CmpE 300 Programming Project
Fall 2016

Due: 27.12.2016 - 23:59

In this project you will implement a simple smoothing and line detecting algorithm using C programming language with the help of MPI (Message Passing Interface) libraries.

Smoothing and Line Detecting

We will give you a text file, which will be a 2D array representation of a grayscale image, as an input. This 2D array will contain integers from 0 to 255 (0 is black, 255 is white, and numbers going from 0 to 255 represent colors that change from black to white in gray tones). Also a python script will be given to you which reads a text file and shows the corresponding image using pillow library. An example grayscale image (5 pixels x 12 pixels) and its 2D integer array (5 rows, 12 columns) representation can be the following:

Image:  (The last line is white)

2D Array:

0	0	0	0	0	0	0	0	0	0	0	0
50	50	50	50	50	50	50	50	50	50	50	50
100	100	100	100	100	100	100	100	100	100	100	100
175	175	175	175	175	175	175	175	175	175	175	175
255	255	255	255	255	255	255	255	255	255	255	255

You should first smooth the given image by using a 3x3 mean filter (which takes the average of 9 numbers). This can be done via successive convolution steps of the 200x200 integer array (which will be the input size given to you) with a 3x3 double array. You can see the 3x3 double array to be used as a mean filter on the right. Doing convolution of this 3x3 matrix with another 3x3 matrix will give a single value.

1	2	3
4	5	6
7	8	9

$\times$

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

$= 5$

Let's name the first matrix's elements as B_{11}, B_{12}, B_{13}, B_{21}, B_{22}, B_{23}, B_{31}, B_{32}, and B_{33} (the first number corresponds to the row number and the second number corresponds to the column number). Name the second matrix's (mean filter's) elements similarly; M_{11}, M_{12}, M_{13}, M_{21}, M_{22}, M_{23}, M_{31}, M_{32}, and M_{33}. The calculation done for convolution is as follows:

$$B_{11} \times M_{11} + B_{12} \times M_{12} + B_{13} \times M_{13} + B_{21} \times M_{21} + \ldots + B_{33} \times M_{33} = R$$

If you look at the mean filter you will see that all the elements are 1/9. As you can see from the convolution formula this process takes the mean of the 9 values of the first matrix.

$$1 \times 1/9 + 2 \times 1/9 + 3 \times 1/9 + 4 \times 1/9 + 5 \times 1/9 + 6 \times 1/9 + 7 \times 1/9 + 8 \times 1/9 + 9 \times 1/9 = 5$$

Now let's see an example smoothing of a 5x5 matrix with a 3x3 mean filter matrix by doing successive convolution steps.

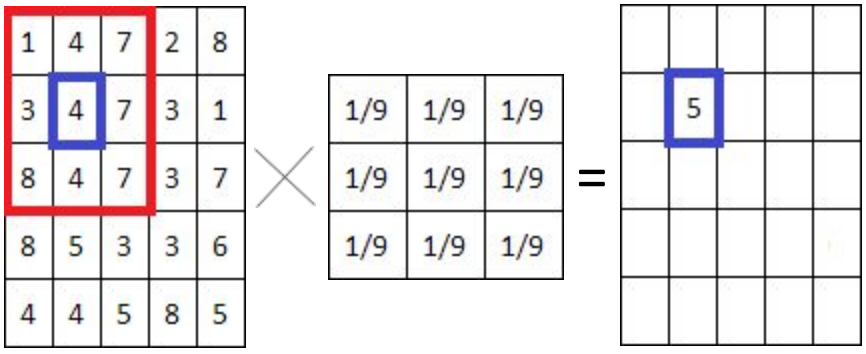
1	4	7	2	8
3	4	7	3	1
8	4	7	3	7
8	5	3	3	6
4	4	5	8	5

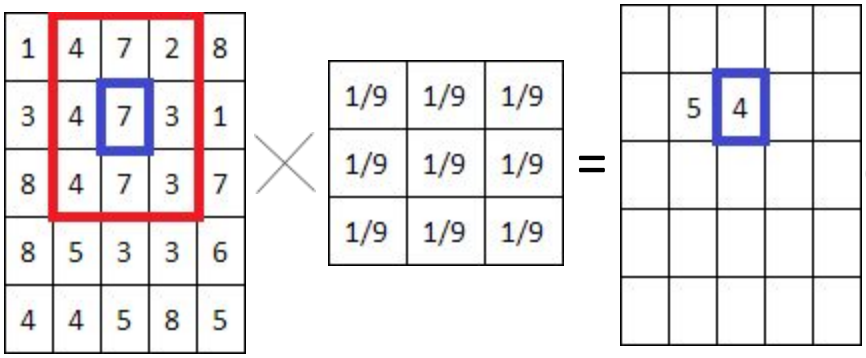
5x5 matrix A

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

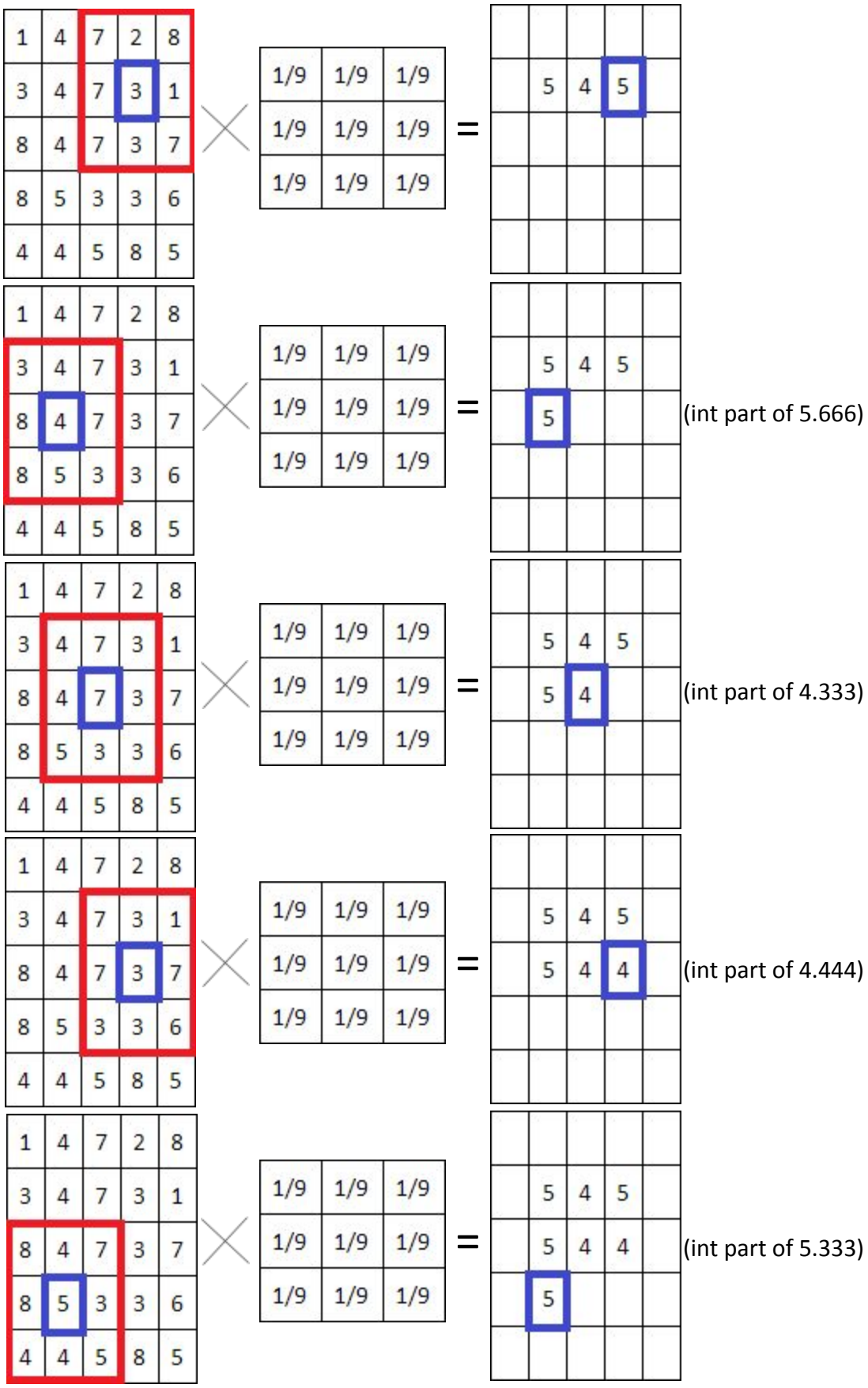
3x3 mean filter matrix

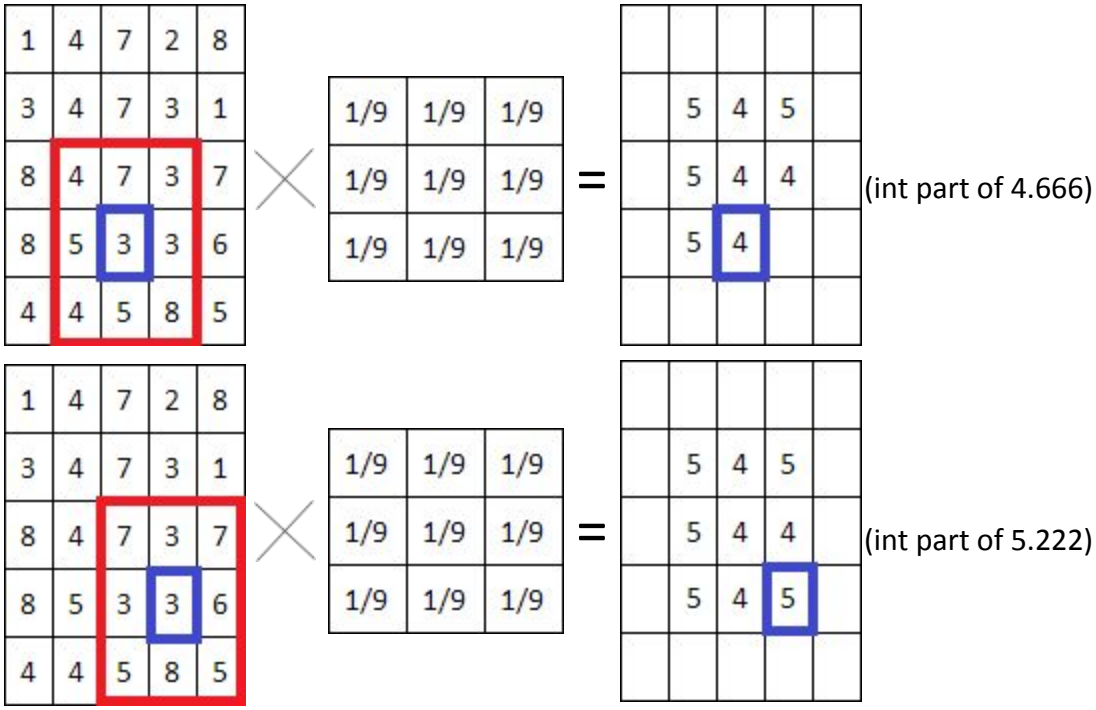
(Convolution of the 3x3 matrix inside the red rectangle with 3x3 mean filter matrix. The resulting number will be written in a new 5x5 matrix B)





(int part of 4.555)


1 4 7 2 8
3 4 7 3 1
8 4 7 3 7
8 5 3 3 6
4 4 5 8 5
1/9 1/9 1/9
1/9 1/9 1/9
1/9 1/9 1/9
5 4 5
5
(int part of 5.666)
5 4
(int part of 4.333)
5 4 4
(int part of 4.444)
5
(int part of 5.333)



At the end you see that only the 3x3 part of the resulting matrix is meaningful. **When you do smoothing to your 200x200 2D integer array you will get a 198x198 2D integer array as a result.**

Next you will do similar convolution procedures (as explained above) with the following four different 3x3 matrices.

a)

-1	-1	-1
2	2	2
-1	-1	-1

b)

-1	2	-1
-1	2	-1
-1	2	-1

c)

-1	-1	2
-1	2	-1
2	-1	-1

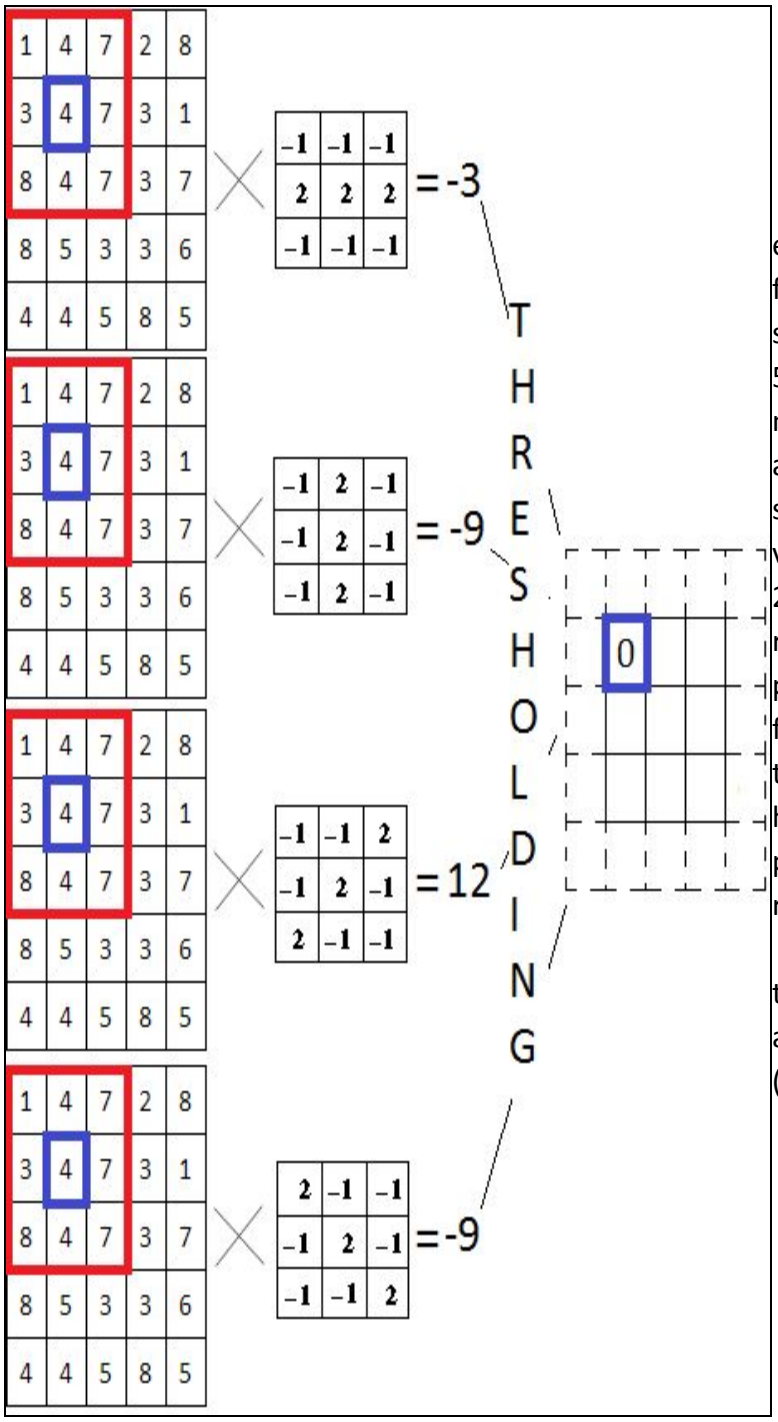
d)

2	-1	-1
-1	2	-1
-1	-1	2

These are the 4 filters for detecting lines in a given image. 'a' detects horizontal lines; 'b' detects vertical lines; 'c', and 'd' detects oblique lines (+45 and -45 degrees).

Just like smoothing procedure explained above, doing successive convolution steps of the horizontal line detector filter (a) on the smoothed image (**198x198** 2D matrix) will increase the pixel values of the horizontal lines in the image (written on a different **196x196** 2D matrix). After that process you will threshold the resulting 2D matrix to create a binary image (only black and white **196x196** image). **Thresholding:** All the values lesser than or equal to the threshold will be set to 0 (black), and all the values greater than the threshold will be set to 255 (white). You can see an example thresholding and a binary image on the following pages.

You should do successive convolution steps of all the four filters (a, b, c, d) **on the smoothed image** (198x198 2D matrix), and then create a single binary image (196x196 2D matrix) by thresholding together (if any of the four filters gives a value greater than the threshold set the value to 255, else 0).



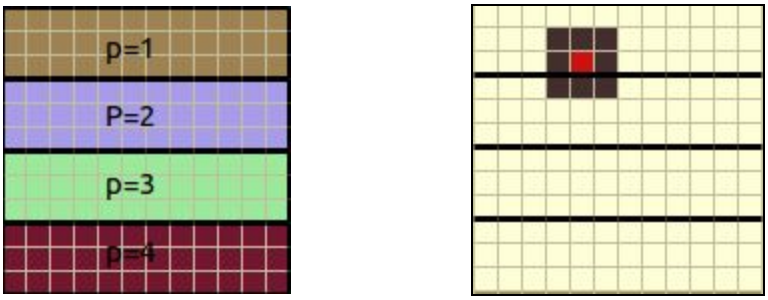
On the left, you can see an example application of the 4 line filters on a single 3x3 part of the smoothed image (assume that the 5x5 matrix on the left is a smoothed matrix of a 7x7 image). After applying (convolving) the four filters separately you will get 4 resulting values. Let's assume the threshold is 25, since none of the filters give a result greater than 25 the resulting pixel is set to 0 (if at least one of the filters had given a result greater than 25 the resulting pixel would have been set to 255). This procedure will continue until the resulting 3x3 2D matrix is filled.

You will apply this procedure to your 198x198 smoothed 2D array and get a 196x196 resulting 2D array (black and white image).

Left: 200x200 RGB (red green blue) building image **Middle:** Grayscale version of the building image (this is the visualization of the 200x200 2D matrix we gave you as an input) **Right:** Resulting binary image (196x196) after applying smoothing and then line detection (with threshold 25)

Parallel Algorithm for Smoothing and Line Detecting

We will assume that the image is a square of size ***n x n*** . There will be 1 master and p slave processors and we will assume that **n** is divisible by the number of slave processors **p**. Each processor is responsible from a group of **n/p** adjacent rows. Each processor works on n/p rows and **n/p*n** cells are stored locally by each processor.



We will explain an example run of smoothing followed by line detection for the case above (with 4 processors and 12x12 2D array as input). The master processor will read the text file and distribute the input equally to the 4 processors (3x12 2D array to each one). Each processor will begin with applying successive convolution steps with the mean filter for smoothing the array they have. For smoothing the boundary pixels, the processors need to communicate. For the example convolution on the right, the processor 1 needs to communicate with the processor 2 to get the 3 values on the boundary pixels to calculate a value for the red pixel for the new 2D matrix. Since the successive convolution steps we explained in the beginning of the project description creates a new 2D array with **(n-2)** height and **(n-2)** width, after smoothing we will be left with 10x10 2D array in total (the processors 1 and 4 will be left with 2x10 2D arrays, the processors 2 and 3 will be left with 3x10 2D arrays after the smoothing operation). After doing smoothing the processors will do 4 separate

filtering steps to detect lines with the given filters in the line detection section. For the example on the right, the processor 1 will get the 3 smoothed pixel values (outside of the processor 1's border) from the processor 2 and apply 4 different filters via convolution and have 4 different values for each convolution. Then by thresholding these values as explained above it will decide the red pixel's value (0 or 255) in the new 2D matrix (which will be the resulting binary image). At the end of these filtering steps the processors 1 and 4 will have 1x8 2D arrays, and the processors 2 and 3 will have 3x8 2D arrays. They will send these values back to the master processor, and the master processor will output the resulting 2D array (8x8 matrix).

Implementation

1. **Read and understand line detection:** http://homepages.inf.ed.ac.uk/rbf/HIPR2/linedet.htm
2. For the MPI environment and tutorials please visit the course page. (https://www.cmpe.boun.edu.tr/courses/cmpe300/2016/fall)
3. The text file, which will be a 2D array representation of a grayscale image must **only** be read by master processor and distributed to slave (rest) processors by the master processor. The whole array should not be stored in each processor locally.
4. Any functioning of the program regarding the whole program such as printing the output should be done by the master processor.
5. When two processors are exchanging information about the boundary cells, be careful to avoid deadlock. Before smoothing you need to share boundary pixels between the processors. But do not start the convolution of the line detector filters, before you made sure that all processors have completed the smoothing procedure. Then you will need to share these new smoothed pixels (on the boundaries) between the processors.
6. Apply three different thresholds (10, 25, 40), and visualize these outputs by using the python script. **Also include these 3 images in your report.**
7. The names of the input and output files will be given on the command line, as well as the number of processors as the following format

   mpiexec -n <Processors> <executable> <input> <output>

   Example: mpiexec -n 2 project.exe input.txt output10.txt
8. Keep in mind that your program will be tested with any 200x200 image with any number of processors that divides 200 without a remainder (2, 4, 5, 8, 10, 20, etc.).

Hint: A python script will be given to you which can be used to read text files and visualize the corresponding images, and also to convert any image to grayscale and output a corresponding text file. This is just to help you to visualize and to let you toy with your own images. Before starting any implementation, we suggest you to run this python script and visualize the "input.txt" to get a better grasp on the project.

Submission

1. The deadline for submitting the project is December 27, 23:59. The deadline is strict. We will then have demo sessions which will be announced later.
2. This is an individual project. Your code should be original.
3. Your code should be sufficiently commented.
4. You should write a header comment in the source code file and list the following information.

   ```
   /* Student Name: Ali Veli
    * Student Number: 2013123456
    * Compile Status: Compiling/Not Compiling
    * Program Status: Working/Not Working
    * Notes: Anything you want to say about your code that will be helpful in the grading
    * process.
   */
   ```

5. You must prepare a document about the project, which is an important part of the project. You should talk about the efficiency of the parallel approach when compared to the sequential one. Follow the guidelines given in the "Programming Project Documentation" link on http://www.cmpe.boun.edu.tr/~gungort/informationstudents.htm
6. Do not forget to comment on the different results of choosing different thresholds (10, 25, 40) in your report (include the resulting 2D arrays' visualizations).
7. Submit your project and document as a compressed archive (zip or rar) by sending an e-mail to metehan.doyran@boun.edu.tr with subject: "CMPE300:Project-<YourStudentNumber>". Your archive file should be named in the following format: "<YourStudentNumber>.zip" or "<YourStudentNumber>.rar". Your archive file should include your code (main.c), your project description, and the three output text files 'output10.txt', 'output25.txt', and 'output40.txt' for thresholds 10, 25, 40. Include the three line detected images with the thresholds 10, 25, 40 visualized by the python script we gave you in your project report.

Important: If you have any further questions, send an e-mail with the subject "[CMPE300 Project] <a brief explanation of your question>" and a well-structured question to cmpe300 mailing list. This way we will be creating a Q&A pool. **Please check the previously asked questions before asking any questions.** If you ask a previously asked question your e-mail will be ignored. **Please do NOT send any e-mails about the project to the assistants or the professor.** We will announce a PS date for the project details, and an introduction to MPI. You can also come to BM 37 on Tuesdays between 15:00-17:00.