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

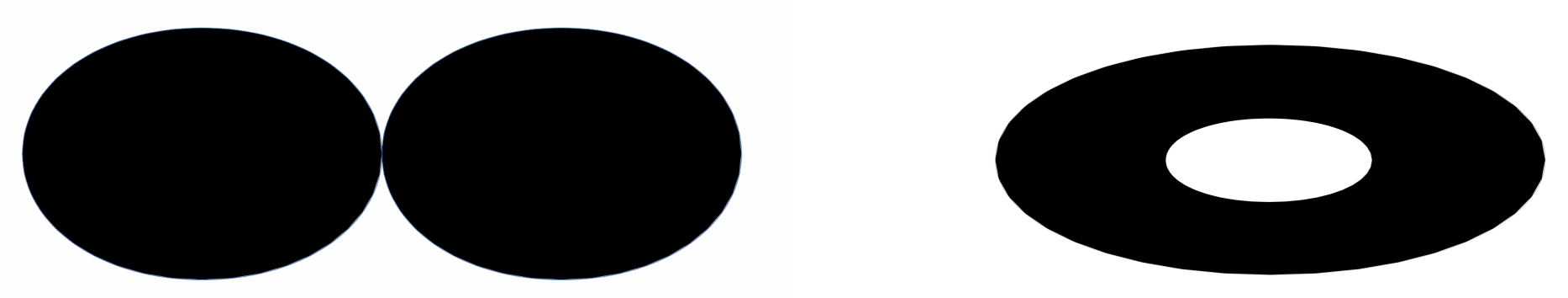
Saurabh Jha, Tejaswi Agarwal and Rajesh Kanna B
School of Computing Sciences and Engineering, VIT University, Chennai, India
saurabh.jha2010@vit.ac.in, tejaswi.agarwal2010@vit.ac.in, rajeshkanna.b@vit.ac.in

1. ABSTRACT

To define and identify a region-of-interest (ROI) in a digital image, the shape descriptor of the 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 computatio n for high resolution images. These issues motivated us to exploit the high level data parallelism available on Graphic Processing Units (GPUs). In this work, we propose a parallel approach to implement yCHG model by exploiting massively 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 an increase in the number of hyperedges in ROI of given size does not impact the performance of the overall algorithm

2. INTRODUCTION AND MOTIVATION

Generic issues in contour tracking:
1. Contour tracking becomes non-deterministic at the point of contour intersection.
2. Contour tracking is difficult when the region is bounded by more than one curve (Region with holes)
3. Contour tracking is highly sensitive to noise around the edges.



With the ever increasing size of images, image processing and representation using the yCHG algorithm becomes costly in terms of memory and computational time.

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.

3. OUR CONTRIBUTION

Our results with the sequential implementation of the yCHG show that:

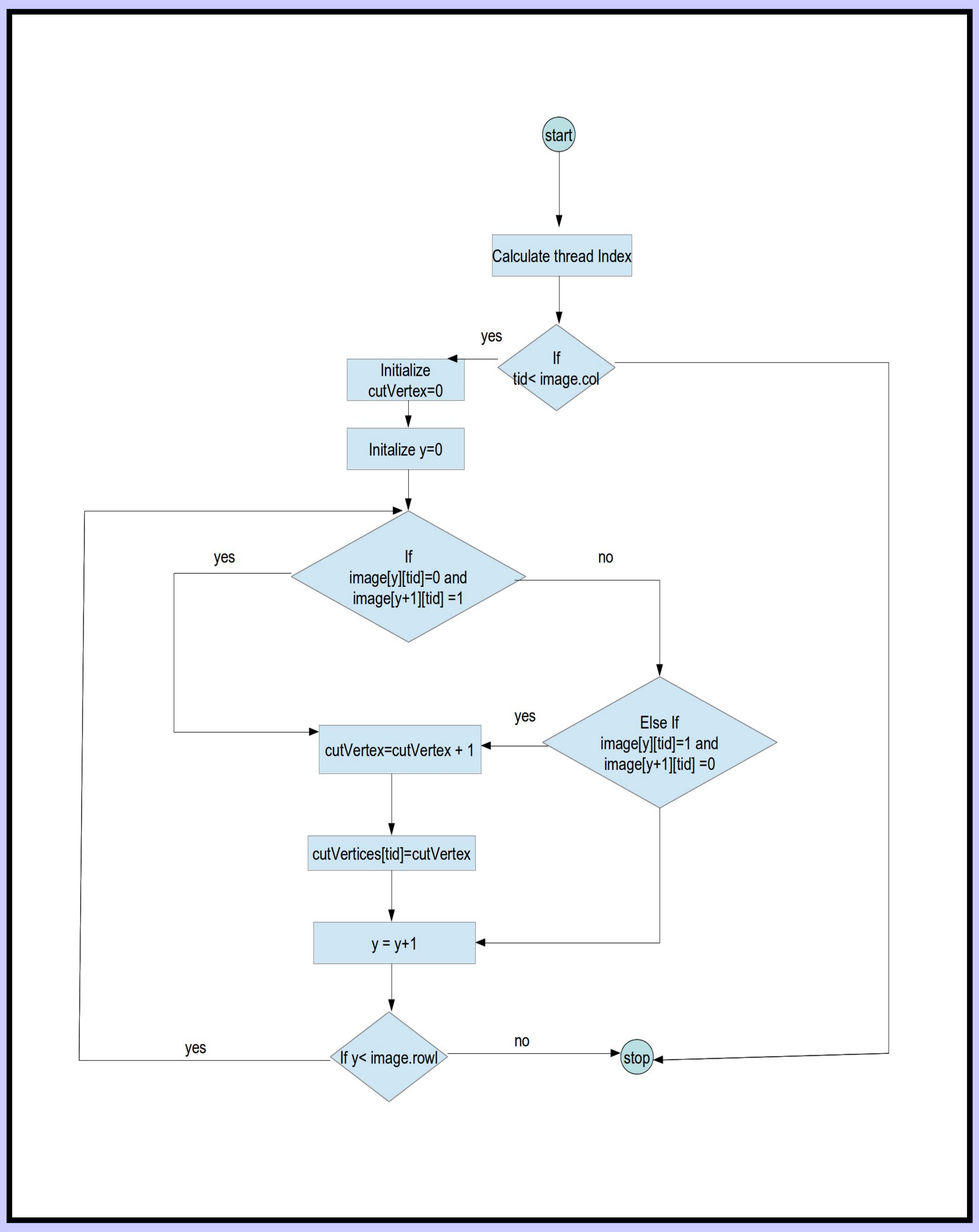
a. The runtime of the yCHG algorithm increases linearly for images up to a resolution of 2000x2000 but a significant change in runtime is observed for images with a higher resolution.

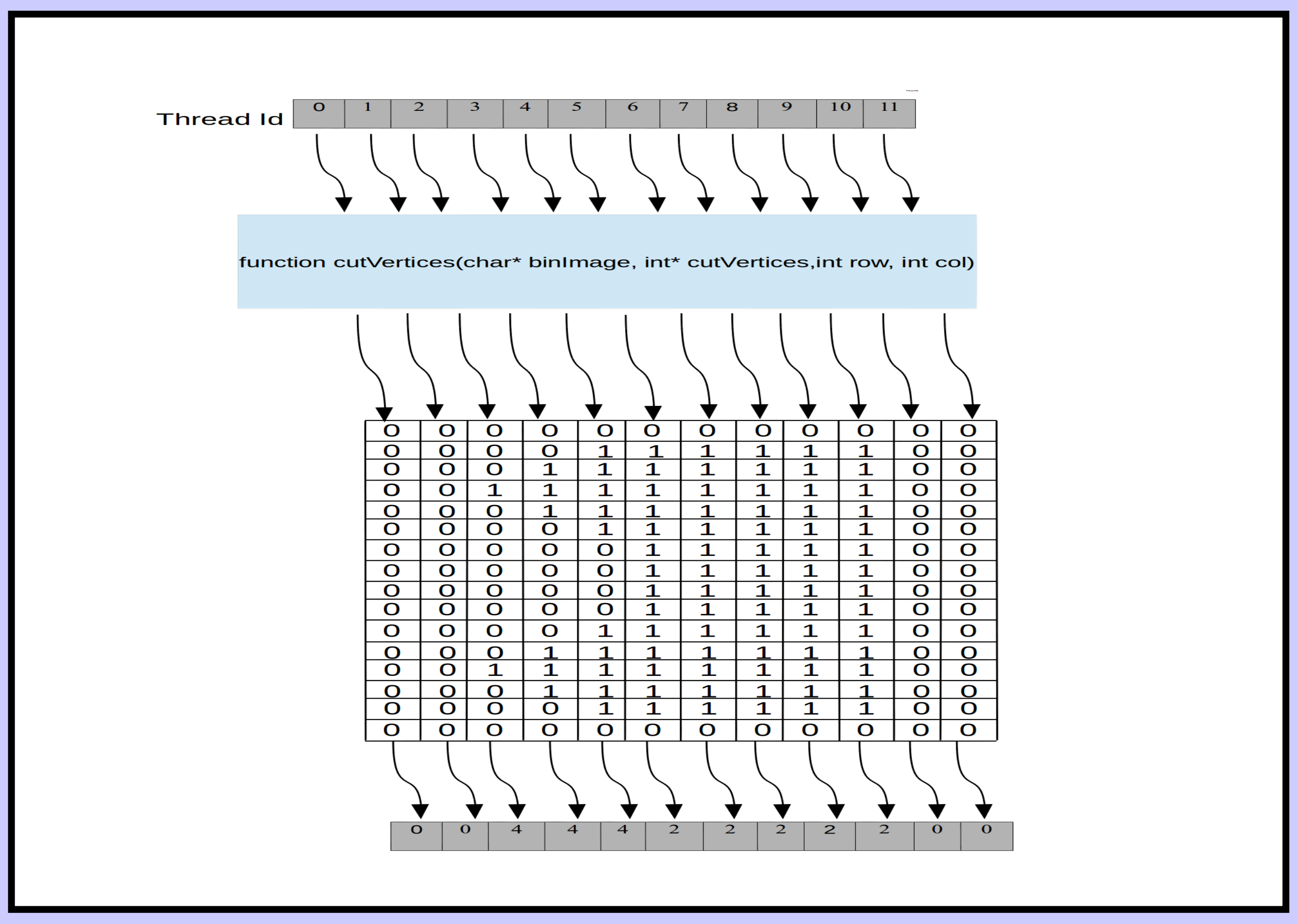
b. The runtime remains constant for images with varying number of hyperedges.

In our contribution, we exploit the inherent data parallelism in the yCHG algorithm on the GPU, which is well suited for parallel processing. 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.

4. PROPOSED ALGORITHM

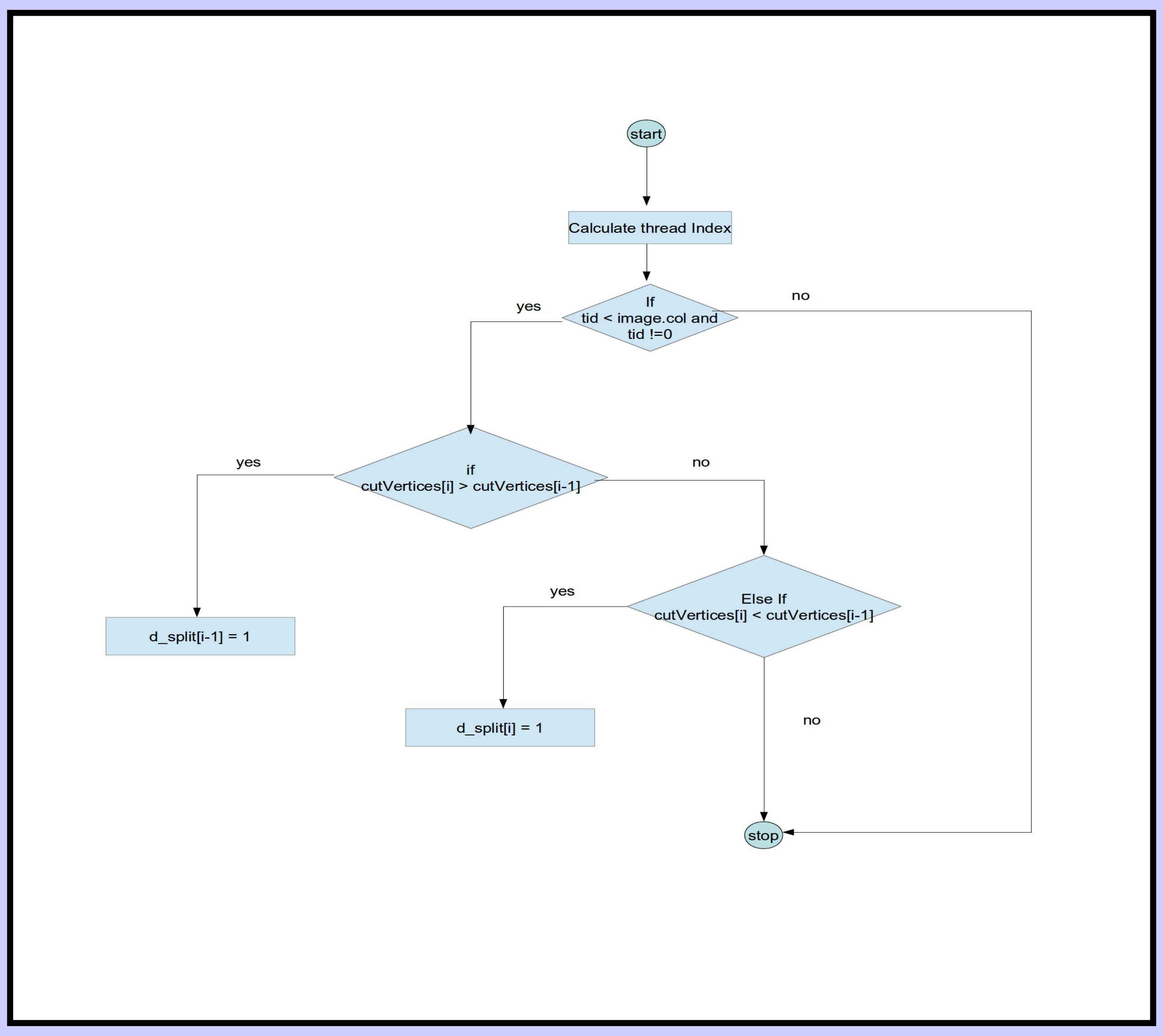
4.1 Finding Cut Vertices

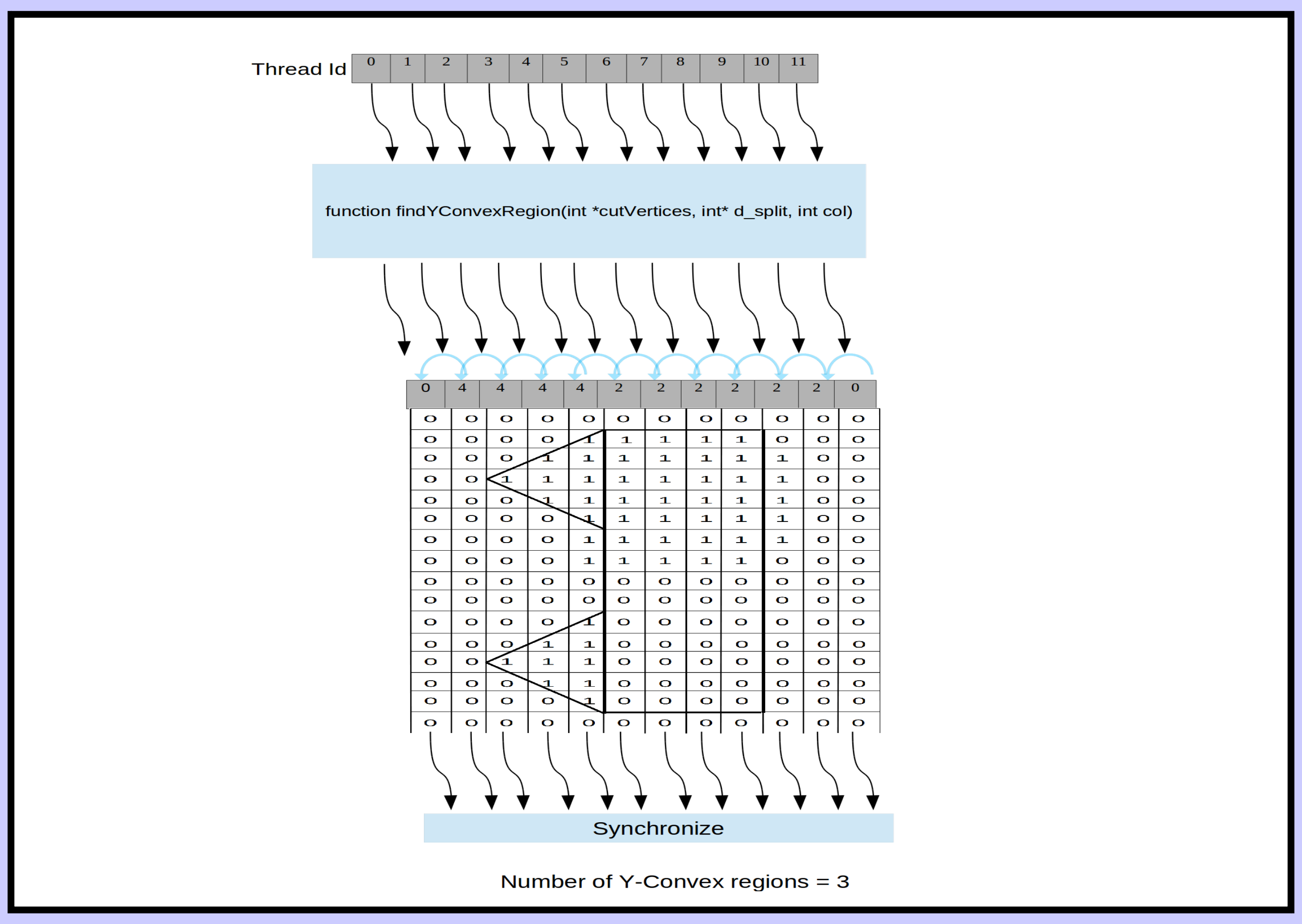



Thread Id
function cutVertices(char* binImage, int* cutVertices,int row, int col)

0	1	2	3	4	5	6	7	8	9	10	11
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	1	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	1	1	0	0
0	0	0	1	1	1	1	1	1	1	0	0
0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	0	0	1	1	1	1	1	0	0
0	0	0	0	0	1	1	1	1	1	0	0
0	0	0	0	0	1	1	1	1	1	0	0
0	0	0	0	0	1	1	1	1	1	0	0
0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	1	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	1	1	0	0
0	0	0	1	1	1	1	1	1	1	0	0
0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	4	4	4	2	2	2	2	2	0	0

4.2 Finding yConvex Regions

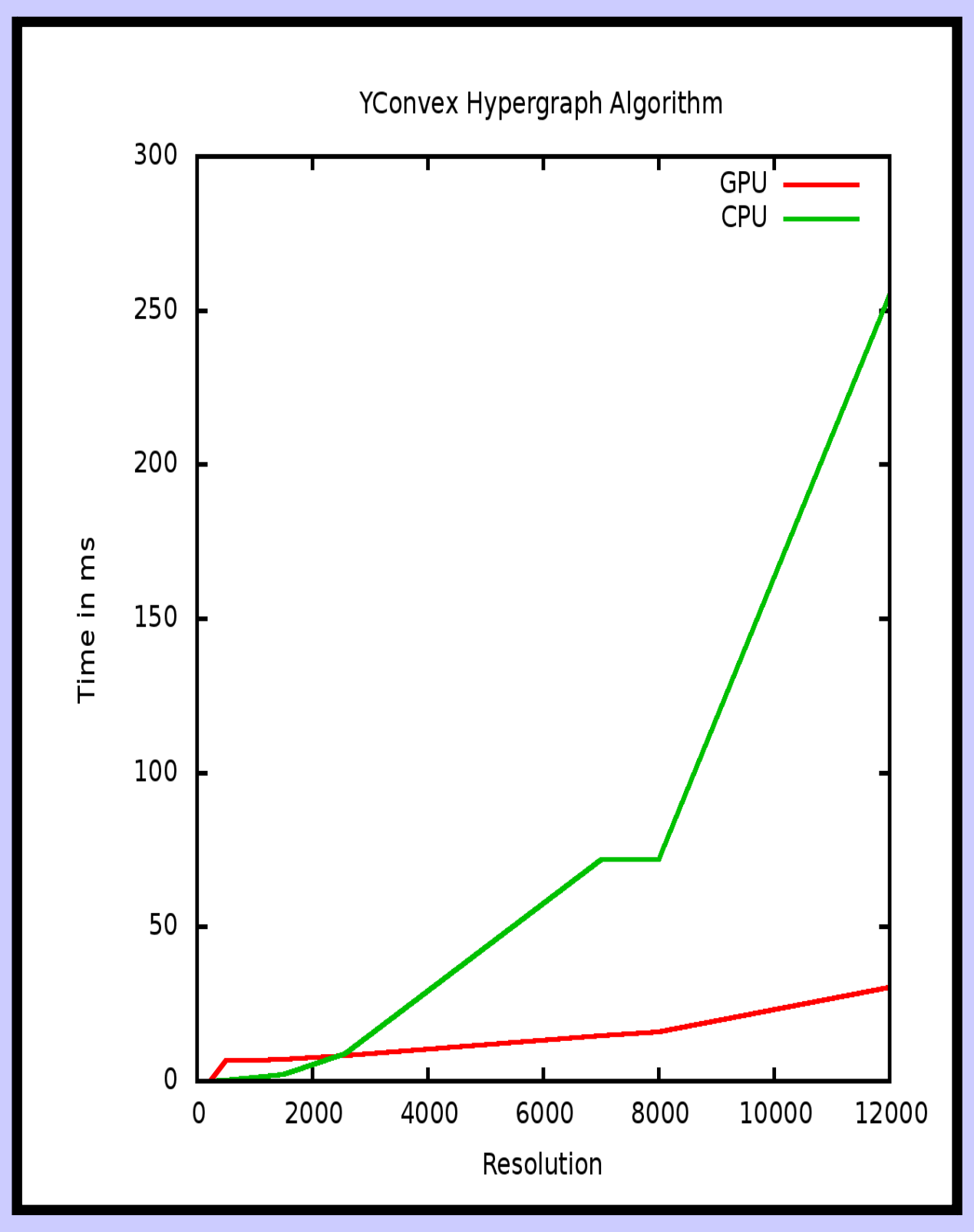

start
Calculate thread Index
If tid < image.col and tid !=0
yes
no
If cutVertices[i] > cutVertices[i-1]
yes
no
d_split[i-1] = 1
Else If cutVertices[i] < cutVertices[i-1]
yes
d_split[i] = 1
no
stop

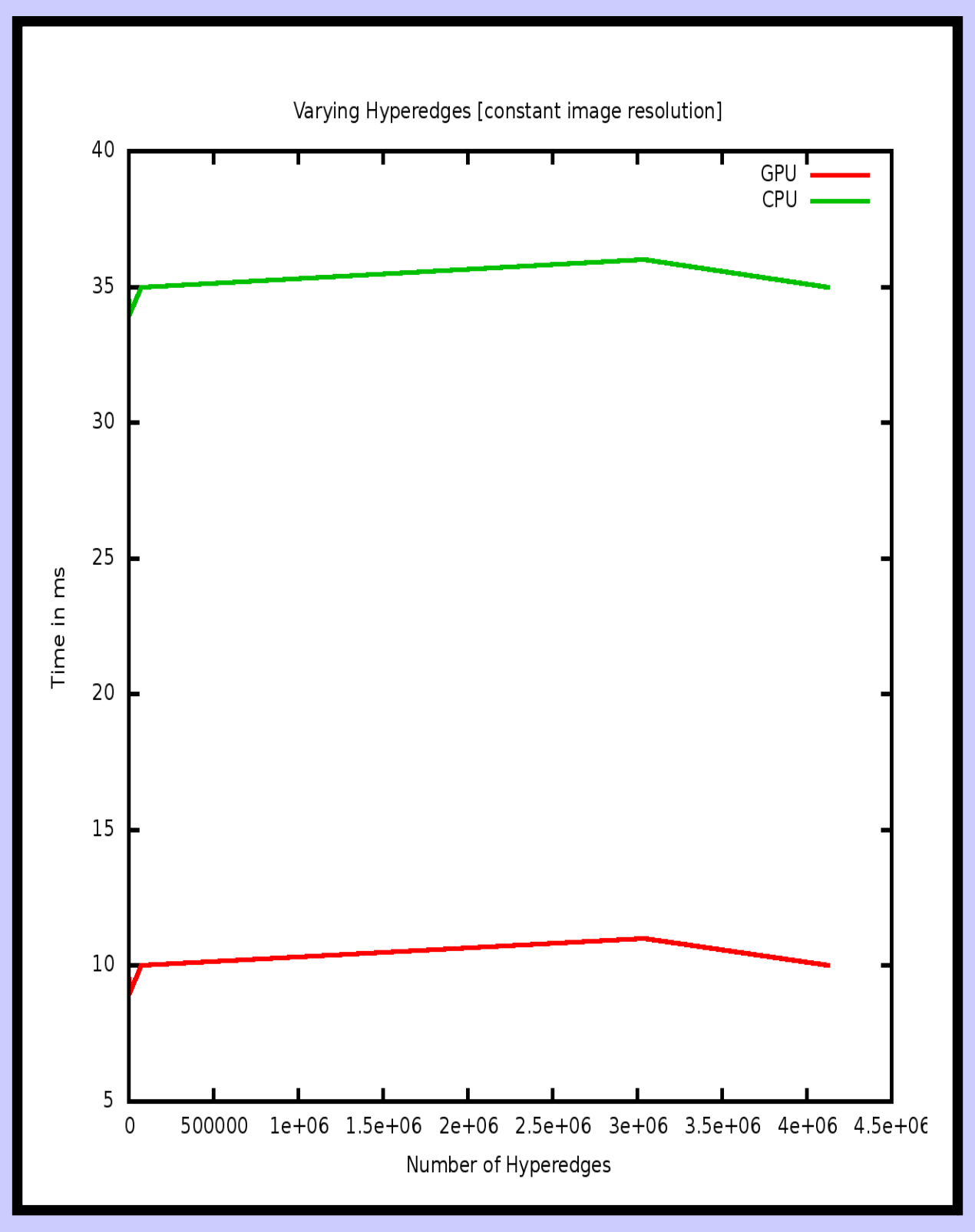

Thread Id
function findYConvexRegion(int *cutVertices, int* d_split, int col)
Synchronize

0	1	2	3	4	5	6	7	8	9	10	11
0	4	4	4	4	2	2	2	2	2	2	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	1	1	0	0
0	0	0	1	1	1	1	1	1	1	0	0
0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	0	1	1	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

Number of Y-Convex regions = 3

5. RESULTS





Result 1

Once the threshold of resolution2000 X 2000 is reached, the parallel yCHG algorithm proposed by us outperforms the serial counterpart by 2 times to 10 times (2x-10x).

Result 2

Varying the number of hyperedges in an image with constant resolution, does not impact the performance of the overall algorithm. This is due to the fact that our proposed algorithm is directly dependent on the resolution of the input image irrespective of other factors.

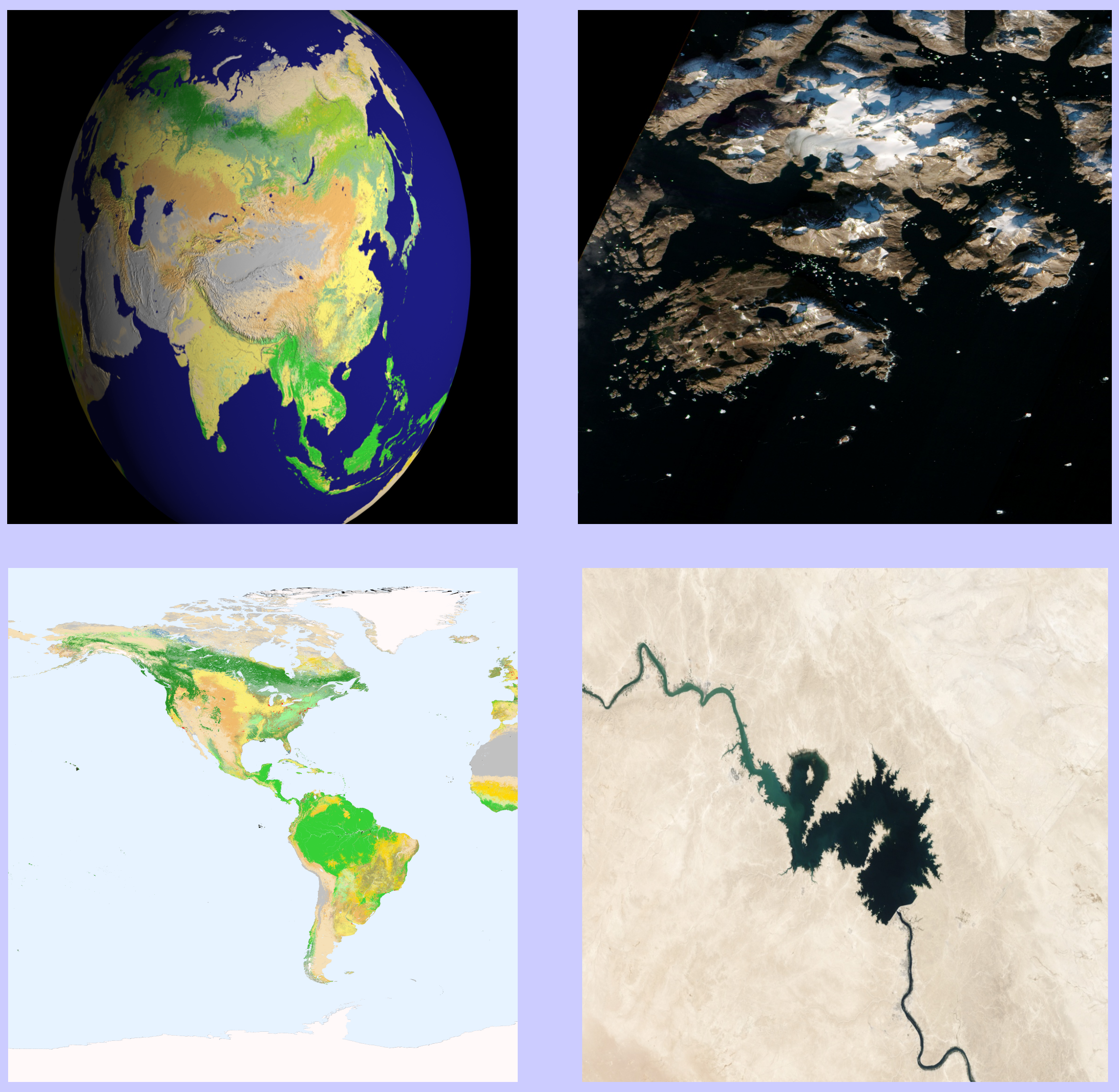
In order to keep constant hyper edges, we take an image of a resolution of 21000x21000 and vary the resolution.

We used the following system specification for our analysis.

		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

On basis of our experiments, we find out the following:
1.For an image with resolution greater than the threshold of 2000x2000, the GPU implementation outperforms the CPU by a factor of 2x -10x.
2. Varying the number of hyperedges in an image with constant resolution does not impact the performance of the overall algorithm. This is due to the fact that our proposed algorithm is directly dependent on the resolution of the input image irrespective of other factors

6. MODIS DATASET



Some of the sample images from MODIS dataset are shown above. 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.

7. CONCLUSIONS

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

REFERENCES

[1] B. Rajesh Kanna, C. Aravindan, and K. Kannan, Development of yConvex hypergraph model for contour-based image analysis, in Proceedings of the 2nd IEEE International Conference Computer Communication and Informatics (ICCCI-2012), 2, 1-5, 2012
[2] B. Rajesh Kanna, C. Aravindan, and K. Kannan, A contour-based scheme for representing arbitrary shapes in digital images, in Proceedings of ACM International Conference and Workshop on Emerging Trends in Computer applications, 1, 535-540, 2011
[3] B. Rajesh Kanna, C. Aravindan, and K. Kannan, Image-based area estimation of any connected region using y-convex region decomposition, AEU -International Journal of Electronics and communications, 66 (2):172- 183, 2012
[4] Hall, Dorothy K., George A. Riggs, and Vincent V. Salomonson. 2006, updated daily. MODIS/Terra Snow Cover Daily L3 Global 500m Grid V005 [November 2000-October 2001]

ABOUT THE AUTHORS

Saurabh Jha is a junior year undergraduate student majoring in Computer Science and Engineering. His research interests are Parallel Computing, Heterogeneous Systems and Distributed Computing.

Tejaswi Agarwal is a junior year undergraduate student majoring in Computer Science and Engineering. His research interests are Parallel Computing, Computer Architecture, and Multi-core Systems.

Rajesh Kanna B is a professor at VIT University, India. His research interests are Image Processing and its applications.