COURSEWORK REPORT

Report for coursework of Introduction to Image Processing

Junyu LIU (20216355)

This is the report for Introduction to Image Processing coursework. Firstly, some specifications of how to run the code is stated. Then, Detailed reports of task 1 and task 2 are explained. Note that all the filters/techniques that are not OpenCV built-in are referenced, and the reference list can be find at the end of this report.

How to Run the Code

In the zip file, all source codes and source images are included. In Task1.ipynb and Task2.ipynb, a main() function is written to call all other functions. Thus, by opening Task1.ipynb and Task2.ipynb, and compiling all code cells until the cell that contains main() function. Below that cell, there is a cell that writes:

```
if __name__ == '__main__':
    main()
```

Run that cell and the results can be shown. (Or call main() function.)

Report for Task 1

A. SNR Table:

The best SNR for each image and corresponding summary of processing techniques are presented in the table below. (Due to the limitation of formatting, the images presented in the table are small. Larger and clearer images are presented in *Image Comparisons* section.)

Image	Best SNR in DB	Brief summary of image processing techniques applied, including key parameters.
Image 1  Before Processing  After Processing	**18.7045**	**Step 1 – Apply Gaussian Blur filter** - **Reason:** The noise pattern in this image follows additive noise pattern (Gaussian noise). - **Function:** cv2.GaussianBlur(img, (kernel size, kernel size), 0) - **Parameter Choice:** - kernel_size: Filter kernel size = 9 is chosen. **Step 2 – Apply bilateral filter** - **Reason:** To preserve the edges in this image when blurred by Gaussian filter. - **Function:** cv2.bilateralFilter(img, d, sigmaColor, sigmaSpace) - **Parameter Choice:** - d: Filtering diameter = 9 is chosen. - sigmaColor: Filter sigma in the color space = 75 is chosen. - sigmaSpace: Filter sigma in the coordinate space = 75 is chosen. **Step 3 – Apply Laplacian Enhancement** - **Reason:** To sharpen the de-noised image to recover details. - **Function:** cv2.Laplacian(img, ddepth, kernel_size) - **Parameter Choice:** - kernel_size: kernel_size = 3 is chosen. - ddepth: ddepth = cv2.CV_16S is chosen.

Image 2  Before Processing  After Processing	**20.74192**	**Step 1 – Apply median blur filter** - **Reason:** The noise pattern in this image is mainly salt&pepper noise, which can be effectively de-noised by median filters. - **Function:** cv2.medianBlur(img, kernel_size) - **Parameter Choice:** - kernel_size: kernel_size = 3 is chosen. **Step 2 – Apply customized intensity transform[2]** - **Reason:** To boost the contrast of the image. - **Function:** customized function - intensity_transform(image, alpha, beta) - **Parameter Choice:** - alpha: 1.1 is chosen to boost the contrast. - beta: 0 is chosen since the brightness is fine.
Image 3  Before Processing  After Processing	**19.50334**	**Step 1 – Equalize histogram** - **Reason:** The contrast of this image is low compared and the image is too bright compared to the ground truth image. - **Function:** cv2.equalizeHist(img) **Step 2 – Apply Gaussian Blur filter** - **Reason:** The noise pattern in this image follows additive noise pattern (Gaussian noise). - **Function:** cv2.GaussianBlur(img, (kernel size, kernel size), 0) - **Parameter Choice:** - kernel_size: Filter kernel size = 3 is chosen. **Step 3 – Apply bilateral filter** - **Reason:** To preserve the edges in this image when blurred by Gaussian filter. - **Function:** cv2.bilateralFilter(img, d, sigmaColor, sigmaSpace) - **Parameter Choice:** - d: Filtering diameter = 9 is chosen. - sigmaColor: Filter sigma in the color space = 75 is chosen. - sigmaSpace: Filter sigma in the coordinate space = 75 is chosen. **Step 4 – Apply customized sharpening filter[1]** - **Reason:** To sharpen the de-noised image to recover details.

		- **Function:** customized sharpening filter - **Parameter Choice:** - kernel: [[0, -1, 0],[-1, 5, -1],[0, -1, 0]]

B. Justification of the technique combinations:

I first discuss the overall approach to identify certain image patterns and choose techniques in overall approach section. All the processing techniques choices follow the overall approach. Then I justify the technique combinations for each image respectively.

Overall approach:

1. Check whether the histogram of the noisy image has significant difference with the ground truth image. If there is significant difference, apply histogram equalization, otherwise the noisy image would have large bias in terms of contrast level or brightness level.
2. Identify the noise pattern in the noisy image and apply corresponding filters to de-noise.
3. Use edge preserving filters to preserve the edges of the blurred image to avoid too much blur on edges.
4. Use sharpening techniques to sharpen the de-noised image to recover more details.

Justification for technique combinations of image 1:

1. The noisy image has similar contrast and brightness in terms of histogram compared with ground truth image. Thus, there is no need to further equalize the histogram.
2. The noise pattern in this image follows additive noise pattern (Gaussian noise). Thus, gaussian blur filter is applied to reduce the noise. The kernel size is chosen to be 3, since this is the best size that can reduce enough noise while not blurring the image too much.
3. Bilateral filter is chosen to preserve edges since bilateral filter modifies Gaussian blur. Although anisotropic diffusion filter can also control the amount of smoothing, it is mainly related to mean filters.
4. Laplacian enhancement has strong response the fine details and is simple to implement. In addition, the image processed after Gaussian blur and bilateral filters are de-noised, which mitigate the weakness of sensitivity to noise of Laplacian enhancement.

Justification for technique combinations of image 2:

1. The noisy image has low contrast in terms of histogram compared with ground truth image. However, if equalize the histogram now, the salt & pepper noise would be enhanced as well. Thus, I choose to do intensity transform later.
2. The noise pattern in this image is mainly salt & pepper noise. Thus, median blur filter is applied to reduce the noise, since it is the most effective filter to deal with salt & pepper noise. The kernel size is chosen to be 11, since this is the best size that can reduce enough noise while not blurring the image too much.
3. Intensity transform [2] is used to boost the contrast since the image is lack of contrast. With experiments, alpha = 1.1 is the best parameter.

Justification for technique combinations of image 3:

1. The noisy image has low contrast and too much brightness in terms of histogram compared with ground truth image. Thus, histogram equalization is applied to boost the image contrast and equalize brightness.

2. The noise pattern in this image follows additive noise pattern (Gaussian noise). Thus, gaussian blur filter is applied to reduce the noise. The kernel size is chosen to be 3, since this is the best size that can reduce enough noise while not blurring the image too much.
3. Bilateral filter is chosen to preserve edges since bilateral filter modifies Gaussian blur. Although anisotropic diffusion filter can also control the amount of smoothing, it is mainly related to mean filters.
4. The image is still soft, thus, a customized filter [1] is used here to enhance the sharpness. With experiments, this filter performs better than the Laplacian enhancement.

C. Image Comparisons

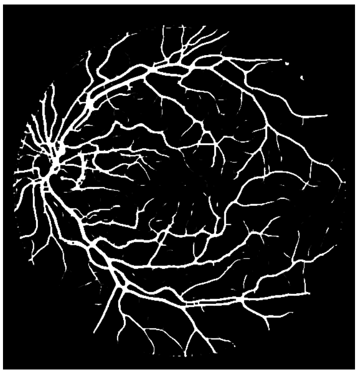
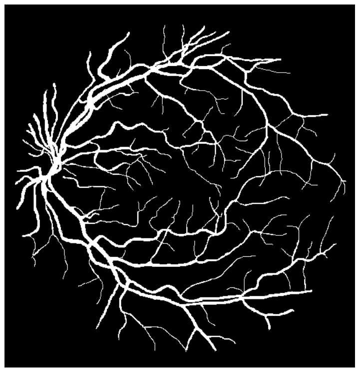
Large and clear comparisons of the original image, processed image and the ground truth image are presented here.

Image Number	Original Image	Processed Image	Ground Truth Image
Image 1			
Image 2			
Image 3			

Report for Task 2

A. Accuracy T:
The accuracy T provided by my algorithm is 95.31%

B. Segmentation result:
The comparison of original image, processed image, and ground truth image is provided here:

Original Image	Processed Image	Ground Truth Image
		

C. Explanation and justification of algorithm:

Algorithm Explanation:
My algorithm can be divided into two stages, pre-processing stage and feature extraction stage. In pre-processing stage, several techniques are used to 1). highlight and contrast the blood vessels from background; and 2). eliminate noise and un-even lighting, which provides a clearer and cleaner image for feature extraction stage. In feature extraction stage, several techniques are used to extract blood vessel features. The step-by-step explanation is provided as follows:

- **Pre-processing stage:**
  1. Separate the green channel of this original image.
  2. Apply Gaussian blur on the separated the green channel to reduce noise.
  3. Apply Contrast Limited Adaptive Histogram Equalization (CLAHE) to equalize images in terms of uneven lighting and low contrast.[3]
  4. Apply gamma adjustment to correct the brightness of the image.[4]

- **Feature extraction stage:**
  5. Apply gaussian match filter to extract the textures of blood vessels in different directions. The gaussian match filter calculates texture is 12 degrees and with a texture detection length L of 9.[3]
  6. Mask out the background using the mask.
  7. Apply grey stretching to enhance the contrast of only the extracted blood vessels. [3]
  8. Apply double thresholding [5] to reduce the false edges while preserving enough true (very thin) blood vessels.
  9. Apply morphological close calculating to fill in the gaps of un-connected holes of thin vessels.

Algorithm Justification:

Based on the algorithm explained above, a step-by-step justification is provided here to justify the rationale for each step. The number corresponds to the step number above.

- **Preprocessing stage:**
  1. By comparing the red, green, and blue channel of the original image, green channel shows the clearest segmentation of blood vessels. Thus the green channel is chosen to convert the original image to grey-level image.
  2. The grey level image contains noise, thus gaussian blur is chosen to reduce the noise. Among different parameter value settings, the kernel size of 5 is the optimal one.
  3. The image has very low contrast and uneven lighting (a dark center and bright eye center), thus Contrast Limited Adaptive Histogram Equalization (CLAHE) [3] is used to equalize images. CLAHE is a variant of Adaptive histogram equalization (AHE) which takes care of over-amplification of the contrast. The key feature of CLAHE is that it operates on small regions in the image (tiles), rather than the entire image. Thus, the contrast of blood vessels can be boost while the background contrast variation can be limited.
  4. The image is still dark after applying CLAHE, thus gamma correction is used to brighten up the image.

- **Feature extraction stage:**
  5. The directions of blood vessel textures are different across the image, thus gaussian match filter [3] is applied to extract features along different directions. Among different parameter value settings, detection angles of 12 degrees and the texture detection length L of 9 are the optimal setting.
  6. Current image suffers from distractions from background and the outline of eyeball, thus multiply the mask onto the image to remove background.
  7. After masking out the background, the image (especially blood vessels) is dark and lack of contrast, thus grey stretching [3] is used to enhance the contrast of blood vessels while keeping the background dark.
  8. Current image shows decent segmentation of bold blood vessels, however, some very thin vessels are missing and there are holes in very thin blood vessels. Thus, inspired by the processing in Canny Filter, double thresholding [5] is used to fill in the gaps of thin blood vessels while mitigating the false detected edges.
  9. Some tiny holes still occur in the image, thus, morphological close calculating (dilate first and then erode) is applied to fill in the gaps of un-connected holes of thin vessels.

D. Advantage and disadvantage of algorithm:

Advantage:
1. My algorithm provides an overall reasonable approach by dividing the whole processing into pre-processing stage and feature extraction stage.
2. In pre-processing stage, my algorithm provides a relatively comprehensive consideration for providing a cleaner and clearer image for feature extraction, by mitigating the issue of noise, un-even lighting and low contrast.
3. In feature extraction stage, my algorithm provides a reasonable combination of texture extraction and thin vessels repairment.

Disadvantage:
1. In pre-processing stage, although noise, un-even lighting and low contrast are mitigated, the un-even lighting is still quite observable (the bright eyeball center), resulting in obvious distraction in the processed image. Other approaches like using homo filters or equalizing histogram multi-times might provide better performance compared to the CLAHE used in my algorithm.

2. In feature extraction stage, although the majority of blood vessels are extracted, some very tiny vessels are either missing or have holes. Some possible alternative approaches are provided here:
   a. Use Gabor filter instead of Gaussian matched filter to extract the blood vessels in different directions. Also, the parameter choices can be further experimented to achieve better result.
   b. Use Laplacian enhancement to enhance the tiny blood vessels which are missing after step 7. Together with the Laplacian enhancement, further noise reduction, low/high pass filters and additional feature extraction might be used, since Laplacian enhancement is sensitive to noise and may introduce artefacts.
   c. Use multiple thresholding in grey stretching and morphological close calculation, to apply different level of enhancement on tiny and bold vessels, ensuring effective highlight for tiny vessels while preventing to over-highlight bold vessels.

Code references

[1] https://www.whcsrl.com/blog/1008390
[2] https://docs.opencv.org/3.4/d3/dc1/tutorial_basic_linear_transform.html
[3]https://blog.csdn.net/qq_40511157/article/details/102770108?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522164916485916780264050983%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=164916485916780264050983&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_ecpm_v1~rank_v31_ecpm-6-102770108.142^v5^pc_search_result_cache,157^v4^control&utm_term=%E6%B8%85%E9%A3%8E_1998&spm=1018.2226.3001.4187
[4] https://docs.opencv.org/3.4/d3/dc1/tutorial_basic_linear_transform.html
[5] https://theailearner.com/tag/hysteresis-thresholding/