# Wavelet-based Edge Detection

Apoorva Mittal
50170557
Ramanpreet Singh Khinda
50169622

**Submitted on December 14, 2015**

## Literature Review

Edge detection is a very prevalent problem in the field of image processing and has been researched a lot. Some edge detection techniques fall in the spatial domain edge detection like Canny and sobel. The other category is edge detection in the frequency domain. Mallat and Zhong in their work[1] proposed a dyadic wavelet and a corresponding wavelet transform. The idea was to detect edges in the wavelet domain by applying wavelet transform to the image in different scales. It was observed that the edge structures were visible in each subband but the noise levels decreased rapidly along these scales.

Xu et al[2] presented a method in which they enhanced significant structures by multiplying the adjacent DWT levels and Sadler and Swami[3] analyzed the multiscale product in step detection and estimation . The paper[4] we are following does something along these lines; we make a multiscale edge detector by multiplying adjacent subbands and making edge maps at different scales and combining them in the end. Edges are determined as local maxima in the product function after thresholding.

The pipeline majorly includes four image processing techniques: convolution of image with a filter as the wavelets used are essentially filters to be convoluted with the images, downsampling the image to half its size by removing alternate rows and columns from the image, element-wise multiplication of image arrays and upsampling the images to twice their sizes by using some kind of interpolation techniques.

## Introduction

i) Typical wavelet based edge detection based techniques involve edge detection in the HL, HH and LH transforms of an image. In their paper Edge detection by scale multiplication in wavelet domain, Zhang and Baol proposed a new method of wavelet based edge detection in which they use a product of two adjacent sub-bands(obtained by convoluting image signal with the wavelet), followed by thresholding to create a combined edge map from different scales. If two or more edges occur in a neighborhood, they may interfere with each other. With a large scale, the dislocation of an edge will occur if there is another edge in the neighborhood. If we select a small scale parameter, the detection result would be noise sensitive. Generally with a single scale it is very difficult to properly balance the edge dislocation and the noise sensitivity. With the scale multiplication, this problem can be largely resolved. Thus scale multiplication enhances image structures, suppresses noise and reduces the interference of neighboring edges.

ii) In this project, we have implemented the above mentioned algorithm and tested its performance in the presence of Gaussian and Impulse(salt and pepper) noises. Two popular wavelet filters are used for the transform- Haar and Coiflet. Performance of the algorithm with and without noise using the two filters is compared in the end of this report.

## Our Approach

i) We followed these steps:

1. For an image, we first calculate its subbands(LL, LH, HL, HH) using wavelet transform. If the original dimensions of the image were m*n, then each of these subband will be of the size m/2*n/2. The HL, HH and LH bands are used for further edge detection in the later steps of the algorithm.
2. The LL subband is essentially a smaller blurred version of the original image. Wavelet transform is now performed on this image to further obtain 4 subbands of size m/4*n/8.
3. We repeat step 2 twice to get to the m/16*n/16 level.
4. Now that we have the edge information(LH, HL, HH) at 4 different scales, we have to combine them to get the final edges. This is done by upscaling one subband image at the lowest scale(m/16*n/16) twice to its size and then adding it to the corresponding subband image at that scale(m/8*n/8).
5. Step 4 is done all the way up to m*n for all the three subbands individually.
6. Now we calculate the magnitude of the LH, HL and HH to obtain our final edges.
7. Steps 1-6 are performed with different noise levels and wavelets.

ii). We have implemented the code in python and used its numpy and openCV2 libraries for image processing. We used our own implementation for wavelet transform. For noises we used an online implementation.

## Outcomes and Deviations

i) We have used two kinds of wavelets for edge detection: Haar and Coiflet. The low pass and high pass filters for the two are :

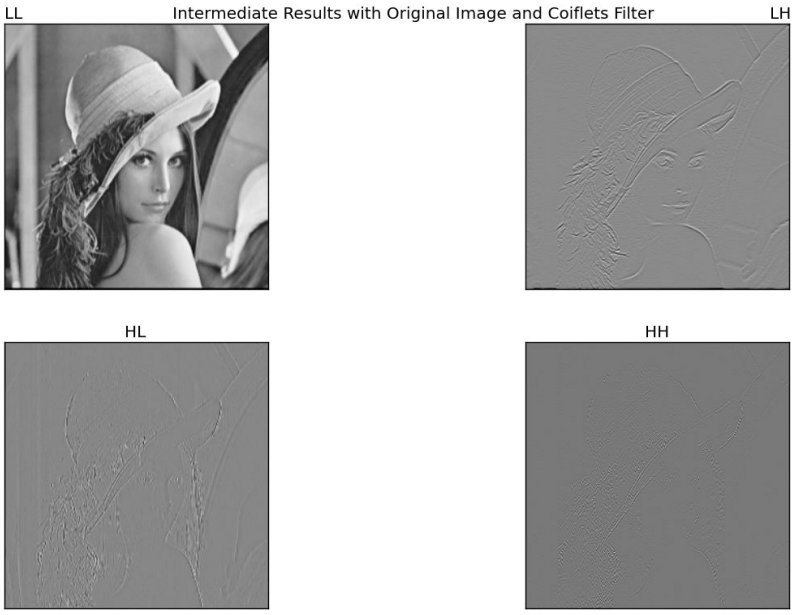
**Haar:**
Low Pass: [1/sqrt(2)  1/sqrt(2)]
High Pass: [-1/sqrt(2)  1/sqrt(2)]

**Coiflet:**
Low Pass: [-0.0157  -0.0727  0.3849  0.8526  0.3379  -0.0727]
High Pass: [0.0727  0.3379  -0.8526  0.3849  0.0727  -0.0157]

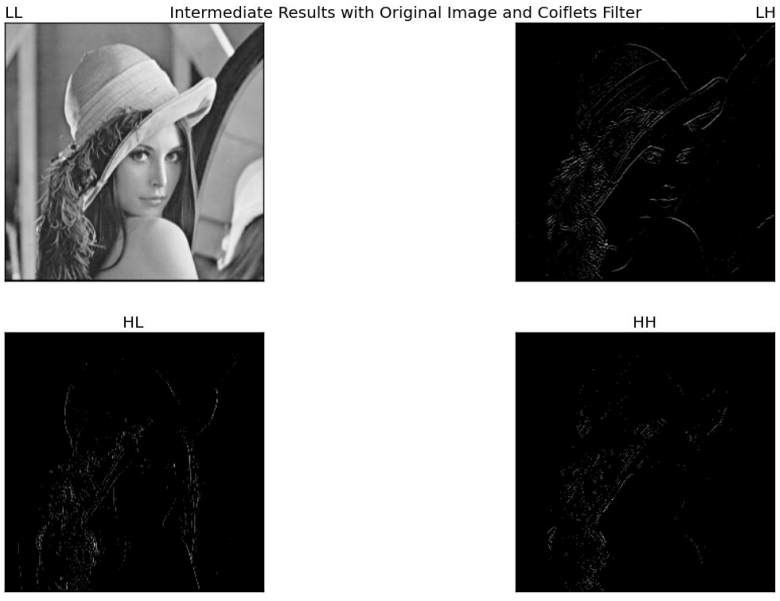
Our code uses two thresholds-one is applied to each of the LH, HL and HH at each level during downsampling (T1). The second threshold is applied to the final edge map in the end (T2).

First the parameters are tuned for edge detection in the **original images**. These were found out to be, on an average for our three test images: **T1= 15, T2= 1e5**

These are the intermediate results obtained for one of the scales for Lena using Coiflet filter:




After applying threshold T1 to the subband, we obtained:




As we can see, the edges are better defined after the threshold is applied.

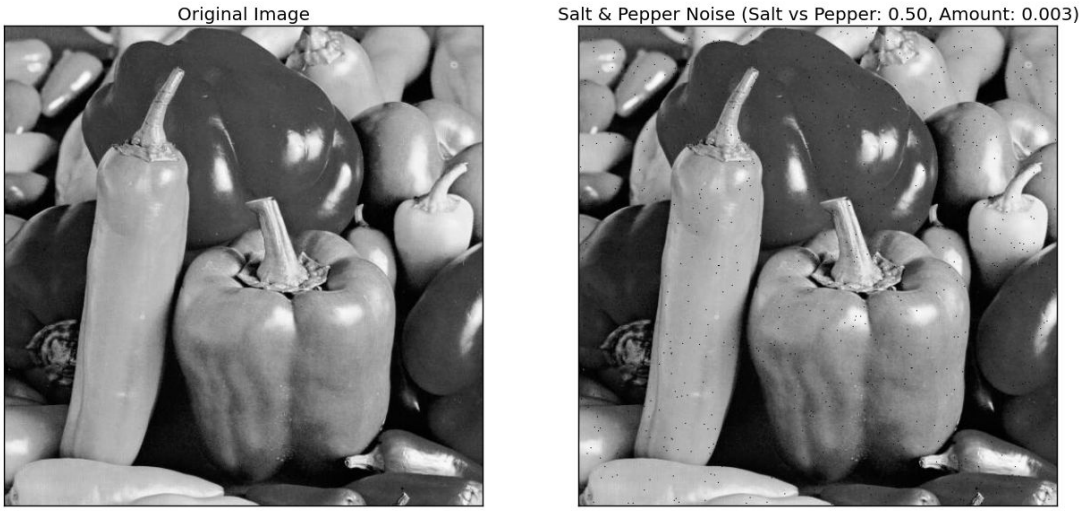
Next we tested how the algorithm is performing when there's noise in the image using two kinds of noises. Both the Gaussian and Impulse noise are characterized by two parameters- mean and variance and the ratio between salt and pepper and the amount of noise.

We ran our system with different values for this noise parameters and tuned our threshold values according to that. In the end we decided to use the following parameter values for the demo because this was significant amount of noise and close to the noise in the referred paper. Also, the results didn't change much when we increase/decrease these noise levels by small amounts.

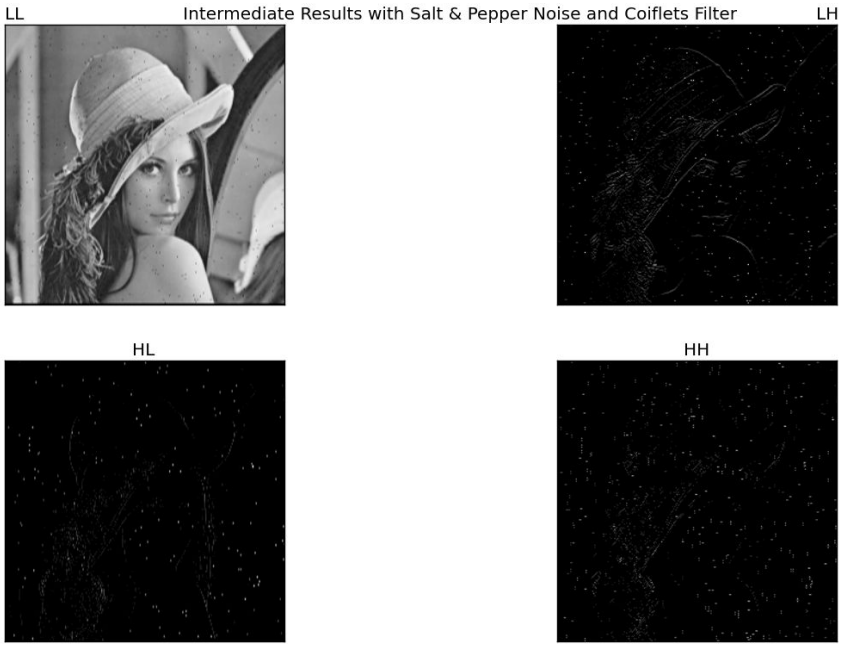
**Gaussian:**
Mean: 20
Variance: 40

**Salt and pepper:**
Ratio: 0.5
Amount: 0.003

Sample images after adding Gaussian and Salt and Pepper Noises:




Similar thresholds are applied for the noisy images as well. With different values.
For the salt and pepper noise with Coiflet wavelet:



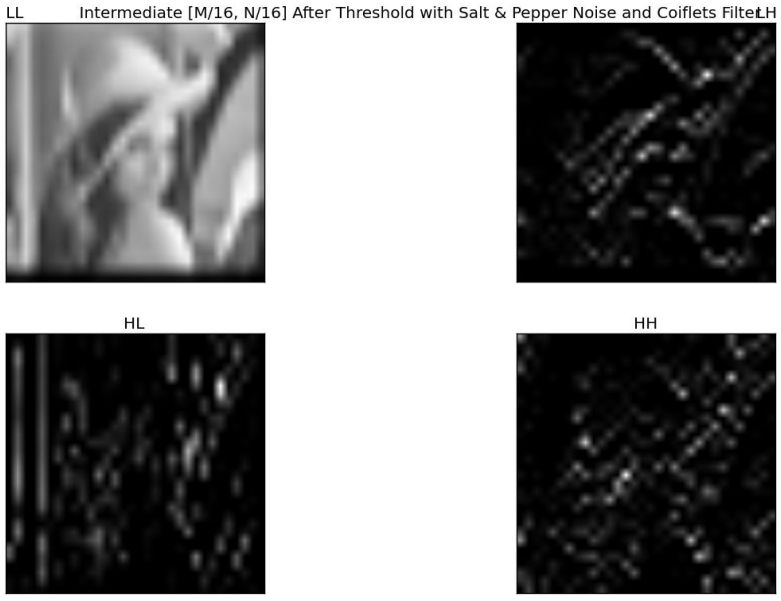
The final results obtained for the three images after applying the thresholds and noise:

**(T1,T2)**

| **Image** | **Original + Haar** | **Original + Cofilet** | **Gauss + Haar** | **Gauss + Coiflet** | **S&P + Haar** | **S&P + Coiflet** |
|---|---|---|---|---|---|---|
| **Lena** | 15, 1e5 | 10, 1e5 | 20, 1e5 | 15, 1e5 | 15, 1e5 | 10, 1e5 |
| **Peppers** | 10, 1e5 | 13, 1e5 | 15, 1e5 | 13, 1e5 | 10, 1e5 | 10, 1e5 |
| **Carriage** | 20, 1e5 | 20, 1e5 | 20, 1e5 | 20, 1e5 | 20, 1e5 | 20, 1e5 |

**Final results obtained from these values are present in the submitted folder.**
**For the final demo, we took the average of these values for each filter and each noise type.**

ii). The algorithm performs well when the image is noise-free. When we run the algorithm on noisy images, as we can see, the noise levels are suppressed when we enlarge the small scale images. So this is the expected behavior.




Since we were not supposed to use any quantitative metrics to analyse the performance of the algorithm, we can only judge by bare eyes. We get good edges with some level of noise in the final result which is much better than what we would have obtained if we would have used any other edge detection technique like LoG.

iii). We always thought of edge detection in noisy images to be an impossible job as noise and edges both are high frequency components of an image. But in this algorithm, the noise is suppressed by downsampling the low frequency image and at the same time, edges are preserved by combining the high frequency images obtained at different scales. So, the algorithm is actually a not-so-complicated way of solving a prevalent research problem in computer vision.

## Software and Program Development

i) As mentioned in the project status report, the pipeline including the downsampling and subband multiplication part was implemented by Apoorva. Ramanpreet implemented the Haar wavelet transform for an image.

The next step was implementing the Coiflet wavelet transform which was done by Apoorva, while Ramanpreet analyzed the results obtained using the Haar wavelet transform and tried different values of thresholds for better results.

Using the implementation for adding noise from stackoverflow, they together analyzed how the results change in the presence of image noise. Finally, the thresholds and noise parameters were fixed to give final results.

ii) As mentioned in the (i) part we implemented most of the code ourselves.
Only the code for adding noise to the image was taken from Internet (http://stackoverflow.com/questions/22937589/how-to-add-noise-gaussian-salt-and-pepper-etc-to-image-in-python-with-opencv).
The filter values for Haar and Coiflet were generated using an inbuilt MATLAB function (http://www.mathworks.com/help/wavelet/ref/wfilters.html)

We did the wavelet transform at different levels obtained on downsampling and also, combining the LH, HL and HH images after upsampling. In the end, it was about experimenting with different threshold values and noise levels.

iii) Understanding the paper and implementing it from scratch was one of the major lessons from the code development in this project.

Earlier we didn't think there was much scope for parallelizing work in the code development. But soon we found ways to contribute to the project individually. Also, we created different functions for most of the things when we realized the need of code reuse in the algorithm. Since the code is pretty long, we also commented it well, for our and the evaluator's convenience. For the demo, we added snippets for showing the intermediate results.

## Summary and Discussion

i) We implemented the technique for edge detection using wavelet transform and their subbands obtained at different levels. We saw how the algorithm handles noisy images by suppressing it in the low sized subbands itself. A major part of the algorithm is thresholding the images obtained at each level to improve the edge maps. We decided on these thresholds by observing the histograms of the images and then fine tuned those by trial and error. The algorithm performs well till a reasonable amount of noise in the image.

ii) Most of the work that we did in the project was taught in the class already. Besides wavelets, convolution and image operations, we have learnt about edge-region-border detection, analyzing and using image histograms, morphological operations, image filtering, working with images in frequency domain: Fourier transform and series, object detection and image understanding.

## Acknowledgement

Besides Dr. Chen for including this interesting project in the coursework, we would like to thank our TA, Bhargava for helping us whenever we were stuck.

## Bibliography

[1] S. Mallat and S. Zhone. “Characterization of signals from -I I multiscale edges, ” IEEE Trans. PAMI, vol. 14, pp. 710-732, 1992.
[2] Y. Xu et a/, “Wavelet transform domain filters: a spatially selective noise filtration technique, ” IEEE Trans. Image Processing, vol. 3, pp. 747-758,1994.
[3] Brain M. Sadler and A. Swami, “Analysis of multiscale products for step detection and estimation, ” IEEE Trans. Information Theory. vol. 45.. D.D.. 1043-1051. 1999.
[4] L. Zhang and P. Bao, “Edge detection by scale multiplication in wavelet domain,” Pattern Recognition Letters, Vol. 23, No.14, pp. 1771-1784, December 2002.