Prof. Marc Pollefeys

Eric Tillmann Bill: Assignment 4

erbill@student.ethz.ch

Task 2: Mean Shift

2.1: Implement the distance Function

To calculate the distance, I make use of the assumption that the radius is set to $+\infty$, which simplifies the calculations as I can then simply calculate the distance from the given point x to all points in X pairwise. In detail, I have decided to use the Euclidean distance and call the following function from the "numpy" library:

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

2.2: Implement the Gaussian Function

The Gaussian function is used to assign a weight to all values in the image that are used in the update function. Specifically, a point is assigned a higher weight if it is close to the point to be updated, while it is assigned a lower weight if it is far away. The choice of what is defined as close is set with the bandwidth parameter, where a larger value results in more points being considered close, while a smaller value has the opposite effect. My implementation is as follows:

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

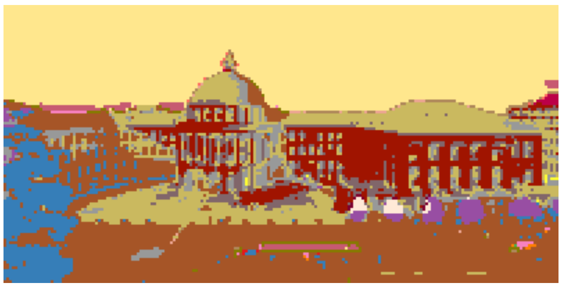
Note that I exclude the scalar `1 / (bandwidth * np.sqrt(2 * np.pi))` as this is not necessary as it is independent of the input (`dist`) and afterward I calculate the normalized weighted sum of all values (in `update_point()`, so the scalar would cancel out anyway. This implementation simply ensures better numerical stability.

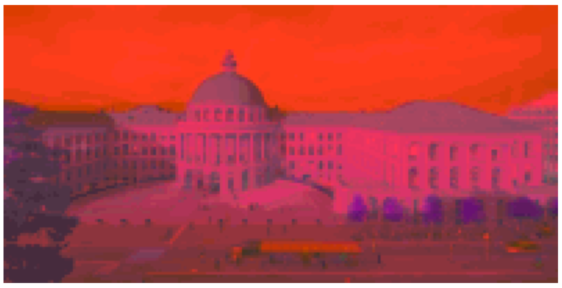
2.3: Implement the update_point Function

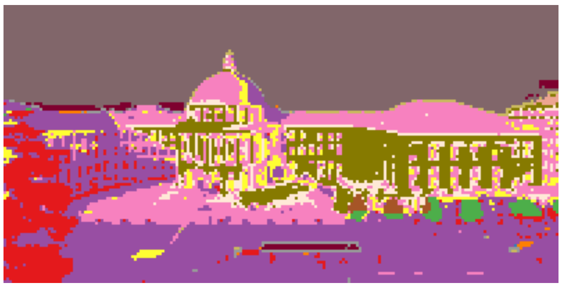
We calculate the new weighted mean by summing each point multiplied by its weight factor and then normalize the entire sum by dividing by the sum of all distances.

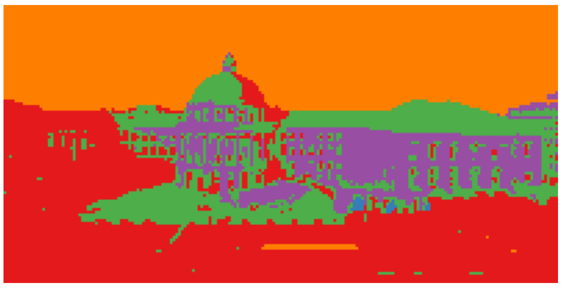
2.4: Experiment with different bandwidth

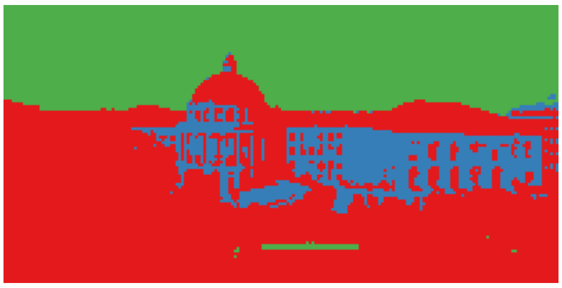
I experimented with different values for the bandwidth parameter. With the exception of image b), it is clear to see that a larger bandwidth results in fewer details in the image segmentation, and larger portions of the picture are assigned the same label. This is to be expected, as the bandwidth parameter, directly controls how close two color-values are. For this example, the optimal bandwidth parameter would be 2.5 or 3.0 as they preserve most of the details, while also being good a segmenting the image.


(a) Bandwidth: 2.5 and Number of steps: 50


(b) Bandwidth: 1.0 and Number of steps: 20 with convergence behavior


(c) Bandwidth: 3.0 and Number of steps: 20


(d) Bandwidth: 5.0 and Number of steps: 20


(e) Bandwidth: 7.0 and Number of steps: 20

Note, for the resulting image when using a bandwidth of 1.0, the number of labels is 282 and therefore exceeds the maximum number of colors (24) available. A possible solution to that is to use the colors of the modes, the mean-shift algorithm converges to. This means, that in the code, we simply use the centroids as color values. You can see the result in image b). This demonstrates nicely that the smaller we choose the bandwidth, the more details we obtain.