ISSN 2277-3061
Volume 15 Number 10
International Journal of Computer and Technology

A REVIEW ON IMAGE SEGMENTATION USING GPU

Gurpreet Kaur [(1)], Sonika Jindal [(2)]
[(1)] Research Scholar, Department of Computer Science Engineering, SBSSTC, Ferozepur
Gurpreet878@gmail.com
[(2)] Assistant Professor, Department of Computer Science Engineering, SBSSTC, Ferozepur
sonikamanoj@gmail.com

ABSTRACT

Image Segmentations play a heavy role in areas such as computer vision and image processing due to its broad usage and immense applications. Because of the large importance of image segmentation a number of algorithms have been proposed and different approaches have been adopted. Segmentation divides an image into distinct regions containing each pixel with similar attributes. The objective of apportioning is to simplify and/or alter the representation of an image into something that is more meaningful and more comfortable to break down. This paper discusses the various techniques implemented for image segmentation and discusses the various Computations that can be performed on the graphics processing unit (GPU) by means of the CUDA architecture in order to achieve fast performance and increase the utilization of available system resources.

Keywords

Segmentation, GPU, Image Processing, OpenCV, Region Growing algorithm, CUDA.

INTRODUCTION

In image segmentation image is divided into some regions and in that region each pixel is similar with respect to some of the characteristic such as the color, intensity, or texture. Adjacent regions are significantly different with respect to the same characteristic(s). Segmentation can be done by detecting edges or points or line in the image. When we detect the points in an image then on the basis of similarities between any two points we can make them into separate regions. And in the case of the line detection technique we use to detect all the lines and the similarities in between those lines then on the basis of the dissimilarities between the lines or curves in the image we can divide the image into two regions. And in the case of edge detection we detect the edges in the image and after finding the edges in the image we can easily divide the image and here we can easily analyze what is inside the image and we can get a better segmented image. even it is the old technique to segment the image now a days these segmentation technique is used to segment the image. We can segment the image by using many techniques but only some techniques gives the better results in image segmentation. Some of the image segmentation techniques won't work for all the images. So till now there are many methods to solve these image segmentation problem. This image segmentation started in 1970 and till now there is no robust and efficient segmentation technique to solve the image segmentation problem. Image segmentation is to segment the image into multiple parts. It is useful everywhere whenever we want to analyze what is inside an image. For example, if we want to find if there is a chair or dog inside an indoor image, we need image segmentation technique to separate the objects in the image and analyze each object individually to check what it is…as we already know that because of image segmentation we can identify the diseases in medical imaging and also in many applications like face detection, iris detection, fingerprint recognition and also in brake light detection technique also we used this image segmentation technique. In image segmentation each technique has its own advantages and also disadvantages, so it's hard to tell which one is better in all the techniques. There are many previous works about the image segmentation, great survey resources could be found from these surveys, we can separate the image segmentation techniques into three different classes.

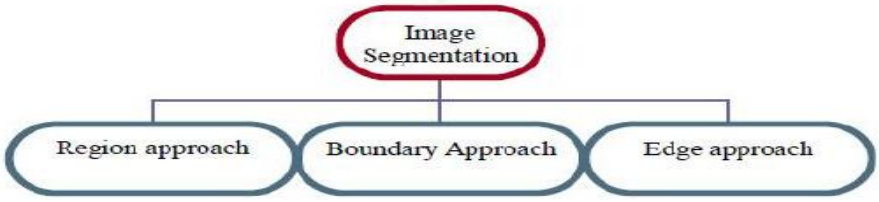

Figure 1. Classes of Image Segmentation

APPLICATIONS OF SEGMENTATION

- Image segmentation is mainly used in Medical imaging.
  - In locating tumors and other pathologies.
  - In measuring tissue volumes.
  - In computer-guided surgery.
  - In diagnosing a disease.
  - In treatment planning.

- To locate objects in the satellite images.
- Face recognition: A facial recognition system is a computer application for identifying or verifying a person from a digital image or a video frame from a video source automatically. We can use this technique by comparing the selected facial features from the image and a facial database. In security systems we can use this face recognition technique to identify the thieves or any security related work. And can be compared to other biometrics such as fingerprint or eye iris recognition systems
- Iris recognition: Iris recognition is method of biometric identification that uses the mathematical pattern-recognition techniques on video images of the iris of an individual's eyes, whose complex random patterns are unique and can be seen from some distance.
- Fingerprint recognition: It refers to an automated method of verifying a match between two or more human fingerprints. Fingerprints are mainly used in identifying individuals and verify their identify .
- Traffic control systems
- Brake light detection
- Agricultural imaging – in detecting the crop disease

RELATED WORK

Abhaya Kumar Sahoo et. al (2015) has stated that image segmentation is the process of dividing a digital image into multiple segments or clusters. The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful for analysis .Image segmentation is typically used to separate the region of interest from the input image. So we have different region of interest for different applications. This research has presented a comparison between serial execution of the Region growing algorithm and Parallel execution of it on CUDA platform (provided by nvidia) integration with MATLAB. Jin Xin Hua et. al (2014) has proposed a GPU-based parallel processing method for real-time image segmentation with neural oscillator network. Range image segmentation methods can be divided into two categories: edge-based and re-gion-based. Edge-base method is sensitive to noise and region-based method is hard to extracting the boundary detail between the object. However, by using LEGION (Locally Excitatory Globally Inhibitory oscillator networks) to do range image segmentation can overcome above disadvantages. The reason why LEGION is suitable for parallel processing that each oscillator calculate with its 8-neiborhood oscillators in real time when we process image segmentation by LEGION. Thus, using GPU-based parallel processing with LEGION can improve the speed to realize real-time image segmentation. Stamos Katsigiannis et. al (2016) has discussed that microarray is a powerful tool for simultaneously studying the expression level of thousands of genes. Nevertheless, the analysis of microarray images remains an arduous and challenging task due to the poor quality of the images which often suffer from noise, artifacts, and uneven background. In this work, the MIGS-GPU (Microarray Image Gridding and Segmentation on GPU) software for gridding and segmenting microarray images is presented. Hanjoo Cho et. al (2014) has proposed a new approach to meanshift based image segmentation that uses a non-iterative process to determine the maxima of the underlying density, which are called modes. To identify the mode, the proposed approach performs a mean-shift process on each pixel only once, and uses the resulting mean-shift vectors to construct links for the pairs of pixels, instead of iteratively performing the mean-shift process. Then, it groups the pixels of the same mode, connected through the links, into the same cluster. Darian M. Onchis et. al (2014) has identified the multiple phases in microstructures images. The procedure is based on an efficient image segmentation using the fuzzy c-means algorithm. Furthermore, the algorithm is accelerated on a GPU cluster in order to obtain optimal computing times for large size images. The results are compared on the same experimental images with the ones obtained from a commercial software and the accuracy of the proposed algorithm is demonstrated. N. Aitali et. al ( ) has proposed a new fine-grained clustering bias field estimation and segmentation algorithm on Single Instruction Multiple Data (SIMD) architecture (GPU). The goal is to accelerate compute-intensive portions of the sequential version. We have implemented this parallel algorithm using Compute Unified Device Architecture (CUDA) on different NVidia GPU cards. Choong Sang Cho et. al (2015) has stated that the topological derivative-based segmentation uses two sparse matrices, and the computational complexity of the segmentation grows up dramatically as the image size increases due to the size of the sparse matrix. Therefore, to provide a fast and accurate segmentation with low complexity, an effective scheme is proposed with keeping the same segmentation performance. To further reduce the computational complexity, the parallel processing structure for the proposed scheme is designed and implemented on graphics processing unit (GPU). Joe Chalfoun et. al (2015) has automated microscopy enables scientists to image an area of an experimental sample that is much larger than the microscope's Field of View (FOV) and to carry out time-lapse studies of cell cultures. An automated microscope acquires these images by generating a grid of partially overlapping images. Arthur D. Costea et. al (2014) has stated that multi-class segmentation assigns a class label to each pixel in an image. It represents a significant task for the semantic understanding of images and has received plentiful attention over the last years. The current state of art is dominated by conditional random field based approaches, defined over pixels or image segments. However, high accuracy segmentation comes at a high computational cost. The best performing methods can barely run at few frames per second and are far from real-time applications. Xiuxia Zhang et. al (2015) has integrated Theano's GPU implementation in CNN system, explored parallelism potential on multi-core CPUs and many-core Intel Phi by testing performance of main kernel functions of CNN. In the future, we will integrate implementations on other two platforms into CNN framework. Mohammed A. Shehab et, al(2015) has proposed to leverage the power of the Graphics Processing Unit (GPU)to improve the performance of such approaches. Specifically,

we focus on the Fuzzy C-Means (FCM) algorithm and its more recent variation, the Type-2 Fuzzy CMeans (T2FCM) algorithm. We propose a hybrid CPU-GPU implementation to speed up the execution time without affecting the algorithm's accuracy. The experiments show that such an approach reduces the execution time by up to 80% for FCM and 74% for T2FCM.

GPU

The advent of multicore CPUs and many core GPUs means that mainstream processor chips are now parallel systems. Furthermore, their parallelism continues to scale with Moore's law. The challenge is to develop application software that transparently scales its parallelism to leverage the increasing number of processor cores, much as 3D graphics applications transparently scale their parallelism to many core GPUs with widely varying numbers of cores. The CUDA parallel programming model is designed to overcome this challenge while maintaining a low learning curve for programmers familiar with standard programming languages such as C. CUDA is a parallel computing architecture developed by NVIDIA in order to execute generalpurpose computation algorithms on GPGPUs. The primary parallel construct in CUDA is a data-parallel, Single Process, Multiple Data (SPMD) kernel function. A kernel function invocation explicitly creates many CUDA threads. The threads are organized into multidimensional arrays that can synchronize and quickly share data, called thread blocks. The CUDA programming model allows the programmer to take advantage of the massive parallel computing power of a graphics processor in order to perform general purpose computation. CUDA benefits from the hundreds of ALUs inside a graphics processor and use massive parallel interfaces in order to connect with its memory.

The CPU or Central Processing Unit is where all the program instructions are executed in order to derive the necessary data. The advancement in modern day CPUs have allowed it to crunch more numbers than ever before, but the advancement in software technology meant that CPUs are still trying to catch up. A Graphics Processing Unit or GPU is meant to alleviate the load of the CPU by handling all the advanced computations necessary to project the final display on the monitor. Originally, CPUs handle all of the computations and instructions in the whole computer, thus the use of the word 'central'. But as technology progressed, it became more advantageous to take out some of the responsibilities from the CPU and have it performed by other microprocessors. In the days before GUIs, the screen was simply a small grid with each box having an 8bit value that corresponds to a character. This was relatively very easy to do for the CPU, but GUIs have greater resolutions with each pixel having a 16bit or 32bit color value. GPUs where originally developed to render 2D graphics; specifically, to accelerate the drawing of windows in a GUI. But as the need for 3D and faster graphics acceleration grew, the GPU became faster and more specialized in its task. GPUs are now generally floating point processors that can easily crunch geometric computations along with texture mapping tasks. Most GPUs have implemented MPEG primitives to make enhance the playback of videos; some even have the capability to directly decode HD video data, taking another task away from the CPU. Hardware wise, GPUs and CPUs are similar but not identical. If we looked at the very building block of each, the transistors, we can see that most GPUs already rival CPUs in transistor count. The specialized nature of GPUs means that it can do its task much faster than a CPU ever can, but it is not able to cover all of the capabilities of the CPU. Multiple GPUs can also be employed to achieve a single goal much like the dual core CPUs currently available. ATI's Crossfire and NVidia's SLI allow users to connect two identical GPU's and make them work as one.

CUDA is a parallel computing platform and programming model made by NVIDIA and implemented by the graphics processing units (GPUs) that they bring forth. A CUDA program consists of server code and device code: the phases that show diminutive or no parallelism are carved in host part and that possess a plentiful amount of parallelism are included in the device path.

OPENCV

The open source computer vision library, OpenCV, began as a research project at Intel in 1998. It has been available since 2000 under the BSD open source license. OpenCV is aimed at providing the tools needed to solve computer-vision problems. It contains a mix of low-level image-processing functions and high-level algorithms such as face detection, pedestrian detection, feature matching, and tracking. The library has been downloaded more than three million times. In 2010 a new module that provides GPU acceleration was added to OpenCV. The GPU module covers a significant part of the library's functionality and is still in active development. It is implemented using CUDA and therefore benefits from the CUDA ecosystem, including libraries such as NVIDIA Performance Primitives (NPP). The GPU module allows users to benefit from GPU acceleration without requiring training in GPU programming. The module is consistent with the CPU version of OpenCV, which makes adoption easy. There are differences, however, the most important of which is the memory model. OpenCV's GPU module includes a large number of functions, and many of them have been implemented in different versions, such as the image types (char, short, float), number of channels, and border extrapolation modes. This makes it challenging to report exact performance numbers. An added source of difficulty in distilling the performance numbers down is the overhead of synchronizing and transferring data. This means that best performance is obtained for large images where a lot of processing can be done while the data resides on the GPU. To help the developer figure out the trade-offs, OpenCV includes a performance benchmarking suite that runs GPU functions with different parameters and on different datasets. This provides a detailed benchmark of how much different datasets are accelerated on the user's hardware.

CONCLUSION

In this paper, a throughout study of the different techniques involved in segmentation have been discussed. Moreover the differences between CPU and GPU have been explained and we came to a solution that GPU aims at accelerating the performance of segmentation and utilize the advantages of GPU computing. In general, the trend in GPU development

has been increasing the number of thread processors, the clock speed and the amount of on-board memory. This allows more data to be processed faster in parallel.

REFERENCES

[1] A. K. Sahoo, G. Kumar, G. Mishra and R. Misra, "A New Approach for Parallel Region Growing Algorithm in Image Segmentation using MATLAB on GPU Architecture," IEEE , pp. 279-283, 2015.

[2] J. X. Hua and M.-H. Jeong, "Real-Time Range Image Segmentation on GPU," International Conference on Control, Automation and Systems (ICCAS), pp. 150-153, 2014.

[3] S. Katsigiannis, E. Zacharia and D. Maroulis, "MIGS-GPU: Microarray Image Gridding and Segmentation on the GPU," IEEE, pp. 1-8, 2015.

[4] H. Cho, S.-J. Kang, S. I. Cho and Y. H. Kim, "Image Segmentation Using Linked Mean-Shift Vectors and Its Implementation on GPU," IEEE, pp. 719-727, 2014.

[5] Darian M. Onchis, D. Frunzaverde, M. Gaianu and R. Ciubotariu, "Multi-phase identification in microstructures images using a GPU accelerated fuzzy c-means segmentation," IEEE, pp. 602-607, 2015.

[6] N. Aitali, B. Cherrad, O. Bouattane, M. Youssfi and A. Raihani, "New Fine-Grained Clustering Algorithm on GPU Architecture for Bias Field Correction and MRI Image Segmentation," IEEE, pp. 118-121, 2015.

[7] C. S. Cho and S. Lee, "Low-Complexity Topological Derivative-Based Segmentation," IEEE, pp. 734-741, 2015.

[8] J. Chalfoun, M. Majurski, T. Blattner, W. Keyrouzl, P. Bajcsy and M. Brady, "MIST: Microscopy Image Stitching Tool," IEEE, p. 1757, 2015.

[9] Arthur D. Costea and Sergiu Nedevschi , "Multi-Class Segmentation for Traffic Scenarios at Over 50 FPS," IEEE, pp. 1390-1395, 2014.

[10] X. Zhang, G. Tan and M. Chen, "A Reliable Distributed Convolutional Neural Network for Biology Image Segmentation," IEEE, pp. 777-780, 2015.

[11] Mohammed A. Shehab, , Mahmoud Al-Ayyoub and Yaser Jararweh, "Improving FCM and T2FCM Algorithms Performance using GPUs for Medical Images Segmentation," IEEE, pp. 130-135, 2015.

[12] Z. Liu, X. Li, P. Luo , C. C. Loy and X. Tang, "Semantic Image Segmentation via Deep Parsing Network," IEEE, pp. 1377-1785, 2015.

[13] Z. Yi, X. Hu, B. Jang and K. K. Kim, "A Robust and Parallel-Friendly Distance Image Based Hand Detection," IEEE, pp. 33-34, 2015.

[14] J. Sirotkovic, H. Dujmic and V. Papic , "Image segmentation based on complexity mining and mean-shift algorithm," IEEE, pp. 1-6, 2014.

[15] A.-R. Baek, K. Lee and H. Choi , "Speed-up Image Processing on Mobile CPU and GPU," IEEE, pp. 79-81, 2015.

[16] Z. Juhasz and G. Kozmann, "A GPU-based Simultaneous Real-Time EEG Processing and Visualization System for Brain Imaging Applications," IEEE, pp. 299-304, 2015.

[17] H. L. Khor, S.-C. Liew and Jasni Mohd. Zain , "A Review on Parallel Medical Image Processing on GPU," IEEE, pp. 45-48, 2015.

[18] A. Asaduzzaman, A. Martinez and A. Sepehri , "A Time-Efficient Image Processing Algorithm for Multicore/Manycore Parallel Computing," IEEE, pp. 1-5, 2015.

[19] A. Fabija and J. Gocławski, "New accelerated graph-based method of image segmentation applying minimum spanning tree," IET Image Processing, pp. 239-251, 2013.

[20] A. S. Baby and Balachandran K, "A Parallel Approach For Region-Growing Segmentation," IEEE, pp. 196-200, 2015.

[21] C. Cho and S. Lee, "Effective Five Directional Partial Derivatives-based Effective Five Directional Partial Derivatives-based," IEEE, pp. 1-9, 2015.

[22] K. Y. Lee, G. Kyung, T. R. Park, J. C. Kwak and Y. S. Koo, "A Design of a GP-GPU based Stream Processor for an Image Processing," IEEE, pp. 535-539, 2015.

[23] M. G. McGaffin and Jeffrey A. Fessler, "Edge-Preserving Image Denoising via Group Coordinate Descent on the GPU," IEEE, pp. 1273-1281, 2015.