Blaine Martinez
Digital Image Processing Final Project

# Comparisons of Adaptive Median Filters

The purpose of this lab is to compare how two different adaptive median filters perform when it is computed on the Central Processing Unit (CPU) of a computer and on a Graphical Processing Unit (GPU). Median filters are used to clean up images with impulse noise such as Salt and Pepper Noise. Impulse noise is defined as noise that appears at random locations in an image with a pixel value that at or close to the largest or smallest possible pixel values of the image. In the case of Salt and Pepper Noise, the impulse values are at both ends of the image's pixels intensities. The 'Salt' noise are the impulses with high pixel intensities, while 'Pepper' noise are impulses with low pixel intensities. Below are two images, Figure 1 is the original image of a circuit board and Figure 2 is image with Salt and Pepper noise applied.

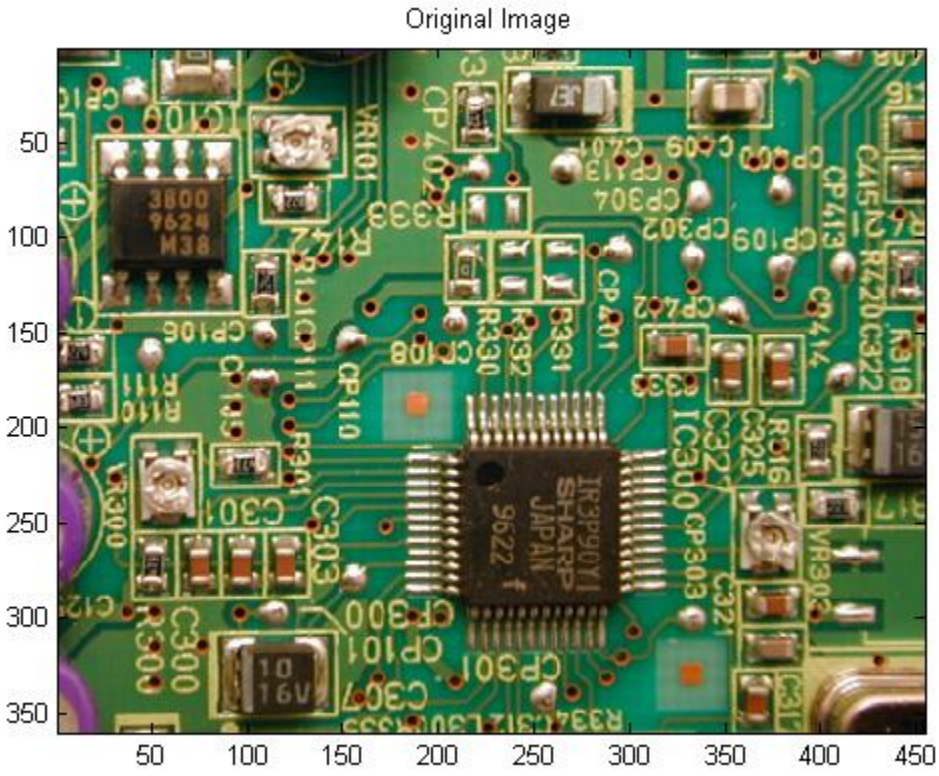


**Figure 1**

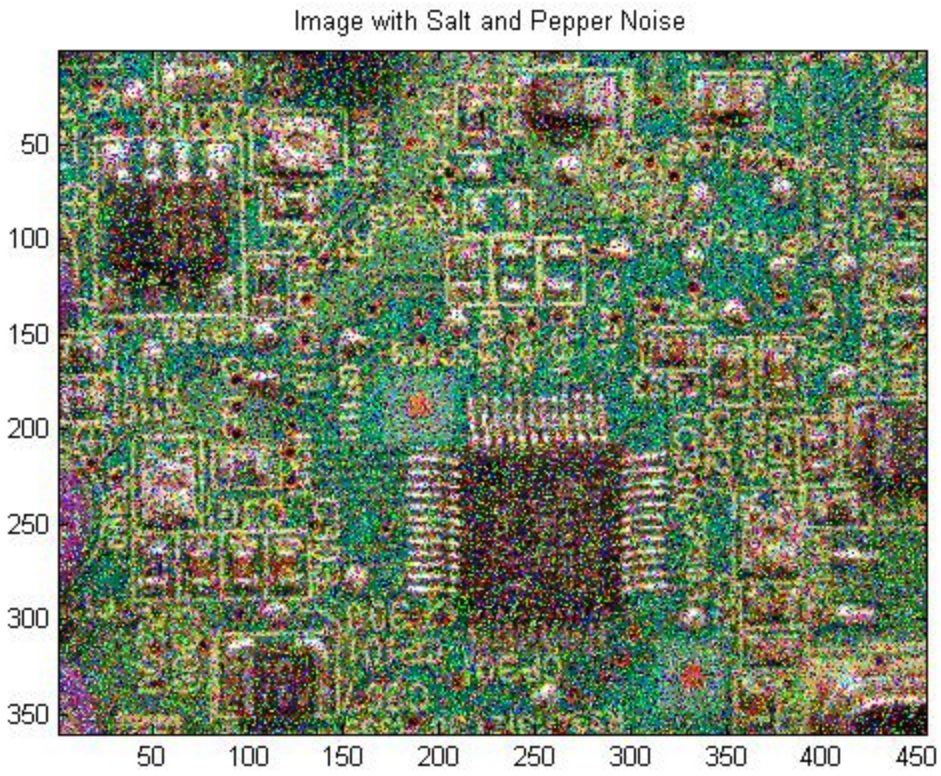


**Figure 2**

The first algorithm works in two phases. The first phase calculates the local maximum, minimum and median pixel values of a small window in an image. The window starts off as a 3x3 mask centered around the pixel being operating on. If the median of that 3x3 window is in between the local min and max, that the algorithm moves into the second phase. If not, the size of the window is increased by one pixel in each direction and the min, max, and medians are calculated again. This process continues until the max window size is reached which then forces the program into the second stage. When the second stage is reached, the pixel being operated is compared to the local min and max values. If the pixel being operated on is not equal to the local min or max values, then the resulting pixel value remains the same. On the other hand, if the pixel being operated on is equal to either the local min or max, then the resulting pixel value is replaced with the local median. The results of this first adaptive are shown below in Figure 3.

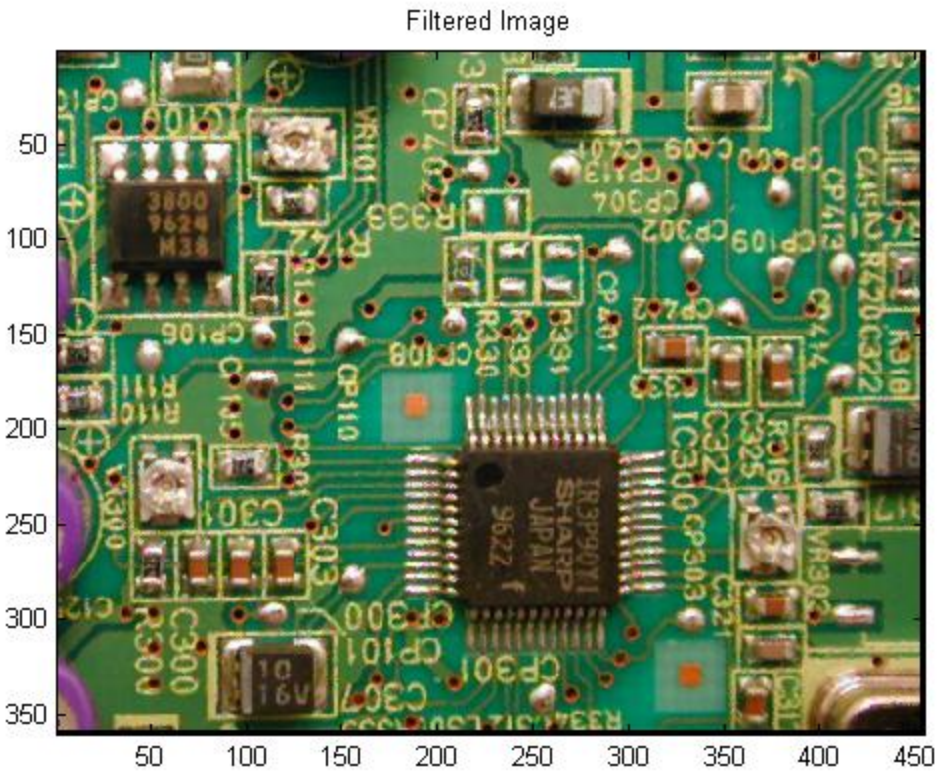


**Figure 3**

The second adaptive median filter I used was works by first identifying the locations of the impulse noise, and then it will apply a local median filter around the impulse location. In order to find the locations of impulse noise, a windowing scheme is used to whether or not impulse noise exists. Figure 4 shows four windows in which a pixel will be compared to in order to determine the location of an impulse pixel. In each of the four windows, the center pixel is compared to two adjacent pixels. In Figure 4, the windows are represented by a 3x3 mask where the pixels that are going to be compared are highlighted in black. The center pixel of this mask is subtracted from the two adjacent highlighted pixels then given a weight. Then the magnitude of the difference between each of these two pixels are added together and divided by the sum of the weights, which yields a weight for each of the four windows form Figure 4. The following two equations are used to calculate the weight of each window and pixel.

$$d^k = \frac{|\sum w_{i+s,j+t} * (f_{i+s,j+t} - f_{i,j})|}{\sum w_{i+s,j+t}}$$

$$w_{i+s,j+t} = \frac{1}{1 + (f_{i+s,j+t} - f_{i,j})^2}$$

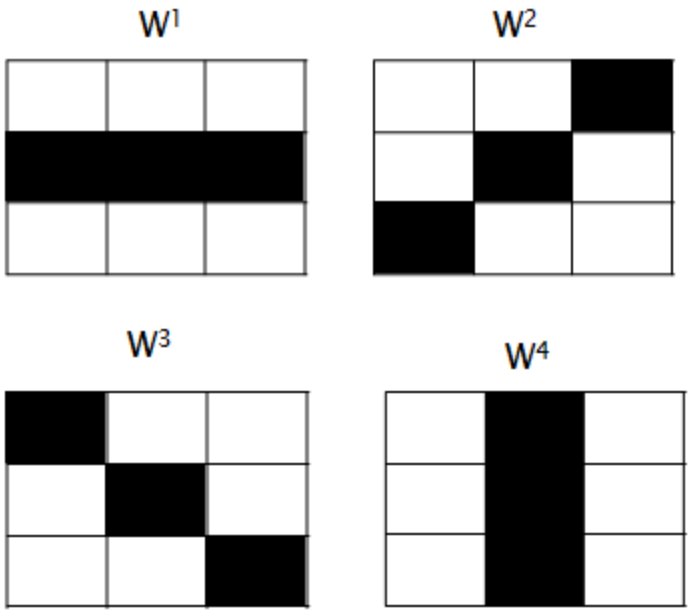


Figure 4

The window weights $d^k$ will be used to determine whether or not the center pixel is an impulse. Of the four $d^k$ window weights the one of with the smallest value is used to compare to a threshold value to determine if the window is centered around an impulse. This technique works because if there was an impulse at the center of the window, the magnitude difference between it and its adjacent two pixels will be large, so if this value is above some threshold value, T, then the center pixel will be reported as an impulse. If the center pixel is not an impulse pixel then the magnitude difference between it and the two adjacent pixels will be very small and as a result, it will stay below the threshold value 'T'. Figure 5 and Figure 6 below show an image and calculated impulse location. Figure 5 is a single color plane of the circuit board image with Salt and Pepper noise applied. Figure 6 shows the identified locations of the impulse values from the image in Figure 5. The way the algorithm is set up is by creating an array the same size of the noisy image full of zeros. The when an impulse has been found, the pixel locations are marked by giving the generated array a 1 in the location of the impulse. So when looking at Figure 6, all the black pixels represent the locations where a 'normal' pixel has been found and all white pixels represent the location where an impulse has been detected.

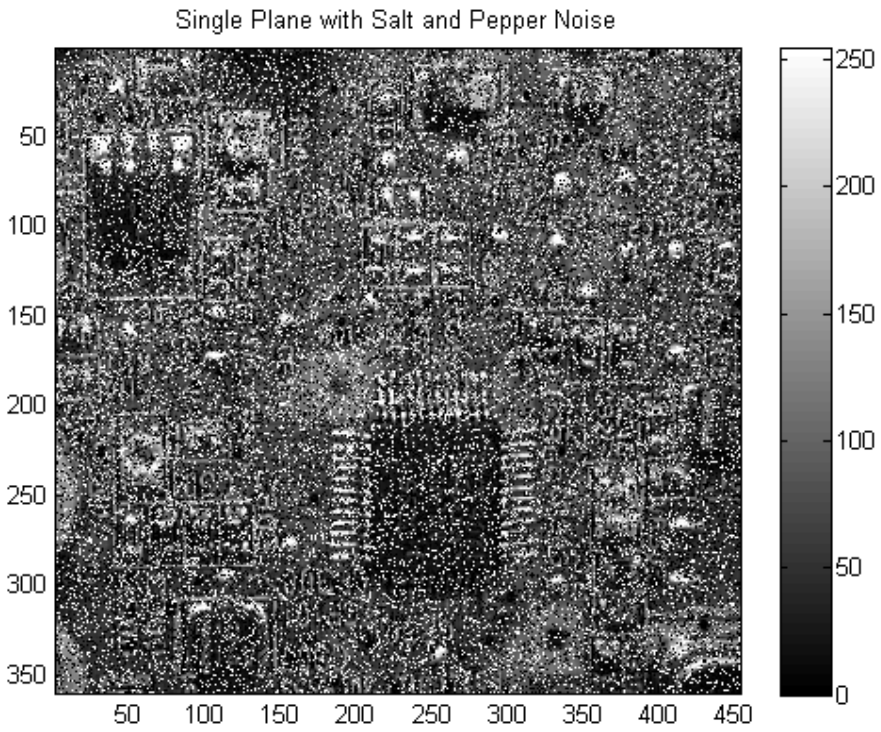


Figure 5

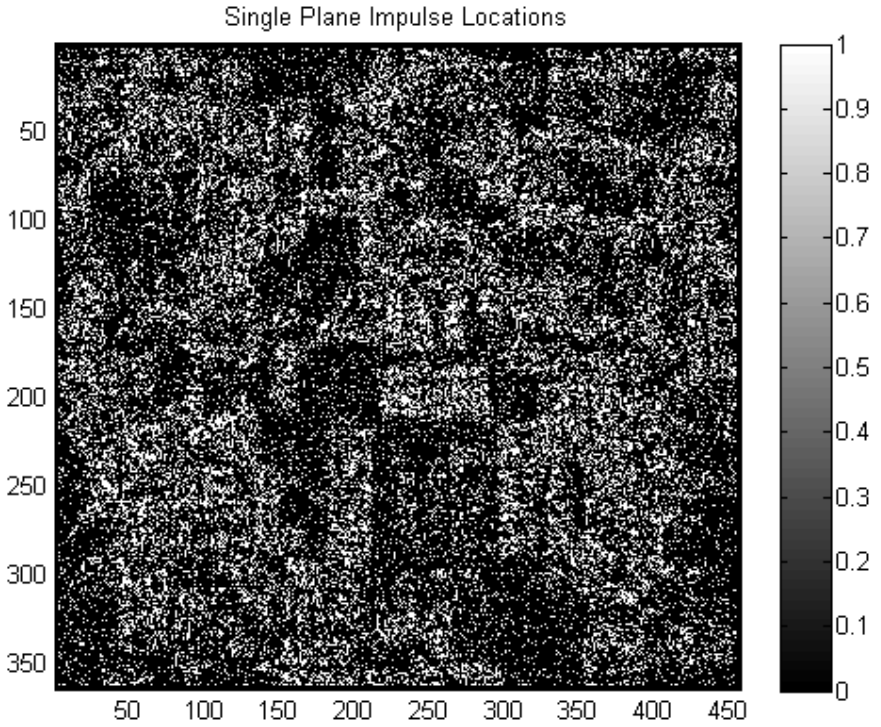


Figure 6

Once all the impulse locations have been found, a local 3x3 median filter is applied around the impulse locations. The results of this field are shown below in Figure 7.

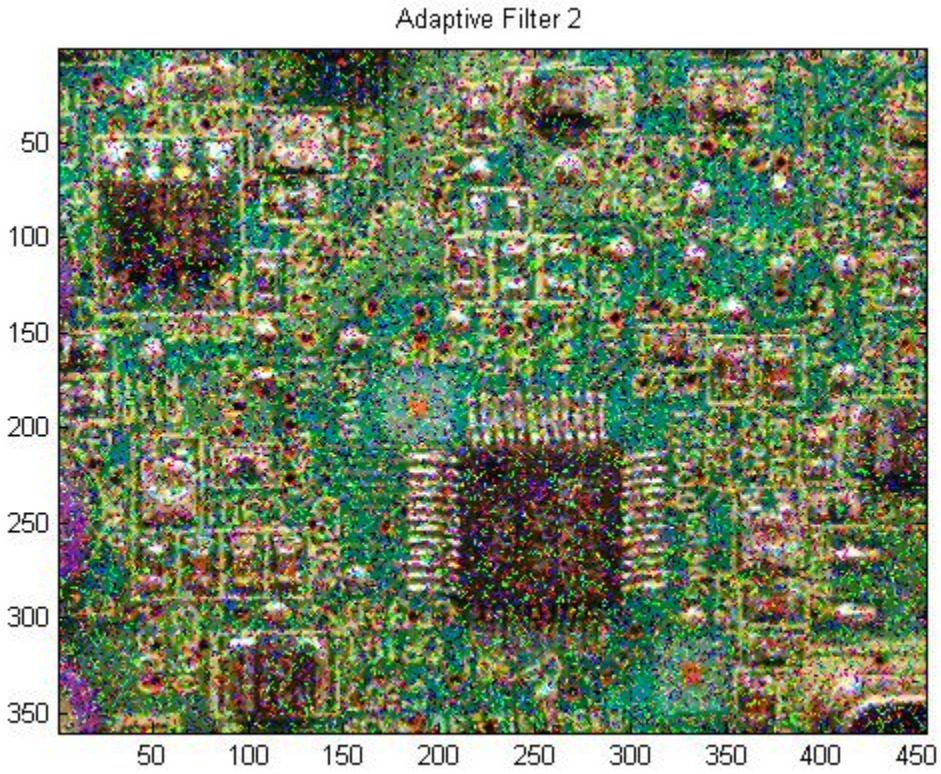


**Figure 7**

As it can be seen in Figure 7, the results of this filter didn't work out as well as the first adaptive median filter. This is due mainly because the original algorithm for this filter required that the impulse pixels be replaced by a certain weighted average of the surrounding pixels. My implementation of this weighted average did not work out so I replaced that portion of the algorithm with a normal 3x3 median filter. From the results in Figure 7, it is shown that this modified algorithm is not very effective in filtering our impulse noise.

The implementations of both of these algorithms were on Matlab using both the CPU and GPU, and the GPU portion used Accelereyes Jacket software. The following times were measured for performing both of these adaptive median filters on the CPU and GPU.

Adaptive Median Filter 1: CPU Time = 16.4 s, GPU Time = 65.6 s, Speed Up Ratio = 0.014

Adaptive Median Filter 2: CPU Time = 9.7 s, GPU Time = 10.01 s, Speed Up Ratio = 0.974

From the results, it can be seen that the filters GPU didn't speed up either of the algorithms. This is due to the fact that the GPU specializes in manipulating large sets of arrays at once, but my implementation of these algorithms performed the operations sequentially because I had trouble trying to embed 'for' loops inside each other with the GPU.

References:

*Digital Images Processing, Third Edition*
Rafael C. Gonzalez, Richard E. Woods
PEARSON, 2008

*Simple Adaptive Median Filter for the Removal of Impulse Noise from Highly Corrupted Images*
Haidi Ibrahim, Nicholas Sia Pik Kong, Theam Foo Ng
IEEE Transactions on Consumer Electronics, Vol 54, No.4 Novermber 2008

*Impulse Noise Removal Using Directional Difference Based Noise Detector and Adaptive Weighted Mean Filter*
Xuming Zhang, Youlun Xiong
IEEE SIGNAL PROCESSING LETTERS, VOL. 16, NO. 4, APRIL 2009

*A New Adaptive Switching Median Filter*
Smail Akkoul, Roger Ledee, Remy Leconge, Rachid Harba
IEEE SIGNAL PROCESSING LETTERS, VOL. 17, NO. 6, June 2010