Exploiting Data Parallelism in the yConvex Hypergraph Algorithm for Connected Region Decomposition using GPGPUs.

Saurabh Jha, Tejaswi Agarwal
Undergraduate Students
School of Computing Science and Engineering
Vellore Institute of Technology
Chennai, India
{saurabh.jha2010, tejaswi.agarwal2010}@vit.ac.in

B. Rajesh Kanna
Advisor
School of Computing Science and Engineering
Vellore Institute of Technology
Chennai, India
rajeshkanna.b@vit.ac.in

***Abstract*— To define and identify a region-of-interest (ROI) in a digital image, the shape descriptor of ROI has to be described in terms of its boundary characteristics. To address the generic issues of contour tracking the yConvex Hypergraph (yCHG) model was proposed by Kanna et al [1]. This yCHG model represents any connected region as a finite set of disjoint yConvex hyperedges (yCHE), which helps to perform the contour tracking precisely without retracing the same contour. We observe that the serial implementation of the yCHG is quite costly in terms of memory and computation for high resolution images. These issues motivated us to exploit the high data parallelism available on Graphic Processing Units (GPU). In this work, we propose a parallel approach to implement yCHG model by exploiting massive parallel cores of NVIDIA Compute Unified Device Architecture (CUDA). We perform our experiments on the MODIS satellite image database by NASA, and based on our analysis we observe that the performance of the serial implementation is better on smaller images, but once the threshold is achieved in terms of image resolution, the parallel implementation outperforms its sequential counterpart by 2 to 10 times (2x-10x). We also conclude that increase in the number of hyperedges in ROI of given size does not impact the performance of the overall algorithm.**

I. INTRODUCTION AND MOTIVATION

Kanna et al, in their paper [1] discussed the generic issues of contour tracking have been in detail. They are:

a. Contour tracking becomes non-deterministicc at the point of contour intersection.
b. Contour tracking is difficult when the region is bounded by more than one curve (Region with holes)
c. Contour tracking is highly sensitive to noise around the edges.

It becomes important to solve contour tracking challenges in multiply-connected regions and regions bounded by non-Jordan curves. To overcome these challenges, Kanna et al. in their paper [1] proposed the yCHG model which could be used to track the contour deterministically.

In the yCHG model, union of all the hyperedges, referred to as yConvex hyperedges (yCHE) gives the original region of interest. A yCHE is one in which no vertical line intersects the boundary curve more than twice. Intersection of any two hyperedges is empty and all hyperedges are simply-connected bounded by either Jordon or non Jordon curve. The primary focus of [1] was proposing an effective model for image decomposition and geometry of images but did not focus on the computational cost complexity and image resolution.

With the ever increasing size of images, image processing and representation using the yCHG algorithm becomes costly in terms of memory and computational time. To explain our motivation clearly, we have studied several databases which require real-time image analysis. Biomedical research involving large images, like the Digital Database for Screening Mammography maintained by the University of Southern Florida contains a large database of images facilitating research in development of computer algorithms to aid in screening.

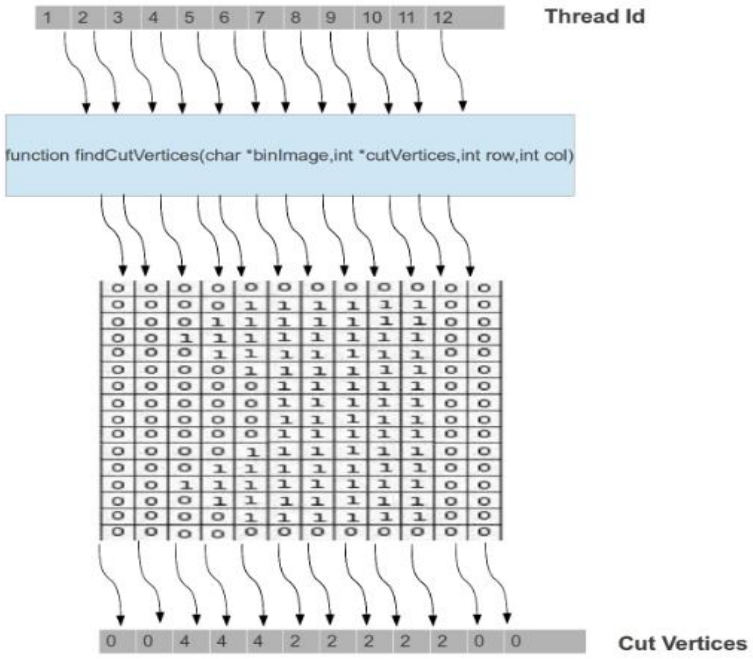
The ASTER satellite image database maintained by NASA contains extensive images of natural hazards, geology and land use of the earth. The MODIS satellite image database, which is a lower spatial resolution to the ASTER, covers the entire earth everyday frequently repeating its coverage to provide real-time images. All the images in the above mentioned databases are of extremely high resolution containing a large number of hyperedges. For our analysis in the rest of the paper, we use the MODIS database as it is available freely for research on the NASA website. Our results with the sequential implementation of the yCHG show that:

a. The runtime of the yCHG algorithm increases linearly for images up to a resolution of 2000x2000 but a significant change in runtime is observed for images with a higher resolution.
b. The runtime remains constant for images with varying number of hyperedges.

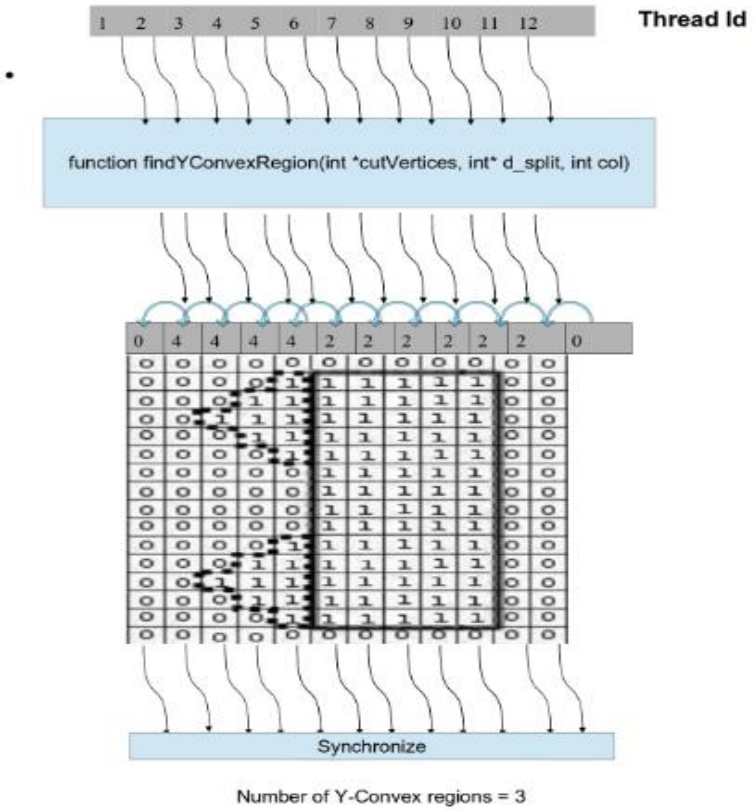
In our contribution, we exploit the inherent data parallelism in the yCHG algorithm on the GPU, which is well suited for parallel processing and accelerating image recognition solutions. Based on the existing yCHG algorithm, we divide an input image to a number of individual column vectors depending on the image resolution, and process each vector on a separate thread on the GPU concurrently. Our implementation results with NVIDIA CUDA show that with an increase in the image resolution the parallel implementation improves the performance of an already fast (relatively) algorithm by 2X-10X, opening up a host of potential new applications that require real time, fast image representation.

II. PROPOSED METHOD:

To remove the data dependencies in the existing algorithm, we divide the algorithm into two steps. The first function computes the number of cut-vertices of an image in parallel. The image is divided into a number of column vectors and each column is scheduled on a separate thread on the GPU. Each thread computes the number of cut-vertices and stores the result in an array. The figure below explains the process graphically.



In the second step of the algorithm, we compute the convex regions by checking the number of cut-vertices in the preceding column vector of the image. Thus, if there is a change in the number of cut-vertices it indicates that there has been a change in the number of convex regions. The figure below explains the second function graphically.



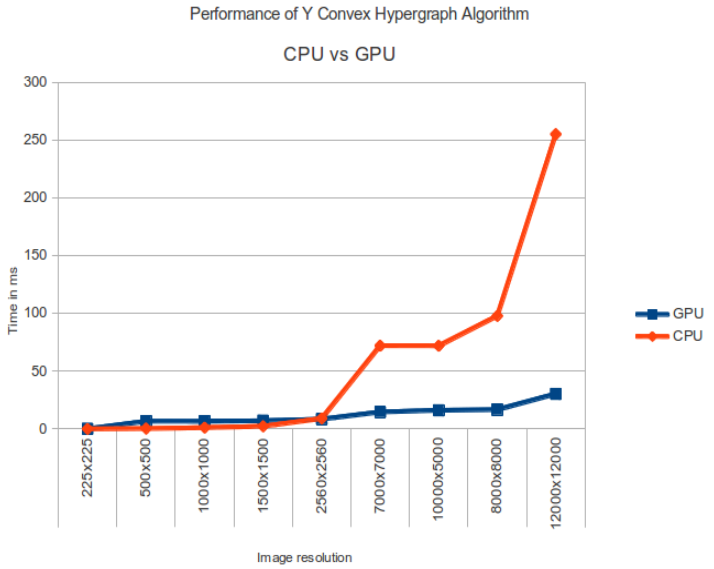
III. RESULTS:

For our experiments, we use the NASA MODIS database, which consists of images with resolutions of up to 21000 x 21000. In order to keep constant hyperedges, we take a single image of a resolution of 21000x21000 and vary the resolution. This image is used as in input to the serial and parallel implementations of the yCHG. Our results are graphically shown below.

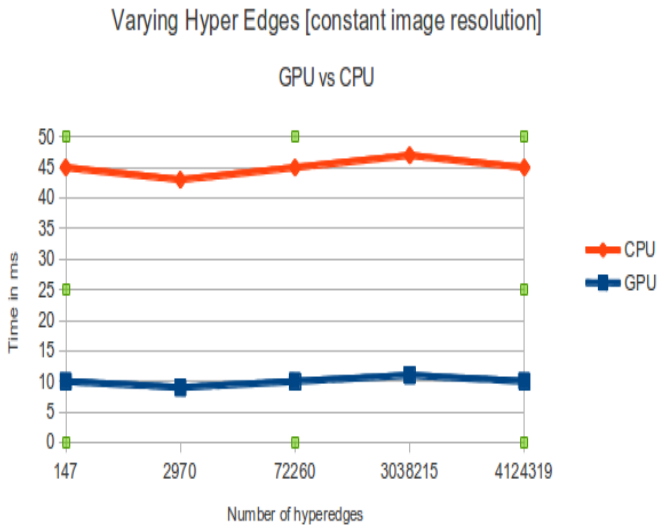
Table: System Specifications:

		Number of cores	**Clock Speed**	**Memory Clock**	**Memory interface width**
CPU	Intel Core i5 480M	2	2660 MHz	1600 MHz	64 bit
GPU	NVIDIA GeForce 310M	16	1468 MHz	790 MHz	64 bit

We understand that while newer hardware (such as cards based on the recently released FERMI architecture) would undoubtedly be faster, we want to show what is possible with only limited hardware investment.



To vary the number of hyperedges, we consider different images from the MODIS database, keeping the same resolution. We observe that the time taken is constant, with the number of hyperedges varying from 147 to 4124319, as shown in the figure below. This was expected as our algorithm is dependent directly on the resolution of the input image irrespective of other factors.



IV. CONCLUSION

In this paper, we exploit data parallelism in an existing yCHG algorithm by enhancing the algorithm to remove the data dependencies. Our implementation results with NVIDIA CUDA show that with an increase in the image resolution the parallel implementation improves the performance of an already fast (relatively) algorithm by 2X-10X, opening up a host of potential new applications that require real time image processing.

V. REFERENCES:

[1] B. Rajesh Kanna, C. Aravindan, and K. Kannan (2012), Development of yConvex hypergraph model for contour-based image analysis. Proceedings of the 2nd IEEE International conference on Computer Communication and informatics - (ICCCI 2012).