Image Correction with Linear and Diffusion Filtering

Kellen Betts

Updated July 31, 2016

Abstract

In this project corrupted images of male supermodel Derek Zoolander are received, and two image processing techniques are implemented to repair the images. The techniques are used for both color and black and white images. Linear filtering is used to denoise a set of images that are corrupted by noise across the entire image. Using both a Gaussian and Shannon filter, moderate success is achieved. A diffusion process is then developed to restore a second set of images that contain noise confined to a small region. The localized nature of the corruption and spatial flexibility of the diffusion process result is near perfect restoration for the second set of images. Finally, a combination of linear filtering and diffusion is explored as a hybrid method for the first images, but only minimal improvement over the individual methods is observed.

1 Introduction

The setup for this project involves two sets of images (Figure 1) of the fictional male supermodel Derek Zoolander that have been corrupted by protestors. One set of images contains noise across the entire image, and one set is corrupted in a localized region near Derek's nose. The first objective (Task 1) is to repair the images with global noise using a linear filter. Both a Gaussian and Shannon filter will be tested. The next objective (Task 2) is to use a diffusion process to repair the images with noise confined to a small region since diffusion is effective when targeting specific regions of an image. Finally, diffusion and linear filtering are compared and their combination is explored as a hybrid method to achieve better results.

Figure 1. Original corrupted images received for the project.

2 Theoretical Background

Linear filtering of images is similar to time-frequency analysis in that the spectral component of data, in this case of an image, is used to apply a linear filter to remove specific frequency components. The basis for the frequency transformation used in this method is the Fourier Transform. For a given function $f(x)$ the transform and its inverse are defined,

$$F(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-ikx} f(x)\, dx \tag{1}$$

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{ikx} F(k) dk \tag{2}$$

where k corresponds to the wave-numbers in the trigonometric identity. With the Fourier transform, the image data is integrated over the domain $x \,\epsilon\, [-\infty, \infty]$ resolving the frequency content. Computationally the transform is implemented over a finite domain $x \,\epsilon\, [-L, L]$ using the Fast Fourier Transform (FFT). One of the important advantages of the FFT algorithm is the low operation count of $O(N \log N)$ using a 2^n discritization. Additionally, given its trigonometric construction, the transform assumes a 2π-periodic domain.

Many different linear filters can be used in image processing. In this project the noise in the image data is shown to be high frequency, so one filter used is a Gaussian (Kutz 13.2.57),

$$F(k_x, k_y) = \exp\left(-\sigma_x (k_x - a)^2 - \sigma_y (k_y - b)^2\right) \tag{3}$$

centered at (a, b) with filter widths σ_x and σ_y . Another filter used is a Shannon based on the step function,

$$F_A(x, y) = \begin{cases} 1 & if\, x, y \,\epsilon\, A \\ 0 & if\, x, y \notin A \end{cases} \tag{4}$$

where A is a targeted area of specified dimensions.

The second method used to remove the noise from image data is diffusion. The use of diffusion can be shown to be equivalent to linear filtering with a Gaussian function by considering a general diffusion equation (Kutz 13.3.58),

$$u_t = D\nabla^2 u \tag{5}$$

where D is a diffusion coefficient, $u \equiv u(x, y)$, and the Laplacian $\nabla^2 = \partial_x^2 + \partial_y^2$. If periodic boundaries are assumed, (5) can be solved using the Fourier transform giving (Kutz 13.3.59),

$$\hat{u_t} = -D\left(k_x^2 + k_y^2\right)\hat{u} \quad \rightarrow \quad \hat{u} = \hat{u_0} \exp\left(-D\left(k_x^2 + k_y^2\right)t\right) \tag{6}$$

which shows that the linear Gaussian filter (3) is equivalent to the solution (6) of the general diffusion equation (5). To implement this method computationally, (5) can be rewritten (Kutz 13.3.60),

$$u_t = \nabla \cdot (D(x, y)\, \nabla u \tag{7}$$

where $D(x, y)$ is a spatial diffusion coefficient that will be used to apply the method on images with a finite region of noise. This differential equation (7) can be discritized using a second-order center difference scheme where the x-domain is (Kutz 13.3.61a),

$$\frac{\partial^2 u}{\partial x^2} = \frac{1}{\triangle x^2} \left[u(x + \triangle x,\, y) - 2u(x,\, y) + u(x - \triangle x,\, y)\right] \tag{8}$$

and a similar scheme is used for $\partial^2 u/\partial y^2$. The discritized linear system for (7) in one dimension is given by (Kutz 13.3.63),

$$\frac{d\mathbf{u}}{dt} = \frac{k}{\triangle x^2}\mathbf{A}\mathbf{u} \tag{9}$$

where the second-order center difference scheme (8) with periodic boundaries gives the Laplacian operator (Kutz 13.3.64),

$$\mathbf{A} = \begin{bmatrix} -2 & 1 & 0 & \cdots & 0 & 1 \\ 1 & -2 & 1 & 0 & & 0 \\ 0 & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & & & & 0 \\ 0 & & 0 & 1 & -2 & 1 \\ 1 & 0 & \cdots & 0 & 1 & -2 \end{bmatrix} \tag{10}$$

To denoise images in a finite region, diffusion is localized by defining the spatial coefficient $D(x, y)$ using a smooth Gaussian similar to the linear filter (3) with,

$$D(x, y) = C \exp\left(-\sigma_x (x - a)^2 - \sigma_y (y - b)^2\right) \tag{11}$$

which is centered at (a, b) with the constant C and widths σ_x and σ_y .

3 Algorithm Implementation and Development

Image Import/Plotting:

The JPG images are imported using the `imread` command. The incoming data is in 8-bit integer format, so it is converted to double before any further operations are performed. Black and white images import as a $x \times y \times 1$ grid, and color images as $x \times y \times 3$ corresponding to the rgb color profile. Each color channel is processed separately, then stitched back together at the end. Plotting of images is done using the `imshow` command requiring conversion back to 8-bit integer format. Several independent plotting functions are developed (see Appendix B) which greatly simplifies the plot code needed to output the results from the various filtering and diffusion processes.

Linear Filtering:

There are four basic steps in the linear filtering procedure:

1. Importing images and initializing vectors/parameters.
2. Transforming image data into frequency domain using FFT.
3. Applying Gaussian or Shannon filter function.
4. Inverse transform of the filtered data back to spatial domain.

The flow of operations is controlled by a main script `hw3_task1.m` . The images are 253x361 so the x, y domain does not match the 2^n discritization necessary to achieve minimum operation count with the FFT. The closest 2^n discritization is $2^8 \times 2^9 = 256 \times 512$ and so rescaling would require significant extrapolation.

The primary filtering operations are run in a separate function `hw3_filter.m` so that exploration of optimal parameters can be automated. In the function, the frequency domain is discritized $1 : x_n$ and $1 : y_n$ because the filter is applied to frequency data that is shifted back to the linear discritization.

1D wave-number vectors are transformed to 2D grids using the `meshgrid` command. The Gaussian filter is calculated using,

$$\texttt{F} = \texttt{exp}(-\texttt{sigma}(\texttt{k}) * (\texttt{Kx} - (\texttt{w}/2+1)).^2 - \texttt{sigma}(\texttt{k}) * (\texttt{Ky} - (\texttt{h}/2+1)).^2)$$

where the sigma and center parameters are explored extensively to achieve best results. A Shannon filter centered at (a, b) is implemented using,

$$\texttt{F}(\texttt{b} - \texttt{width} : 1 : \texttt{b} + \texttt{width}, \texttt{a} - \texttt{width} : 1 : \texttt{a} + \texttt{width}) = 1$$

with the remaining regions of F filled with zeros and width parameter explored until best results are achieved.

The 2D image data (individual color channel if rgb) is transformed and shifted in frequency domain using the `fft2` and `fftshift` commands respectively. The frequency data is then multiplied by the specified filter. Finally, the data is inverse shifted/transformed and returned to the main script for plotting and analysis.

Diffusion:

There are four basic steps in the diffusion procedure:

1. Import of images and initialization of vectors/parameters.
2. Building a sparse 2D differentiation matrix (L).
3. Building a sparse coefficient matrix $D\,(x, y)$ for localized diffusion.
4. Solving the linear system with the ODE solver.

The flow of operations is controlled by a main script `hw3_task2.m`, with primary operations run in the separate function `hw3_diffusion.m`. The spatial domains (x, y) are linearly discritized. Sparse derivative matrices are built using the `spdiags` command for each 1D domain (x, y) corresponding to the A matrix (10). The 1D operators are then stitched together to make a sparse 2D Laplacian operator (L) using the `kron` command.

For diffusion applied to a localized region, the sparse coefficient matrix $D\,(x, y)$ is built using the `spdiags` command as well. The matrix is discritized using a smooth Gaussian function,

$$\texttt{D2}(\texttt{jx}, \texttt{jy}) = \texttt{C} * \texttt{D2}(\texttt{jx}, \texttt{jy}) * \texttt{exp}(-\texttt{sigma} * (\texttt{jx} - \texttt{a}).^2 - \texttt{sigma} * (\texttt{jy} - \texttt{b}).^2)$$

which gives the diffusion operation spatial localization centered at (a, b). The sigma and center (a, b) parameters are explored extensively to achieve best results.

Next, the 2D image data (individual color channel if rgb) is passed to the ODE solver `ode113` (see Appendix A) with the 2D Laplacian operator (L), coefficient matrix $D\,(x, y)$, and parameters for the time interval steps. The ODE solver calls a separate function `hw3_rhs.m` which contains the right-hand-side operation of the linear system,

$$\texttt{rhs} = \texttt{coeff} * \texttt{L} * \texttt{u}$$

Having multi-channel data (rgb color) means output from the ODE solver has to be handled carefully to ensure images are stitched back together correctly.

4 Computational Results

Task 1:

The objective in Task 1 is to repair the set of images (Figure 2) with global noise using a linear filter. The images both appear to have similar noise across the entire image. Visualizing the frequency component of the image on a log scale (Figure 3, left) clearly shows frequencies in the unfiltered image are concentrated at low frequencies, so the objective is to remove the effects of high-frequency noise.

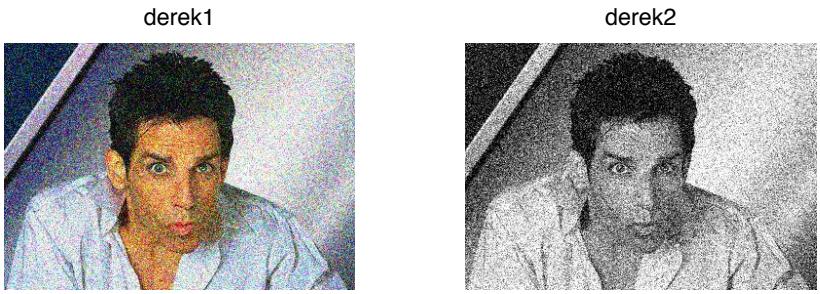

Figure 2. Original images showing noise corruption across the entire x, y spatial domain.

For the Gaussian filter, different values of the width parameter were explored (Figure 4) with a value $\sigma \approx 0.0025$ showing the best balance of noise reduction and blur. The frequency plot for the filtered image (Figure 3, right) shows that the Gaussian filter with $\sigma = 0.0025$ spreads the distribution. When processing the images, each channel is isolated and filtered separately (Figure 5). The best results achieved for the Gaussian filter with both the color and black and white images are seen in Figure 6. These images indicate the Gaussian filter is able to moderately reduce the noise in the original images with slight image blur.

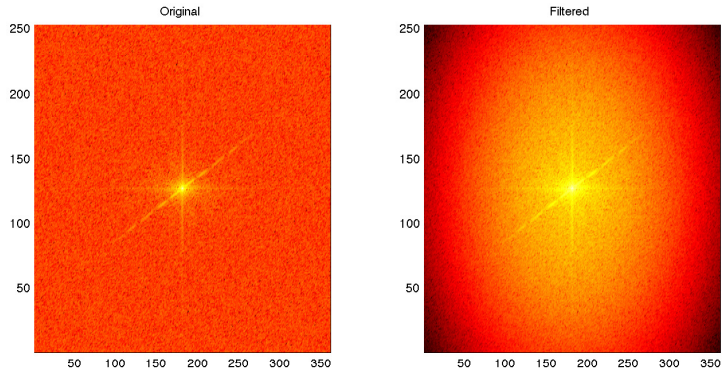

Figure 3. Comparison of the log of the Fourier transform for the original noisy image (left) and Gaussian filtered with $\sigma = 0.0025$ (right). The filter appears to improve the frequency distribution moderately.

Figure 4. Series of images exploring Gaussian filter width (σ) values. The optimal value at $\sigma \approx 0.0025$ best balances the tradeoff between noise reduction and blurring.

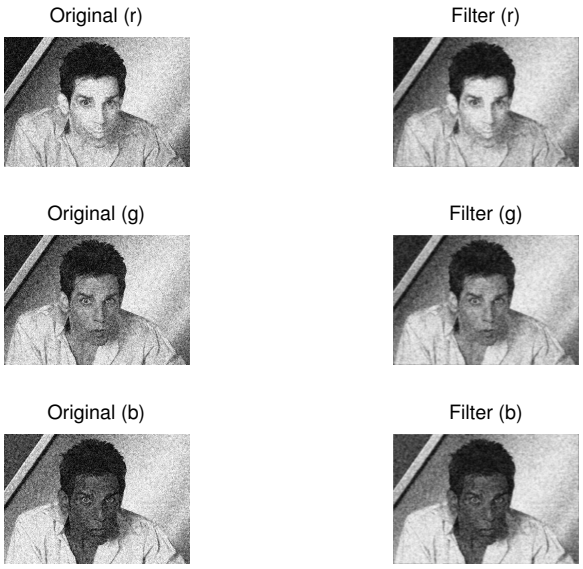

Figure 5. The Gaussian filter $(\sigma = 0.0025)$ is applied separately for each channel of the color image. The channels are then stitched back together to build a filter color image.

Figure 6. Comparison of the original noisy image (left) with the best results achieved for the Gaussian filter (right) with $\sigma = 0.0025$. The filter appears to moderately remove the noise in the image while slightly reducing image sharpness.

For the Shannon filter, testing different values of the width parameter shows that a width of 50 pixels has the best balance of noise reduction and blur. The effect of the Shannon filter on the frequency distribution is seen in Figure 7 (right), where all frequencies that fall outside the window are zeroed out. Figure 8 shows the best results for both the Gaussian and Shannon filters. The results indicate that both filters moderately removed the noise corrupting the images. The Shannon filter appears to better preserve the sharpness of the image, which agrees with its step-function construction. Conversely, the smoothly varying Gaussian appears to remove more noise (it can reach inside the Shannon window), but also slightly reduces image sharpness meaning some necessary frequency content is lost.

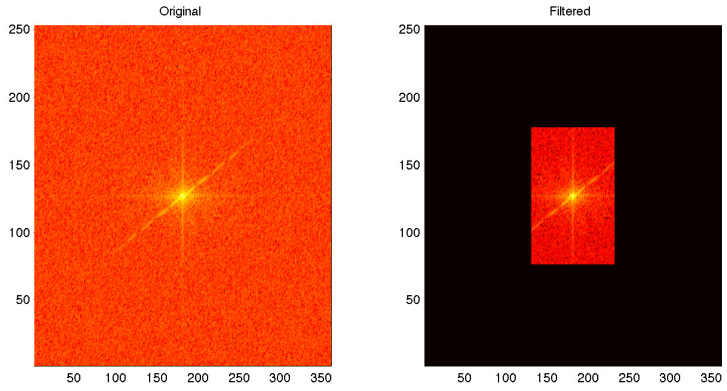

Figure 7. Comparison of the log of the Fourier transform for the original noisy image (left) and Shannon filtered with a window width of 50 pixels (right).

Figure 8. Comparison of the original noisy image (left) with the best results for the Gaussian filter (center) with $\sigma = 0.0025$ and the Shannon filter (right) with a width of 50px. The Shannon filter appears to better preserve the sharpness of the image, while the Gaussian appears to remove more noise but also slightly reduce image sharpness.

Task 2:

The objective of Task 2 is to repair a set of images (Figure 9) with noise corruption localized to a small square region. Both images appear to have the same corruption. The noise patch in the images appears slightly lower and left of image center, so the (a, b) coordinates were iterated starting from image center until the optimal coordinates $(155, 162)$ were determined. The time interval parameter is extensively explored, and as seen in Figure 10 a value of $t \approx 0.03$ best balances the tradeoff between removal of the local noise and preservation of adjacent image detail. For the black and white images, a slight higher value ($t \approx 0.05$) is optimal. The best results achieved for denoising with the diffusion process (Figure 11) show the noise patch is effectively removed with minimal disturbance to adjacent detail. Since the patch is removed with the diffusion process, additional filtering is unnecessary.

Figure 9. Original images showing noise corruption localized to a square region near the nose.

Figure 10. Series exploring time interval parameter. The value at $t \approx 0.03$ best balances the tradeoff between removal of the local noise and preservation of adjacent image detail.

Figure 11. Comparison of the original (left) with localize noise patch and the best result for denoising with the diffusion process (right). These show the noise patch is effectively removed with minimal disturbance to adjacent detail.

Additional Tests:

Using the techniques developed in Task 1 and Task 2, the diffusion and filtering processes can be further explored by applying diffusion to the images from Task 1 and comparing the results with the filters. First, the diffusion process is optimized for the images from Task 1 using a constant coefficient D since the targeted spatial region is the entire image and testing the time interval parameter. The optimal values for these images are $D = 0.005$ at $t \approx 0.01$ which best balances the tradeoff between removal noise and loss of image sharpness. Comparison (Figure 12) of the diffusion process at $t = 0.01$ with the Gaussian and Shannon filters shows that diffusion achieves similar results with a moderate reduction in noise. This is not surprising since the linear Gaussian filter (3) and Fourier solution to the diffusion equation (6) are shown to be equivalent (see section 2).

Finally, sequential denoising techniques with both filtering and diffusion are tested using the techniques and parameters previously established. If the original noisy images are filtered using the Gaussian then subsequently sent through the diffusion process (Figure 13), only moderate improvement in the final image quality is seen. Similar results are seen if the sequence is reversed with diffusion applied first (Figure 14).

Figure 12. Comparison of the original noisy image (upper left) with denoising by diffusion (upper right), Gaussian filter (lower left), and Shannon filter (lower right). Diffusion is similar to both filter methods, falling in the middle for the tradeoff of noise removal and image sharpness.

Figure 13. Comparison of the original noisy image (left) with the results from the Gaussian filter (center) and a sequential denoising by diffusion (right). Moderate improvement in image quality is achieved with this technique.

Figure 14. Comparison of the original noisy image (left) with the results from denoising by diffusion (center) and sequential Gaussian filter (right). Moderate improvement in image quality is achieved with this technique as well.

5 Summary and Conclusions

Ultimately, in this project image processing techniques, including linear filtering and diffusion, are developed and applied to two sets of images corrupted by noise. One set is corrupted by noise across the entire image, and linear filtering with both Gaussian and Shannon filters are shown to have moderate success. The second set of images has a patch of noise localized to a small square region. These images are repaired using a diffusion process with a targeted spatial coefficient matrix. The results show that diffusion is highly effective at removing the localized noise, but when it applied to the images with noise across the entire image the results are comparable to linear filtering. Additionally, hybrid techniques with both linear filtering and diffusion show minimal improvement.

One of the more substantial challenges faced in this project was building the targeted coefficient matrix for the diffusion process. The initial version I made caused matrix multiplication errors when used in the rhs function. Fortunately, one of my classmates provided a suggestion on the discussion board that resolved the dimensioning problem I was having, and I am indebted to their contribution.

Appendix A

MATLAB commands used:

abs Takes the absolute value.

ceil Used to round an operation toward ∞.

eye Builds identity matrices for given dimension. Used when building 1D derivative matrices and 2D Laplacian operator.

fftn The all important FFT function, which performs a discritized Fourier Transforms. This version of the function transforms n-dimensional data. The `fft` and `fft2` versions transform 1- and 2-dimensional data respectively.

fftshift The output of the `fft` algorithm is shifted (butterfly algorithm), so data in the frequency domain is shifted back using this function before plotting.

imread Load the data in an image file into a matrix that can then be used for subsequent data manipulations.

imshow Used to plot image data.

kron Builds a large matrix filled with all possible element-wise products for given matrices. Used to build the 2D l=Laplacian operator from 1D derivative matrices.

load Loads data from an external file.

length Used to get the length of vectors.

linspace Used to build a linear vector with $n + 1$ points for the spatial domain. The vector is then trimmed to n points due to the periodic boundaries.

meshgrid Transforms the domain of given vector input to multi-dimension matrices.

nextpow2 Used to find the next power of 2 for a given input. Since FFT is optimal with a 2^n discritization, the datasets are trimmed to a 2^n length.

num2str Used to convert a number to a string.

ode113 Non-stiff ODE solver use for time evolution of linear system. Has a default relative tolerance of 10^{-3} and absolute tolerance of 10^{-6}.

ones Used to build vector/matrices pre-filled with "1".

pcolor Used to plot the 3D spectrograms.

plot Used to plot the various parameters agains time or frequency.

real Returns $\mathrm{Re}\,(z)$ for a given input.

reshape Reshapes vector or matrix for given dimensions.

spdiags Builds sparse diagonal matrices from given vector and indices. This is a critical tool for lower operation count when solving linear systems.

strcat Used to concatenate a string.

subplot Used to produce plot arrays.

switch Condition structure used to vary code execution based on a specified tag.

tic/toc Used to time the operations.

uint8 Used to convert data to 8-bit integer format.

zeros Used to build vectors and matrices filled with "0".

Appendix B

See the project root for files.

Task 1:

Main control script `task1.m` and linear filtering function `filter.m`.

Task 2:

Main control script `task2.m`, diffusion filtering function `diffusion.m` and right-hand-side function `rhs.m`.

Plotting and related:

- `plot_imgPick.m`
- `plot_array22.m` (note: `plot_array23.m` is essentially identical but plots a 2x3 subplot array)
- `plot_diffSeries.m`
- `plot_filtered.m`
- `plot_fourier.m`