadcast later, I need to keep the dim.

```
def distance(x, X):
    return np.sqrt(np.square(x - X).sum(axis=1, keepdims=True))
```

Weight Calculation

Then I plug the distance into the gaussian kernel function to calculate the weights corresponding to all other points respectively.

```
def gaussian(dist, bandwidth):
    w = (1 / (np.sqrt(2 * np.pi) * bandwidth)) * np.exp(-0.5 * (dist / bandwidth)
** 2)
    weight = w / w.sum(axis=0)
    return weight
```

New Center Calculation

Use the weight calculated to weight every data points to get the new center data point.

```
def update_point(weight, X):
    return (weight * X).sum(axis=0)
```

Experiment with different bandwidth

Why

I try different bandwidth and the results are shown below. I can find that the bandwidth is important for clustering. The bigger the bandwidth is, the less the number of cluster is, the number of color of the saved image is, which means the final range of the cluster is becoming bigger, including more data points. That can be explained by the kernel function:

$$K(distance, bandwidth) = \frac{1}{\sqrt{2\pi} \times bandwidth} \exp -0.5 \times (\frac{distance}{bandwidth})^2$$

When the bandwidth is small, the kernel function is mainly influenced by the distance, which means the close data point are influencing the center more when generating new center. When bandwidth is big, the distance is not that important to the weight, which means considering about more data points instead of only the neighbor points when generating new center. To sum up, the bigger bandwidth is, the more global is, vice versa, which leads to the final number of clusters directly.

The results below prove the conclusion above, then bandwidth is 1 or 2.5， the number of cluster is very big, which is more than 24 of the color amount, leading out of bounds bug. Then bandwidth is 3, 5 , 7, bug is missing and the amount of color becomes less and less.

- 1

```
Traceback (most recent call last):
  File "D:\OneDrive\school\ETH\study\HS23\CV\Lab\Lab5\exercise\mean-shift\mean-shift\mean-shift.py", line 56, in <module>
    result_image = colors[labels].reshape(shape)
IndexError: index 251 is out of bounds for axis 0 with size 24
```

- 2.5

```
Elapsed time for mean-shift: 297.48637104034424
Traceback (most recent call last):
  File "D:\OneDrive\school\ETH\study\HS23\CV\Lab\Lab5\exercise\mean-shift\mean-shift\mean-shift.py", line 56, in <module>
    result_image = colors[labels].reshape(shape)
IndexError: index 24 is out of bounds for axis 0 with size 24
```

- 3

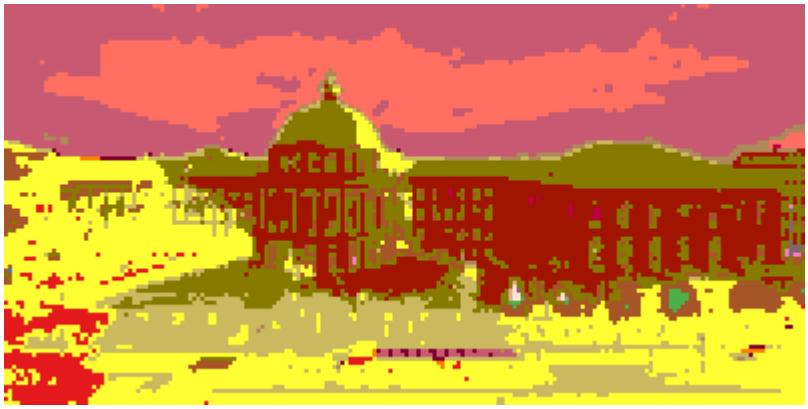
- 5

- 7

Possible Solutions

Apart from adding the bandwidth to make the clusters less, through modifying the divisor, it can also be solved. The bigger the divisor is, the less the number of cluster is.

For bandwidth equals 1, I modify the divisor to 25:



```
centroids, labels = np.unique((X / 25).round(), return_inverse=True, axis=0)
```

Of course, if I can add the number of color in colors.npz, I don't need to modify these hyper parameters, I just need to add the number of color(24) to the number of clusters.