Implementation Of Adaptive Median Filtering Method For Impulse Noise Detection

Jansi T
Assistant Professor, Dept of Computer Application
Sri Paramakalyani College, Alwarkurichi.

Abstract- *This paper describes an image denoising filter for salt & pepper noise is proposed. An adaptive median filter is a great tool to have to remove salt and pepper noise. The problem with implementing the adaptive median filter is the amount of time it takes to perform all the necessary calculations on all the layers of the image. One method to help decrease the amount of time to complete all processes is to implement the algorithm on a GPU rather than a CPU. Results acquired show that there was an improvement in processing time, but the improvement was not as much as expected. To evaluate the performance of proposed filter, both quantitative and qualitative techniques are used and a comparison is carried out between proposed filter and other standard filters, it is observed from experimental results that proposed filter performs remarkably well in filtering and preserving the image detail as compared to well known standard filters.*

Keywords- Adaptive Median Filter, Impulse Noise, denoising.

I. INTRODUCTION

Digital images get contaminated by impulse noise during image acquisition/transmission. The intensity of impulse noise has the tendency of being either relatively high or relatively low. Thus, it could severely degrade the image quality and cause great loss of information details. So it becomes important to suppress this noise in the images before some subsequent processing, such as edge detection/extraction, image segmentation and object recognition. In the literature, various filtering techniques have been proposed for removing impulse noise and it is well-known that linear filters could produce serious image blurring. The adaptive median filter was implemented using MATLab. In order to have MATLab perform at its peak performance all the code had to be vectorized. Once all the code was vectorized, GPU software was selected. The selected GPU software was Jacket by Accelerates since the software was easy to implement with MATLab. Jacket is a GPU software package that allows the user to implement MATLab on the GPU. Jacket supports many MATLab functions and includes the image processing toolbox with pre-written MATLab functions for use. Jacket makes the transition from performing calculations on the GPU and the CPU extremely easy. To perform the processing on the GPU the key 'g' is put in front of most functions. This filter is to be robust in removing mixed impulses with high probability of occurrence while preserving sharpness. Three new methods based on SAM, Circular SAM (CSAM), Weighted SAM(WSAM),Weighted CSAM(WCSDAM).The switching median(SM) filters is more effective than uniformly applied methods, in which a median and weighted median (WM) filter is usually used to detect impulses. However, the median based detector fails to distinguish thin lines from impulses. The basic operation involves two processes,

1. A Filtering Process
2. Adaptation Process

In which aims to adjust the filter parameters (filter transfer function) to the (possibly time varying) environment. Separable 2-D median filter preserves 2-D edges.

A. Purpose

- Remove impulse noise
- Smoothing of other noise
- Reduce distortion, like excessive thinning or thickening of object boundaries.

B. Impulse noise

Impulse noise affects image pixel by pixel and not the whole area of an image. This noise is introduced due to acquisition/transmission errors. Impulse noise can be of two types:

- Random Values Impulse Noise (RVIN)
- Salt & pepper noise

The Salt and Pepper (SP) noise is also called as fixed valued impulse noise, it will take a gray level value either minimal (0) or maximal (255) (for 8-bit monochrome image) in the dynamic range (0-255). It is generated with the equal probability. In the case of salt and pepper noise, the image

pixels are randomly corrupted by either 0 or 255. That pixel is replaced by either white value (255) or black value (0) that's why it is called as salt & pepper noise. For each image pixel at location (i,j) with intensity value O(i,j), the corresponding pixel of the noisy image will be X(i,j), in which the probability density function of X(i,j) is

$$P(x) = \begin{cases} p/2 \text{ for } x = 0 \\ 1 - p \text{ for } x = 0_{i,j} \\ p/2 \text{ for } x = 255 \end{cases}$$

Random Valued Impulse noise (RVIN) is also called as variable type impulse noise which also replaces some pixels with random values like salt & pepper noise but in the range [0 255](in case of 8 bit gray scale image). The algorithm has three main purposes:

(a) To remove 'Salt and Pepper' noise.
(b) To smoothen any non impulsive noise.
(c) To reduce excessive distortions such as too much thinning or thickening of object boundaries.

II. IMPLEMENTATION AND TESTING

An adaptive median filter algorithm is a great tool to have to remove salt and pepper noise from images due to the simplicity of the algorithm, the robustness of the algorithm, and the conditionals that are inside the algorithm. The conditionals are simple enough to calculate quickly and easily, but each conditional calculation has to be made at each iteration in the algorithm so the overall computing cost can be quite expensive when dealing with larger images and all layers of the image. The GPU could decrease the time of overall filtering because the GPU is amazing at doing parallel computing. The GPU will calculate each loop iteration and every layer of the image simultaneously rather than the CPU which calculates everything sequentially. The adaptive filter works on a rectangular region $\mathbf{S_{xy}}$. The adaptive median filter changes the size of $\mathbf{S_{xy}}$ during the filtering operation depending on certain criteria as listed below. The output of the filter is a single value which the replaces the current pixel value at (x, y), the point on which $\mathbf{S_{xy}}$ is centered at the time. The following notation is adapted from the book and is reintroduced here:

$\mathbf{Z_{min}}$ = Minimum gray level value in $\mathbf{S_{xy}}$.

$\mathbf{Z_{max}}$ = Maximum gray level value in $\mathbf{S_{xy}}$

$\mathbf{Z_{med}}$ = Median of gray levels in $\mathbf{S_{xy}}$

$\mathbf{Z_{xy}}$ = gray level at coordinates (x, y)

$\mathbf{S_{max}}$ = Maximum allowed size of $\mathbf{S_{xy}}$

The adaptive median filter works in two levels denoted Level A and Level B as follows:

Level A: A1= $\mathbf{Z_{med} - Z_{min}}$

A2= $\mathbf{Z_{med} - Z_{max}}$

If A1 > 0 AND A2 < 0, Go to level B

Else increase the window size

If window size <= $\mathbf{S_{max}}$ repeat level A

Else output $\mathbf{Z_{xy}}$.

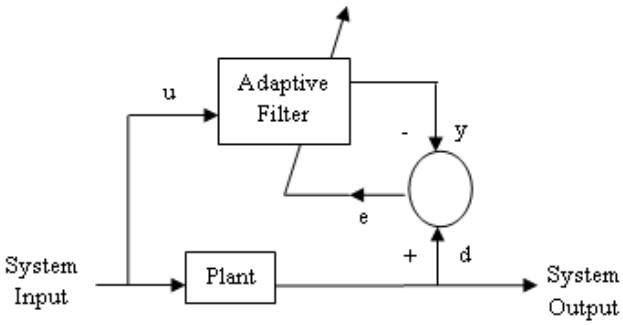
Level B: B1 = $\mathbf{Z_{xy} – Z_{min}}$

B2 = $\mathbf{Z_{xy} – Z_{min}}$

If B1 > 0 And B2 < 0 output Z_{xy}

Else output Z_{med}.

III. SYSTEM ARCHITECTURE



Adaptive Median Filter

IV. RESULTS

The first test was to compare the final results of the GPU calculations to the CPU calculations to make sure the algorithm was still being performed correctly. The final results show that there is no visual difference between the two results so the algorithm still works as it is supposed to. The filtered image results of one layer of the original image are shown in Figure 1.

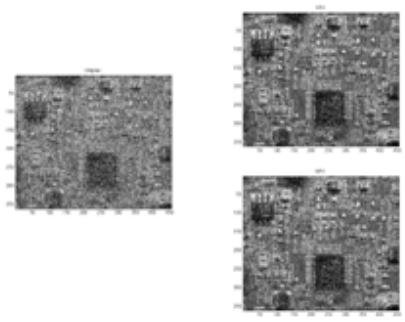


Figure 1 : original(left);CPU (upper right);GPU (lower right)

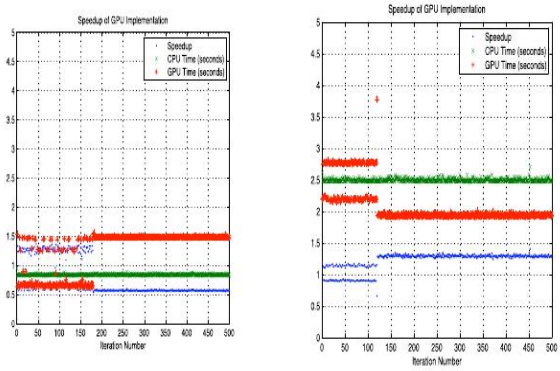


Figure 2: Total Time of Calculations

V. CONCLUSION

In conclusion, Jacket is great software to use for GPU processing and has much potential. The reason why this experiment did not produce the results claimed by Accelereyes (10 to 20 times speedup) is because a GPU that was used was not a dedicated device for processing data. It was the general GPU for the entire computer. Also, the way the filter was implemented required sending data back to the CPU for computation before continuing on the GPU. This data transferring cost a lot of process time on the GPU, if we could implement all the functions on the GPU, eliminating the data transfer problem, the process would be much faster. Overall, the GPU is faster at producing similar results to the CPU due to its ability to process several algorithms simultaneously instead of the sequentially.

REFERENCES

[1] Gonzalez, Rafael C., and Richard E. Woods. *Digital Image Processing*. Upper Saddle River, NJ: Prentice Hall, 2008. Print.

[2] Gonzalez, Rafael C., Richard E. Woods, and Steven L. Eddins. *Digital Image Processing Using MATLAB*. Upper Saddle River, NJ: Pearson Prentice Hall, 2004. Print. *Jacket Documentation - Fast GPU Software for MATLAB and C/C*. Web. 18 Apr. 2011. <http://wiki.accelereyes.com/wiki/index.php?title=Main_Page>.

[3] P. Perona and J. Malik, Scale-space and edge detection using anisotropic diffusion, IEEE Trans. Pattern Anal. Mach. Intell., vol. 12, no. 5, pp. 629–639, May 1990.

[4] C. Tomasi and R. Manduchi, Bilateral filtering for gray and color images, in Proc. IEEE Int. Conf. Computer Vision, 1998, pp. 839–846.

[5] K. J. Overton and T. E. Weymouth, A noise reducing preprocessing algorithm, in Proc. IEEE Computer Science Conf. Pattern Recognition and Image Processing, Chicago, IL, 1979, pp. 498– 507.

[6] H. Lin and A. N.Willson Jr., Median filters with adaptive length, IEEE Trans. Circuits Syst., Vol. 35, no. 6, pp. 675–690, Jun. 1988.

[7] T. Sun and Y. Neuvo, Detail-preserving median based filters in image processing, Pattern Recognit. Lett., vol. 15, pp. 341–347, Apr. 1994.

[8] A. C. Bovik, T. S. Huang, and D. C. Munson, A generalization of median filtering using linear combinations of order statistics, IEEE Trans. Acoust., Speech, Signal Process., vol. ASSP-31, no. 6, pp. 1342–1350, Dec. 1983.