**Running the Program**

Requirements:  
 1. Matlab

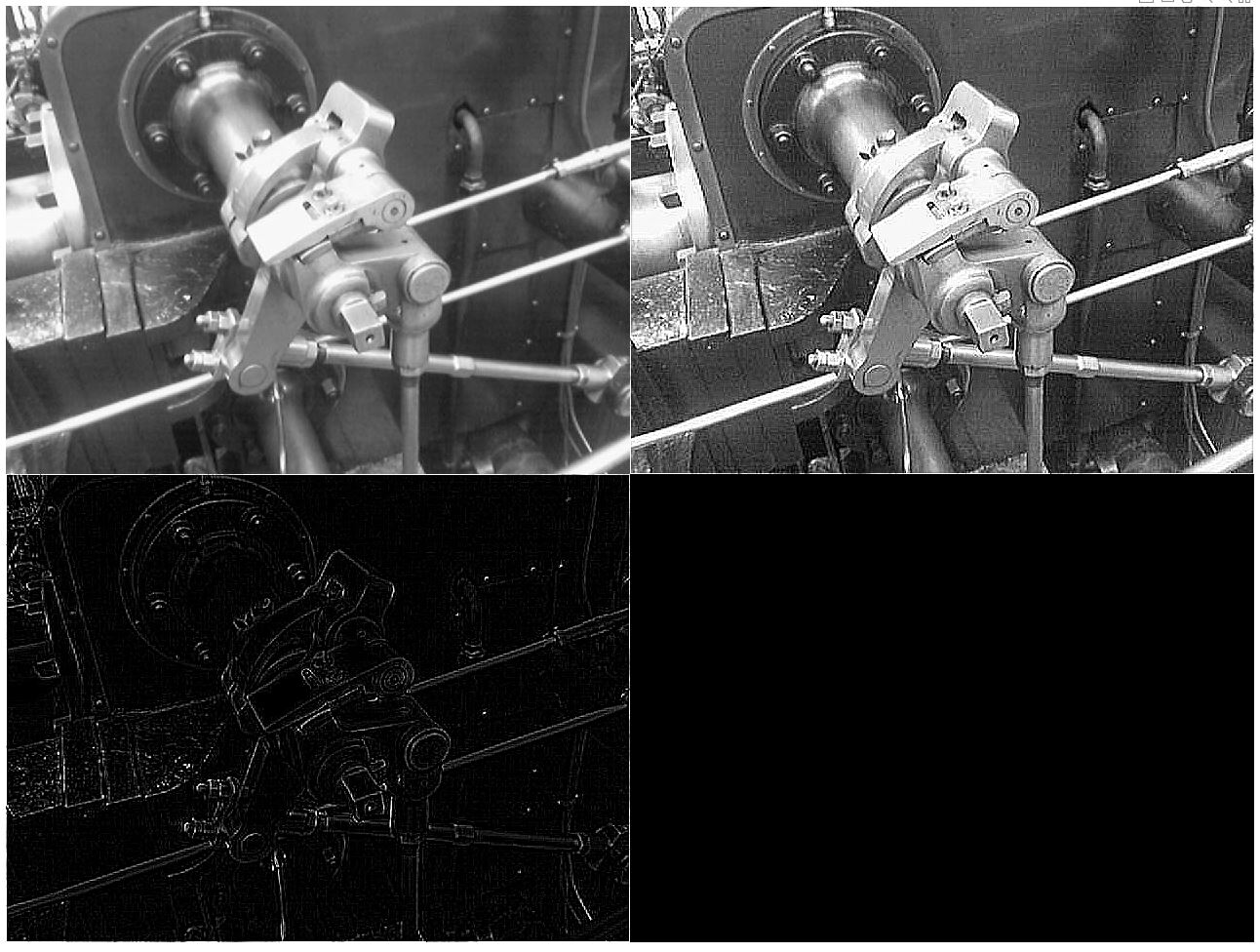
2. M5atlab Libraries

-None

Steps:

1. Open zip file and choose where to put the .m file
2. Make sure that there is a folder in the same directory names “pictures”.
3. Open matlab and make sure that the same directory where you put both the .m file and the pictures is selected in the directory bar.
4. If the code is in the Editor section, the main part we write in within matlab, and the directory is in the correct spot, showing both the “HomeworkOne.m” file and “sobel.png”, you may then run the program by pressing the green arrow.

Program Part1(filters):

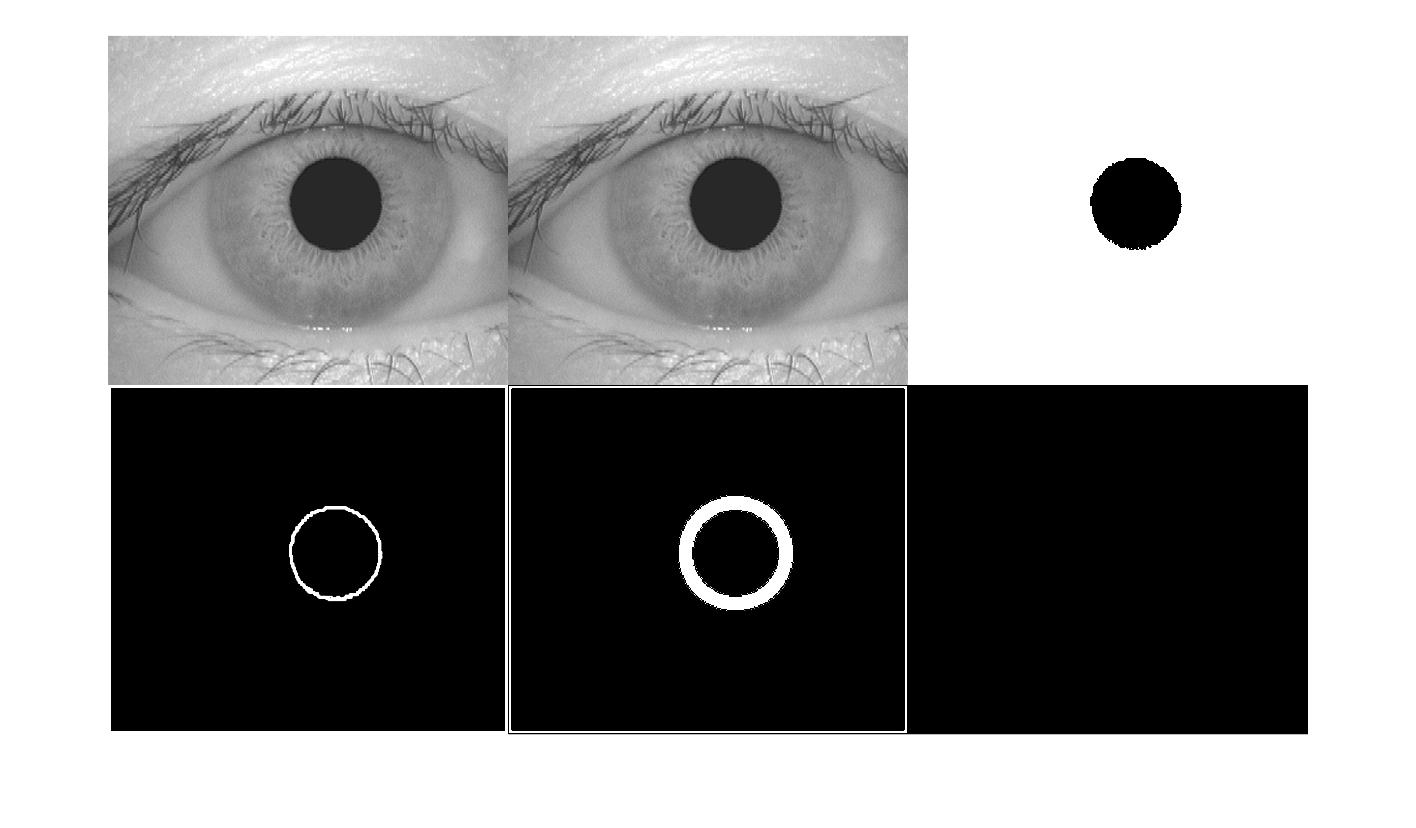
1. Get picture from the same directory as the .m file and give it a variable name “pic”.
2. Show the picture and then get the size information, specifically the number of color channels information to run through an if statement to determine if it’s already a gray image or not. If it’s not already gray, convert to grayscale with “rgb2gray”.
3. Afterwards take the gray image and send it to my function “avgFilter” to get the average filter of the gray scale. Inside the function the mask is created and then divided by nine to make it average throughout the mask. I then convoluted the gray scale image with the mask to get the average filtered image, and show an image of the result.
4. I then create a grayscale picture with salt and pepper noise to filter out with my median filter.
5. Send grayscale pic to “medFilt” which is the median filter to get salt and pepper noise out. Inside “medFilt” the program gets the size information. Then it pads zeros to use for a new image later. It then loops through the original image with salt and pepper, and filters out each pixel using variables “i” and “j”. Which are the center, and finds the median between the center and the 8 pixels that make up the neighborhood using the matlab function “median”. Then returns the picture and displays the Salt and Pepper image next to the cleaned up image.
6. Next the program sends the picture to the “sobFilter” function to find the edges within the image. Once the program is in the function, the program creates two kernels/masks. One is for the x direction while the other is in the y direction for detection. The program then convolutes the grayscale image with the x kernel first. Then convolutes the image with the y direction. Afterward the two output images are both squared respectfully and added. The summed result is then square rooted and returned back to the main and displayed.
7. Finally for part 1, the program sends the grayscale image to it’s final function “laplacianFilter” to get the laplacian filter on it. The first thing is that the kernel is created with the center being 8 and the rest of the neighborhood being -1. The image is then convoluted with the kernel. Afterwards, the result is added to the original picture to get a clearer image. The same thing is done again, except it's not added to the original to show that this filter tracts all the edges, which is better than sobel in my opinion. Finally, the program displays the original grayscale image, the sharpened image, and the edge detection that it really is all side by side as seen below.

Output Part 1:

1. Original
2. GrayScale
3. Average Filter
4. Montage of picture with salt and median filter picture
5. Sobel Filter Picture
6. Montage of Gray Scale picture, laplacian Sharpening, and laplacian filter.

Program Part 2(Iris):

1. Get the picture from the same directory as the .m file labeled “iris.bmp”.
2. Check if it’s a gray scale image already and if not, convert it to gray scale.
3. Then I did a histogram equalization to find a decent threshold point manually.
4. Used the threshold to get rid of everything and have just the pupil.
5. Then I masked with sobel found the edges to get the boundary, well, just about.
6. Then masked with the median filter to get rid of any small salt and pepper noise that I can and can’t see.
7. Then use both the thresholded image and the grayscale image to find the center of the circle do to the fact that these were the only two with the same sized matrixes.
8. Plot the circle in it’s own figure
9. Then do two for loops to show the boundary on the meanPic. First do the larger radius of 45 using the circle with center formula and make what it finds white. Afterwards, do the same thing with the radius of 35 but make it black, this shows the boundaries. Unfortunately, my program didn’t find the exact center of the meanPic, so I had to adjust center using my eyes.
10. Finally, montage original pic, grayPic, binaryImage, edgePic, meanPic.



Output Part 2:

1. Montage of original pic, grayPic, binaryImage, edgePic, meanPic.

Image Citation:

“Sobel Operator.” *Wikipedia*, Wikimedia Foundation, 21 July 2019, https://en.wikipedia.org/wiki/Sobel\_operator.