Math Research for Freshmen: Applications in Image Restoration

By Katrina Palmer and Lisa Perrone

Have you ever wondered how scientists identify objects in space, how doctors interpret MRI and PET scans, or how police detectives identify suspects using surveillance cameras? If so, then you've dabbled in an application of our research called image restoration. In the spring of last year, we were given a rare opportunity to share this research: we taught one module in a five-module course that was designed to offer interested freshmen a glimpse at various ongoing math and science research projects at Emory University. Together with other graduate students, post-docs, and faculty representing various areas of science, our goal was to present our research in a way that freshmen could understand, to have students learn by discovery rather than lecture, and to emphasize the inter-connectivity between math and science that might go unnoticed in traditional single-disciplinary course work. In this brief work, we describe what we did in our math module to accomplish these goals. For full access to our course materials, go to Freshman Seminar Spring 2003, Image Restoration module, at *www.cse.emory.edu/sciencenet/coll_curr/order/index.html*.

Strictly speaking, our research is in numerical linear algebra with applications in image restoration, that is, we take blurred, noisy images and use mathematics to try to make them clearer. We've worked with images from satellites, microscopes, MRIs, and digital cameras. For the class we taught, we wanted to present both the mathematical theory and the imaging applications, so we divided our module into the following three major concept areas: convolution, linear algebra and Matlab, and experimental restorations. Details of each concept area appear below.

Introduction to the Image Restoration Problem. In order to capture students' interest, we began with a dynamic introduction to our problem. We presented a noisy, blurred image (with a brief discussion of what 'blurred' and 'noisy' might mean), and then displayed the sequence of images that appeared while using an iterative method to restore the blurred image. As seen in Figure 1, the image quality grew sharper for a time, then at some point it became poorer at each iteration, until ultimately the "restored" image was unrecognizable. We then asked students what they thought might have happened. The remaining class time proved to be a rich learning experience where we talked about things like random background noise, true randomness, iterative methods, and image restoration in general.

Convolution. The first real hands-on work for students began when we started talking about convolution. Convolution is a mathematical operation that represents the blurring process. More precisely, in convolution, each pixel of a blurred image is a weighted sum of neighboring pixels from the original scene. The weights are defined by something called a kernel. Below, we provide an example that illustrates the convolution operation.

Let X denote the input information, and K denote the kernel. In particular, suppose:

$$X = \begin{array}{|c|c|c|c|c|}\hline 17 & 24 & 1 & 8 & 15 \\ \hline 23 & 5 & 7 & 14 & 16 \\ \hline 4 & 6 & 13 & 20 & 22 \\ \hline 10 & 12 & 19 & 21 & 3 \\ \hline 11 & 18 & 25 & 2 & 9 \\ \hline \end{array} \quad \text{and} \quad K = \begin{array}{|c|c|c|}\hline 8 & 1 & 6 \\ \hline 3 & 5 & 7 \\ \hline 4 & 9 & 2 \\ \hline \end{array}$$

An important part of convolution is that the "center" (or origin) of the kernel must be specified. By convention, for kernels with odd dimensions, it is usually the middle pixel value. Thus, in our example, 5 is the center of the kernel.

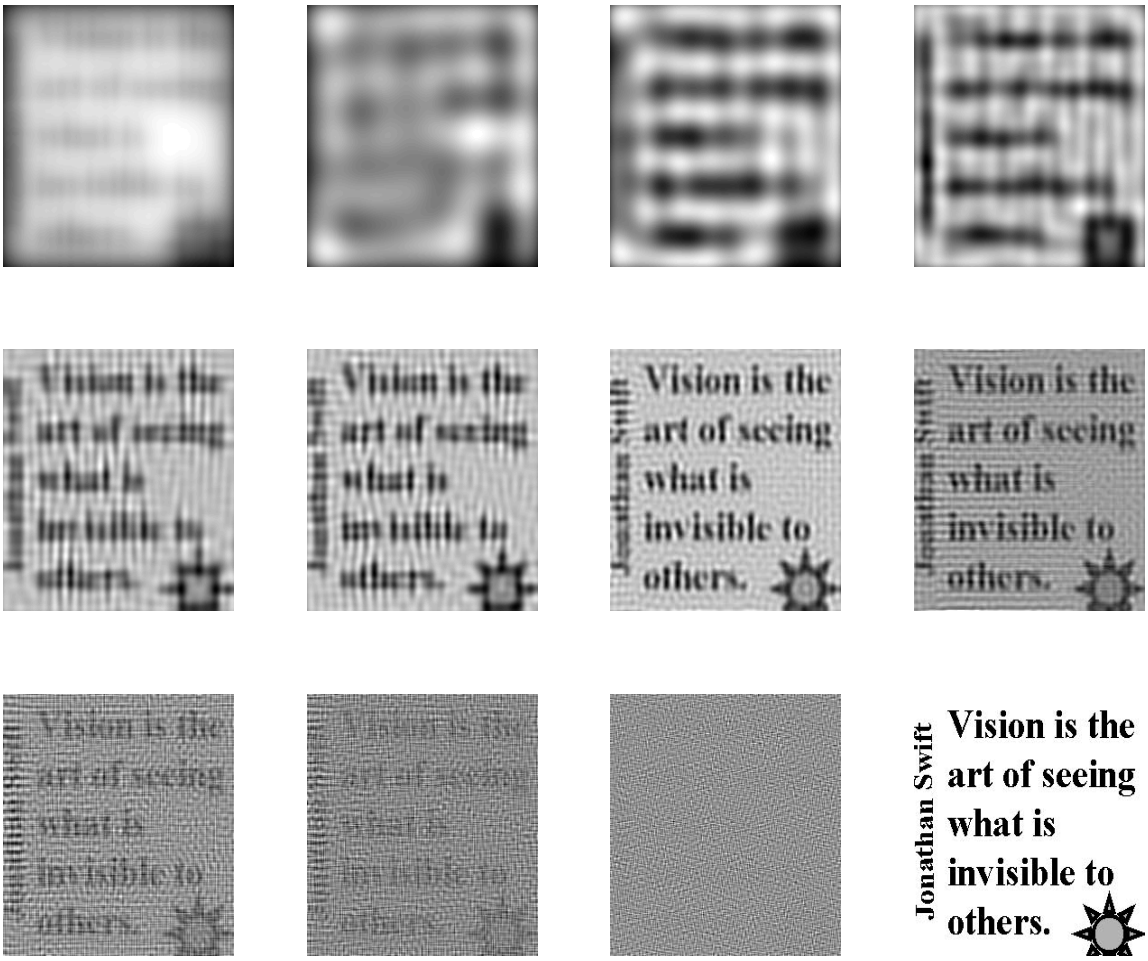


Figure 1: Introductory example. The first eleven images show the progression from degraded to restored to over-processed image. The last image in the series is the non-blurred, non-noisy, original scene.

With this notation, the convolution operation is performed as:

1. Rotate the kernel, K, 180 degrees about the center to get

$$\tilde{K} = \begin{array}{|c|c|c|} \hline 2 & 9 & 4 \\ \hline 7 & 5 & 3 \\ \hline 6 & 1 & 8 \\ \hline \end{array}$$

2. Place $\tilde{K}$ on top of X, so that the center of $\tilde{K}$ lies on top of a specific pixel element of X.
3. Multiply each weight in $\tilde{K}$ by the pixel of X underneath it.
4. Sum up the individual products to get a pixel element of the output (blurred) image.

For example:

- The (2,2) pixel of the output image is obtained by putting the center of $\tilde{K}$ over the (2,2) entry in X as follows:

17 ^{2}	24 ^{9}	1 ^{4}	8	15
23 ^{7}	5 ^{5}	7 ^{3}	14	16
4 ^{6}	6 ^{1}	13 ^{8}	20	22
10	12	19	21	3
11	18	25	2	9

  Then, multiplying and adding, the (2,2) pixel of the output image is:

$$17 * 2 + 24 * 9 + 1 * 4 + 23 * 7 + 5 * 5 + 7 * 3 + 4 * 6 + 6 * 1 + 13 * 8 = 595.$$

- The (2,3) pixel of the output image is obtained by putting the center of $\tilde{K}$ over the (2,3) entry in X as follows:

17	24 ^{2}	1 ^{9}	8 ^{4}	15
23	5 ^{7}	7 ^{5}	14 ^{3}	16
4	6 ^{6}	13 ^{1}	20 ^{8}	22
10	12	19	21	3
11	18	25	2	9

  Then, multiplying and adding, the (2,3) pixel of the output image is:

$$24 * 2 + 1 * 9 + 8 * 4 + 5 * 7 + 7 * 5 + 14 * 3 + 6 * 6 + 13 * 1 + 20 * 8 = 410.$$

Students were given practice with small kernels and input images. For details and examples, we refer the reader to the website containing course material and assignments.

Linear Algebra and Matlab. The goal in the Linear Algebra and Matlab component of the course was to develop and hone the students' understanding of convolution into technical and computational abilities. For the basics, students learned about vectors and arrays, and how to refer to entries of an array by the corresponding row and column number. Then, using Matlab, students built arrays of various sizes, then they experimented on their arrays using array operations (+, -, *, transpose) in order to glean an understanding of each operation and the dimensional restrictions thereof. We did detailed examples of matrix-times-vector operations on the board, with students working on their own examples at their desks, since this particular operation would play a central role in the material to come.

Once students were comfortable with array manipulation, we entered into a group discussion to introduce the linear algebra model of image blurring, $A\mathbf{x} = \mathbf{b}$. In this model, A is an array that represents the blurring operation and $\mathbf{x}$ and $\mathbf{b}$ are column vectors that represent the original (X) and blurred (B) images, respectfully. The vectors are obtained by storing the images as arrays, with array entries indicating light intensity, then stacking rows of the arrays into single column vectors. Once students understood this, we challenged them to figure out the size and structure of A that would make the multiplication $A \cdot \mathbf{x}$ equivalent to the convolution operation of X with a given kernel.

As their example, they supposed image X and kernel K were given by

$$X = \begin{bmatrix} 17 & 4 & 15 & 6 \\ 3 & 7 & 11 & 20 \\ 14 & 2 & 8 & 12 \\ 9 & 5 & 1 & 10 \end{bmatrix} \quad \text{and} \quad K = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}.$$

Since the vector $\mathbf{x}$ had length 16, and the vector $\mathbf{b}$ representing the blurred image B had to be the same size, the blurring array A needed 16 columns and 16 rows, respectively, in order to multiply with $\mathbf{x}$ and produce $\mathbf{b}$. Thus, A was required to be a 16×16 array, and the $A\mathbf{x} = \mathbf{b}$ model looked like

$$\begin{bmatrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{bmatrix} \begin{bmatrix} 17 \\ 4 \\ 15 \\ 6 \\ 3 \\ 7 \\ 11 \\ 20 \\ 14 \\ 2 \\ 8 \\ 12 \\ 9 \\ 5 \\ 1 \\ 10 \end{bmatrix} = \begin{bmatrix} b_{11} \\ b_{12} \\ \vdots \\ \\ \\ \\ \\ \\ \vdots \\ \\ \\ \\ \\ \\ \vdots \\ b_{44} \end{bmatrix}.$$

16×16 array

Figuring out the entries of A was the biggest challenge, and for this, we revisited the convolution model. The first entry of vector $\mathbf{b}$, corresponding to b_{11} in the blurred image, resulted from the convolution of K and X with the center of K lying on top of x_{11}. Assuming that elements of K lying outside the image contributed nothing to the blur, the arithmetic was

$$0 * 1 + 0 * 2 + 0 * 1 + 0 * 2 + 17 * 4 + 4 * 2 + 0 * 1 + 3 * 2 + 7 * 1 = b_{11}.$$

Moreover, this first entry of $\mathbf{b}$, from a matrix-times-vector standpoint, was the result of the first row of A times the vector $\mathbf{x}$. Putting these two facts together, students determined that the first row of A was comprised of zeros and the appropriate elements of the kernel, in the appropriate order. That is, the first row of A looked like:

$$\begin{bmatrix} 4 & 2 & 0 & 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Likewise, since the convolution arithmetic to produce b_{12} was

$$0 * 1 + 0 * 2 + 0 * 1 + 17 * 2 + 4 * 4 + 15 * 2 + 3 * 1 + 7 * 2 + 11 * 1 = b_{12},$$

the second row of A had to be

$$\begin{bmatrix} 2 & 4 & 2 & 0 & 1 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Using similar reasoning, the remaining rows of A were determined, and A looked like

$$A = \left[\begin{array}{cccc|cccc|cccc|cccc} 4&2&0&0&2&1&0&0&0&0&0&0&0&0&0&0\\ 2&4&2&0&1&2&1&0&0&0&0&0&0&0&0&0\\ 0&2&4&2&0&1&2&1&0&0&0&0&0&0&0&0\\ 0&0&2&4&0&0&1&2&0&0&0&0&0&0&0&0\\ \hline 2&1&0&0&4&2&0&0&2&1&0&0&0&0&0&0\\ 1&2&1&0&2&4&2&0&1&2&1&0&0&0&0&0\\ 0&1&2&1&0&2&4&2&0&1&2&1&0&0&0&0\\ 0&0&1&2&0&0&2&4&0&0&1&2&0&0&0&0\\ \hline 0&0&0&0&2&1&0&0&4&2&0&0&2&1&0&0\\ 0&0&0&0&1&2&1&0&2&4&2&0&1&2&1&0\\ 0&0&0&0&0&1&2&1&0&2&4&2&0&1&2&1\\ 0&0&0&0&0&0&1&2&0&0&2&4&0&0&1&2\\ \hline 0&0&0&0&0&0&0&0&2&1&0&0&4&2&0&0\\ 0&0&0&0&0&0&0&0&1&2&1&0&2&4&2&0\\ 0&0&0&0&0&0&0&0&0&1&2&1&0&2&4&2\\ 0&0&0&0&0&0&0&0&0&0&1&2&0&0&2&4 \end{array}\right].$$

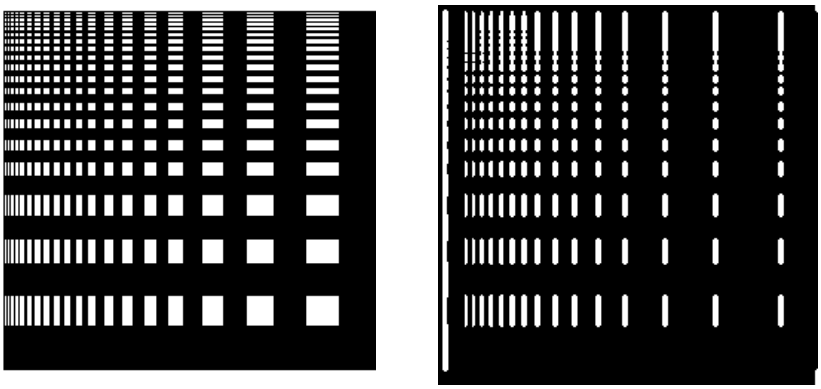
Finally, once we determined A by hand, we went back to Matlab to check that our $A \cdot \mathbf{x}$ really did produce the (vectorized) result of convolution. It did indeed, and students were pleased and satisfied, but exhausted. Of course, the victory was short-lived; they realized that they had been studying image *blurring*, not image *restoration*! We contented them by letting them know that they would be working on image restoration soon.

Experimental Restorations. Our goal in this area was to give students a sense of accomplishment for all the new mathematics they had learned. In that vein, we sought after visual results by embarking on the study of edge detection and computer-based restorations. The problem of edge detection is to determine locations in an image where there is a sudden variation in the gray level. These sudden changes can sometimes be detected by convolving the image with certain kernels. Using Matlab, students experimented with various kernels to see which type(s) of edge each kernel would detect. An example of a kernel used by students is

$$K = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix},$$

which convolves with the image in Figure 2a to produce the result in Figure 2b. After careful inspection of the input and output data, students determined that this particular kernel detects and highlights edges where color changes from light (large pixel values) to dark (small pixel values) going from left to right. Upon making this determination, then looking back at the kernel structure, some students were able to guess the effects of other edge-detection kernels without relying on convolution by Matlab. In many cases, a quick Matlab-based convolution confirmed their guesses.

For our work with computer-based restorations, we relied heavily on two things: one, an image restoration package called RestoreTools [1], [2]; two, careful presentation and discussion of the concept of iterative methods of image restoration. In an iterative method for solving $A\mathbf{x} = \mathbf{b}$, an initial guess at the true solution $\mathbf{x}$ is made, often $\mathbf{x}_0 = \mathbf{b}$, then updates $\mathbf{x}_k$, $k = 1, 2, \ldots$ are repeatedly computed until achieving a result that is sufficiently "good." One way to check the quality of an intermediate solution $\mathbf{x}_k$ is to compute the size of $A\mathbf{x}_k - \mathbf{b}$ at each step of the iterative



(a.) True Image (b.) Convolution with K

Figure 2: Edge Detection Example .

routine. As long as this quantity, called the residual, gets smaller at each iteration, the updating process continues. Once it begins to grow, the process is halted, and the previous solution is considered the best. Students understood this well. Indeed, in theory, with the exact solution $\mathbf{x}$, $A\mathbf{x} - \mathbf{b}$ would be the zero vector.

We had some theoretical discussions regarding the wisdom of the residual-minimizing approach. The crux was that, because there is random noise in real data, then even if the true $\mathbf{x}$ were available, $\mathbf{b}$ would not be exactly $A\mathbf{x}$. Thus, why should we try to minimize the size of $A\mathbf{x}_k - \mathbf{b}$? The ensuing discussion was short, thought-provoking, but (as expected) without a satisfying end. Many methods exist for addressing the image restoration problem, but no method is ideal, and no method is unconditionally better than the others. Experience may provide intuition as to the "best" restoration method for any given image, but even then, image restoration is not an exact science. The noise in an image is truly random, a condition which our greatest computers are unable to duplicate. Nevertheless, given these limitations, minimization of the residual is at least an intuitively good place to start.

Once students had a reasonable (conceptual) grasp on how iterative methods performed, and because they already possessed such a good understanding of convolution, we gave them a final assignment: given a left-motion-blurred image, create a kernel of any size and shape that might represent the blurring process. Then use RestoreTools to build A from the suspected kernel, and attempt a restoration. They were up to the task, and although there were many different kernels invented, each student did finally come up with some type of left-motion-blurring kernel, and each student was able to obtain a restoration. The module website has an example of a left-motion-blurred image used for these experiments. Students seemed to enjoy this day the most. They found it very rewarding to make images clearer.

Conclusions. Overall, this module was a success. We were able to present a taste of our math research to freshmen, and they were able to digest and enjoy it! As evidenced on the website, we spent more time on statistics and Matlab than this report indicates; topics included basic statistical knowledge (including probability distributions), the Monte Carlo method, and Matlab script-writing. For readers interested in integrating the image restoration module into a broader course where such extra topics are needed, we recommend the material posted on the website, as it provides a basic but solid foundation. In particular, during the science modules covered later in the semester, the students benefited from having learned statistics and Matlab within our module.

Nevertheless, whether offering Image Restoration as a stand-alone short course or as part of a series on Math and Science Research, undergraduates –even freshmen– can enjoy and benefit from the tractable, concepts-first, details-later approach to Image Restoration offered here. Try it and see!

For additional background reading and information, see [3], [4], and [5]. Course materials can be found at the website specified in the first paragraph of this work.

Acknowledgements. We would like to acknowledge professor David Lynn of Emory University for his vision in conceiving the Origins of Order project, and for his tireless support and exceptional insights that helped bring the course to life. The Origins of Order Freshman Seminar Project was funded by a grant from the Howard Hughes Medical Institute.

References

[1] J. Nagy, K. Palmer, and L. Perrone. *Iterative Methods in Image Restoration: An Object-Oriented Approach.* Num. Algor., v. 36, pp. 73–93, 2003.

[2] RestoreTools: An Object Oriented Matlab Package for Image Restoration, 2002. http://www.mathcs.emory.edu/∼nagy/RestoreTools.

[3] D. O'Leary and J. Nagy. *Image Deblurring: I Can See Clearly Now.* Computing in Science & Engineering, IEEE CS and AIP pub., D. O'Leary ed., pp. 2–4, 2003.

[4] Using MATLAB, Version 6. The Math Works, Inc., 2000.

[5] M. Bertero and P. Boccacci. *Introduction to Inverse Problems in Imaging.* IOP Publishing Ltd., London, 1998.