CV: Lab 05 Writeup

Benedict Armstrong
benedict.armstrong@inf.ethz.ch

Mean-Shift

The mean-shift algorithm can be split into four steps which are repeated until convergence:
- Calculate distances between pixels
- Calculate weights for each pixel based on distance
- Calculate the mean of the pixels weighted by the weights
- Update the pixels to the mean

Calculating Distances

I simply used numpy's `linalg.norm` function to calculate the distances for all pixels from a given pixel.

Calculating Weights

We use the Gaussian kernel to calculate the weights for each pixel. The kernel is defined as:

$$K(x) = e^{\frac{x}{\sqrt{2b}}^{2}}$$

where b is the bandwidth and x is the distance between the pixels.

Calculating the Mean and Updating Pixels

The mean can be easily be calculated using the following code:

```
np.sum(weight.reshape(-1, 1) * X, axis=0) / np.sum(weight)
```

this result is then used to update the pixels.

Results

Now for the interesting part, the results. I ran the algorithm on the provided image of the ETH main building. I tried a few different bandwidths with 15 steps each to see how they affect the result.

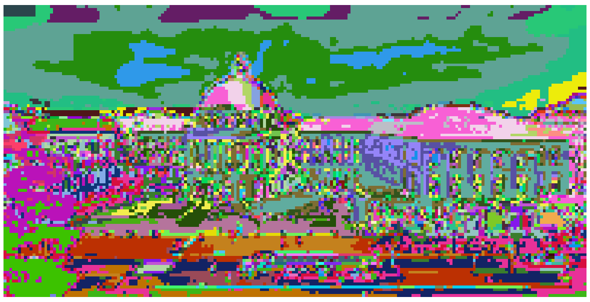
Figure 1: Mean-shift with bandwidth = 1

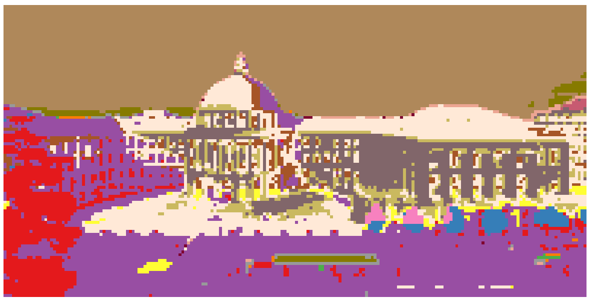
Figure 2: Mean-shift with bandwidth = 3

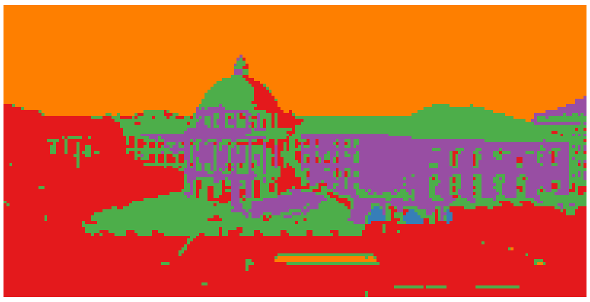
Figure 3: Mean-shift with bandwidth = 5

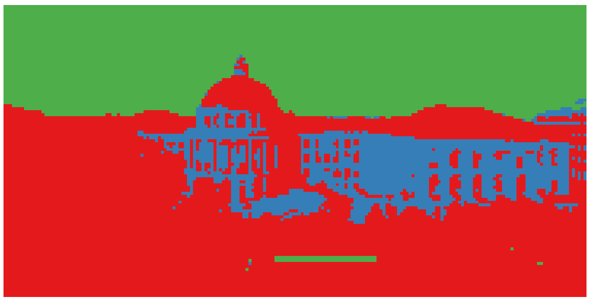
Figure 4: Mean-shift with bandwidth = 7

Bandwidths of 1 and 7 seem to be too small and too large respectively. For $b = 1$ the image is too noisy and for $b = 7$ the image is too smooth and we loose detail such as the left side of the building.

A bandwidth 3 and 5 seems pretty good but somewhere in the middle might be the sweet spot. After playing around abit I found that a bandwidth of 4.5 with 20 steps gives the following (best) result:

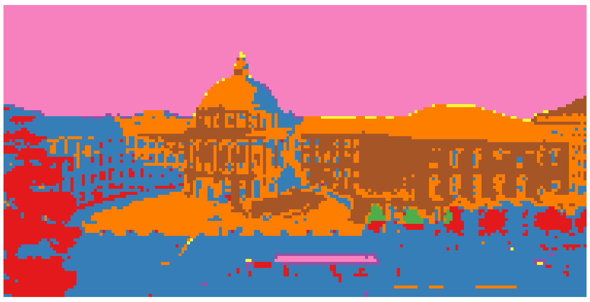
Figure 5: Mean-shift with bandwidth = 4.5 and 20 steps

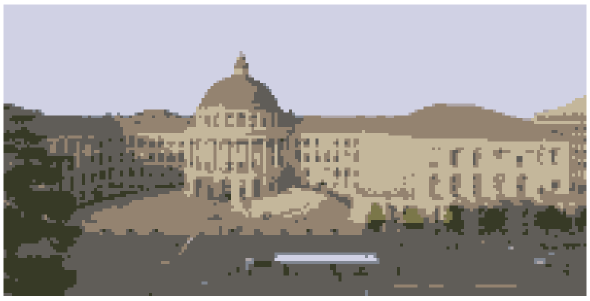
Figure 6: Mean-shift with bandwidth = 4.5 and 20 steps (using color of centroids)

Which seems to be a good balance between the two. There was issue I noticed where `colors.npz` only contained 24 colors. So if the algorithm generated more than 24 centroids it would fail. My solution was to simply create a new `colors_2.npz` file with 500 (random) colors.

Color space

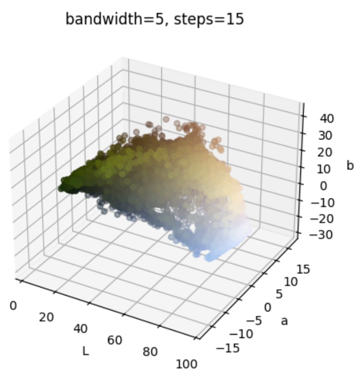

Figure 7: 3D Scatter plot of the colors before mean-shift

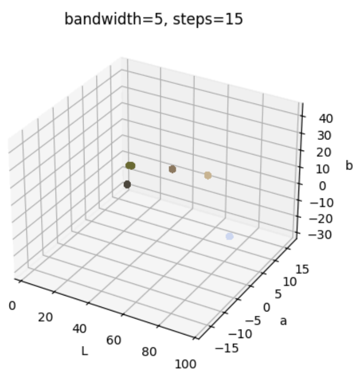

Figure 8: 3D Scatter plot of the colors after 15 steps of mean-shift with bandwidth = 5

We can plot the color of each pixel on a 3D scatter plot to see how the colors are grouped. The first image shows the colors before mean-shift is applied, and the second shows the colors after 15 steps of mean-shift with a bandwidth of 5. We can see that the colors are grouped into 5 clusters.

I've also made a few little GIFs showing the evolution of the algorithm (including the scatter plot shown above).