AMATH 482/582: HOMEWORK 2

JONATHAN MCCORMACK

Applied Mathematics Department, University of Washington, Seattle, WA
jrmack20@uw.edu

Abstract. Image processing is a rich field of data analysis with broad industrial applications as images contain meaningful features that can be highlighted and quantified. In this report we accept various benchmark, grayscale images and attempt to remove the regions with smooth gradient to reveal an edge map with only the lines from the original image. The algorithm is based on combining a Gaussian pre-filter to remove noise and a multiscale two-dimensional Discrete Wavelet Transform to remove flat regions. This process can be incredibly useful in presenting the key, structural features of an image.

1. Introduction and Overview

In image processing, the two-dimensional Fast Fourier Transform (FFT) decomposes an image using frequency to describe how the intensity of the image changes at the pixel level. Higher frequency regions represent edges, where low frequencies represent smooth areas or gradients. The drawback of the FFT is that the whole image is decomposed as one object. This limits the FFT's ability to describe details unique to any particular region of an image[2]. Additionally, the use of sine and cosines as basis functions also limits the abillity to manipulate the sharpest regions of an image.

The Discrete Wavelet Transform (DWT) is intended to solve these two particular limitations of the FFT by evaluating custom basis wavelets at every point in the spatial domain independently. This allows images to be filtered uniquely at different regions of the image in the spatial domain. We intend to apply these features of the DWT to remove low-amplitude high frequency information from the image that is assumed to be created from noise. We also intend to remove all low frequency information to reveal the image's line structure.

2. Theoretical Background

Wavelet analysis is based on the inner products of the function of interest, $f(t)$, and a family of wavelets, $\psi_{a,b}(t)$ (1), based on the mother wavelet, $\psi(t)$. The mother wavelet is a mean-zero function the spatial variable, t. The family of wavelets differs by increasing a, lowering the frequency, or increasing b, shifting the window of the wavelet. The one-dimensional Continuous Wavelet Transform (2) generates $W_\psi(f)(a,b)$ that displays the funtion of interest, $f(t)$, as a combination of wavelets at every frequency at every point in the spatial domain[1]. This CWT is a natural extension of the one-dimensional Fourier Transform, but where each point of the domain of the function is being transformed independently.

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\frac{t-b}{a} \tag{1}$$

Date: February 2, 2026.

$$W_\psi(f)(a,b) = \int_{-\infty}^{\infty} f(t)\overline{\psi}_{a,b}(t)dt \tag{2}$$

Similarly, the one-dimensional Discrete Wavelet Transform(3) applies the wavelet family along a discrete set of data. However, the result of the DWT is unique in that to conduct the inner product of the data with the wavelet requires multiple data points. For every iteration of DWT, the scaling variable increases and frequency decreases, causing the number of wavelet coefficients to halve. This forces each iteration of DWT to produce two one-dimensional arrays that contain half the length of data; one is a high-pass filter and the other is a low-pass approximation. This is what makes the DWT an effective tool to compress data as every pass lowers the information by half in each dimension. The two-dimensional DWT extends the DWT in a way that can be naturally applied to images by conducting a horizontal and vertical pass for each iteration. The result is a low-pass approximation image with 3 additional images giving the Horizontal, Vertical, and Diagonal (double high-pass filter) edges with the frequency at that level of DWT.

$$W_\psi(f)(j,k) = \int_{-\infty}^{\infty} f(t)\overline{\psi}_{j,k}(t)dt \tag{3}$$

where $\psi_{j,k}$ is a discrete family of wavelets

$$\psi_{j,k}(t) = \frac{1}{a^j}\psi\frac{t-kb}{a^j} \tag{4}$$

Although the FFT is not the subject of this report, the algorithm makes use of the FFT to apply a Gaussian pre-filter when the user is analyzing a image that appears to have noise. The Gaussian filter applies a bell curve centered on the zero-frequency and eliminating all frequencies outside of a standard deviation denoted by σ. To apply the Gaussian filter, the data is transformed into the fourier domain and multiplied element-wise with the filter. The filtered data is inverse transformed and then the algorithm proceeds. The below equation is the two-dimensional Gaussian filter(5) as used in the algorithm.

$$g(x,y) = e^{\frac{-(x^2+y^2)}{2\sigma^2}} \tag{5}$$

3. Algorithm Implementation and Development

The data was provided as a benchmark 512x512 grayscale image. In order to create an unbiased algorithm, the code only has modifications based on if the user did or did notice noise in a provided, hypothetical image.

For all images, after 3 levels of DWT, the array of low-pass information was zeroed out to eliminate smooth surfaces that lacked edges. If the image had noise, the high-pass information from each level of the DWT was filtered by an increasing number of standard deviations above the mean for the higher frequency levels to avoid including noise. If an image did not have noise, the threshold was lowered by 2x standard deviations at every level to include more fine details without the risk of including noise.

After the algorithm was applied to the original image without noise, it was also tested against various noise levels, various wavelets, different stock images, and user-provided high- and low- light images.

Pseudocode

(1) If the image is noisy, apply the Gaussian filter with $\sigma = 50$
(2) Use pywt.wavedec2 to conduct a 3-level decomposition using the 'Haar' wavelet
(3) Zero all coefficients in the LL array from the last level
(4) Threshold for each level:
    - Normalize the LH, HL, and HH arrays for that level
    - Calculate $\sqrt{LH^2 + HL^2 + HH^2}$
    - Threshold that level using 2x, 4x, or 6x standard deviations above the mean for the increasing frequencies respectively
(5) Use pywt.waverec2 to reconstruct the image from the filtered data
(6) Make all nonzero values equal to one to generate a binary edge map

4. Computational Results

Using the original image with no noise applied and the above algorithm, the below Figure 1 was generated. Although, there is a clear outline of the man and the camera, none of the features in the background are clearly distinguishable. The base algorithmic filter is too aggressive to retain the meaningful high detail edges which human eyes can detect. However, the threshold including the Gaussian pre-filter was clearly able to remove the noise and produce a similar image for $\sigma = 0.1$ and 0.25 in Figure 2. The edge map produced from the image having $\sigma = 0.5$ has all of the same features as the edge maps from the less noisy images, but clearly includes granular noise in the field and sky of the original image. This is probably the best way of demonstrating that a filtering step should be added to identify smooth regions (i.e., the sky and field) and zeroing out this data as any edges in these regions are clearly noise.

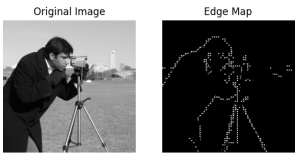

Figure 1. Proof of concept of the edge detection algorithm on the benchmark image with no noise.

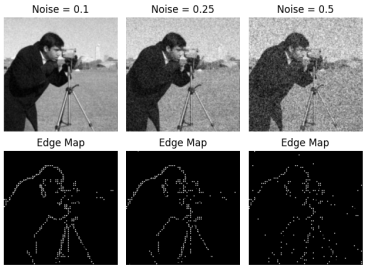

Figure 2. Edge detection algorithm applied to noisy images with σ =0.1, 0.25, and 0.5 respectively.

Next, we attempted the same algorithm on the original image using the 'Daubechies 8' and 'Biorthogonal 2.2' mother wavelets Figure 3. Although 'Daubechies 1' and 'Biorthogonal 1.1' are recommended for edge detection, we chose 'Daubechies 8' and 'Biorthogonal 2.2' because those wavelets are use for smoothing and image compression. In both cases, the smoother wavelet caused blurring, because the edges were detected at the lower frequency transforms and then reconstructed to a larger image. One key takeaway is that for the finer details, a smoother, multi-point wavelet family could be used to remove more noise, but a different, sharper wavelet must be used at the lower frequency to avoid this blurring effect.

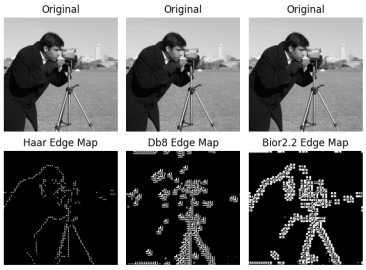

FIGURE 3. Edge detection algorithm using the 'Haar', 'Daubechies 8', and 'Biorthogonal 2.2' mother wavelets.

In order to test the resolution of the edge detection, the below 'house' and 'brick wall' stock images were ran through the above edge detection algorithm, but with the lower threshold as discussed above. The edge maps in Figure 4 clearly show that even with the issue of reconstructing the lower frequency images, straight lines are clearly detectable. This suggests that the threshold needs to be applied to the Vertical, Horizontal, and Diagonal edge maps independently as the edges in the 'camera' photo are not as clearly defined as the 'house' or 'brick wall'. Additionally, we used a 'low light' and 'high light' image of reflective kitchen surfaces to see if the algorithm could differentiate between actual edges and mirror edges in Figure 5. In the 'low light' image the lack of amplitude caused no discernable edges, but in the 'high light' image, the algorithm clearly discerned edges, even if the edges weren't seamlessly connected. This showed the importance of rescaling the image to create more amplitude difference before the DWT to more clearly distinguish the edges.

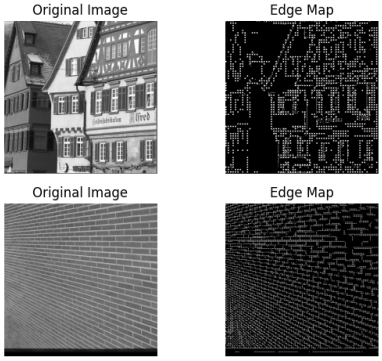

FIGURE 4. Edge detection using stock images that have multiple edges at different scales, showing the algorithms capability without noise.

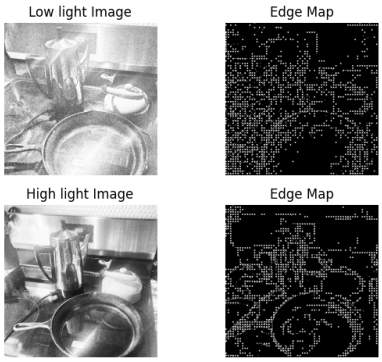

FIGURE 5. Edge detection of highly reflective kitchen surfaces using a phone camera in different lighting conditions.

5. Summary and Conclusions

Given a noisy, grayscale image, the algorithm described above was able to detect the most prominent edges. As it currently stands, the algorithm is most capable with data that has edges spread evenly over the image. However, the the current threshold is too rigid and doesn't adapt to an image with a combination of smooth and non-smooth regions. The reconstruction method also puts more weight on lower frequency (i.e., thicker) edges, which distorts the cumulative image.

Future improvements to the method should be focused on improving how the noise is adaptively filtered at each level and how to reconstruct the image so that the compressed lower-frequency edges are not blurred after reconstruction. Also, a key improvement is to better take advantage of the Discrete Wavelet Transform's ability to filter spatial regions of the image differently as to identify edges that are in the foreground and background, not just which ones have the highest magnitude. Lastly, the image should be decomposed using a combination of different wavelet families for the different frequencies to balance the noise elimination and edge detection.

Acknowledgements

I would like to thank my classmate Samuilad for our discussions about scientific computing and data analysis. I would also like to thank Professor Hosseini for his lectures, notes, and code examples.

References

[1] S. L. Brunton and J. N. Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2 edition, 2022.

[2] J. Kutz. *Data-Driven Modeling & Scientific Computation: Methods for Integrating Dynamics of Complex Systems and Big Data*. OUP Oxford, 2013.