Motion Deblurring for Car Plate Recognition

Shengrui Lyu
Hong Kong University of Science and Technology
slvaa@connect.ust.hk

Abstract

The report is for ELEC 4130 Moving Car License Plate Reading. The project can be separated into three parts, deblurring, car plate locating and digit recognizing. In this report I will introduce the method for the first part, deblurring.

1. Introduction

License Plate Recognition (LPR) became one of the focus on image processing sector. This was due to number of increases in vehicles in our society and its license plate is like the representation of its own identity. The security camera on road or parking lot could recognize the license plate image and store the data in the database. This can be used for locating the personal, tracking stolen cars,
The difficulty of LPR is that with the security camera of parking lot or highway, it is highly likely the cars are moving. It causes occurrence of motion-blur of the captured image. There must be some careful pre-process on the image to restore the original image. Our project research grouped worked on the LPR of a Chinese car license plate.

1.1. Deblurring

For motion deblurring, we need to detect the blurring direction and length to calculate the Point Spread Function (PSF) in order to apply Wiener filter. Firstly, I use the Fourier transform, histogram equalization, binarization and morphological method and Radon transform to calculate the blurring length. Next, I rotate the image to let the blurring length lie on the horizontal direction and detect the blurring length by the autocorrelation function. Finally, I calculate the PSF and apply the Wiener filter to recover the image.

1.2. Car Plate Segmentation

After we recover the blurred image, the next step is to locate the car plate and provide the segmented image to digit recognition module. This part is done by my teammate, Donghong Du.

1.3. Digit Recognition

After we get the clear car plate image, we are going to segment each character out and recognize each character by deep learning model. This part is done by my teammate, Lee Hyun Seung.

2. Implementation of Motion Deblurring

The deblurring is completely base on Matlab.

2.1. Blurred Images Generation

I used add_blur.m to add the motion blur, the blur length is a random value between 0 to 20. And the blur direction is a random value between 0 to 90.

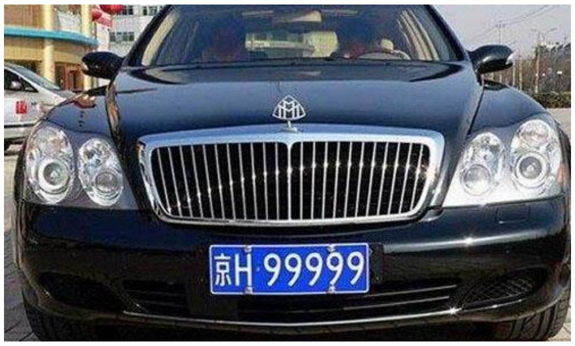

Figure 1: Original image(Blur Length: 15, Blur Degree: 30)

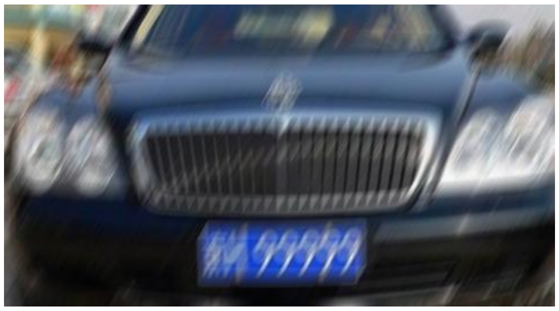

Figure 2 : After serious motion blur (blur length is 20 pixels; blur direction is 60 degree)

2.2. Blurring Equations

The blurring can be viewed as the following equation, where h is the Point spread function (PSF).

$$g(x,y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} h(x,\alpha,y,\beta) f(\alpha,\beta) d\alpha d\beta$$

The PSF can be written in the following form[1]:

$$h(x,y) = \begin{cases} \frac{1}{L} & if \ \sqrt{x^2+y^2} \le \frac{L}{2} \ and \ \frac{x}{y} = -\tan\varphi \\ 0 & otherwise \end{cases}$$

Which is a sinc function in frequency domain. And the dark gaps in the frequency domain is where the PSF has zero value, which provides the foundation for our blurring direction estimation.

2.3. Blurring Direction Detection

As mentioned in 2.2, the direction of motion blur in original image domain will result in the direction of the gaps in the frequency domain.

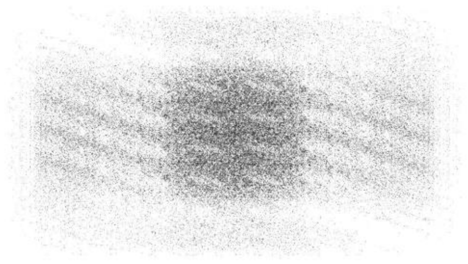

Figure 3: The frequency domain image of Figure 2

From figure 3, we can roughly see there is gaps in frequency domain along 60 degree, however, it is not very clear. We can use histogram equalizer to uniform the distribution of the pixel value, then apply an adaptive median filter to connect the graph and increase the signal noise ration of the frequency image.

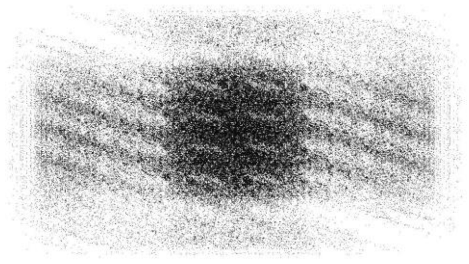

Figure 4: Histogram equalization of Figure 3

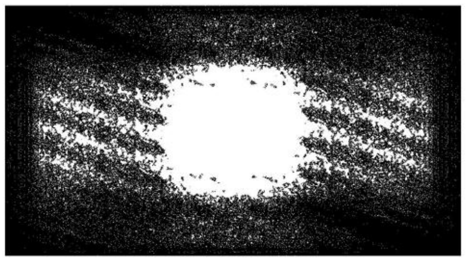

Figure 5: Figure 4 after adaptive median filter [2] and thresholding

To further reduce the noise white dots around the main frequency shape, I choose to apply an opening.

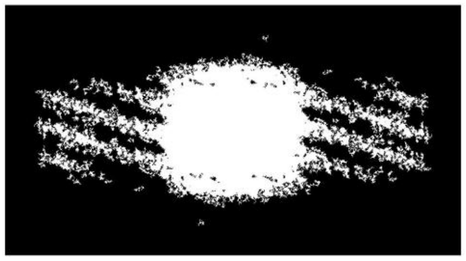

Figure 6: Figure 5 after opening

Up to this step, we can clearly see the frequency gaps from figure 6, next, to find the direction of the edges, I choose to use Randon transform.

However, the larger cluster located at the center of the frequency image may dominate the result of Randon transform, to solve this issue, I used the following method: Firstly, since the image is symmetrical along horizontal direction, I only use the left half of image. Then, I use a Canny edge detector to find the edge of the frequency image to better show the gap in frequency domain.

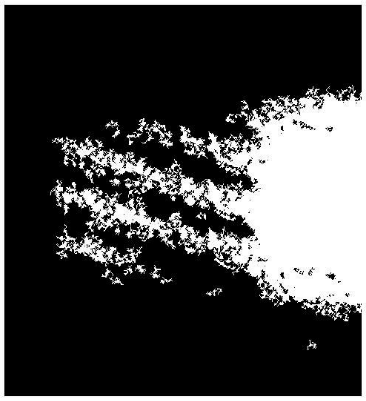
Figure 7: Half of the frequency domain in figure 6

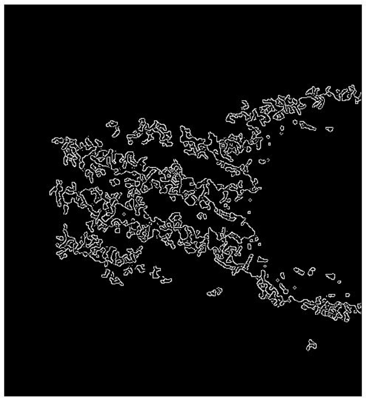
Figure 8: after canny edge detection on figure 7

Then we can apply Radon transform on the processed frequency image. The transform will calculate the integral of pixel value projected along directions from 0 to 90 degree. And the direction with largest integral is the direction parallel with the frequency gaps, which is the desired motion blur direction.

2.4. Blurring Length Calculation

Firstly, we rotate the image by the degree calculated from section 2.2, to let the motion blur purely in horizontal direction. Then we only choose the central part of the image to avoid the zero-paddings around the image due to the rotation.

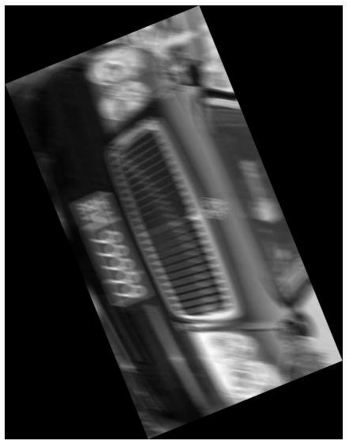
Figure 9: Example of a rotated plate mage

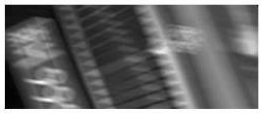
Figure 10: After cutting the surrounding part of figure 9

If the motion blur is along the horizontal direction only, then it can be viewed as: [3]

$$h(x) = \begin{cases} 1 & 0 \le x \le t_0 \\ 0 & otherwise \end{cases}$$

Therefore, if the components in the image do not have much correlation with each other, the autocorrelation of the blurred image will have the maximum value at x=0, and the minimum value at x=t0, which is the blurring length.

We can calculate the autocorrelation of the image itself to find the blurred length, i.e., the distance between the peak the valley.

In order to reduce the impact of other periodical patterns in the original image, such as the car wheel or the tree in the background, I choose the mode value of all rows to avoid the impact of special pattern in some rows. Additionally, I process the image with a patch to calculate the first difference, to reduce the impact of the large periodical patterns.

2.5. Motion Deblurring by Wiener filter

With the motion blur distance and direction, we can simply apply Wiener filter to deblur the image and suppress the noise.

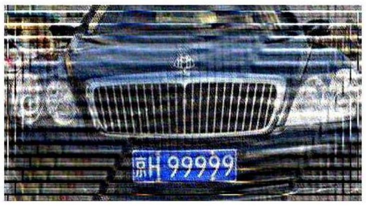

Figure 11: Deblurred image

3. Results and Conclusions

The evaluation of my part can be done in two ways.

Firstly, we can evaluate the parameter estimation accuracy. I calculate the relative error of blurring angle and length estimation for each of 90 images and compute the average value. Additionally, I also calculate the relative error for the 77 images whose car plate can be located in the next part of our group project.

For all 90 images:

Relative Blurring Length Estimation Error	11.6%
Relative Blurring Direction Estimation Error	1.5%

For 77 successful images:

Relative Blurring Length Estimation Error	4.8%
Relative Blurring Direction Estimation Error	1%

From those figures, we can find out that the failure of the motion deblurring for the 13 images is mainly because of the large blurring direction estimation error. Once the direction estimation has high error, the blurring length estimation will definitely fail. Additionally, since many of our images are of low resolution, it contains only several hundreds of pixels, which will negatively affect the blurring direction estimation.

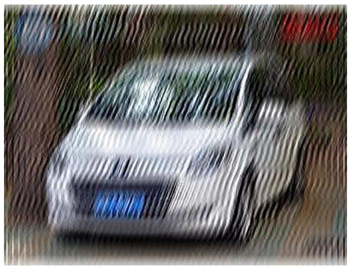
Figure 12: Example of a failure image due to wrong direction estimation

Secondly, my part is the first part of the project and the aim of my part is to assist the car plate segmentation and character recognition, it is reasonable to use the statistics of the following part to evaluate my result.

The result of car plate location: 77/90 = 0.855

The result of digit recognition:

Number of Input Photos	Rate of successful segmentation
40	0.625

Number of successful segmented digits	Rate of successful recognition
88	0.840909091

The advantage of my part is to find the motion blur length and direction automatically, which is an open area for a long time. The problem I solved including: effectively finding the motion blur direction by applying various image processing techniques, such as 2D Fourier transform, histogram equalization, adaptive median filter, canny edge detection and Radon transform; finding the motion blur length by using auto-correlation and using the statistics of the result to be more robust; applying Wiener filter with the parameters to achieve motion deblur.

From this project, I learned to apply the knowledge I learned from lectures into real-world applications, as mentioned in the paragraph above. Additionally, I realized how to address the practical issues occurred when applying different algorithms and how to compare and choose similar algorithms.

The limitation of this project is as following: Firstly, since the motion blur is manually generated in this project, so it has more randomness than the real-world cases. Therefore, the recovery image is not very satisfying viewed by human eyes. In the real-world application, since the camera has fixed specs and the road has fixed direction, the randomness of the motion blur will be lower, and it will be easier to estimate the blurring length and direction with prior knowledge. Secondly, the method may have slightly lower accuracy when dealing with white car deblurring comparing to the black car deblurring since the frequency image of white car has less bright points.

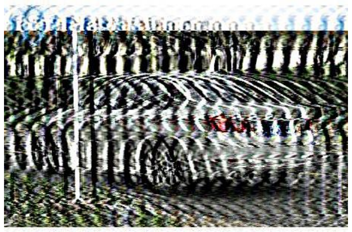

Figure 13: Example of a failure due to the white car (the blurring length estimation failure)

The future work includes reducing the computation time by using more effective algorithms, utilizing probabilistic models of the noise to achieve higher accuracy and shorter computing time and estimating the blurring distance and length with a probabilistic prior if it is used in real-world application.

References

[1] Mohammadi J., Akbari R. and Mehdi K. Vehicle speed estimation based on the image motion blur using RADON transform 2010

[2] Kintali K. Adaptive Median Filter (MATLAB Code) 2016 https://www.mathworks.com/matlabcentral/fileexchange/30068-adaptive-median-filter-matlab-code

[3] Yu S., Dang J. and Lei T. Motion blur parameters estimation based on frequency and spatial domain analysis 2012