LARGE SIZE IMAGE PROCESSING USING DIVISIBLE LOADING TECHNIQUE

Surapong Uttama

School of Information Technology, Mae Fah Luang University, Chiang Rai, Thailand
Email: usurapong@gmail.com

ABSTRACT

Recent advance in image technology induces more numbers of large size images which require greater space and time resources. In this paper we propose a novel image model and a new image processing method called a divisible loading technique to cope with possible problems of loading and processing large size images. We experiment our proposed method on various aspects including image loading and spatial image processing such as computing image statistics and image convolution. The result indicates that the novel technique outperforms the normal one in terms of memory usage, loading time and region of interest processing especially for very large size images.

Index Terms—*Large size image; large image processing; divisible loading; subimage processing*

1. INTRODUCTION

With the ongoing growth of the image acquisition technology and computer processing power there are more production and needs on large size images. Recent development in a charge-coupled device (CCD) provides a capability of a digital compact camera to easily record 10 million pixel images with approximately 3500x2500 pixels resolution and 24 bits depth. This single image after compression could require more than 10 Megabytes (MB) on a secondary storage and even ten times larger in computer memory. Another illustration of large size image is in the filed of remote sensing when researchers need images covering very large areas and containing more details as possible. A medium size of a landscape image may have a resolution of 6000x5000 pixels requiring 14 MB disk space. A large size of a celestial map could require 9000x12000 pixels and 30 MB. Various huge remote sensing images could have storage sizes measured in Gigabytes or even Terabytes.

It is clearly seen that large size images will lead to at least two major problems in image processing. Firstly, loading these images requires large memory space and may result in the insufficient computer memory. Most image processing application reserves the memory, usually the heap, of limited space. Therefore, loading entirely a very large image at once could issue the unexpected end of application due to the inadequate memory. Secondly, processing image especially in spatial domain relies on pixel or subimage operation which is an iterative and memory consuming process. Certainly this would bring about the insufficient memory. Moreover, in image processing we often focus on specific regions of interest (ROI). Consequently it could be better not wasting time and memory to load a whole large image before a ROI operation.

Thus in case we need to apply a spatial image processing in pixel or region level to a very large image, it is possible to load and process a smaller subimage progressively. Therefore, our contribution in this paper is to develop an image model and an image processing technique corresponding to this concept for large size images.

The paper is organized as follows. In Section 2 we present the theory and related works on large size image processing. Section 3 provides computational details including the algorithm and its implementation. Then we describe our experimental set-up, results and discussion in Section 4. Finally, in section 5, we draw conclusions and suggest some possible future works.

2. THEORY AND RELATED WORKS

Large size image processing is a well recognized topic especially in the remote sensing area. However, it is not a very hot issue in image processing as seen from very few related literature. One possible reason could be because the computer running the remote sensing application and dealing with very large image is usually a computer server with powerful performance. Nevertheless, we found that there are attempts on large size image processing in two aspects: hardware and software.

Using hardware to help improving large size image processing is found in [1] and [2]. Both literatures focus on how to distribute the image processing load to many computers in a network. Thus they developed algorithms to partition an image and scheduled the image processing tasks to many network computers while checking load balancing. The result shows that this technique improves the image processing speed provided that the network has good topology and the load distribution is optimized.

Regarding software aspect, some literatures [3-6] are found. Different techniques were used to enhance processing speed such as finite element [3], wavelet [4] [5], and training data [6]. All results go along in the direction that satisfies the reduction of memory usage and computation time. Some suggest that the speed will be better with parallel processing.

It is observed from the literature both in hardware and software directions that though they can improve image processing speed and reduce memory usage, they focus on the processing steps not on the image loading process. That means the algorithms in the literature are implemented after loading a whole image into the memory. Therefore, if the input image is too large to fit into the memory, those algorithms will not be implemented and we can not process our image.

Here we propose a novel concept of large image processing. Instead of loading an entire image to the memory at once, we will read a smaller subimage progressively. We name this method "divisible loading technique" which refers to the loading of a divided piece of image one by one. Then for each subimage we apply image processing algorithm and merge the results. The loading process is in physical level based on the knowledge that an image file always contains a piece of information telling image's sizes, encoding algorithm, metadata etc. Knowing this information helps us to load and decode a part of an image rather than a whole image.

In order to load a part of an image and process it properly, firstly we have to define an image model. An image can be modeled to many subimages in the following two schemes.

1. Distinct subimages: each subimage is independent to one another. There is no overlap between any two subimages.

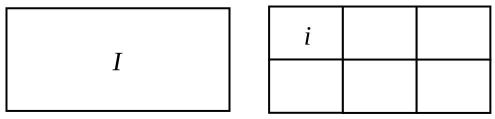

Figure 1. An image and its distinct subimages

An image (I) is denoted by the disjoint union of many subimages (i).

$$I = \bigsqcup_{j \in J} i_j \tag{1}$$

where J is the set of numbers of subimages.

2. Overlapped subimages: there are overlaps between two neighboring subimages.

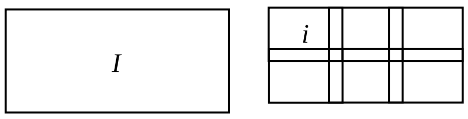

Figure 2. An image and its overlapped subimages

In this model, an image (I) is denoted by the union of many subimages (i).

$$I = \bigcup_{j \in J} i_j \tag{2}$$

where J is the set of numbers of subimages.

The reason why we propose these two models is to prepare the image structure to be relevant with the image processing operations. The distinct subimage model is suitable for pixel processing or distinct block operation such as finding simple image statistics, image histogram and image negative, etc. In contrast, the overlapped subimage model is appropriate for sliding neighborhood operation such as an image convolution.

3. COMPUTATIONAL DETAILS

For any large size image that we need to process, the computational steps are as follows.

1. Read image information (sizes, encoding, metadata, etc.) for image loading preparation
2. Choose image processing operation and its corresponding image structure (distinct or overlapped)
3. Define a subimage size or a region of interest (ROI)
4. Load a subimage according to the size or ROI in step 3.
5. Apply image processing algorithm to the subimage.
6. Repeat step 4 and 5 for the next subimage if necessary
7. Merge the outputs if necessary

To verify our proposed principle, we set up experiments with various image processing operations which are:

1. Image loading
2. Simple statistic: arithmetic mean
3. Image histogram
4. Image convolution: 32x32 Smoothing filter
5. ROI operation: 512x512 square region on an image center and computation of arithmetic mean

For a new proposed method, a progressive load of a 512x512 square subimage is chosen.

The testing environment is Windows XP, Intel Core2Duo E8400 CPU, 2 GB memory. The algorithm is implemented on JAVA programming language having 256 MB heap size. Three large images in grayscale JPEG encoding format are chosen and the details are as follows.

1. Landscape image: 6679x4724 pixels, 8 bits per pixel, Disk size 13.74 MB, Memory size 30 MB.

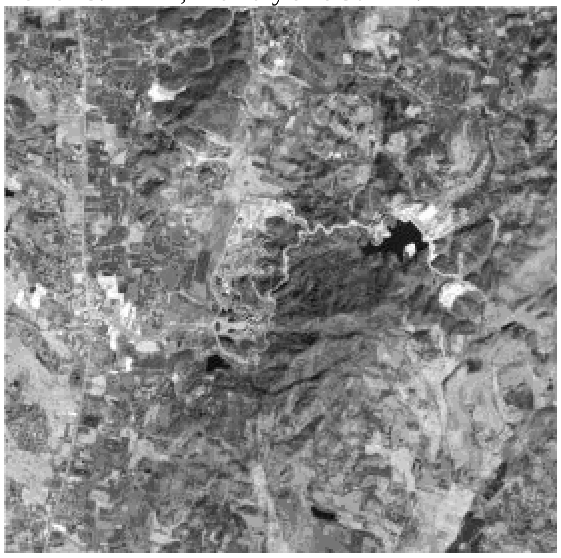

Figure 3. Landscape image

2. Galaxy image: 8858x11811 pixels, 24 bits per pixel, Disk size 28.36 MB, Memory size 99.8 MB.

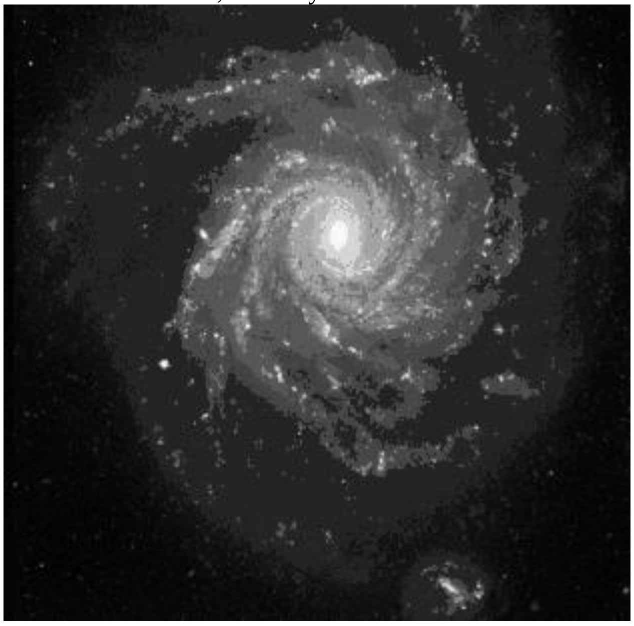
Figure 4. Galaxy image

3. Earth image: 12700x11592 pixels, 8 bits per pixel, Disk size 31.45 MB, Memory size 140.4 MB

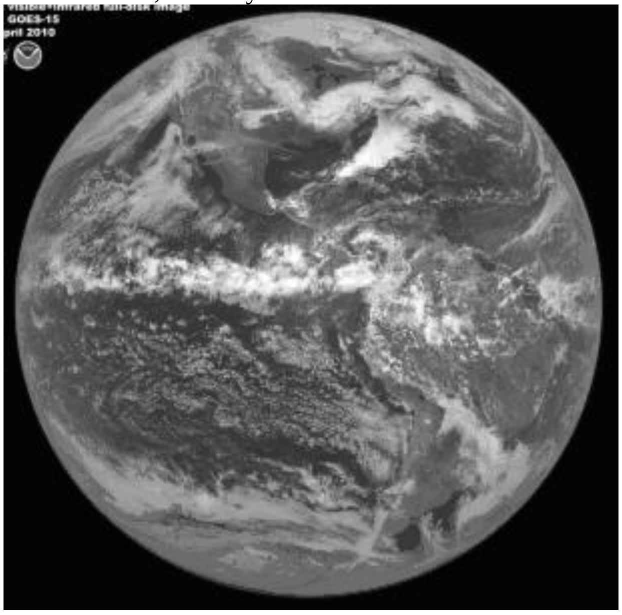

Figure 5. Earth image

4. RESULTS AND DISCUSSION

We perform the test on three large images and compare the results between an old or normal approach (an entire image loading and processing) and a new or proposed method (a progressive image loading and processing). The comparison criteria are processing speed and remaining memory space. For a first image processing operation which is an image loading, it is evident that the new method is faster because it reads just only a small part of an image. However, if comparing the time to load until we get the whole image, the old method is superior. This is certainly due to the iterative loading process of the new method which is slower. However, in case we do not have enough predefined heap space, the old loading method is impossible while the new one can.

The results of other image processing operations are shown in Table 1. Here a √ symbol signifies a better performance. We notice that firstly the new method is better for ROI operation for all images. Surely this is because the new method load and process only ROI while the old one wastes time loading a whole image before ROI processing. Next if we observe the Landscape image, the old method displays better performance. This happens from the fact that accessing data in a computer memory is much faster than in a secondary storage i.e. a hard drive. More times of image loading results in lower processing speed.

Regarding a larger image Galaxy, the old method is more effective for computing image mean and histogram but less effective for smoothing. We can explain that this image commonly consumes quite a large memory due to its large size. By adding the smoothing convolution, it will abruptly consume more memory resource.

For a very large image Earth, the new operation seems to be more efficient in terms of memory use and speed. The old method wastes lots of times loading an entire image which is very large and spends too many memory space. This could cause an insufficient memory problem when computing smoothing convolution.

Table 1. A comparison between a normal and a proposed image processing. √ symbol means better performance.

Operation	Landscape		Galaxy		Earth	
Method	old	new	old	new	old	new
Mean	√		√			√
Histogram	√		√			√
Smoothing	√			√		√
ROI		√		√		√

One thing that should be noted from the experimental result is that image size plays an important role in choosing the image processing method. Our new proposed method is likely to be superior for bigger images.

5. CONCLUSION

Our novel proposed method namely divisible loading technique is developed to improve the performance of large size image processing. We recognize that large size images such as remote sensing images require a great amount of memory space and loading time. For a personal computer with limited resource, it is hardly possible to do anything with these images. So we develop a new image loading and processing technique based on smaller subimages. This approach acquires a part of an image progressively from disk storage. Thus it requires really smaller amount of memory comparing to the normal image loading method which loads a whole image at once. This will permit us to tackle very large images with least amount of memory and computing resources. Nonetheless, it trades off between less resources and more complexity. The proposed method has higher complexity in terms of iterative loading and processing. Hence it will not be recommended for small to medium size image but will outperform in case of larger images.

A future development of this work is to design more rigorous experiments with various sizes of images i.e. up to GB or TB and also with color images. One should try more image processing operations and set up more rigid criteria to evaluate the performance quantitatively.

REFERENCES

[1] C. K. Lee and M. Hamdi, “Parallel image processing applications on network of workstations,” *Parallel Computing*, vol. 21, pp. 137–160, 1995.

[2] B. Veeravalli and S. Ranganath, “Theoretical and experimental study on large size image processing applications using divisible load paradigm on distributed bus networks,” *Image and Vision Computing*, vol. 20, pp. 917-935, 2002.

[3] T. Preußer and M. Rumpf, “An Adaptive Finite Element Method for Large Scale Image Processing,” *Journal of Visual Communication and Image Representation*, vol. 11, pp. 183-195, 2000.

[4] G. Uytterhoeven, D. Roose, and A. Bultheel, “A Wavelet Toolbox for Large Scale Image Processing,” *ACPC’99, LNCS1557*, pp. 337-346, 1999.

[5] D. Chaver, M. Prieto, L. Piñuel and F. Tirado, “Parallel Wavelet Transform for Large Scale Image Processing,” *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPSí02)*, 2002.

[6] N. R. Pal and J. C. Bezdek, “Complexity Reduction for Large Image Processing,” *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, vol.32, no.5, pp. 598-611, October 2002.