Computer Vision 1: Assignment 4
(Due date: 10.01.2022)

Submission instructions:

- For each programming task, submit a `.py` source code file or a Jupyter/Colab notebook file. Do not include any images or data files you used.
- For each pen & paper task, submit a `.pdf` file. Type your solution in LaTeX, Word, or another text editor of your choice and convert it to PDF. Do not submit photographs or scans of handwritten solutions!
- In all submissions, include at the top names of all students in the group.
- Choose one person in your group that submits the solution for your entire group.

Task 1: SIFT (pen & paper)

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

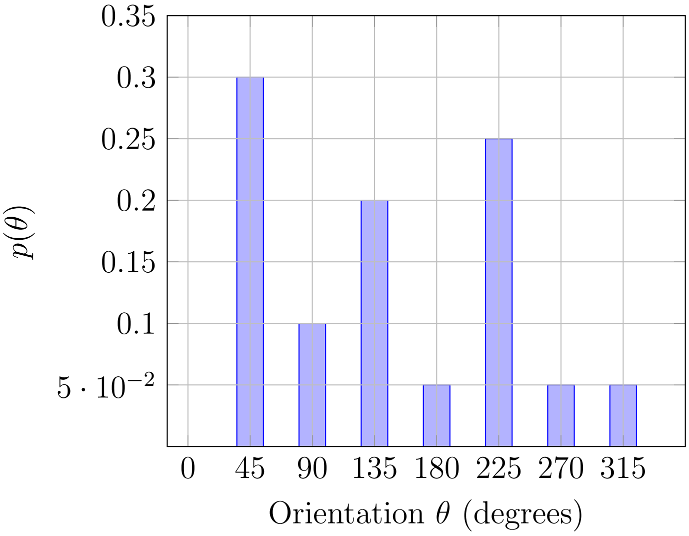

Figure 1: A normalized orientation histogram of a SIFT keypoint.

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

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

Task 2: Undersegmentation error (pen & paper)

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

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

error for a ground truth segment G_j is defined as

$$\mathrm{UE}(G_j) = \frac{\left[\sum_{\{S_i | S_i \cap G_j \neq \emptyset\}} \mathrm{Area}(S_i)\right] - \mathrm{Area}(G_j)}{\mathrm{Area}(G_j)} \tag{1}$$

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 2. 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 3.

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

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

Figure 3: Two ground truth segments G_1 (left) and G_2 (right) highlighted in gray.

Task 3: Undersegmentation error (programming)

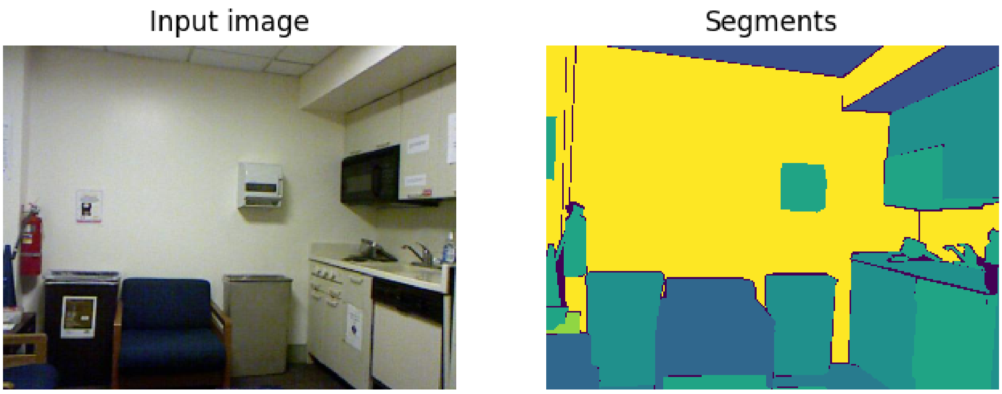

Figure 4: Input image (left), and a ground truth segmentation result (right), where the colors indicate the segments. Segments correspond to semantic classes such as “wall”. Source: NYUv2 dataset `https://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html`.

- Download the input image `0001_rgb.png` and the ground truth segmentation `0001_label.png` from Moodle. These images are shown in Figure 4.
- In the label image L, the pixel intensity value indicates which ground truth segment a pixel belongs to. For example, if $L(x, y)$ has a value of i, the pixel (x, y) belongs to the ith segment.
- Use the implementation of the SLIC algorithm from `skimage.segmentation.slic` to segment the color image. Select reasonable values for the parameters n (number of segments) and c (compactness). Visualize the segmentation and revise the parameters if necessary. Leave the other parameters at their default values. **Note:** Because SLIC is not a *semantic segmentation* method, you should not expect its output to match that shown in Figure 4.
- Implement calculation of the undersegmentation error as defined in Eq. (1) for each ground truth segment in L. Compute and print out the undersegmentation error for every ground truth segment, and the average undersegmentation error over all ground truth segments.
- How does the average undersegmentation error change when you increase the desired number of superpixels n? **Why? Answer in 1-2 sentences as a comment in your code.**

Hints:

- Use comparisons such as `L==i` to obtain binary masks areas of the label image L that belong to segment i.
- Use logical indexing with binary masks to extract corresponding areas of the segmentation result.
- Use `np.unique` and `np.count_nonzero` to find overlapping superpixels and to calculate their areas, respectively. You may also use `np.unique` to discover all labels that are present in L.
- Note that the label image has value 0 for pixels that are unlabeled (e.g., parts of boundaries). You can either ignore this or skip the label 0, as the undersegmentation error for this segment is not very meaningful.