Computer Vision - Lab Assigment Report
Image segmentation

Tuan Mate Nguyen (tunguyen@student.ethz.ch)

Mean-shift

First I calculated the L2 distances of the current color to all the other pixels' color (as magnitude of the difference vector). The implementation of the Gaussian weight function is the same for both versions (already vectorized). To update the current point first the color vectors of all points are multiplied with their weights and divided by the sum of the weights (normalization). Then the sum of these normalized color vectors yield the new color vector.

The vectorized implementation had equivalent steps exept that in the beginning I created a copy array of all the points (X) for each point to allow for parallel processing.

Bandwidth

As expected, increasing bandwidth results in fewer distinguished segmented object. The reason is that the weights of more distant (in color-space coordinates) colors - contained in the image pixels - are higher relative to closer colors' weights than in the small bandwidth case. As a result the mean color value assigned to the current pixel will be closer to the global mean (mean color of the full image). Consequently these mean values will also be closer to each other with each step until being merged into a set of colors with only few different colors (e.g. 3 colors for $bandwidth = 5$ case).

On the other hand, in case of a smaller bandwidth value, basically only the close neighborhood would be taken into account. If the color clusters (segments) are well-separated and far apart, then the found mean will better approximate the cluster mean.

If, however, the bandwidth is too small, the new color will basically remain unchanged (other colors have low weights so they have no effect) with every pixel representing a segment, resulting in an image very similar to the original. Actually the script crashes if there are too many colors (probably not enough labels).

(a) $bandwidth = 1.8$ (b) $bandwidth = 2.5$ (c) $bandwidth = 3.8$ (d) $bandwidth = 5$

Figure 1: Segmented images for different bandwidth values

Performance

The script was run on a desktop machine with a GeForce GTX 1080 Ti graphics card. (note: on my laptop I got warnings about insufficient amount of RAM). The runtime was chosen to be the fastest out of 5 measurements. As expected the CPU version using Python loops was the slowest and the batched (vectorized) version on the CPU was second slowest. On the GPU the non-vectorized and vecorized implementations achieved around 2x and 16x speed-up, respectively, compared to the best CPU result.

CPU	32.41s
Batched CPU	18.62s
GPU	7.41s
Batched GPU	1.17s

Table 1: Timing results