Image Deblurring in MATLAB® Using LU Decomposition

Caleb M. Gannon

McMaster University

Table of Contents

Abstract

The focus of this project was to find efficient (memory and time) solutions to linear systems of a large order. The developmental environment of choice for this assignment was MATLAB®. The method of choice is partial-pivot LU decomposition. To create the desired blurred images, a variety of techniques are used.

Horizontal Blur

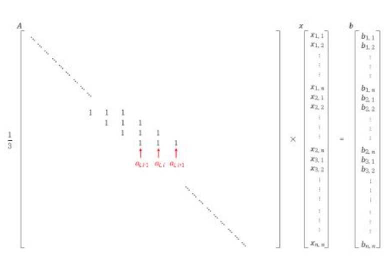
When horizontal blur is applied, it is done by averaging N amount of pixels, as though the camera had been moving. A blur matrix is created to represent this movement, and applied to the image. This is done by solving the linear system:

$$Ax = b \qquad (1).$$

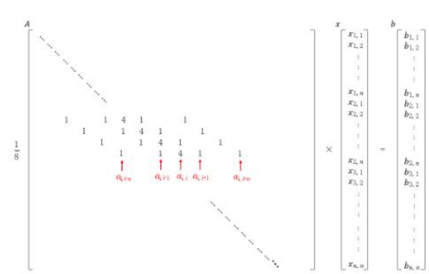
In this system, *A* is the known blur matrix, derived from the blur kernel. The vector *b* holds the blurred image information. The vector *x* is the desired deblurred image.

Vertical and Out of Focus Blur

The vertical and Out of Focus blurred images are generated by directly using a generated blur matrix, in accordance with project design.



Horizontal motion blur



Out-of-focus blur

The vertical motion blur matrix is similar to horizontal, but the x image matrix is stacked by column, rather than row.

LU Decomposition

This system is easily solved using LU Decomposition. This method breaks any square matrix (A) into a lower (L) and upper (U) triangular matrix, such that:

$$LU = A \qquad (2).$$

Due to the large scale of the arrays, I decided upon the partial-pivot method of LU decomposition, which permutates the rows in order to use the smallest possible pivot, thus reducing the risks of error when the order of A is very large. The permutation information is stored in the matrix P, such that:

$$P'LU = A \qquad (3).$$

$$LUx = Pb \qquad (4).$$

Where *P'* is the transpose of P.

Deblurring an Image

With LU decomposition available to solve a large linear system, all we have to do is find the x in equation (1) given *L, U and P.* This involves forward and backwards substitution. Forward and backward substitution is used to solve for b using:

$$Ly = Pb \qquad (5).$$

Given $Ux = y \qquad (6).$

Equation (5) can be solved for y using forward substitution, and then Equation (6) is solved for b using backward substitution.

Once b is found, it is reshaped into the original dimensions, and if the deblurring kernel is correct, it will be deblurred. It is very important that the deblurring kernel be correct. Small errors in the kernel can lead to large differences in the final image.

Discussion

Efficiency

The efficiency of my LU Decomposition can be calculated by analysing the loops and operations in the code. This function contains a nested for-loop of order 2, running from 1 to *n,* with *n* being the size of the blur matrix. Therefore, T[myLUD()] = $\Theta[\, n^2 \,]$. The forward/backward substitution methods use a nested for loop of order 2, therefore:

$$T[deblur] \;=\; \Theta[(length * width)^2\,] \quad (7).$$

Upon analysing the code, I calculated the number of operations required to deblur the image to be $3n^2 + 2 * \frac{n(n+1)}{2} \; + \; n^2 \; + \; 3n$ in the order of loop completion. Thus, solving a 20x20 image would take 801600 steps.

Effects of Errors in Blur Kernel

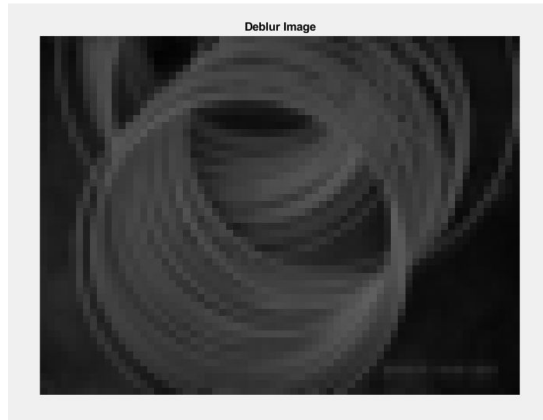


Figure 1. Figure 2.

Adding an offset of 1 to the main diagonal of the blur matrix caused the deblurred image to lose significant brightness.

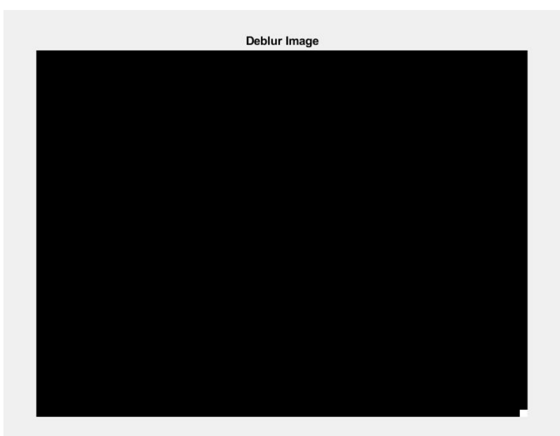


Figure 3. Figure 4.

Adding an offset of 1, along the diagonal offset by +10 from the main diagonal renders the returned image empty (Figure 3). Adding ten singular 1's, evenly spaced along the diagonal offset by +10, creates visible tears in Figure 4. Ten small changes to an array containing 11222500 elements completely disturbes the image. This is the effect of errors in the kernel.

Results

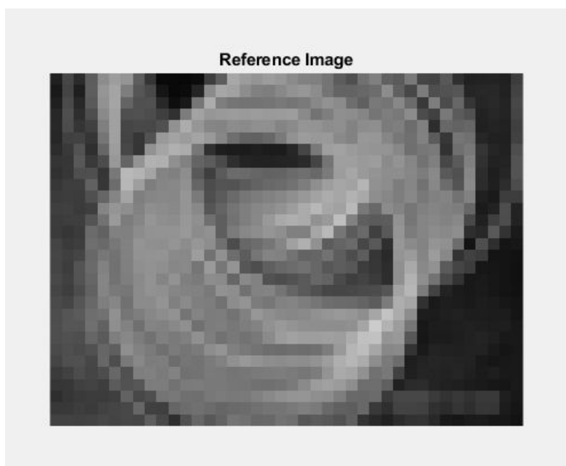


Figure 5.

This image is to be used as reference for all blurring and deblurring results.

Horizontal Blur

Figure 6.

The image above is generated using an out of focus blur kernel.

It is deblurred using LU Decomposition.

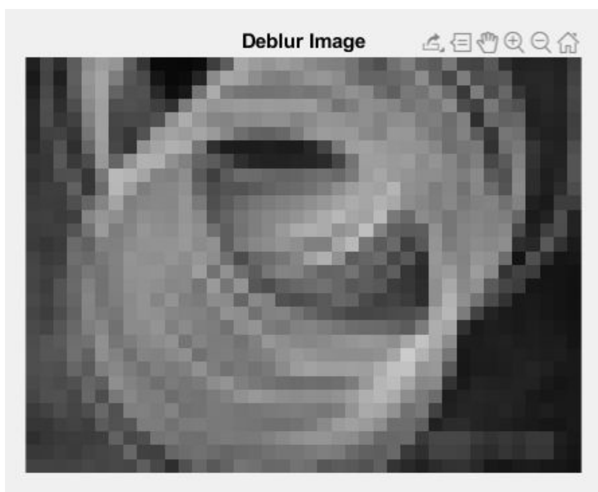


Figure 7.

Vertical Blur

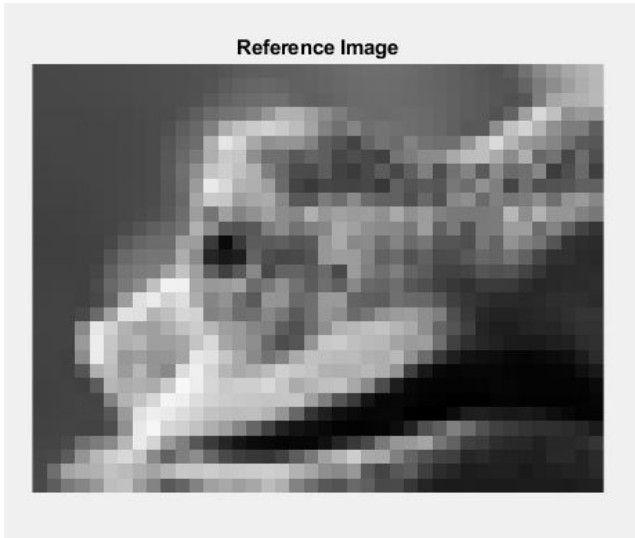


Figure 8.

This image is to be used as reference for vertical blurring.

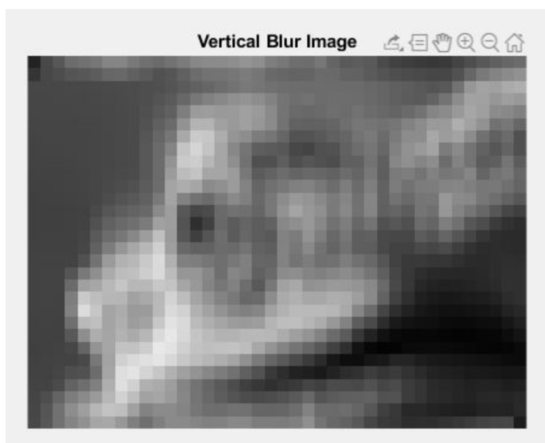


Figure 8.

The image above is generated using a vertical blur kernel.

It is deblurred using LU Decomposition.

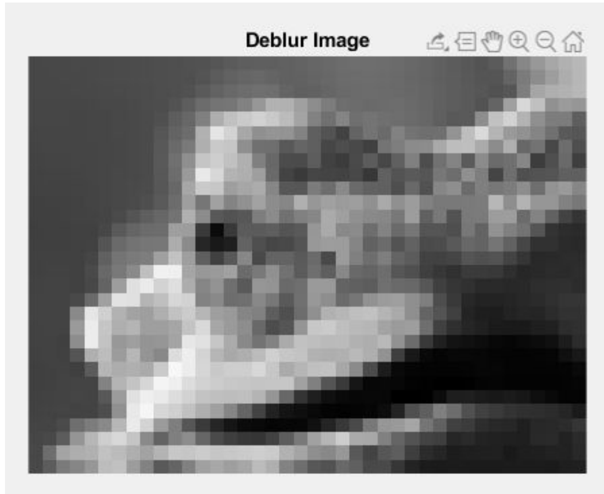


Figure 9.

This section highlights the ability of the code to perform deblurring on different images.

Out of Focus Blur

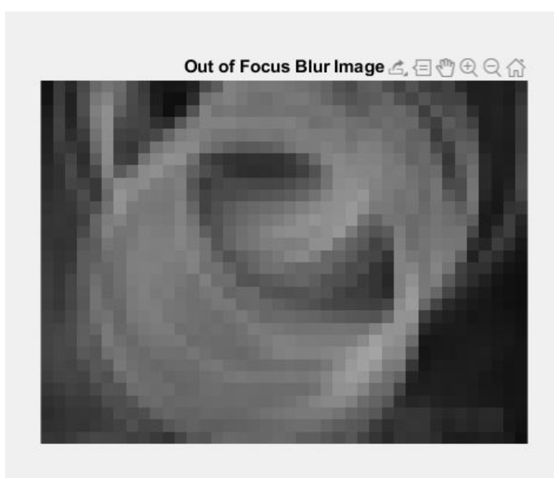


Figure 10.

The image above is generated using an out of focus blur kernel.

Once again with reference to Figure 5.

It is deblurred using LU Decomposition.

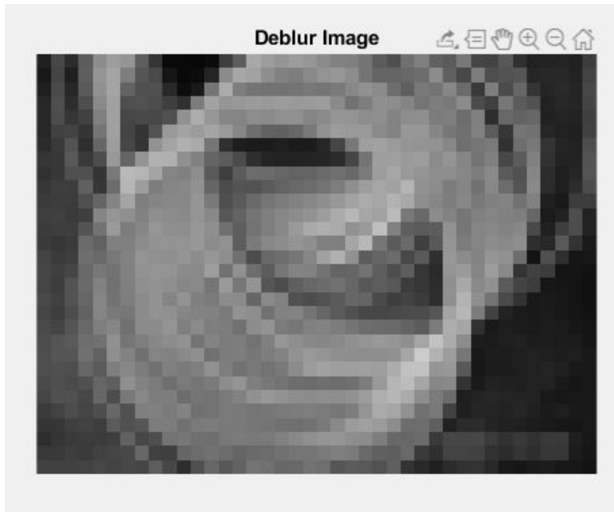


Figure 11.

References

LU Factorization with Partial Pivoting. (n.d.). Retrieved from https://chepusht.mathcs.wilkes.edu/LUwithPivoting.pdf

Linear Algebraic Equations (Chapters 9,10,11,12). (n.d.). Retrieved from https://www.ece.mcmaster.ca/~xwu/part3.pdf