IMAGE DEBLURRING - COMPUTATION OF CONFIDENCE INTERVALS

Author: Victoria Taroudaki, tarvic@math.umd.edu

Advisor: Dianne P. O'Leary*, oleary@cs.umd.edu

*Professor, Computer Science Department and Institute for Advanced Computer Studies, University of Maryland

Abstract

Pictures that cameras take can be blurred images of the original object. The restoration-deblurring of the image is not a trivial procedure and it can be very expensive depending on the size of the image. In this project we are trying to efficiently compute confidence intervals for the digital values that represent the image and visualize them so that the viewer can distinguish truth from uncertainty.

1. Project Background/ Introduction

People always wanted to keep snapshots of their everyday life for reference at a later time or for research and educational purposes. Cavemen drew on the walls of the caves using colors made from nature. Later artists painted their houses, graves and other buildings, objects or paintings with various scenes. More recently, cameras were invented, first engraving, then analog cameras and at last digital cameras. In none of these cases is the object represented exactly in the image. But as technology progresses, the accuracy of the representation increases. Digital cameras give us very good representations of the true image but due to the procedure that the image passes through, blurring occurs. This blurring can be caused by the machine errors in transforming the image into data in the camera and from the background and the way of taking the picture. Having clear images is not a luxury. Sometimes it is a matter of life and death, like in surgeries where the doctor needs to know exactly where to operate, or in weather forecasts.

An image is divided into pixels which have several values which denote the color of that pixel. A grayscale image, which we will use for simplicity, has one value for each pixel, an integer in the interval [0, 255]. 0 is the black color and 255 is the white color. Blurring occurs when the values of the pixels are distorted. In this project, we will assume that this distortion is caused by a linear transformation from the camera.

Let's now make the following notation:

Symbol	Size	Explanation
K	$m\times n$	Matrix defined through the Point Spread function (PSF) in the case of a linear problem
X		Original Clear Image
x	$n\times 1$	Vector containing the values corresponding to the pixels of the image X
B		The blurred image we measure
b	$m\times 1$	Vector which contains the values of the pixels of the blurred image B
e	$m\times 1$	Noise vector

With the above notation, the model is described by the equation:

$$b=Kx+e$$

In general, the goal is given the vector b and the matrix K and also given a distribution for the noise such that the mean value is 0 and the variance is a nonsingular matrix S^2, we need to compute confidence intervals (i.e. intervals in which the values of the object we calculate fall into the intervals with a certain confidence) for the quantities $\varphi_k^*=w_k^Tx$ for $k=1,2,\ldots,p$, where w_k are given vectors. If w_k is a column from the identity matrix, then we obtain a confidence interval for a single pixel.

Two different types of confidence intervals exist. The one-at a time confidence intervals where we compute confidence intervals for each of the φ_k^* individually contain the true values with a specific probability α ($\alpha\%$ confidence). $\Pr\{l_k\leq\varphi_k^*\leq u_k\}=\alpha$, $k=1,2,\ldots,p$

The other type is the simultaneous confidence intervals where the φ_k^* are all treated simultaneously and we compute intervals that contain all of the true values with a probability greater than α. $\Pr\{l_k\leq\varphi_k^*\leq u_k,\ k=1,2,\ldots,p\ \}\geq\alpha$

Suppose that the noise is normally distributed, and that $\hat{x}$ is an unbiased estimate of the true solution x. Then, given α in (0,1), there is a $100\alpha\%$ probability that the true value of w_k^Tx is contained in the interval $[l_k,u_k]$ where

$l_k = \min_x\{w_k^T x: \|K(x-\hat{x})\|_S^2 = \kappa^2\}$ and $u_k = \max_x\{w_k^T x: \|K(x-\hat{x})\|_S^2 = \kappa^2\}$,

where $\|K(x-\hat{x})\|_S^2 = [K(x-\hat{x})]^T S^{-2}[K(x-\hat{x})]$ and κ is such that $\alpha = \int_{-\kappa}^{\kappa} n(x;0,1)dx$ where $n(x;0,1)$ is the normal distribution of x with mean value 0 and variance 1.

Since we know that the true pixel values lie between 0 and 255, we want to use this information to reduce the length of the confidence intervals.

With the same assumptions as before and also assuming that x is nonnegative and less than 255, as well as that the matrix S is nonsingular and symmetric, the computation of the lower and upper bounds of the confidence intervals have also been done. In [4] we are given that: "Under the above assumptions, the probability that φ is contained in the interval $[l_k,u_k]$ is greater than or equal to α where $l_k = \min\left\{w_k^T x: \left\|K(x-\widehat{x})\right\|_S \le \mu, 0 \le x \le 255\right\}$ and $u_k = \max\left\{w_k^T x: \left\|K(x-\widehat{x})\right\|_S \le \mu, 0 \le x \le 255\right\}$, $\text{rank}(K)=q$, $\int_0^{\gamma^2} \chi_q^2(\rho)d\rho=\alpha$,

$r_0 = \min_{x\ge 0}\left\|K(x-\widehat{x})\right\|_S^2$, $\mu^2 = r_0 + \gamma^2$ and χ_q^2 is the probability density function for the chi-squared distribution with q degrees of freedom.

Other tools like using Chebyshev's Inequality are useful for problems where the noise is not normally distributed, but this is not the main purpose of this project and so it will not be examined here unless there is some more time at the end than expected.

2. Approach

In general, the matrix K can be experimentally measured. We can construct an initial image which contains only one white point (of value 255) in one specific pixel, say the (i,j) pixel of the image X, or the ij element of the vector x corresponding to that clear image and black anywhere else (value 0). Then, we measure the blurred vector b which corresponds to the blurred image we take. This vector b is going to be the ij column of the matrix K. If we know that the blur is spatially invariant and it usually is as it affects only neighboring points of the source point, then having measured only one column is enough to determine the matrix as we will know all the columns of K. If the blur is spatially variant, we need to move the source point to all the pixels of the image and measure the blur to compute all the columns of the blurring matrix.

In this project we will use a function K which will be given by a square matrix. This matrix can be very large depending on the size of the image. To reduce the expense, we need to partition

it into sub-matrices and so the whole problem into p smaller problems. These sub-problems are much easier to solve.

Let's consider again the problem: $b=Kx$ where K is the $n\times n$ PSF matrix, x is the vector corresponding to the original image and b the vector corresponding to the blurred image. If we want to make a sub-problem of size $r\times c$ (r rows and c columns on the sub-image defining the sub-problem), we proceed as follows. Using a matrix E with n rows and rc columns, with columns that are unit vectors corresponding to the pixels in the sub-image, and a matrix $\overline{E}$ that corresponds to the other n-rc unit vectors we have :

$$E^T b=E^T Kx=(E^T K[E\ \overline{E}])\left(\begin{bmatrix}E^T\\ \overline{E}^T\end{bmatrix}x\right)=E^T\begin{bmatrix}\widehat{K_s} & \widehat{K_t}\end{bmatrix}\begin{bmatrix}x_s\\ x_t\end{bmatrix}=[K_s\ \ K_t\ ]\begin{bmatrix}x_s\\ x_t\end{bmatrix}=K_s x_s+K_t x_t$$

where x_S is the vector corresponding to the sub-image of the original image. Ignoring the second term of the right hand side we have that $b_s=K_s x_s$ which is a sub-problem we need to solve. E is a matrix of unit vectors that we choose but in this project we will use the matrix which is part of the identity matrix corresponding to the sub-problem, i.e., if we have $x_s=\begin{pmatrix}x_1\\ x_2\end{pmatrix}$, then we will take E to be the first two columns of the identity matrix with size $n\times n$ so that the matrix K_s is going to be 2×2.

The methods used in the introduction can now be applied to these smaller problems. We compute the confidence intervals for these. To verify that the computed intervals are correct, we can take a blurred image and deblur it. We can have many different samples of deblurred images. Then we can display these deblurred images and see if see if the values of the pixels of the deblurred image are in the confidence interval with a probability that is defined by the construction of the confidence interval.

3. Implementation

The implementation of the codes will be done in Matlab. There are some already completed Matlab programs that may be used and are included in the Image Processing toolbox. These are mostly input, output and display tools. The problem involves matrices which are easy to be handled using the available linear algebra tools of Matlab. The problem can also be divided into smaller in size problems which can be solved more easily. If the blur is spatially invariant, these sub-problems involve the same matrix. These sub-problems can be solved using parallel computing. So even if Matlab's optimization tools may be used, they will need to be modified so that they can also be used in parallelizing the problem.

4. Databases

A big image database is the USC-SIPI Image Database which belongs to the Signal & Image Processing Institute of the University of Southern California and can be accessed here: http://sipi.usc.edu/database/. The database includes grayscale and color images saved in TIFF-format. They are of different sizes, 256×256, 512×512 and 1024×1024. The grayscale pictures have 8 bits/ pixel whereas the color images have 16 bits/pixel.

Also a variety of image databases can be found through the Image Processing Place by the link: http://www.imageprocessingplace.com/root_files_V3/image_databases.htm

5. Validation

In order to validate the code, we will need to run the program using data which when used, will give us a known result. As data, we mean blurred images that we know their origin, i.e. the clear image. In detail, we will take some images which will be considered as the clear images. The confidence intervals that we will compute should give us images that approximate these clear images. Then, we will blur these images to obtain the input in our code. Noise will also be added to the blurred images. Then, deblurring of the images will take place. We will also compute the confidence intervals using our code and we will count how many samples fall within the intervals. As the same code is going to be used to compute the 95% confidence intervals for all the pixels in the image, it is enough that close to 95% of the pixel values of the same image are between the computed values. If not, we should check every part of the code and the correspondence of the theory as well as some special circumstances which may be developed from the image and affect the outcomes. The images that will be used will be taken from the databases mentioned above. At the beginning, 256×256 images are to be used. The maximum size of the images will be determined by the Matlab memory.

6. Testing

A good code should give the right results in a relatively short time without much cost in memory. The time may depend on the size of the image, its format and the way that the values of the pixels are stored. The same affects the cost in memory. Also, for various types of blurring (Point Spread Function) the time for the same image may be different. All of the above are going to be studied and compared.

7. Project Schedule

September:	Study the literature, get familiar with the image toolbox and the commands of Matlab for images, write and present project proposal.
October:	Study the literature- understand all the aspects of the problem- write code.
November:	Write the code and validate it.
Early December:	Write and present midyear report.
Late January:	Test various images, begin to exercise on parallel computing.
February:	Work on the parallel part of the code.
March:	Test various images, validate and correct code if necessary.
April:	Validate and correct code if necessary, write final report.
May:	Present final report.

8. Milestones

End of September:	Having a good understanding of the literature, having used Matlab image toolbox to get familiar with it, having prepared and presented the project proposal.
End of October:	Having finished studying the basic literature, having set a database- having started writing the code.
End of November:	Having finished the basic writing of the code. (Ideal: Having finished the Matlab part of the code). Having validated it and corrected it if needed.
Middle of December:	Having written and presented the midyear report.
End of January:	Having tested several images, getting acquainted with parallel programming.
End of February:	Having chosen open MP or MPI and having worked on the parallel computing part of the code. (Ideal: having finished with parallel programming)

End of March:	Having tested various images, having validated the code. If in need, having corrected the code (Ideal: by the end of March the code works producing the correct results in little time)
End of April:	Having an efficiently working code. Having written the biggest part of the final report.
Middle of May:	Having presented the project and turned in code, final report and validation results.

9. Deliverables

Code, report and validation results

10. Bibliography

[1] Tony F. Chan and Jianhong (Jackie) Shen, “Image Processing and Analysis”, SIAM, Philadelphia, 2005

[2] Per Christian Hansen, James G. Nagy and Dianne P. O’Leary, “Deblurring Images Matrices, Spectra, and Filtering”, SIAM, Philadelphia, 2006

[3] James G. Nagy and Dianne P. O’Leary, “Image Restoration through Subimages and Confidence Images”, Electronic Transactions on Numerical Analysis, 13, 2002, p 22-37

[4] Dianne P. O’Leary and Bert W. Rust, “Confidence Intervals for inequality constrained least squares problems, with applications to ill-posed problems”, SIAM Journal on Scientific and Statistical Computing, 7, 1986, p 473-489

[5] Bert W. Rust and Dianne P. O’Leary, “Confidence intervals for discrete approximations to ill-posed problems”, The Journal of Computational and Graphical Statistics, 3, 1994, p 67-96