Parallel Computing Accelerated Image Inpainting using GPU CUDA, Theano, and Tensorflow

Conference Paper · July 2018

DOI: 10.1109/ICITEED.2018.8534858

3 authors:

Heronimus Tresy Renata Adie

2 PUBLICATIONS **9** CITATIONS

SEE PROFILE

Ignatius Aldi Pradana

Universitas Atma Jaya Yogyakarta

1 PUBLICATION **4** CITATIONS

SEE PROFILE

Pranowo Pranowo

Universitas Atma Jaya Yogyakarta

74 PUBLICATIONS **267** CITATIONS

SEE PROFILE

Parallel Computing Accelerated Image Inpainting using GPU CUDA, Theano, and Tensorflow

Heronimus Tresy Renata Adie
Department of Informatic Engineering
Universitas Atma Jaya Yogyakarta
Yogyakarta, Indonesia
heronimustra@gmail.com

Ignatius Aldi Pradana
Department of Informatic Engineering
Universitas Atma Jaya Yogyakarta
Yogyakarta, Indonesia
ign.aldi.pradana@gmail.com

Pranowo
Department of Informatic Engineering
Universitas Atma Jaya Yogyakarta
Yogyakarta, Indonesia
pran@mail.uajy.ac.id

***Abstract*—Image inpainting refers to image restoration process that reconstruct damaged image to obtain it lost information based on existing information. PDE-based approach is commonly used for image interpolation especially inpainting. Since PDE process express convolution and continuous change, the approach may take a lot of computational resources and will run slow on standard computer CPU. To overcome that, GPU parallel computing method for PDE-based image inpainting are proposed. These days, some handy platform or frameworks to utilize GPU are already exist like CUDA, Theano, and Tensorflow. CUDA is well-known as parallel computing platform and programming model to work with programming language such as C/C++. In other hand Theano and Tensorflow is a bit different thing, both of them is a machine learning framework based on Python that also able to utilize GPU. Although Theano and Tensorflow are specialized for machine learning and deep learning, the system is general enough to applied for computational process like image inpainting. The results of this work show benchmark performance of PDE image inpainting running on CPU using C++, Theano, and Tensorflow and on GPU with CUDA, Theano, and Tensorflow. The benchmark shows that parallel computing accelerated PDE image inpainting can run faster on GPU either with CUDA, Theano, or Tensorflow compared to PDE image inpainting running on CPU.**

Keywords—Image Inpainting, PDE, Parallel Computing, GPU, CUDA, Theano, Tensorflow

I. INTRODUCTION

Recently, digital image processing already had significant amount of uses and application. Image inpainting is one of the problem case appears in image processing, which useful for repairing or filling damaged part of image. The terms of inpainting in digital image processing was firstly introduced by Bertalmio, et al who brings up image inpainting method based on a third-order partial differential equation [1]. The purpose of inpainting itself is to repair the damaged area of image with existing information gained from the surrounding damaged part of image. In terms of visual effect, the repaired image need to maintain it rationality to look like as same as original image [2].

There are many approaches to achieve state-of-the-art result of inpainting that have already published. Generally based on the algorithm there are three group of images inpainting method [3], which is Partial Differential Equation (PDE), texture synthesis, and convolution-filter based algorithms. After Bertalmio, et al introduced BSCB (Bertalmio-Sapiro-Caselles-Ballester) with third-order PDE based method, there are many other like Perona Malik [4] anisotropic diffusion, Total Variation Wavelet Inpainting [5] which uses an Euler-Lagrange equation, Oliveira's Algorithm [6] that proposed fast reconstruction for small damaged portions of images using convolutional operation, and many other approaches including the Schönlieb modern PDE technique using fourth-order PDE based method [7], and current Fang Zhang, et al [2] PDE method based on image characteristic that improve the protection of important characteristics of the image during inpainting process.

Each of the algorithm has their own weak point such as slow computational speed, poor visual effects and disability to reconstruct textures. Although the current algorithm like Schönlieb fourth-order PDE, or Fang Zhang method based on image characteristic has improved the major problem on computational speed and visual effect result, there are still a room for improvement especially to accelerate the computational speed and simplify it application and practice. In principle, the heavy computational load on some inpainting method come from algorithm like the PDE equation and convolutional-filter based algorithm that take a lot step to compute. Current hardware can pretty much handle the computation from small size of image, but the size of image exist today are getting larger up to 4k resolution and may increase more. In that case, computation process of single image matrix will be slower.

Current technique of parallel computing can be applied along with current specialized hardware for computing like GPU. Recent parallel computing with the help of CUDA platform can speed up the simulation model of inpainting to about 40x faster than CPU [8], but the uses of parallel computing on CUDA platform is a bit challenging because it requires good understanding of low level programing language and it will be hard to reproduce or applied the algorithm since all code need to be built from scratch every time. To overcome that drawback, machine learning framework are used to improve to ease the code implementation for image inpainting. Machine learning framework bring code simplification on high level API on Python layer, so it allows the users to easily deploy the computation either on CPU or GPU without any change of the code.

In this paper we will use generic CUDA, Theano, and Tensorflow to apply the GPU parallel computation on Image Inpainting process. CUDA undoubtedly is the most powerful platform for parallel programming with widely supported hardware. Theano and Tensorflow are slightly different than CUDA, both of them is a high levels API framework that also

support speed optimizations for numerical data computation using both CPU or GPU and has largely used for machine learning purpose running on Python programming language [9,10]. Theano and Tensorflow will represent the use of framework for code simplicity. Theano is selected among the other frameworks because it was the pioneer of the computational graph and one of oldest library that remains popular in the research community for deep learning and machine learning in general. Tensorflow also selected because it ranks as most popular deep learning library for data science with significant developer and large community support [11]. The final result of this research is an accelerated image inpainting program using the capability of parallel computing on GPU.

II. Prevorious and Related Work

Prananta et al [8] have applied parallel computing on image Inpainting using native GPU CUDA programming. The use of that method can actually reduce the heavy computation problem of fourth order PDE and successfully employ the use of GPU in parallelism to gain computation speed up to 36x in the simulation model and speedup to 48x in the simulation picture. At the same year Kuo et al [12] also employ the graphic processing unit to speedup Inpainting-based multi-view synthesis algorithms. The research show that proposed method has 51 times speeded up for synthesis virtual view with better quality.

Both of the related work used only native GPU implementation for parallel programing with CUDA. The method show that computational speed can successfully speeded up, but neither of both work use the framework like Theano or Tensorflow to employ the flexibility and simplify the development of the image-inpainting method.

III. Proposed Image Inpainting Method using CUDA, Theano, and Tensorflow

The main idea of this proposed image inpainting method using CUDA, Theano, and Tensorflow is to optimize the computational calculation speed using the capability of each platform to make use of GPU to do parallel computation, and also to make uses the flexibility to simplify the development of image inpainting application. Since the goal is to prove the capability of parallel programming with GPU to accelerated image inpainting process, we will use PDE-based image inpainting that involves heat equation because PDE can mainly represent most image inpainting algorithm exist and also involved the convolution process that can be parallelize, despite of it lacking ability to generate texture and do not preserve edges so the filling part will mostly blurry.

Platform or framework to simulate the parallel computation process on GPU we will use in this research is CUDA, Theano, and Tensorflow. The PDE-based image inpainting program will be execute on both CPU and GPU. Theano and Tensorflow natively support CPU ad GPU switching, but the CUDA platform will only run on GPU so we will add C++ based version as comparison to run on CPU.

A. Image Inpainting Algorithm

PDE appear as a natural way to smooth images, that become one of effective way of image inpainting. The main concept of PDE can be draws as an analogy between the image inpainting process and the diffusion [13]. Images can be comparable to heat, fluid, and gas which spontaneously move from the area of high concentration to the area of lower concentration, that concept inspired the image inpainting process of filling missing part of damaged image. Our method makes use the following heat equation:

$$\frac{\delta u}{\delta t} = \alpha \left(\frac{\delta^2 u}{\delta x^2} + \frac{\delta^2 u}{\delta y^2} \right) \quad (1)$$

Therefore, we will applied some minor modification to that equation. The intensity value from the masked image will be added to the equation. The purpose of that modification is to limit the restoration of image only to damaged/masked area of the image. We will fill the masked area with the average value or heat diffusion from the surrounding pixel value, finally the equation is calculated as follow:

$$\frac{\delta u}{\delta t} = \alpha \left(\frac{\delta^2 u}{\delta x^2} + \frac{\delta^2 u}{\delta y^2} \right) + \lambda (u - I) \quad (2)$$

Where the u is the intensity of the image pixel, t is for time value, x and y is pixel index from the image that represents as matrix, λ is constant value from the masked image, and I represent as intensity from original damaged image.

The detailed step of image inpainting process using heat equation are given as follow:

1) Read the data of damaged image and it mask, then convert the image data to float32 matrix.
2) Initialize the value for reconstruction image (u).
3) Run the step of PDE
    a) Update the u value based on (2)
4) Repeated step 3 until fair image condition reached
5) Show the final image inpainting result (u)

B. Uses of Parallel Computation

The main uses of parallel computation here is to deal with computing process of the image inpainting. One of the heavy computing process come from the heat diffusion for filling the masked area with the average value from the surrounding pixel value based on (1). Larger image dimension will take more time to do that process because it done pixel by pixel. The parallel computing platform like CUDA, or framework like Theano and Tensorflow can accelerated the computation process using it's built in function.

While CUDA can reproduce the image inpainting algorithm manually, Tensorflow and Theano need some adjustment because they process the data as tensor format. Tensor value on Theano and Tensorflow cannot be accessed by it index, the tensor can only be processed by the built-in function of its framework. That tensor limitation makes processing the PDE

based image inpainting algorithm need some work around with Theano and Tensorflow. The heat diffusion part on PDE based image inpainting algorithm can be treat as a convolutional process to achieve the same goal, since Theano and Tensorflow came with the simple built-in function to do convolution process, it can be used to calculate the heat diffusion part. Convolution function in Tensorflow and Theano came with optional parameter to determine the kernel filter, strides, and padding. Kernel filter is a weighted value that determined how the image value will be updated during the convolution process to produce new updated value, strides determine how much the window shifts by in each of the dimensions, and padding determined did the input tensor dimension will be padded with new value or not. The function can be found at *theano.tensor.nnet.conv2d* for Theano, and on Tensorflow it located at *tensorflow.nn.conv2d*. Both function have similar way to use, only some minor format on the input data needed. The important input for the convolutional process on this framework is the kernel filter. Equation (1) can be presented as 3x3 matrix kernel for filtering shown in the Fig 1. Fig. 1 also show the process of calculated new value for filling the damaged area by using the value of its neighborhood value (top, bottom, right, and left) around it missing pixel. The snippet of programming code function for convolution is shown on Fig. 2 for Tensorflow, and on Fig. 3 for Theano.

0	1	0
1	4	1
0	1	0

Fig. 1. Kernel filter to represents heat equation for convolution process.

```
result = tf.nn.depthwise_conv2d(input, kernel,
                                [1, 1, 1, 1],
                                padding='SAME')
```

Fig. 2. Snippet code of convolutional function using Tensorflow

```
theano_convolve2d = th.function([input, filters],
                                T.nnet.conv2d(input, filters,
                                border_mode='half',
                                subsample=(1, 1)))
```

Fig. 3. Snippet code of convolutional function using Theano

IV. Experimental Results

The proposed parallel computing accelerated image inpainting using CUDA, Theano, and Tensorflow was evaluated with four examples of damaged image with each image have three different image dimension which is 800 x 600 (SVGA standard), 1600 x 1200 (UXGA standard), and 3600 x 2400 (QUXGA standard) pixels dimension that represent small, medium, and large image size format. To evaluate the proposed parallel computing accelerated image inpainting using CUDA, Theano, and Tensorflow we performed comparative experiment to run the image inpainting on 3.4 GHz Intel i7-3770k PC with 8 GB Nvidia GTX 1080 general purpose GPU. Comparative experiment on this image inpainting model will be tested on CPU with C++ platform, CPU with Tensorflow framework, CPU with Theano framework, GPU with CUDA platform, GPU with Tensorflow framework, and GPU with Theano framework. Latency time of the image inpainting process to run on single image will be recorded as the main indicator to compare, and on GPU testing the GPU memory allocation will also add to the result. We also look the similarity index of the inpainted image using SSIM method to see did the framework or platform can take effect to the results of inpainting process. The damaged image and it mask that used for the input are shown on Fig. 4 and Fig. 5. In Fig. 6 the result image of 2000 iteration of inpainting process is shown, the 2000 iteration of inpainting process is average optimum number that can produce the best result, more than that the inpainted image result will just look same.

The final qualitative result is shown by Table I and Table II. Total time of each test elapsed during the image inpainting process in Table I show many variative result. Based on that result, the native C++ program is the slowest and took average 22.338 second for 800 x 543 images dimension. Tensorflow and Theano based image inpainting process was slightly faster than native C++ program, that's because the convolution process on both framework is more effective using the built-in function rather than the primitive manually written convolution process on native C++ program. Tensorflow on CPU score 28.185 second for 800 x 543 images dimension. Theano on CPU score 15.777 second for 800 x 543 images dimension. Theano is the faster platform/framework on our CPU comparison. The GPU test shown in Table II shows that CUDA platform take the lead as the faster platform on GPU. CUDA result only 0.177 second for 800 x 543 image dimension or also 200 times faster than the native C++ running on CPU. Tensorflow score 0.942 second for 800 x 543 image dimension or 29.913 times faster than Tensorflow that running on CPU. Theano is the slowest among the other competitor. Theano score 4.214 second on GPU, that only 3.744 times faster than the Theano that running on CPU. On the Table III we can see the speed up comparison for each GPU platform and framework for three different image dimensions. CUDA and Tensorflow speed up increase on bigger image dimensions, only Theano that the speed up didn't increase on bigger image dimension. On the Table IV we can also see the GPU memory allocation of each framework during the inpainting process. CUDA use only 220 Mb of GPU memory, Theano use 289 Mb of GPU memory, then the Tensorflow take up to 623 Mb of GPU memory. Based on the GPU memory allocation, CUDA take the smaller GPU memory even the speed

benchmark shows that CUDA is way faster than the other. The similarity index (SSIM) of inpainted images shown on Table V is same for all the platform/framework, that's prove that all the platform/framework run same process and calculation of inpainting.

Overall result show that parallel computing using CUDA platform on image inpainting method is faster than Theano and Tensorflow. On CPU without the parallel computing Theano and Tensorflow framework is slightly faster than the native C++ program because they actually build and compiled first and provide better convolution process. On GPU test the CUDA platform is way faster than Theano and Tensorflow because the Theano and Tensorflow itself using the CUDA library as their backend to utilize the GPU. The main advantage of Theano and Tensorflow at this case is their compatibility to run on CPU and GPU without changing the code also have some useful built-in function.

Fig. 4. Damaged images examples for inpainting process.

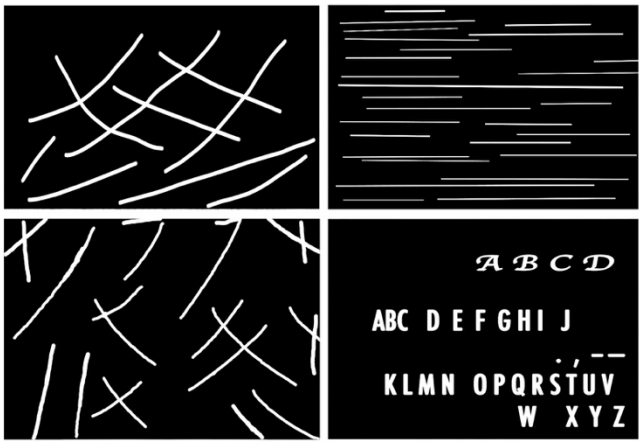

Fig. 5. Mask of damaged area of images from Fig. 4.

Fig. 6. Inpainting result on four image examples after 2000 iteration.

TABLE I. TIME REQUIRED FOR INPAINTING ON CPU

Image Dimension	Running Time on CPU (s)[a]		
	C++	Theano	Tensorflow
800 x 543	22,338	15,777	28,185
1600 x 1068	93,258	82,712	116,565
3600 x 2043	723,575	399,793	585,754

[a] Average time from four testing images on Fig. 4 after 2000 iteration.

TABLE II. TIME REQUIRED FOR INPAINTING ON GPU

Image Dimension	Running Time on GPU (s)[a]		
	CUDA	Theano	Tensorflow
800 x 543	0,117	4,214	0,942
1600 x 1068	0,460	23,571	2,452
3600 x 2043	2,336	109,465	11,448

[a] Average time from four testing images on Fig. 4 after 2000 iteration.

TABLE III. SPEED UP FOR INPAINTING PROCESS ON GPU

Image Dimension	GPU Speed Up Compared to CPU		
	CUDA[a]	Theano	Tensorflow
800 x 543	191,738	3,744	29,913
1600 x 1068	202,955	3,509	47,539
3600 x 2043	309,816	3,652	51,168

[a] CUDA performance compared to C++ on CPU

TABLE IV. GPU MEMORY ALLOCATION FOR INPAINTING PROCESS

Platform / Framework	*Memory Allocation On GPU (Mb)*
CUDA	220
Tensorflow	623
Theano	289

TABLE V. STRUCTURAL SIMILARITY INDEX OF TESTED IMAGES

Image Dimension	SSIM (%)[a]
800 x 543	91,39%
1600 x 1068	94,02%
3600 x 2043	96,32%

[a.] Average SSIM value from four image examples.

V. CONCLUSION

After evaluated the proposed parallel computing accelerated image inpainting using CUDA, Theano, and Tensorflow, here is our main observations:

- The uses of parallel computing platform or framework using GPU can speed up the computational process time of image inpainting.
- The uses of Theano and Tensorflow framework prove that both framework can simplify the code with their built-in function and ease the application of parallel programming without changing the code when switching from CPU to GPU.
- Theano is faster framework for image inpainting run on CPU rather than Tensorflow and native C++ program.
- CUDA is faster parallel computing platform for image inpainting run on GPU rather than Theano or Tensorflow.
- Even CUDA is faster among the other, CUDA also take smaller amount of GPU memory than Theano, and Tensorflow.

REFERENCES

[1] Bertalmio, M., Sapiro, G., Caselles, V., Ballester, C.: Image inpainting. In: 27th International Conference on Computer Graphics and Interactive Techniques Conference, ACM Press, Los Angeles, 2000, pp. 417-424.

[2] Zhang F. et al., "Partial differential equation inpainting method based on image characteristics," in: Zhang YJ. (eds) Image and Graphics. ICIG 2015. Lecture Notes in Computer Science, vol 9219. Springer, Cham, 2015.

[3] H. Noori, S. Saryazdi, and H. Nezamabadi-pour, A Convolution based image inpainting, 1st International Conference on Communications Engineering, 2010, pp. 130-134.

[4] Perona, P. Malik, J. Scale-space and edge detection using anisotropic diffusion. IEEE-PAMI 12, 1990, pp. 629-639.

[5] Chan, T., Shen, J. Mathematical models for local deterministic inpaintings. UCLA CAM TR 00-11, March 2000.

[6] M. Oliveira, B. Bowen, R. Mckenna, Y. S. Chang, "Fast digital image inpainting," in proc. VIIP2001, pp. 261-266, 2001.

[7] C, Schönlieb. Modern pde techniques for image inpainting. Doctoral dissertation. University of Cambridge. 2009.

[8] Prananta E, Pranowo, Budianto D. GPU CUDA accelerated image inpainting using fourth order PDE equation. TELKOMNIKA. 2016; 14(3): 1009-1015.

[9] M, Abadi., et al. TensorFlow: a system for large-scale machine learning. Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16), 2016, pp. 265-283.

[10] J, Bergstra., et al. Theano: A CPU and GPU math compiler in Python. Proc. of The 9th Python in Science Conf. (Scipy 2010). 2010.

[11] M. Pascal. Ranking popular deep learning libraries for data science. https://blog.thedataincubator.com/2017/10/ranking-popular-deep-learning-libraries-for-data-science. Accessed: 2018-03-20.

[12] P. Kuo, J. Lin, B. Liu, J. F. Yang, Inpainting-based multi-view synthesis algorithms and its GPU accelerated implementation, 9th International Conference on Information, Communications & Signal Processing, Tainan, 2013, pp. 1-4.

[13] M, Hardik. Implementation of image inpainting using heat equation. CiiT International Journal of Digital Image Processing. 2012.