Fast Paralleled Removal Method of Impulsive Noise Using Edge Strength

Yusuke Koshimura^{a*}, Takashi Miyazaki^a, Yasuki Yokoyama^a, Yasushi Amari^a, Hiroaki Yamamoto^b

^aNational Institute of Technology, Nagano College, 716 Tokuma, Nagano city, Nagano Prefecture and 381-8550, Japan

^bShinshu University, 4-17-1 Wakasato, Nagano city, Nagano Prefecture and 380-0928, Japan

*Yusuke KOSHIMURA: 17906@g.nagano-nct.ac.jp

Takashi MIYAZAKI: miya@nagano-nct.ac.jp

Abstract

The Median Filter is known as a method for effectively removing impulsive noise. However, this method occurs deteriorating image quality since the filtering process is also applied to non-noise pixels. Therefore, a method called the Switching Median Filter, improved capable of distinguishing between noise pixels and non-noise pixels has been proposed. Our Multi-directional Switching Median Filter is one of which has been improved such as same. As the first step in this method, the detection and removal processing of pixels considered as noise are performed by using a 2x2 detection operator from four detections. As the second step, each density value of the noise pixel is corrected to restore the image by averaging process. However, this method as same as other methods takes time to remove noises in the high resolution image. Also, it requires an optimal threshold in order to make the image quality better for each image.

In this paper, we propose a novel method that is parallelized to improve processing speed by allocating pixels of this MSMF to the threads of GPU, and whose threshold value is calculated according to the average value of edge strength around the target pixel. And we report the effective performance of image denoising for the difference in image noise ratio compared some conventional methods.

Keywords: image denoising, Switching median filter, GPU, parallel algorithm.

1. Introduction

In recent years, the super-high resolution digital images such as 4k and 8k are remarkably improved, so that the more effective noise removal methods are required. When digital images are captured with C-MOS or CCD due to influence

of the variation in sensitivity among light receiving cells, impulsive noise is superimposed on the images and deterioration is likely to occur.

The Median Filter method (MF) is effective to remove such as these noises ^(1,2), but this method performs processing on all pixels, so degrades non-noise pixel. As a method solving this problem, the Switching Median Filter method (SMF) using switching type noise detection filter has been proposed. Therefore, degradation of non-noise pixels caused by MF is able to be suppressed and both denoising and edge preservation are able to be suppressed and both denoising and edge preservation are able to be achieved ⁽³⁻¹²⁾. From the perspective of hardware implementation, a simpler algorithm is preferred. However, most of methods based on SMF tend to become a complicated algorithm.

Therefore, we have proposed the Multidirectional scanning SMF (MDSM), which has both original signal preservability and noise elimination capability, by using a difference by the 2x2 size as a noise detection operator ⁽⁸⁾. The feature of the MDSM is suitable for parallel processing, we have reported the improved parallel algorithm of MDSM, called GPUSMF by using GPGPU ⁽¹³⁾.

In this paper, we proposed a novel paralleled method that improved GPUSMF by using variable threshold of noise detection calculated based on edge strength. We report the experiment results of noise removing performance using this new developed method.

2. Conventional method

Multi-directional SMF is a simple algorithm that the density difference of interest pixel calculated using a small detector of 2x2 size, which is characterized by a small calculation amount. Multidirectional SMF consists of two

steps: one is a processing of noise detection and removal with a MF, the other is an average processing for improving image quality of the restored image.

2.1 First step

At the time of noise removal, since the restored pixel is reflected on the original image, there is a noise removed area and a noise pre-removal area in the MF window. Therefore, in the noise detector of 2x2 size, other pixels except the target pixel are non-noise pixels or noise removal pixels. Further, since the pixel information used for each scanning direction is different, removal is performed from a plurality of scanning directions, thereby reducing bias.

Duplicate target image deteriorated due to noise is duplicated by the number of scanning direction. The proprietary SMF with 2x2 noise detection operator is recursively applied to the duplicated images from 4 different scanning directions. Calculation of the density difference value is calculated by multiply-add operation with the noise detection operator. The density different value g_{ij} of the target pixel is expressed by equation (1) and (2). Where X_{ij}^{WD} is the set of pixels in the window of the detector, D is the detecting operator, and $*$ is the product sum operator.

The comparison is made by comparing the density difference value g_{ij} with the set threshold value (T_D).

$$D = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \quad (1)$$

$$g_{ij} = |X_{ij}^{WD} * D| \quad (2)$$

2.2 Second step

The noise detection result in the first step is different for each scanning direction, and the obtained noise reduction image is also different. This is because the noise detection window is small relative to the MF window. For these different noises reduced images, a more excellent removed image is obtained by calculating the density average of the positions. By performing this processing, it is possible to suppress the possibility that the effect of erroneous detection, which is likely to occur in scanning in only one direction, becomes an average processed pixel in noise determination in which noise due to multidirectional scanning and non-noise.

3. Outline of GPGPU

3.1 CUDA

This section describes CUDA (Compute Unified Device Architecture)^(14,15). CUDA is a general purpose parallel computing platform and programming model for GPU. We call GPGPU (General-Purpose computing on Graphics Processing Units) a technology that uses GPU for general purpose computation. CUDA's architecture includes new components designed for GPU computing, and the limitations so far have been overcome. This architecture is not specialized in graphics, it is designed to use an instruction set prepared for general purpose computation. CUDA C is cross-platform and has openness to accommodate calls from various languages. While GPU is good at simple and parallel executable calculations, it has a feature that is not good at calculation with many branches and calculation with low parallel degree. In this paper, we develop programs with CUDA 9.0.

3.2 Program composition

The CUDA program performs calculation using both CPU and GPU. A CPU called a host reads a program or data stored in the main memory, processes the data according to an instruction described in the program, and writes the result in the main memory. On the other hand, when the GPU performs processing, it reads the program and data from the video memory, and writes the result to the video memory after processing. GPU and video memory as "device", programs to run as GPU is called "kernel". Hosts and devices are handled independently of each other. In other words, in processing by CUDA, data is first recorded in the host memory, and then its contents are transferred to the corresponding device memory.

3.3 Thread structure

Execution on the device side starts by calling the kernel function from the host side. A large number of threads are generated in the device, and statements defined by the kernel function are executed by each thread. A thread is the minimum unit for executing a program when the kernel is operated.

In the proposed method in this paper, parallelization is performed by associating one pixel of an image with this thread.

4 Parallel Denoising Algorithm using GPGPU

In this algorithm, each thread corresponds to each pixel in the image, and since the kernel statement is executed for

each thread, the processing of removal noise is performed almost simultaneously for all the pixels. However, this parallel processing algorithm makes it difficult to perform recursive processing and sequential removal processing of the algorithm processed by using only CPU such as Multi-directional SMF. Therefore, we try to improve the restored image by using iterative processing instead of that two processes.

4.1 GPUSMF

In the parallelization method GPUSMF using GPU proposed by Koshiyama and others, in order to reproduce the operation of the Multi-directional SMF on the GPU, the noise image is duplicated on four sheets and the processing by the noise detection operator corresponding to the Multi-directional SMF. The reason why duplication is made on four images is to perform the same processing as the Multi-directional SMF from four directions at the same time. After that, iterative processing is performed 15 times, and the four different removal results obtained are averaged to obtain a restored image.

4.2 DegMF

In GPUSMF, the number of times to use MF for noise removal increases and processing time increases due to averaging processing.

In this method, called DegMF (Degree Median Filter), instead of raster scanning for noise detection from different directions of Multi-directional SMF, detection operators corresponding to four directions are used for the target pixel at the same time as the parallel processing. When one or more detection operators detect the noise from these four directions, the MF processing is performed with the target pixel as the center pixel. The features of this method are reduction of the four directional scanning processing of MF executed in the GPUSMF method and elimination of the averaging processing carried out for improving the quality of the restored image. By deinstalling these processes, it is possible to simplify the algorithm and improve the processing speed.

4.3 Proposed method

This method consists of two steps of reducing noise by preprocessing and then removing noise using edge information. Details of each step are shown below.

4.3.1 First step

The first step plays a role to remove noise using

algorithm of GPUSMF. Outline of the first step is shown Fig. 1. Noisy image is copied to memory area corresponding to each of the four scanning directions. Then, noises are removed from these copied images by the Multi-directional SMF corresponding to each direction.

In this step, the iteration process is not used because of main denoising process in next step, namely, the first step is pre-processing of noise reduction.

4.3.2 Second step

Fig. 1 shows outline of the first step. The strength of the edge is calculated in the region centered on the target pixel x_{ij} of the images obtained from first step. This region is a rhombus with a diagonal line as 2 pixels. A part of rhombus is used corresponding to scanning direction, not all of the rhombus pixels. The used region for calculating the edge strength is shown Fig. 2. It shows example of scanning direction “upper left”, and the region colored with gray pixels is used for calculation. And determine value of edge strength in target pixel by averaging each edge strength, in this calculate using noise detection operator too. Edge strength is multiplied by edge loading (W_G) and then set threshold value (T_S) is added. By this processing, this threshold value is able to change based on information of around the target pixel.

The main part of algorithm example is shown below.

Algorithm (i, j)

```

1:  $d_1 \leftarrow |x(i, j) + x(i - 1, j - 1) - x(i - 1, j) - x(i, j - 1)|$ 
2: if  $d_1 \geq T_D$  then
3:    $y_1 \leftarrow \text{MED}(X_{i,j})$ 
4: else
5:    $y_1 \leftarrow x(i, j)$ 
6: end if
7:  $\text{edge}(i, j) \leftarrow \text{AVERAGE}(\text{rhombs})$ 
8:  $T_U = W_G \times \text{edge}(i, j) + T_S$ 
9: if  $d_1 \geq T_S$  then
10:   $y_1 \leftarrow \text{MED}(X_{i,j})$ 
11: else
12:   $y_1 \leftarrow x(i, j)$ 
13: end if

```

Fig. 1. Outline of first step.

Fig. 2. Outline of Second step.

5 Experiments and Results

5.1 Experimental method

- Superimpose a random value as impulsive noise on the test image
- Remove noise by the proposed method and remove noise by the conventional methods.
- Evaluate the speed of removal processing.
- Evaluate the quality of restored image.

First, low resolution images (256x256) shown Fig.3 are used to research on effectiveness corresponding to noise ratio of proposed method in the case of low resolution image.

In addition, in order to investigate the differences and the effectiveness of each method accompanying the increase in resolution, prepare three sizes of test images shown Fig.4 512x512, 1024x1024, 2048x2048. For the three kinds of

Fig.3. Low resolution images (256x256).

Fig.4. High resolution images.

noise overlap rates of 10%, 30%, and 50%, the differences of the restored image is examined by the differences of the image resolution.

In this paper, fix the threshold value used in each method with 40.

5.2 Superimposed noise and evaluation index

We describe a type of noise used in the experiment. In the work of noise removal, it is normally used two types of noise, one is salt-and-pepper noise, and the other is random value noise. This session shows two types of noise model which have the density variation rate of the noise. One is the salt-and-pepper noise which has only two values of 0 or 255 as a density value of superimposed noise, and the other is the random value noise which takes the value from 0 to 255. The

random noise level V within a certain gradation is expressed by

$$V = v(U_{max} - U_{min}) \quad (3)$$

Here, v is the gradation variation rate, U_{max} is the maximum gradation value and U_{min} is the minimum value that can be expressed in the image. Therefore, this type noise is a model close to actual noise.

When the pixel position is (i, j) , the signal of the degraded image is $x(i, j)$, and the signal of the original image is $x_0(i, j)$, the noise model is defined by

$$x(i, j) = \begin{cases} x_0(i, j) & : p = q - 1 \\ \text{RND}(U_{max}, -V) & : p = q/2 \\ \text{RND}(U_{min}, V) & : p = q/2 \end{cases} \quad (4)$$

Here, $\text{RND}(c, d)$ is a uniform random number that takes the value of the interval $(c, c + d)$, p is the possibility that the value is selected, and q is the noise rate. That is, the salt-and-pepper noise is $v = 0.0$.

Next, we describe the evaluation index in the evaluation of restored image quality. PSNR (Peak Signal to Noise Ratio) is used in this experiment. The formula used for PSNR calculation is shown as

$$\text{PSNR}[\text{dB}] = 10 \log_{10} \frac{255^2}{\text{MSE}} \quad (5)$$

$$\text{MSE} = \frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N (x_{ij} - y_{ij})^2 \quad (6)$$

Here, MSE (Mean Square Error) is the mean square error between the original image and the restored image. The number of pixels in the horizontal direction is M , the vertical direction is N , the current pixel density of the coordinates (i, j) is x_{ij} , and the restored pixel density is y_{ij} . Image quality estimation using PSNR which is popular in image processing research takes a higher value as it is closer to the error between the origin image and the restored image.

5.3 Experiment results

The results of comparing the average image quality of the noise removed image by proposed method and some other methods are shown in Fig. 5.

Fig. 6 shows the results of comparing the average processing speed of denoising by using GPU methods.

Figures 7-9 show restored image quality, and figures 10-12 show denoising speed.

From Fig. 5, it can be seen that the restored image quality tends to decrease more than the method excluding MF in the proposed method, when the noise rate increases.

That is a reason why the noise ratio is increased and the noise cannot be removed sufficiently in the preprocessing which affects the calculation of the edge strength.

However, it can be seen that the increase in processing speed is suppressed compared to GPUSMF as shown in Fig. 8.

Fig.5. Result of comparing image quality.

Fig.6. Result of comparing processing speed.

Fig.7. Noise ratio 10%.

Fig.10. Noise ratio 10%.

Fig.8. Noise ratio 30%.

Fig.11. Noise ratio 30%.

Fig.9. Noise ratio 50%.

Fig.12. Noise ratio 50%.

From these results, the processing speed of the proposed method is roughly midway between GPUSMF and DegMF. GPUSMF is developed in the previous research, and DegMF is the speed improved method of GPUSMF. And it can be said that the image quality of the proposed method has considerable superior restoration at low noise rate. It can be said that restoration can be done more effectively for high resolution images, the efficient restoration can be expected. However, as the noise rate becomes higher, the amount of noise eliminated by one noise removal in preprocessing decreases. Therefore, the calculation of edge strength is hindered, so that it is conceivable that image quality restoration degree is somewhat inferior compared with other methods.

However, by increasing the number of noise removal processing, which is a preprocessing, it is possible to reduce the influence of noise on this edge strength detection and to obtain better results. Fig.13 shows a partial enlargement images of the obtained removal result images in each method at 10% of noise ratio, and the effectiveness of the proposed method is most confirmed. The restored image of each method can see the state of restored image quality tendency shown in Fig.7.

In DegMF, the noise cannot be completely removed, and it can be seen that the tip of the short hand and some parts of the edge of the numeral are missing. In MDSM, edges can be preserved with a high degree of accuracy, but it can be seen that the amount of remaining noises is high correspondingly. Similarly in GPUSMF, there is little edge defect, but it also shows that noises are, not as much as DegMF, also left.

Also, although MF seems to be unlikely to be a noise-like pixel, it can be seen that there are too many edge defects in terms of deterioration of the original pixels.

On the other hand, the proposed method shows that noise elimination is performed with high precision, and edge preservation is improved as compared with MF. From these points, it can be confirmed that edge preservation and noise elimination are compatible with each other by changing the threshold value using the edge strength.

Fig.13. A partial enlargement restored images.

6 Conclusions

We proposed the improved method of GPUSMF method with the variable threshold by using the edge strength. This proposed method is possible to advance of noise elimination rate and to preserve edges at the same time by making the threshold variable. However, in this proposed method, there are some values to have to set for reducing noises, preprocessing noise removal threshold values, edge loads, and variable threshold. It is conceivable that optimum threshold value for each image, it can be expected to further improve the restored image quality. In the preprocessing, it is the purpose of reducing noise and it is not completely eliminated, so that the noise that cannot be eliminated increases as the noise rate increases.

Therefore, we can expect effectiveness even for images with high noise rate, so we aim for further improvement, by changing the number and method of preprocessing for each image.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number 16K00260 and 17H00383.

References

- (1) Kazuhiro FUJITA, and Mitsuji MUNAYASU, Nobuyuki NAKAMORI, Hiroshi KONDO, Riti NAKURA, Nobuyuki WATANABE, Ryo HONEN: "Restoration of deteriorated image · Improvement of image quality by noise removal", Torikeppusu, Vol. 1, No. 1, pp. 50-63, 2008.
- (2) Narendra, M.Patrenahalli: "A Separable Median Filter for Image Noise Smoothing", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-3, No. 1, pp.20-29, 1981.
- (3) Image processing standard textbook editing committee: "Image processing <Image processing standard text book>", CG-ARTS, Vol. 1, No. 1, pp. 165-181, 2010.
- (4) Tong Sun and Yrjo Neuvo: "Detail-preserving median based filters in image processing", Pattern Recognition Letters, Vol. 15, No. 4, pp.341-347, 1994.
- (5) Z. Wang and D. Zhang: "Progressive switching median filter for the removal of impulse noise from highly corrupted images", IEEE Trans. Circuits Syst. II CAS II, vol.46, No.1, pp.78-80, 1999.
- (6) S.Zhang and M.A.Karim: "A new impulse detector for switching median filters", IEEE Signal Process. Lett., Vol9, No.11, pp.360-363, 2002.
- (7) B.Xiong and Z.Yin: "A universal denoising framework with a new impulse detector and nonlocal means", IEEE Trans. Image Process., Vol.21, No.4, pp.1663-1675, 2012.
- (8) Yasuki YOKOYAMA, Takashi MIYAZAKI, Mitsuo SONE, et al.: "Switching Median Filter Using Multi-direction Scanning 2x2 Noise Detector and Averaging Method", IEICE Trans(A), Vol. J95-A, No. 10, pp.737-750, 2012.
- (9) Yuhei HASHIMOTO, Yoshinobu KAJIKAWA, Yasuo NOMURA: "Removal of Impulse Noise Using Impulse Noise Position Information by High Performance Impulse Detector", IEICE Trans(A), Vol.J84-A, No.6, pp.759-768, 2001.
- (10) D.R.K. Browning: "The weighted median filter", Comm. ACM, Vol.27, No.8, pp807-818, 1984.
- (11) R.H.Chan, C.-W.Ho, M.Nikolova : "Salt-and-Pepper Noise Removal by Median-Type Noise Detections and Detail-Preserving Regularization", IEEE Trans. Image Processing, Vol.14, No.10, pp.1479-1485, 2005.
- (12) S.-S Wang, C.-H Wu: "A New Impulse Detection and Filtering Method for Removal of Wide Range Impulse Noises", Elsevier. Pattern Recognition, Vol.42, Issue9, pp.2194-2202, Jan.2009.
- (13) Shota KOSHIYAMA, Hiroaki YAMAMOTO, Takashi YAMAZAKI, Yasuki YOKOYAMA, Mitso SONE: "A Fast GPU-Based Parallel Multi-Direction Switching Median Filter", IEICE Trans(A), Vol.J99-A, No.1, pp69-72, 2016.
- (14) J. Sanders and E. Kandrot, CUDA by Example: CUDA by Examples, An Introduction to General-Purpose GPU programming, Addison-Wesley, 2011.
- (15) Masatomo INUI: "Introduction to GPU parallel graphics processing - Introduction and utilization of CUDA · OpenGL", Technical commentary, pp114-136, 2014.