CS 484/555
Fall 2024
Homework Assignment 2

Due: 30 November 23:59

1 Line Based Object Matching: [50 points]

The goal of this assignment is to match objects by comparing their line patterns. In particular, you will try to match books according to the line orientation histograms computed from the images of their covers. You are given a data set of books in which each book has two images.

One of them is the original image of the book cover, and the other one is a rotated version of the cover with respect to an arbitrary angle between 0 and 180 degrees. The goal is to find a match between the original and rotated books, and then, to find the angle of rotation. Fig. 1 shows the original images of 15 books. Fig. 2 shows the rotated images of these books.

The approach can be summarised in terms of the following steps.

- *The data*: the images come as two folders along with the homework assignment in folder named **part1**. Half of the images belong to the original books, and the remaining belong to the rotated ones.
- *Perform edge detection*: run the Canny edge detector on each image after converting it to a greyscale image. The output of this step is a binary edge image that marks pixels that represent a significant change. You have to experiment with the parameters to obtain important edges that will be useful in the following steps. After evaluating different parameter values for a subset of the data set, you must fix them and use the same values for all images. Fig 3 shows some examples.
- *Perform line fitting to find line segments*: find straight line segments in the edge detector outputs using Hough transform. After performing the Hough transform, you can find the bins that accumulated the most points in the Hough array. You can play with the minimum number of peaks to find a reasonable number of lines. Fig. 4 shows some examples.
- *Compute line orientation histograms*: the orientation values are typically in the range $[-\pi, +\pi]$. You can divide this range into uniform bins, and compute a histogram of line orientations weighted by line lengths. That is, a line should contribute to its corresponding bin by its length instead of just 1. You have to experiment with the number of bins to find a good representation.
- *Find a match and compute the angle of rotation for each rotated book*: in this step, your task is to find the original book of each rotated book using the orientation histograms. Rotated book histograms can be considered as shifted versions of the corresponding original book histograms. The number of bins in a rotated book histogram that has to be shifted to match the corresponding original book histogram should be approximately proportional to the angle of rotation. Hence, the angle of rotation can be estimated by finding how many bins the second histogram has to be shifted so that the two orientation histograms are similar. The original book of each rotated book can be found by shifting the rotated book histogram to one bin at each iteration and calculating the Euclidean distances between the shifted histogram and the original book histograms. As a result, the original book that results in the minimum Euclidean distance can be a reasonable match. Also, the amount of shift needed can be used to find the approximate angle of rotation. Note that you should use circular shifts when you compare the histograms.

Figure 1: Original book images.

- *Discuss your results*: you should discuss the results of each step and compare how the performance is affected by different parameters (i.e., Canny parameters, Hough transform parameters, number of bins).

For this part in the submission include:

- In the report:
  - The results for edge detection for different parameters. You can use edge detection code from other sources but you must cite the source that you used.
  - The results for line detection in which detected lines are overlayed on the original images. You can use Hough transform code from other sources but you must cite the source that you used.
  - Example line orientation histograms. You must provide results for different numbers of bins.
  - Results for matching the rotated books to the original books as well as the estimated rotation angles. You must provide a result for each book, i.e., 15 results. You can provide additional results for different parameter settings.
- A well-documented script that runs the particular sequence of operations and reproduces the result presented in your report for computing the line orientation histogram using the line detection results and for matching the line orientation histograms for different books using circular shifts and Euclidean distances. You must write your **OWN CODE** for this part.

Figure 2: Rotated book images.

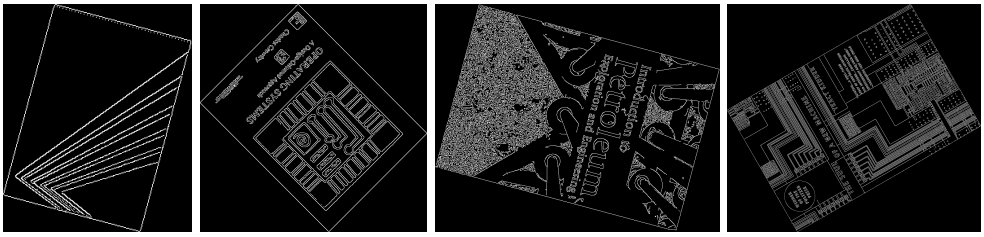

Figure 3: Example edge maps on rotated images.

2 Image Segmentation with Superpixels: [50pts]

Image segmentation is a fundamental problem in computer vision. The goal of this assignment is to obtain an accurate segmentation of image texture information. A common problem in segmentation algorithms is that it is usually hard to find a global set of parameters that work well for all images in a data set. The same set of parameters may provide an over-segmentation for some images while they may cause under-segmentation for others. In this assignment, we will try to obtain a good segmentation by, first, over-segmenting the images, and then, combining the regions that have common texture characteristics using clustering. The data set for this assignment consists of 10 images, shown in Fig. 5, that are taken from a database at the University of Washington, Department of Computer Science and Engineering (http://imagedatabase.cs.washington.edu/groundtruth/). These images are also given as a part of the assignment in the folder named **part2**

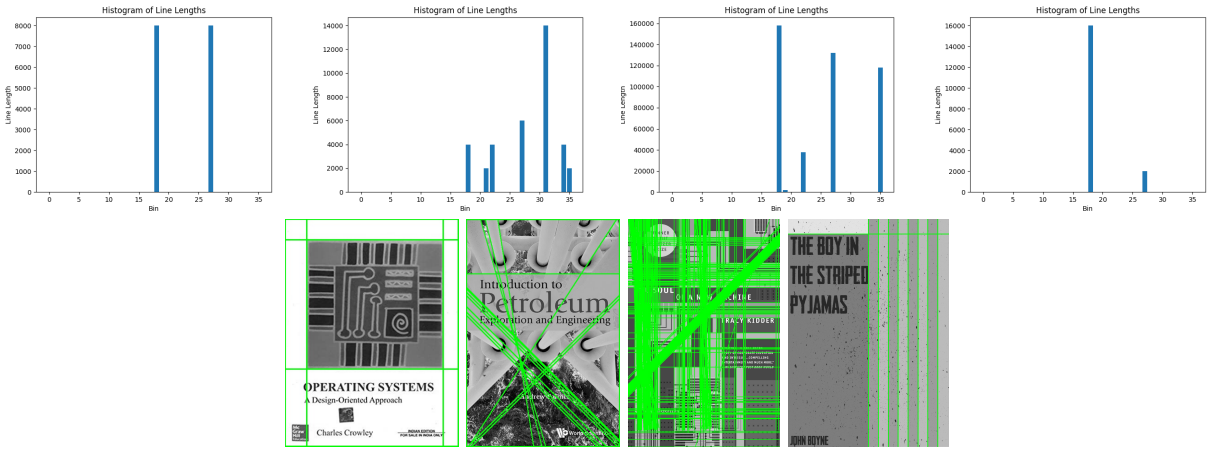

Figure 4: Lines overlayed on sample images and their corresponding histograms.

Figure 5: Ten images in the data set of this assignment.

2.1 Part 1

The first step is to obtain an over-segmentation for each image. First, read the following paper (the pdf is provided along with this assignment):

R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, S. Susstrunk, "SLIC Superpixels Compared to State-of-the-art Superpixel Methods," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34, num. 11, pp. 2274–2282, May 2012.

The algorithm described in the paper can be used to obtain an over-segmentation of an image into superpixels which are locally homogeneous groups of pixels that preserve the object boundaries. Oversegmentation using superpixels can be used as a preprocessing stage that provides an abstraction and simplifies the computation at later stages. Second, use the code for the above paper available here to obtain superpixels for each image in your data set. There are two versions (SLIC and SLICO). You can use either version. Read the instructions for the version you choose carefully. The output that will be necessary in the rest of the homework assignment is a matrix that contains an integer label for each pixel where each label indicates the corresponding pixel's superpixel id. You may need to experiment with the parameters in the code to obtain a meaningful over-segmentation. After trying the code on several images, fix the parameters and use the same settings for segmenting all images. The output of this part for each image is an integer label image where each pixel stores the id of the region it belongs to.

2.2 Part 2

The second step is to compute Gabor texture features for all images. You can use the Gabor filter implementation in Matlab's image processing toolbox which can be accessed from the links below:

1. the Matlab implementation by Dr. Peter Kovesi
2. gaborconvolve.m

. You can adjust the parameters so that you obtain a filter bank of 4 scales and 4 orientations. You can use the suggested default values for most of the parameters. Note that Gabor filter responses should be computed on the grayscale version of the input images, and the magnitude of the output response should be used in the rest of the assignment. At this step, for each pixel, you will have one response for each filter, e.g., you will have a total of 16 responses per pixel if you use 4 scales and 4 orientations. Once you have Gabor representations for each pixel in an image, compute Gabor features for all superpixels by simply computing the average of Gabor features of the pixels inside a particular superpixel. Assume that you have Ni superpixels extracted from an image i, after this step you will have an $N_i \times 16$ feature matrix for image i.

2.3 Part 3

Now, each superpixel is represented as a point in the 16-dimensional feature space. You can use any clustering algorithm (e.g., k-means) to group the superpixels. You can use clustering code from other sources but you are required to know the details of the particular implementation used. The clustering algorithm must be applied to the whole data set (not separately for each individual image), i.e., feature vectors of all superpixels of all images (or a random sample if you have memory issues) should be given to the clustering algorithm. The output at this step is a matrix that contains an integer (cluster) label for each superpixel. You can use these labels to produce a pseudo-colour representation of the clustering result by colouring the pixels corresponding to each superpixel according to the cluster label of that superpixel. You need to experiment with the design parameters (e.g., superpixel size, Gabor filter bank size, number of clusters) to obtain a meaningful clustering. You should provide an output pseudo colour image for each image in the data set.

2.4 Part 4

The final step is to build a contextual representation for each superpixel and repeat the clustering process. First, you will need to find the neighbouring superpixels for each superpixel. We define two levels of neighbourhoods. For a given superpixel, the first level neighbours consist of the superpixels that have a number of pixels in a ring around the given superpixel. In Fig.6 , this first level ring is represented by the yellow circle. That is, the superpixels that are in the first level neighbourhood of the blue superpixel are the ones that have a number of pixels within the yellow circle (except the blue superpixel). Given the expected size of a superpixel and the diameter of its equivalent circle, the yellow ring corresponds to the circle which is centred at the centroid of the blue superpixel and has a radius of 1.5 times this diameter. You can set a threshold for the number of pixels required to accept a superpixel to be inside a given ring. The second level neighbourhood is defined similarly and is shown by the green ring in Figure 2. The superpixels that have a number of pixels within the ring bordered from outside by the green circle and from inside by the yellow circle are the ones in the second-level neighbourhood of the blue superpixel. The radius of the green circle can be set as 2.5 times the expected diameter of a superpixel. Once the neighbouring superpixels are identified, you should compute an average feature vector from the vectors of the superpixels within the first-level neighbourhood and a separate average feature vector from the vectors of the superpixels within the second-level neighbourhood. Then, the final contextual feature representation for the superpixel is constructed by concatenating the feature vector of this superpixel, the average vector computed for the first-level neighbourhood, and the average vector compared for the second-level neighbourhood. For example, when the feature vectors for superpixels have lengths of 16, the concatenated feature vector will have a length of 48. You should repeat the clustering process by using the contextual feature vectors. You need to experiment with the design parameters e.g., the threshold for accepting a superpixel to be within a ring, and possibly the radius of the first and second-level neighbourhoods) to obtain meaningful clustering. You should provide an output pseudo colour image for each image in the data set.

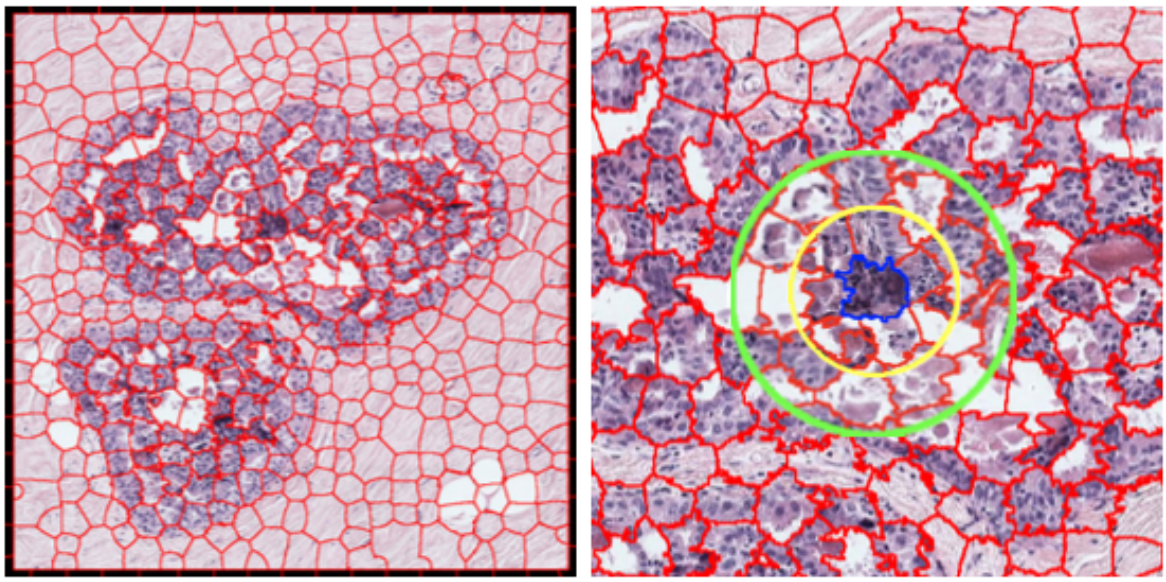

Figure 6: left: Superpixels by the SLIC0 algorithm; right: Circular neighbourhoods around the blue superpixel. **Note that this image is only given as a demonstration, your dataset does not contain breast cancer images.**

For this part in the submission include:

- In the report
  - Description of the parameters used for superpixel segmentation.
  - Segmentation results for all images using the superpixel segmentation algorithm (SLIC version) that you chose. Note that you must produce over-segmentations using the same parameters for all images.
  - Description of the parameters used for Gabor texture feature extraction.
  - Gabor texture feature examples (e.g., the output of the imgaborfilt or gaborconvolve functions) for several scales and orientations for at least two different images.
  - Clustering/segmentation results for all images. You must show the obtained regions by overlaying segment boundaries on each image or overlaying the pseudo-colouring result with some transparency on the image as shown in the lecture slides.
  - Citation for any external code used.
  - Detailed discussion of the results with respect to different parameter settings (including all parameters listed above and descriptions of how and why you chose the particular values).
- Any source code that you wrote for the assignment.

Notes

- You should submit your solutions as a single archive file that contains your code and report (a pdf file that contains the description of the methodology, the resulting images, descriptions of how you obtained them, and discussion of the results) to Moodle.
- For any questions, please contact yigit.ekin@bilkent.edu.tr via email.