Computer Vision 1: Homework 9

Important: Mark the homeworks you solved in the homework sheet and bring your solutions with you to the exercise class. For each homework problem, one student will be chosen at random to present their solution.

Programming tasks.

Download and read `Elbphilharmonie.jpg`.

- Segment the color image by using the function `skimage.segmentation.slic`. Use the parameters $n = 500$ and $c = 10$.
- Draw the resulting segmentation by both
  - coloring the area of each superpixel by the corresponding segment's average RGB value in the image. Use `skimage.color.label2rgb` with argument `kind='avg'`, and
  - drawing the boundaries of superpixels. Use `skimage.segmentation.mark_boundaries` to get an image with the overlaid borders.

  Examples of both visualizations are shown in Figure 1.
- Features extracted from individual superpixels can be useful in many applications. As an example of computing such a feature, use logical indexing to calculate the average RGB value of all pixels in the cluster with index 3.

  Hints: First, use a comparison to get a binary array indicating where the clustering result array is equal to 3. Then, use the binary array to index the color image to extract the RGB values of all relevant pixels. Finally, calculate the mean R, G, and B values of all relevant pixels.

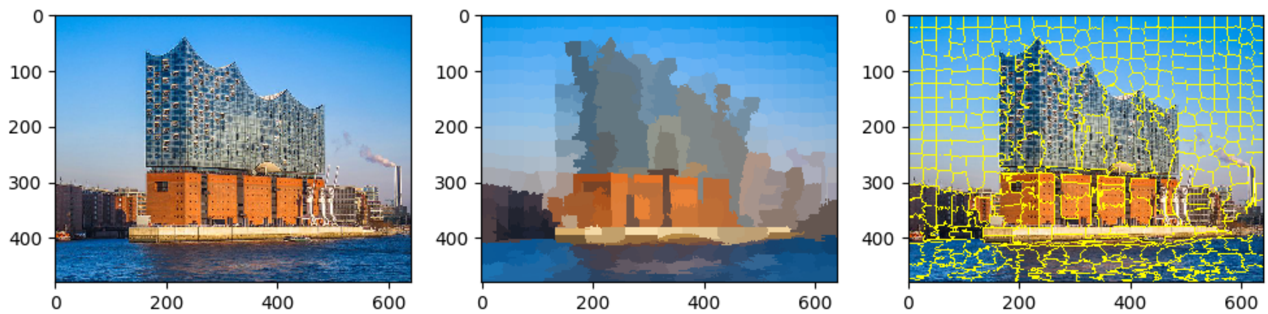


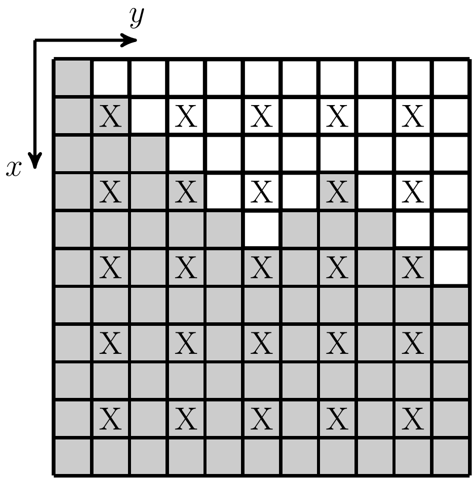
Figure 1: Original image (left) and SLIC superpixels ($n = 500$, $c = 10$) visualized by average colors (middle) or by drawing superpixel boundaries (right).

Other tasks.

1. Keypoint orientation in SIFT is used to represent a keypoint relative to its orientation thus achieving invariance to image rotation. The orientation is calculated by using sample points around the keypoint. Given a Gaussian smoothed image L, the gradient $\nabla f(x, y) = \begin{bmatrix} g_x & g_y \end{bmatrix}^T$ at pixel location (x, y) is calculated using the central difference filter:

$$g_x = L(x + 1, y) - L(x - 1, y)$$
$$g_y = L(x, y + 1) - L(x, y - 1).$$

Consider the image shown below, where white pixels have the value 1 and pixels with gray background have value 0. The axes labeled with x and y indicate the directions in which the respective pixel coordinates increase, so that the upper left pixel is at coordinate $(0, 0)$, and the lower right pixel is at coordinate $(10, 10)$.



For each sample point marked by a symbol "X" in the picture, calculate

- the gradient $\nabla f(x, y) = \begin{bmatrix} g_x & g_y \end{bmatrix}^T$,
- the magnitude of the gradient $m(x, y) = \sqrt{g_x^2 + g_y^2}$, and
- the angle of the gradient $\theta(x, y) = \arctan\left(\frac{g_y}{g_x}\right)$.

2. Consider Figure 2, which shows a normalized orientation histogram for a SIFT keypoint after weighting[1].

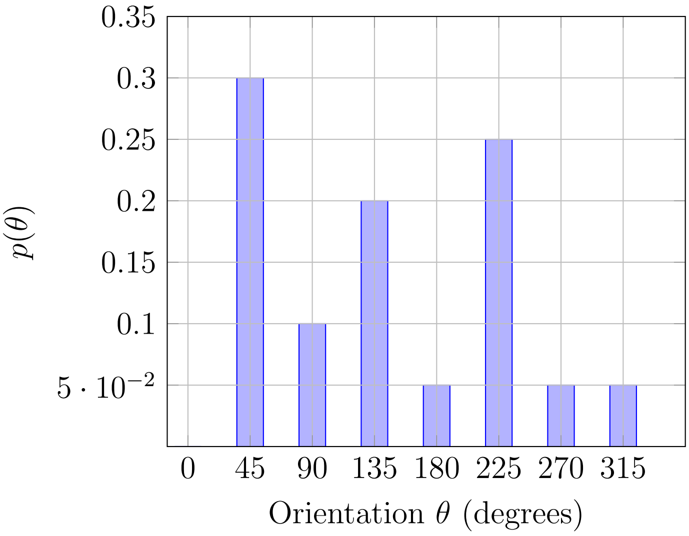


Figure 2: A normalized orientation histogram of a SIFT keypoint.

(a) What is the dominant local direction of the keypoint?

(b) How many new keypoints will be created, and why? What are their orientations?

[1]For simplicity, we consider an 8-bin orientation histogram. In the original SIFT algorithm, 36 bins are used.

3. We have developed a segmentation algorithm that has partitioned an image into segments S_i, $i = 1, \ldots, n$ as described on the lecture slides. For the same image, we have a *ground truth segmentation* G_j, $j = 1, \ldots, m$, which specifies how the image ideally should be partitioned.

   Intuitively, the *undersegmentation error* for an individual ground truth segment G_j measures the amount of "bleeding" that a segmentation exhibits beyond G_j. The undersegmentation error for a ground truth segment G_j is defined as

$$\mathrm{UE}(G_j) = \frac{\left[\sum_{\{S_i | S_i \cap G_j \neq \emptyset\}} \mathrm{Area}(S_i) \right] - \mathrm{Area}(G_j)}{\mathrm{Area}(G_j)}$$

   where the summation is over all segments S_i which have any overlap (non-empty intersection) with G_j, and the function Area returns the area, i.e., the number of pixels, of the corresponding segment.

   Our image has 12 pixels in it, and our segmentation algorithm produces the segments S_i shown in Figure 3. Calculate the undersegmentation errors $\mathrm{UE}(G_1)$ and $\mathrm{UE}(G_2)$ of the ground truth segments G_1 and G_2 shown in Figure 4.

1	4	4	4
1	2	4	4
1	2	3	3

Figure 3: A segmentation S_i, $i = 1, 2, 3, 4$, with i indicated on each pixel.

Figure 4: Two ground truth segments G_1 (left) and G_2 (right) highlighted in gray.