Digital Image Processing
School of Information Science and Technology
ShanghaiTech University

Assignment 3

An Lihua 2022131001

Due time: 23:59, May 5th, 2023

---

1 Notes

This homework has **80 points** in total.

Please submit your homework to blackboard with a zip file named as **DIP2023_ID_Name_hw3.zip**. The zip file should contain three things: **a folder named 'codes' storing your codes**, **a folder named 'images' storing the original images**, and **your report named as report_ID_Name_hw3.pdf**. The names of your codes should look like **'p1a.m'** (for (a) part of Problem 1), so that we can easily match your answer to the question. Make sure all paths in your codes are relative path and we can get the result directly after running the code. Please answer in English.

Please complete all the coding assignments using MATLAB. All core codes are required to be implemented by yourself (without using relevant built-in functions). Make sure your results in the report are the same with the results of your codes. Please explain with notes at least at the key steps of your code.

2 Policy on plagiarism

This is an individual homework. You can discuss the ideas and algorithms, but you can neither read, modify, and submit the codes of other students, nor allow other students to read, modify, and submit your codes. Do not directly copy ready-made or automatically generated codes, or your score will be seriously affected. We will check plagiarism using automated tools and any violations will result in a zero score for this assignment.

Problem 1 (30 pts)

(a) Please implement the Basic global thresholding on ”flower.tif”. (start with T = 0.001) (10pts)

(b) Please implement the Otsus method on ”caster_stand.tif”. (10 pts)

(c) Please implement the edge tracking using Canny Edge Detector on ”fingerprint.tif”. (10 pts)

Solution:

(a) The algorithm step is from the course slides.

➢ **Steps:**

1. Select an initial estimate of the global threshold **T**;
2. Segment the image using **T** to two groups G_1(>T) and G_2(⩽T) ;
3. Compute average intensity m_1 and m_2 for G_1 and G_2 respectively;
4. Compute new threshold **T**=(m_1 + m_2)/2;
5. Repeat 2-4 until the difference between **T** in successive iteration is smaller than requirement.

Figure 1: Algorithm step.

The results are shown below, the thresholding level is 0.4326 in this case:

Figure 2: Basic global thresholding: (a) origin image, (b) after BGT.

(b) Otsus method needs to calculate the CDF of the histogram of image first, then get the average intensity and global variance. Next is to find between-class variance σ_B and choose the maximum value as the thresholding T. The result is shown below, the thresholding level using Otsus is 0.3412:

Figure 3: Ostus method: (a) origin image, (b) after Otsus thresholding.

(c) Canny edge detector method has four main steps:

1. Smooth the input image with a Gaussian filter;
2. Compute the gradient magnitude and angle images;
3. Apply non-maxima suppression to the gradient magnitude image;
4. Use double thresholding and connectivity analysis to detect and link edge.

So in this problem, I first use Gaussian filter to smooth the image and then use sobel mask to detect the edge of the image.

The purpose of non-maximum pixel gradient suppression is to eliminate the spurious response caused by edge detection. The basic method is to compare the current pixel gradient intensity with the gradient intensity of adjacent pixels along the positive and negative gradient directions. If it is the maximum, the pixel is retained as the edge point. If it is not the maximum, it is suppressed and not considered as the edge point. For more accurate calculation, linear interpolation is usually used between two adjacent pixels crossing the gradient direction to obtain the pixel gradient to be compared.

For the final step, it is necessary to define a high threshold and a low threshold, and in this problem is 20 and 80 respectively. Pixels with gradient intensity lower than the low threshold are suppressed and not considered as edge points; pixels above a high threshold are defined as strong edges and retained as edge points; those between high and low thresholds are defined as weak edges and are left for further processing. Generally, weak edge pixels caused by real edges will be connected to strong edge pixels, while noise response will not be connected. By viewing weak edge pixels and their 8 neighboring pixels, one can make a judgment based on their connection with strong edges.

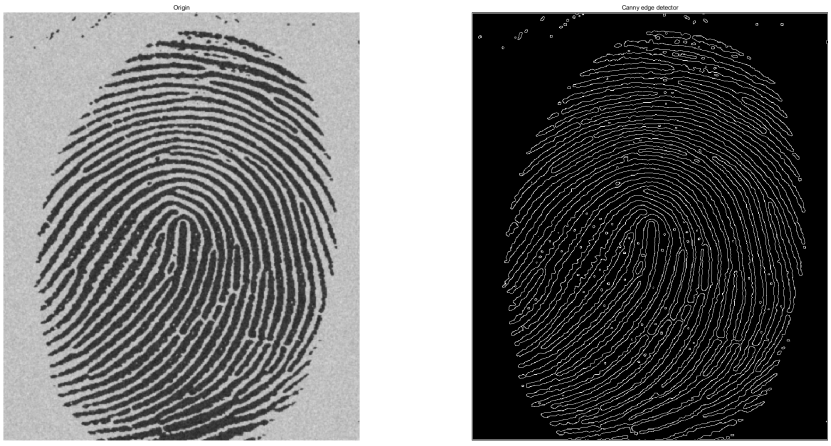

Figure 4: Canny edge detector: (a) origin image, (b) after Canny.

Problem 2 (20 pts)

Figure 1 shows an image of some texts titled with an unknown angle. For better readability, the titling angle must be found and used to correct the orientation of the image. Hough transform is a good choice for this application. 'tilted.png' is one of the texts, your tasks are:

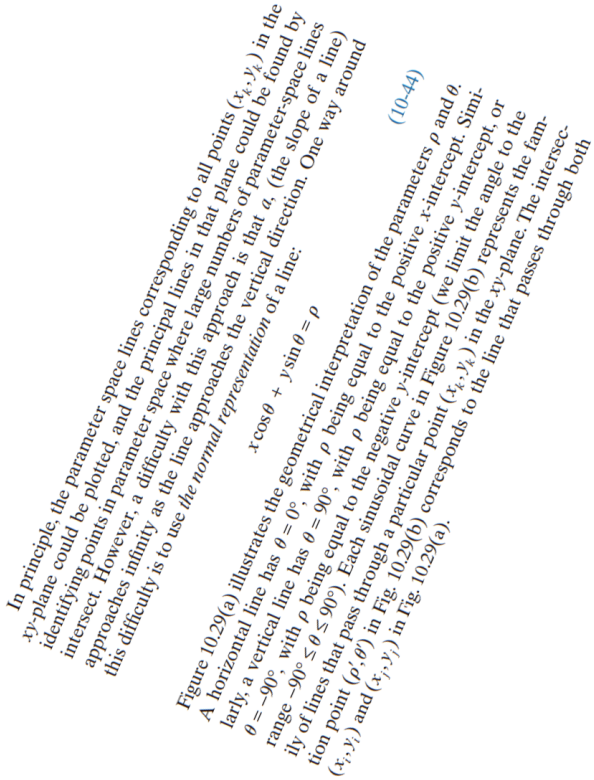

Figure 5: tilted text

(a) Implement Hough transform and perform it on 'tilted.png', show its $\rho\theta$ -plane image. The input of Hough transform is a binary image, so you should do some thresholding before performing Hough transform. (15 pts)

(b) Using result obtained by Hough transform, correct the tilt angle, show the corrected image. (5 pts)

Solution:

(a) Hough transform is a very important method for detecting the shape of discontinuity boundaries. It achieves the fitting of lines and curves by transforming the image coordinate space into the parameter space. The main steps can be summarized as follows:

1. Read color images from RGB three channels and convert them into grayscale images;

2. Using operators such as prewitt and sobel to perform edge detection on grayscale images and binarize the images;

3. Using the Hough transform detection method, count the number of occurrences of each data point after Hough transform, take the (rho, theta) corresponding to the largest data points as the parameters of the line equation, and calculate the line equation;

4. Draw a line equation image in the original image to achieve image line detection.

So in this problem, I choose sobel mask as the detector to get the edge of the image, then calculate the parameter θ and ρ by $x\cos\theta + y\sin\theta = \rho$.

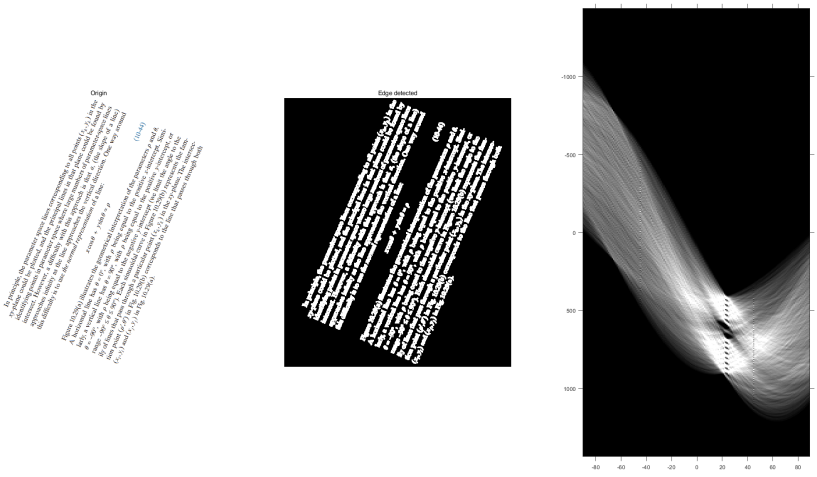

Figure 6: Hough transform: (a) origin image, (b) edge detected, (c) $\rho\theta$ -plane.

(b) Since we have got θ and ρ from (a) and combined them to a list H, then sort H and match the maximum value which is the angle we need, in this problem we need to rotate -66 degreed to correct it.

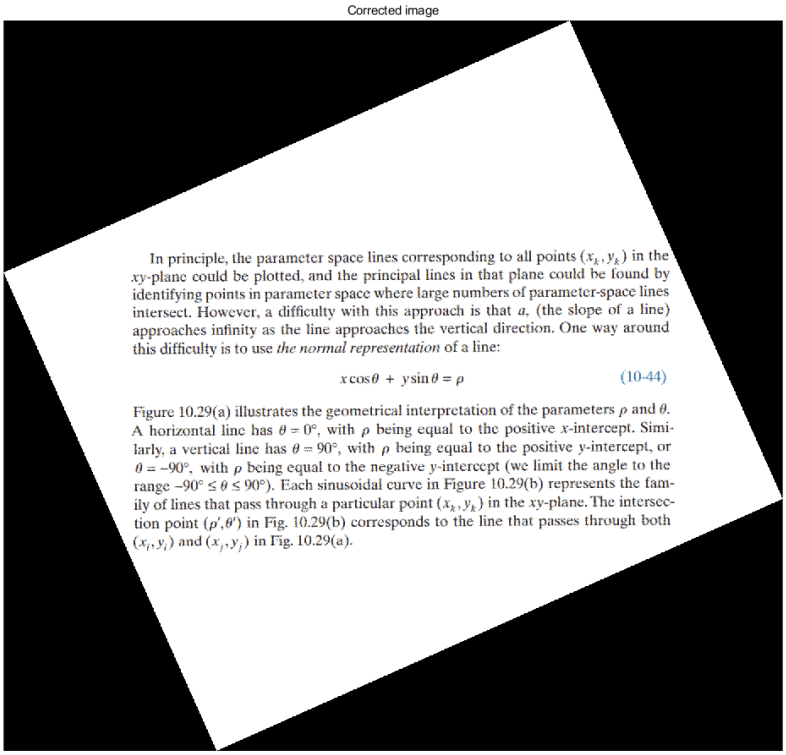

Figure 7: Hough transform: (a) origin image, (b) corrected image.

Problem 3 (30 pts)

Super pixel is a method that turns a pixel-level picture into district-level picture, which is an abstraction of basic information elements. A super pixel is a small area composed of a series of adjacent pixels with similar characteristics such as color, brightness, and texture. Most of these small areas retain effective information for further image segmentation, and generally do not destroy the boundary information of objects in the image.

In this problem, you need to turn 'sea house.jpg' to super pixel style using SLIC algorithm with 100, 500 and 1000 cluster centers (In practice, number of cluster center can be slightly different from these values) and show the result images.

Reference, doi:10.1109/TPAMI.2012.120

Algorithm 1 SLIC superpixel segmentation

/ Initialization */*
Initialize cluster centers $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ by sampling pixels at regular grid steps S.
Move cluster centers to the lowest gradient position in a 3×3 neighborhood.
Set label $l(i) = -1$ for each pixel i.
Set distance $d(i) = \infty$ for each pixel i.

repeat
  / Assignment */*
  for each cluster center C_k **do**
    for each pixel i in a $2S \times 2S$ region around C_k **do**
      Compute the distance D between C_k and i.
      if $D < d(i)$ **then**
        set $d(i) = D$
        set $l(i) = k$
      end if
    end for
  end for
  / Update */*
  Compute new cluster centers.
  Compute residual error E.
until $E \leq$ threshold

Figure 8: SLIC Algorithm

Figure 9: sea house.jpg

Solution: For SLIC algorithm, we first need to initialize the cluster center, then get the pixels and connect the each part with correct label.

```
step = floor(sqrt(M*N/num_cluster)+0.5);
% Initial the cluster center
[cluster,num_cluster] = Initial_Cluster(step, M, N, R, G, B);
% get super pixels
[~,labels] = Get_Super_Pixel(cluster, num_cluster, M, N, step, corr, R, G, B);
% connect
k = floor(M*N/(step*step));
labels = Connect(labels, M, N, k);
```

Figure 10: Main steps of SLIC.

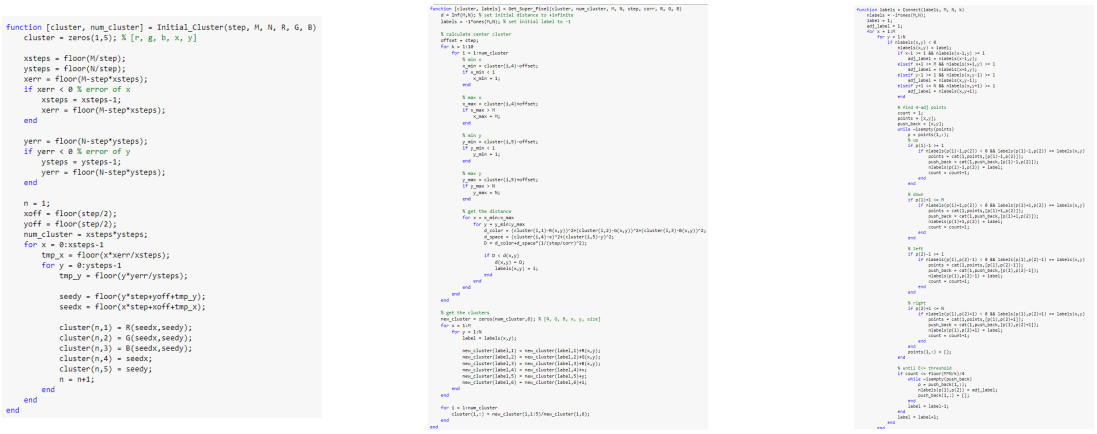

Figure 11: Code of each functions: (a) Initial_Cluster(), (b) Get_Super_Pixel(), (c) Connect().

At last, the results are shown below:

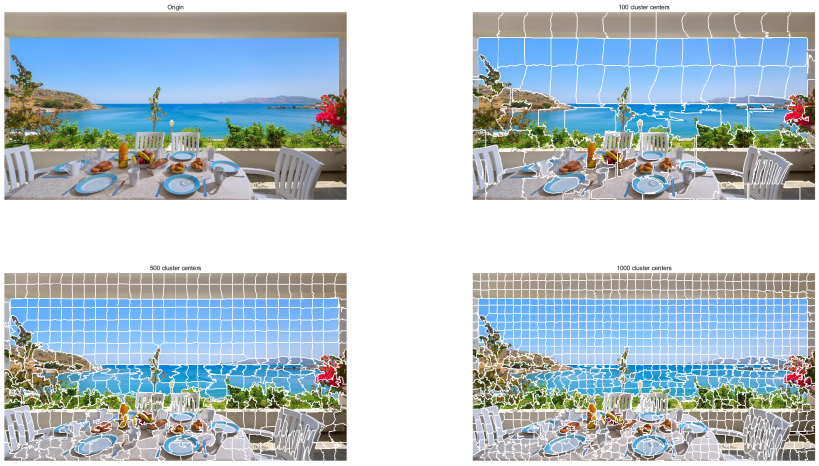

Figure 12: SLIC: (a) origin image, (b) 100 cluster center, (c) 500 cluster center, (d) 1000 cluster center.