CS 4299
Senior Project

Final Report

Enhancement of Photographs Taken with a Moving Smartphone

Submitted to

Project Advisor

Asst. Prof. Dr. Dobri Atanassov Batovski

Project Committees

Chayapol Moemeng
Asst. Prof. Paitoon Porntrakoon

Submitted by

Mr. Xiaohan Yu Adm. No. 571-8303
Mr. Jian Pan Adm. No. 572-8309

Assumption University of Thailand
May, 2018

Assumption University of Thailand

Vincent Mary School of Science and Technology

Department of Computer Science

Project Title:	Enhancement of Photographs Taken with a Moving Smartphone	
Presented by:	Mr. Xiaohan Yu	Adm. No. 571-8303
	Mr. Jian Pan	Adm. No. 572-8309
Project Advisor:	Asst. Prof. Dr. Dobri Atanassov Batovski	
Project Committees:	Chayapol Moemeng	
	Asst. Prof. Paitoon Porntrakoon	

Academic Year: 2/2017

The Department of Computer Science, Vincent Mary School of Science and Technology, Assumption University of Thailand, had approved this Senior Project Proposal of the three-credit course, CS 4299 Senior Project, for the Bachelor Degree of Science in the major of Computer Science (BS CS).

Approved by:

_____

Asst. Prof. Dr. Dobri Atanassov Batovski

Project Advisor

_____	_____
Chayapol Moemeng	Asst. Prof. Paitoon Porntrakoon
Committee Member	Committee Member

_____

Piyakul Tillapart

Program Director, BC CS Program

Abstract

The enhancement of blurred photographs is a challenging task in digital image processing. In some cases, the motion blur can be reduced in order to recognize symbolic content, which is of particular interest in investigative journalism and forensic science. The design and testing of an algorithm for selected practical cases should result in a smartphone app for real-time deblurring.

Table of Contents

Titles **Page**

Table of Contents

Titles

Page

Table of Contents

List of Figures

Titles

Page

List of Tables

List of Abbreviations

PSF	Point Spread Function
RL	Richardson-Lucy
MAP	Maximum Posteriori
MLE	Maximum Likelihood Estimator
EM	Expectation Maximization

Chapter 1 Introduction

1.1 Introduction

The digital cameras have massive used in the modern daily life in worldwide nowadays, for example, car-DVR, Closed-circuit television, especially ubiquitous smartphones, it provides amount of useful cases for investigative journalism and forensic science. However, as lots of unpredictable situations and/or deliberate actions, the photograph that be provided are usually ill-pose images, blurred images in the other words. Image blur is caused either by the camera motion or by the object motion [1]. To solve this problem, the mode is usually employed as a linear degradation process

$$I = L \otimes \mathrm{f} + \mathrm{n}, \tag{1.1}$$

where I is deblurred image, L is unblurred image, n is noise. $\otimes$ is the convolution operator and f is an unknown linear shift-invariant point spread function (PSF). The noise and PSF are of fundamental importance in restoration and enhancement of the blurred and noise images.

The field of image deblurring focus on reconstruction or estimation of the PSF of the blurred images. There are various techniques can be found as plenty of researches, such as the Lucy-Richardson method, the Van Cittert's itration, the Wiener Filter etc. Since the development of machine learning, the neural networking technique is more and more popular in this field.

The goal of this project is that try to find a decent way for employing known techniques to capture key information using a smartphone in real-time, particularly, using iPhone or iPad. Because of its unity of products, IOS-device is a better platform for testing.

Chapter 2 Project Overview and Background

2.2 Project Overview and background

Figure 1 source: blurred road sign clue to saving girl, published by CNN, Jun 19, 2013.

https://edition.cnn.com/2013/06/19/us/cfp-us-race-against-time/index.html In the news policemen confused the number on sign was 203 or 200. **"**So Cole and the other agents got back on the road and started driving **every highway in Kansas that starts with a 2."**

This project roots CNN news as shown in Figure 1. The story highlights show

1.Federal agents raced against clock trying to save girl in internet image,

2.Using high-tech kit, they found a single clue, tracked it down and saved the girl

3. Homeland Security Investigations use combination of computers and old fashion detective work.

Obviously, the "high-tech" is not quite high-tech.

Following the technology of image processing and computer version development, now we need not to" go back on the road and started driving every highway in Kansas that stats with a 2".

This project does not try to give perfect restoration of blurred image, instead of trying to find the key information that people want to know. Especially, providing important evidences and clues investigative journalism and forensic science.

The project through investigating classic algorithms using some criteria and implementing these algorithms in Mathematica, at first. Then try to adopt them to IOS. The most important and difficult point is using very limited efficiency to fix a quite dreadful problem. The balance between performance and running time on smartphones is matter of sovereign importance in this project.

Figure 2 the image using our method to restore with Mathematica in 1.36 seconds

Chapter 3 Literature Review

3.1 Image Deblurring and non-Blind deblurring

The whole family of deblurring methods is divided into two types: blind and non-blind deblurring. [2] Early works about image deblurring focuses on non-blind deblurring, assuming the blur function (point spread function, or PSF) is known. A lot of them rely on the classic algorithm, Lucy-Richardson, Wiener or Tikhonov filter to perform the deconvolution operation and obtain blur function estimate. As the PSF is not known, a considerable amount of research has been dedicated to the estimation of the PSF from the image itself. [1972-1974], as the two papers [11] and [12] published, a method which computes the blurred image with the assumption that its pixel intensities conform to a Poisson distribution. [13],Donatelli et al. [2006] use a PDE-based model to fix a blurred image with reduced ringing by incorporating an anti-reflective boundary condition and re-blurring step [2]. Most of these papers lack experiments about real images, and lots of them attempt to use proprietary programming languages, examples include MatLab, Wolfram language. Especially, few of them attempt to model error in the estimated kernel. Most non-blind deconvolution methods assume that the kernel with no errors (noise).

3.2 Blind deblurring

Before 1996-1998 the general belief was that blind deconvolution was not just impossible, but that it was hopelessly impossible. [2] The main difficulty in solving blind deconvolution is that the problem is ill-posed, so most of the existing algorithms rely on image heuristics and assumption on the sources of the blur.

[6], provided that the motion is shift-invariant, at least locally, and the blur function (PSF) that caused the blur is known.

These field of methods addresses the blur caused by camera shake by considering blur to be uniform across the image [2].

First the camera motion is estimated in term of the induced blur kernel, and then the effect is reversed by performing a deconvolution operation.

The deblurring can be posed as a convex optimization problem. Neither the kernel nor the sharp image is known. We need to recover both the blur and the sharp image. The problem is non convex. Use an alternating minimization algorithm.

However, [5] as illustrated by their examples, the blur kernels induced during camera shake do not have simple forms, and often contain very sharp edges. After 2006, as starting with the success of Fergus et al.[5], using an assumption on the statistical property of the image gradient distribution to approximate the unblurred image, a lot of derivative methods has been developed over the last ten years [7] [8] [9] [10]. [5], this brilliant paper solved this intractable problem. That is a milestone in image processing field. Their result is better than what the popular image processing software done, for example, Photoshop's "unsharp mask". And their appendix with pseudo code and the implementation with MatLab codes are the foundation stone of image deblurring, many related researches are based their pseudo code and implementation. However, the method based on iterative approach, which improve the estimate of the motion kernel and sharp image on each iteration by using parametric prior models. Obviously, the running time is a significant issue, as well as the stopping criterion.

3.3 Convolutional neural networks

With the success of deep learning, over the last few years, there appeared some approaches based on convolutional neural networks (CNNs). [16] use CNN to estimate blur kernel, [17] use fully convolutional network to move for notion flow estimation. Such previous methods use CNN to find the unknown blur kernel.

Recently, [18] provide a method about a kernel-free end-to-end approaches.

[2] present an approach, which is based on conditional GAN and content loss. The quality of the deblurring model is also evaluated in a novel way on a real-world problem – object detection (YOLO) on blurred images. It provides models, training code and dataset.

3.4 The approach with hardware

The approaches to solve motion blur problem with hardware, which are more general than software methods. The first approach uses optically stabilized lenses for camera shake compensation. These lenses have an adaptive optical element, which is controlled by inertial sensors, that compensates for camera motion [14].The second approach uses specially designed CMOS sensors. These sensors prevent motion blur by selectively stopping the image integration in areas where motion is detected. However, it does not solve the problem of motion blur due to camera shake during long exposures.

[15] utilize inertial sensors (accelerometers and gyroscopes) in modern smartphones to detect exact motion trajectory of the smartphone camera during exposure and remove blur from the resulting photography based on the recorded motion data even during long exposure.

Chapter 4 Methodology

In this part, comparison about the methods that previous paper provided except CNN and hardware approach, because they are not the goal of this project.

4.1 The deconvolution

Mathematically, convolution is operation on two functions to produce a third function. It is an integral that express the amount of overlap of one function g as it is shifted over another function f. It therefore "blend" one function with another function. (Bracewell 1965, p. 25) with the variable implied, and also occasionally written as

$$f \otimes g = h \tag{4.1.1}$$

Deconvolution, mathematically, is an algorithm- based process used to reverse the effect of convolution.

Physical measurements, the situation is usually closer to

$$(f \otimes g) + \varepsilon = h \tag{4.1.2}$$

in this case ε is noise that has entered our recorded signal.

According to Fourier Transform

$$F(\omega) = \mathcal{F}|f(t)| = \int_{-\infty}^{\infty} f(t)e^{-iwt}\,dt \tag{4.1.3}$$

Using Mathematica to show how Fourier Transform work:

$$\text{FourierTransform}[\text{Sin}[10\text{^}3t]\text{Exp}[-\text{t}10]\text{UnitStep}[t], t, w]$$

Deconvolution is usually performed like this:

$$F = \frac{H}{G} \tag{4.1.4}$$

F, G,and H being the Fourier Transform of f, g, and h respectively. If we plus noise, the system will be:

$$y(t) = (\mathrm{h} * x)(t) + n(t) \tag{4.1.5}$$

Where * denotes convolution and:

$x(t)$ is some original image at time t.

$\mathrm{h}(t)$is unknown image kernel.

$n(t)$is the noise.

$y(t)$is known blurred image.

4.2 Richardson-Lucy

This algorithm can be stated like this:

According to the deconvolution formula with Fourier Transformation:

$$HF = G \tag{4.2.1}$$

We can rewrite the above formula to:

$$\sum_j h_{ij} f_j = g_i,\ h_{ij}, f_j, g_j > 0. \tag{4.2.2}$$

Where h_{ij} is the PSF, f_j is the pixel value at location j in the blurred image, and g_i is the observed value at pixel location i.

Following the Richardson-Lucy's paper, the f_i is assumed to a Poisson distribution, then rewrite the formula to:

$$f_i = f_i \sum \left(\frac{h_{ki} g_k}{\sum_k h_{kj} f_i}\right) \tag{4.2.3}$$

Obviously,

$$f^{n+1} = f^n H^* \left(\frac{g}{H f^n}\right) \tag{4.2.4}$$

Here H^* is the adjoint operator of H. g/Hf^n denotes the vector obtained by component wise division of g by Hf^n.

For programming implementation easy, this can be written more generally (for more dimensions) in terms of convolution,

$$f^{t+1} = f^t * \left(\frac{g}{f^t \otimes \mathrm{h}} \otimes \hat{\mathrm{h}}\right) \tag{4.2.5}$$

Where $\hat{\mathrm{h}}$ is the flipped PSF (point spread function), such that

$$\hat{\mathrm{h}}_{mn} = h_{(i-n)(j-m)}, 0 \le n \le i, 0 \le n \le j, 0 \le m \le i, 0 \le m \le j$$

As we can see,

Richardson-Lucy algorithm does not focus on noise, obviously, when the amount noise issue is faced, this algorithm works not well.

4.3 Wiener Deconvolution

Our goal is trying to find

$$x(t) = (g * y)(t) + n(t) \tag{4.3.1}$$

Where $x(t)$ is an estimate of $x(t)$ that minimizes the mean square-error.

The Wiener deconvolution filter provides such a $g(t)$. The filter is most easily described in the frequency domain:*

$$G(f) = \frac{H*(f)S(f)}{|H(f)|^2\, S(f)+N(f)} \tag{4.3.2}$$

$$\mathrm{X(f)} = \mathrm{G(f)Y(f)} \tag{4.3.3}$$

The operation of the Wiener filter becomes apparent when the filter equation above is rewritten:

$$G(f) = \frac{1}{H(f)}\left[\frac{H(f)^2}{|H(f)|^2 + \frac{N(f)}{S(f)}}\right]$$

$$= \frac{1}{H(f)}\left[\frac{H(f)^2}{|H(f)|^2 + \frac{1}{SNR(f)}}\right]$$

As mentioned above, we want to produce an estimate of the original signal that minimizes the mean square error, which may be expressed:

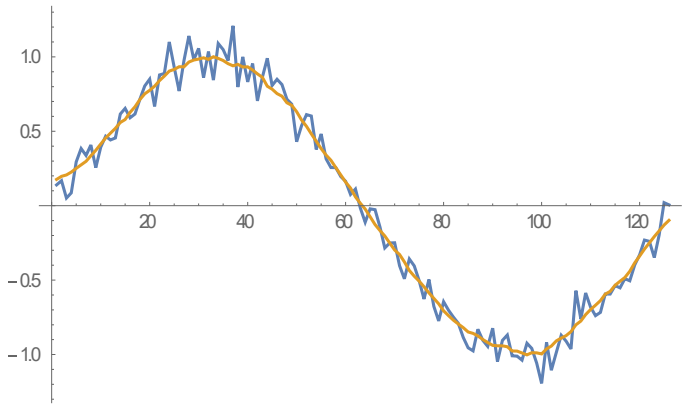
$$\varepsilon(f) = \mathbb{E}|X(f) - X(f)|^2 \qquad (4.3.4)$$

**the formula derivation and interpretation come from [21]*

Not like RL, Wiener Filter is such a noise friendly method. To use Mathematica denoise with Wiener Filter:

```
data=Table[ Sin[x],{x,0,2 Pi, 0.05}];
noisy=data+0.1*RandomReal[NormalDistribution[],Dimensions[data]];
filtered=WienerFilter[noisy,6,0.1];
ListPlot[{noisy,filtered},Joined->True]
```

The output will be:



4.4 Experiments in Mathematica

In this part the experiments in Mathematica of the deblurring algorithm.

4.4.1 Some preparations

In digital image processing, we can simply think the image is a 2-D array, so the kernel can be simulated by Mathematica.

At first, we built an image with a 2-D array

$$Img = Table[If[x^2 + y^2 < 20^2, 0.5, 0], \{x, -40, 40\}, \{y, -40, 40\}];$$

Then the image will be blurred by Gaussian Filter (kernel of pixel radius 4) with Mathematica:

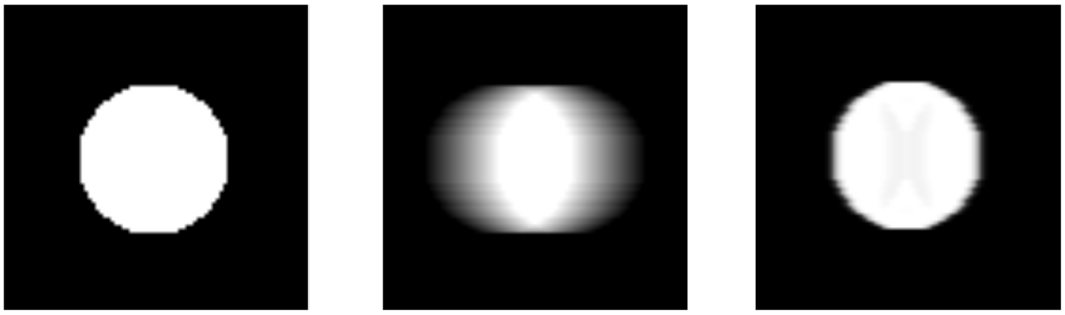
$$GauusianFilter[image[img], 4]$$

4.4.2 Richardson-Lucy

Now, deblurred image will be deployed by Richardson-Lucy algorithm in Mathematica:

```
RichardsonLucy[img_?MatrixQ, psf_?MatrixQ, maxIter_?NumberQ] :=
  Module[{res, mid, i}, mid = Ceiling[Dimensions[psf] / 2];
    = im;
   For[i = 1, i ≤ maxIter, i++,
    res = res ListConvolve[psf, im / ListConvolve[psf, res, mid], mid];];
   res];
```

The result is after 15 iterations: figure 1



a. original b. the blurred c. restored image

Figure 3 show the result of experiment in Mathematica a) is an original image. b) is the result after convolution with blurred kernel. c) is the deblurred image with Richardson-Lucy Algorithm after 15 times.

4.4.3 Wiener Algorithm

we deploy the same method in above image with Wiener Filter Algorithm: figure 2

Figure 4 shows the result of restoration with wiener filter is worse than Richardson-Lucy.

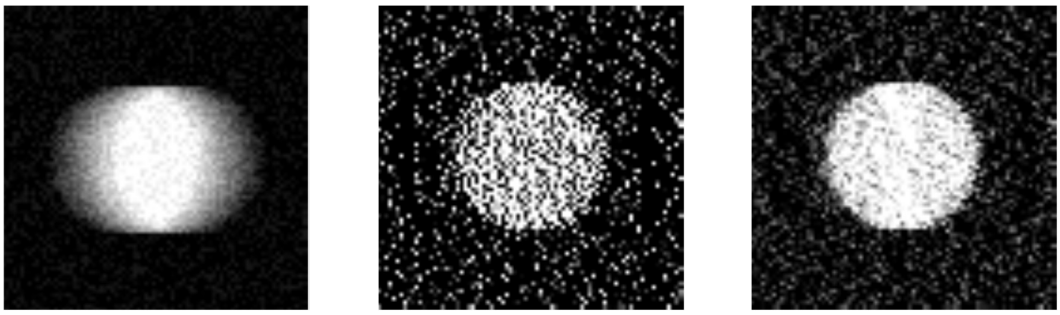
However, Richardson-Lucy algorithm is quite sensitive with noises. So the next step will show the performance about motion blurred image with noise in Richardson-Lucy Algorithm and Wiener Filter.

At first, adding noise to the motion blurred image with Mathematica.

```
ImageEffect[MotionBlurredImage, {"Noise", 0.1}]
```

Then, applying Richardson-Lucy and Wiener to noised image, respectively.

Show the result in Figure 3



a. noised b. RL deblur c. Wiener deblur

Figure 5 shows the result of restoration with Mathematica. a) is noised image with built-in noise function b) is the deblurred image with Richardson-Lucy Algorithm c) is the deblurred image with Wiener Filter. Obviously, image c is better than image b, though RL algorithm's performance is more better than Wiener Filter in Figure 1 and Figure 2.

4.4.4 Other algorithms

The project also implements some other algorithms and considers some results worked by [20], for example, Poisson MAP,Van Crittert, Landweber with Mathematica. These can be find in Appendix A.

Chapter 5 Results and Analysis

5.1 Evaluation and Analysis

At last chapter we discussed RL and Wiener Filter, we using observation find that RL works well in noise-free motion kernel deblurring, however, in noise environment, Wiener Filter is better choice. To prove this observation, we still need solid scientific evidence.

An image histogram is a type of histogram that acts as a graphical representation of the tonal distribution in a digital image [22].

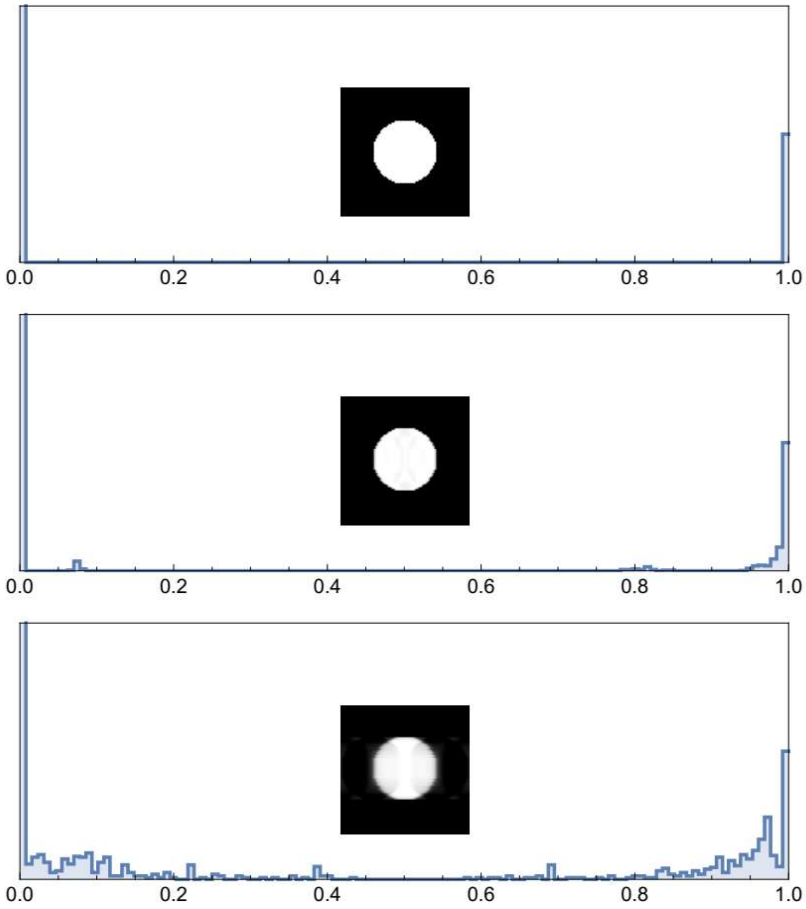


Figure 6, the histogram of original image, RL-blurred image, and Wiener Filter deblurred image. We can find the histogram of RL-blurred is better than Wiener's.

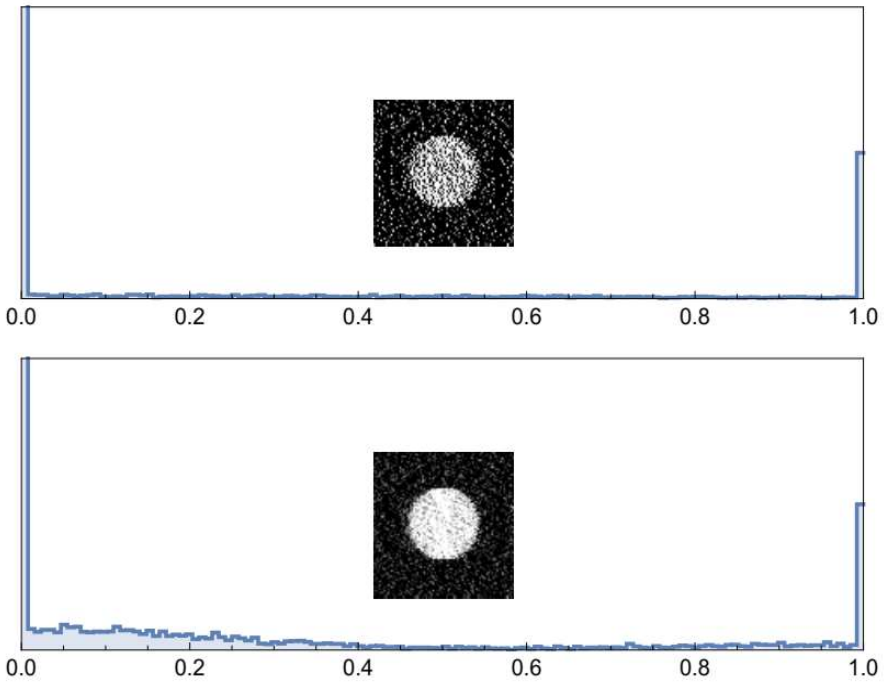


Figure 7, shows the restorations of a motion-blur image with noise.

By looking at the histogram for a specific image a viewer will be able to judge the entire tonal distribution at a glance.

From Figure 6, we find our obversion is correct, RL method is more better.

But, Figure 7, makes us quite surprised. This is unexpected. Obviously, the bottom one is more better, however, its histogram does not accept our viewpoint. We must understand image histogram plots the number of pixels in the image (vertical axis) with a particular brightness value (horizontal axis). The bottom one (Wiener Filter) can denoise, it makes many bright-noise be grey. That is the reason that why we can find the grey pixels become more. The Top one exactly proof RL does not have powerful ability to solve blurred image with lots of noise.

As this project will be applied in mobile phone, so the appropriate run-time and efficiency is very important. The experiment above use a very simple and small image, however, in the real world the image is more complicate and big. Following the mobile devices development, the camera embed on the cell phones can capture very huge pixel picture, even 4K.

The image that the is provided by the news will be used for evaluation.

we iterated the RL and Wiener Filter using that image from 1 to 100 times, then comparing the running time and quality of results.

Iteration	1	5	10	15	20	25	30	35	40	45
Second	0.16	0.48	0.92	1.36	1.78	2.3	2.69	3.09	3.55	4.11

Iteration	50	55	60	65	70	75	80	85	90	100
Second	4.55	4.88	5.3	5.8	6.16	6.66	7.11	7.66	8.08	8.89

Table 1 shows the relationship between iteration and running-time. This test-data base on a desktop that built in CPU: Intel(R)_Pentium(R)_CPU_G4400_@_3.30GHz, RAM: 8.00GB, OS: Microsoft(R) Windows 10(R).

Figure 8 shows the results about iteration 1 time, 10 times, 50 times and 100 times, respectively.

According to Figure 8 and the Table 1, we can find more iterations are not a good idea, obviously. The more iterations, the more times. Thought from naïve thinking, the more iterations should make more better and precise result, the Figure 8 shows the result is not like what we thought.

Obviously, 10 iterations are a suitable choice, for both its result and running time.

5.2 Implementation on IOS

We choose IOS as the test platform and use OpenCV as the software API.

For OpenCV does not provide a motion built-in function. We design our own Motion Blur function as estimating the blur kernel. we also implement the Richardson-Lucy Algorithm and Wiener Filter by ourselves.

This application not only provide image deblurring function, but also provide basic and popular functions about image processing for users, such that rotation, Resizing, sharpen, Image Denoise, Negative Image, Colour Covert, Gaussian blur and some filters, etc.

The programming language is Swift and Object-C.

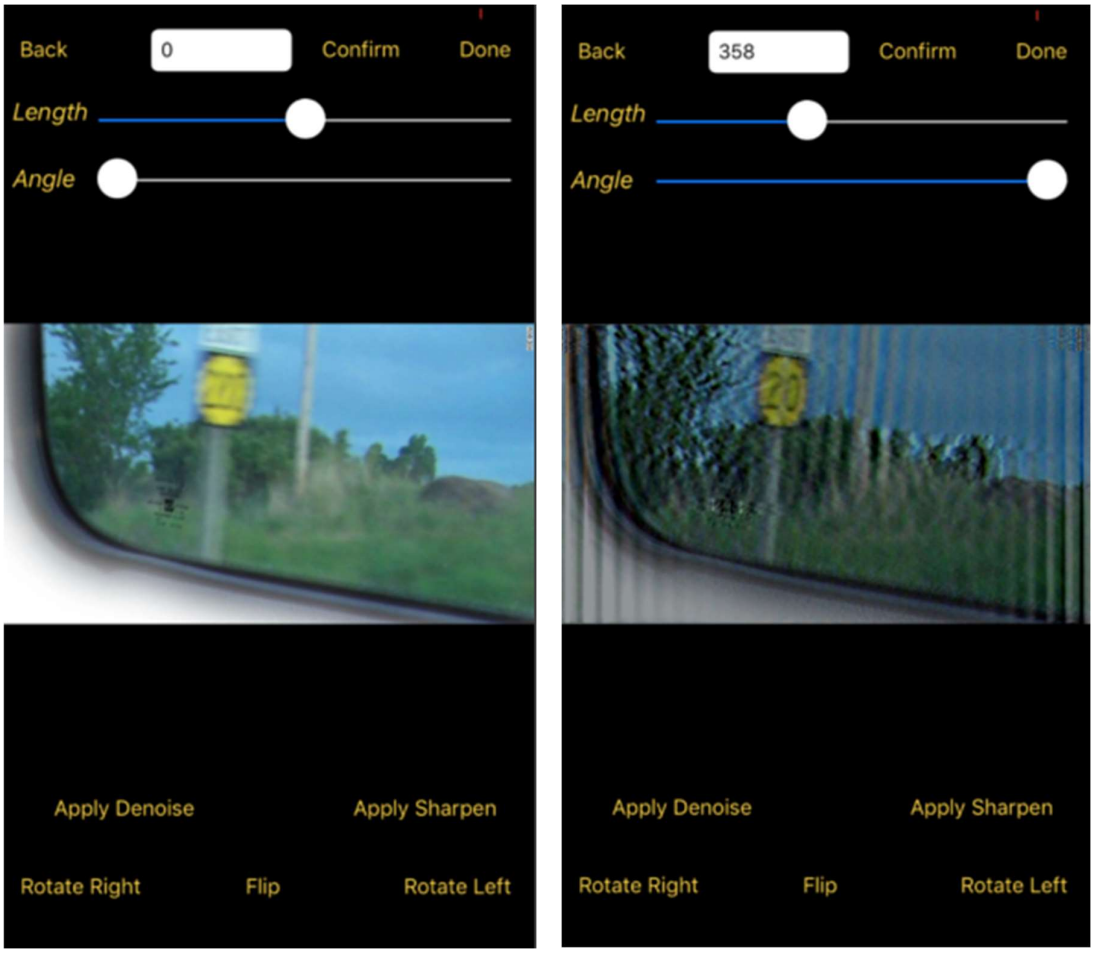


Figure 9 To use our smart phone IOS App restore the blurred key clue in 2.56 seconds

Chapter 6 Conclusion and Future Work

The conclusion of algorithm is that we should choose appropriate algorithm to be used. In different circumstances different algorithms should to be needed.

When no noise is occurred, the Van Cittert algorithm gives best results, however, this situation is a rare bird, so we discard that method. Although Richardson-Lucy performed worse than Wiener Filter, it still is suitable choice for its better result. Wiener Filter is better solution when the image has a lot of noise.

This outcome of the comparison made is not fully and well as stated in the literature. But it can be as a evidence that is provided to prove some classic paper is correct. In this project, we do not provide any new way. We just use Mathematica re-construct some results that has been proved by formers. The meaning of above work is try to find a suitable for mobile phone app. Because of the results provided by some papers are usually worked by powerful computers and some math tools, it cannot be used for commercial app. Our experiment and Demo provide a feasibility that using classic image-deblurring algorithm on mobile phone platform.

The next work we want to using CNN and GAN methods for image deblurring on smart phone. Machine learning gives another thought to restore the image without estimate the kernel.

.

.

Acknowledgements

We are indebted to Dobri Atanassov Batovski, Chayapol Moemeng and Paitoon Porntrakoon for their insights and suggestions. We are most grateful to Tapanan Yeophantong, Kwankawol Nongpong and Rachsuda Setthawong, for their constructive comments.

References

[1] Jianya Jia. Single image motion deblurring using transparency. *10.1109/CVPR.2007.383029,* 2007.

[2] DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks.

[3] RASKAR, R., AGRAWAL, A., AND TUMBLIN, J. 2006. Coded exposure photography: Motion deblurring using fluttered shutter. *ACM Transactions on Graphics 25, 3, 795–804*

[4] Multichannel blind image deconvolution using the Bussgang algorithm: spatial and multiresolution approaches. 10.1109/TIP.2003.818022 .IEEE

[5] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman. Removing camera shake from a single photograph. *ACM Trans. Graph., 25(3):787–794, July 2006.*

[6] P.A. Jansson, Deconvolution of Image and Spectra, second ed. *Academic Press, 1997.*

[7] S. D. Babacan, R. Molina, M. N. Do, and A. K. Katsaggelos. Bayesian blind deconvolution with general sparse image priors. *In European Conference on Computer Vision (ECCV), Firenze, Italy, October 2012. Springer*

[8] D. Perrone and P. Favaro. Total variation blind deconvolution: The devil is in the details. *In EEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.*

[9] L. Xu and J. Jia. Two-phase kernel estimation for robust motion deblurring. *In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics),* 2010.

[10] L. Xu, S. Zheng, and J. Jia. Unnatural L 0 Sparse Representation for Natural Image Deblurring.

[11] Richardson, William Hadley (1972). "Bayesian-Based Iterative Method of Image Restoration". *JOSA. 62 (1): 55–59. doi:10.1364/JOSA.62.000055*

[12] Lucy, L. B. (1974). "An iterative technique for the rectification of observed distributions". Astronomical Journal. *79 (6): 745–754. Bibcode:1974AJ..79.745L. doi:10.1086/111605*

[13]DONATELLI, M., ESTATICO, C., MARTINELLI, A., AND SERRACAPIZZANO, S. 2006. Improved image deblurring with antireflective boundary conditions and re-blurring. *Inverse Problems 22, 6, 2035–2053.*

[14] Canon Inc., www.canon.com/technology/optics/shakecorrect_shift/index.html, 2003.

[15] Ondrej Sindelar, Filip Sroubek, Image deblurring in smartphone devices using built-in inertial measurement sensors. *Electron Imag.22(1)011003doi:10.1117/1.JEI.22.1.011003 Journal of Electronic Imaging Volume 22, Issue 1*

[16] J. Sun, W. Cao, Z. Xu, and J. Ponce. Learning a Convolutional Neural Network for Non-uniform Motion Blur Removal.

[17] D. Gong, J. Yang, L. Liu, Y. Zhang, I. Reid, C. Shen, A. Van Den Hengel, and Q. Shi. From Motion Blur to Motion Flow: a Deep Learning Solution for Removing Heterogeneous Motion Blur

[18] S. Nah, T. Hyun, K. Kyoung, and M. Lee. Deep Multi-scale Convolutional Neural Network for Dynamic Scene Deblurring. 2016.

[19] M. Noroozi, P. Chandramouli, and P. Favaro. Motion Deblurring in the Wild. 2017

[20] S.H.M Allon, M.G.Debertrand, B.T.H.M. Sleutjes, Fast Deblurring Algoithrms , *Semantic Scholar, 2006.*

[21] Wiener, Norbert (1949), Extrapolation, Interpolation, and Smoothing of Stationary *Time Series. New York: Wiley. ISBN 0-262-73005-7*

[22] Ed Sutton. "Histograms and the Zone System". *Illustrated Photography.*

APPENDIX A

1. Van Cittert*

```
VanCittert[im_List, psf_List, cVal_, maxIter_] :=
  Module[{est, mid2, i}, mid2 = Ceiling[Dimensions[psf] / 2];
   est = im;
   For[i = 1, i < maxIter, i++, est = est + cVal (im - ListConvolve[psf, est, mid2]);];
   est];
```

*the codes provided by [20]

2. Motion Blur Function with OpenCV

```
< pre class = "cpp" name = "code" > void genaratePsf (Mat & psf, double len, double angle)
{double half = len / 2;
 double alpha = (angle - floor (angle / 180) * 180) / 180 * PI;
 double cosalpha = cos (alpha);
 double sinalpha = sin (alpha);
 int xsign;
 if (cosalpha < 0) {xsign = -1;} else {if (angle == 90) {xsign = 0;} else {xsign = 1;}} int psfwdt =
  1;
 int sx = (int) fabs (half * cosalpha + psfwdt * xsign - len * EPS);
 int sy = (int) fabs (half * sinalpha + psfwdt - len * EPS);
 Mat_ < double > psf1 (sy, sx, CV_64F);
 Mat_ < double > psf2 (sy * 2, sx * 2, CV_64F);
 int row = 2 * sy;
 int col = 2 * sx;
```

3. Richardson-Lucy core codes with OpenCV and our Motion Blur filter.

```
cv::MitionBLur (Y, reBlurred, cv::Size (winSize, winSize), dim, degree);
 // applying motionfilter
reBlurred.setTo (FLT_EPSILON, reBlurred ≤ 0);


cv::divide (wI, reBlurred, imR);
imR = imR + FLT_EPSILON;


cv::MotionBLur (imR, imR, cv::Size (winSize, winSize), dim, degree);
 // applying motionfilter

J2 = J1.clone ();
cv::multiply (Y, imR, J1);

T2 = T1.clone ();
T1 = J1 - Y;
```