Complex Geometric Primitive Extraction on Graphics Processing Unit

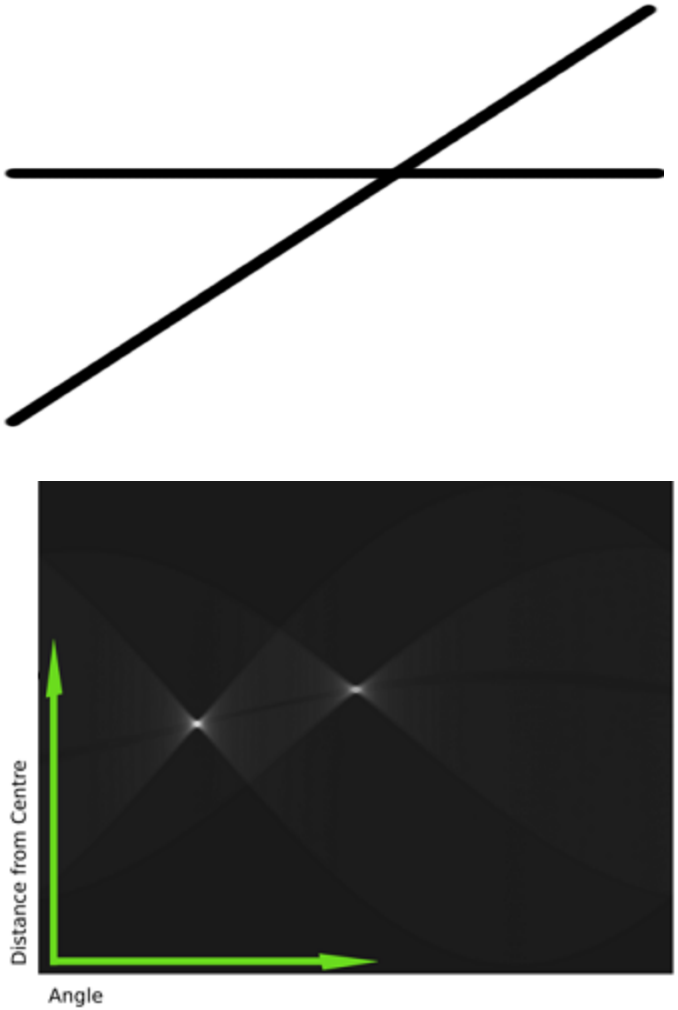
Mert Değirmenci
Department of Computer Engineering,
Middle East Technical University, Turkey
mert.degirmenci@ceng.metu.edu.tr

Outline

- Geometric primitive extraction
- Genetic algorithm context
- Motivation for a GPU implementation
- Process of primitive extraction
  - Edge detection on GPU
  - Genetic algorithm on GPU
- Results & Discussion

Geometric primitive extraction

- Extraction of low level information from the image. Higher level of information can be built on clustering appropriate lower levels of information.
- Hough Transform is one approach to extract a primitive equation of which is known in advance. The idea is to perform a mapping from image space to parameter space in order to obtain a function. Optima's of this function correspond to instances of primitives.

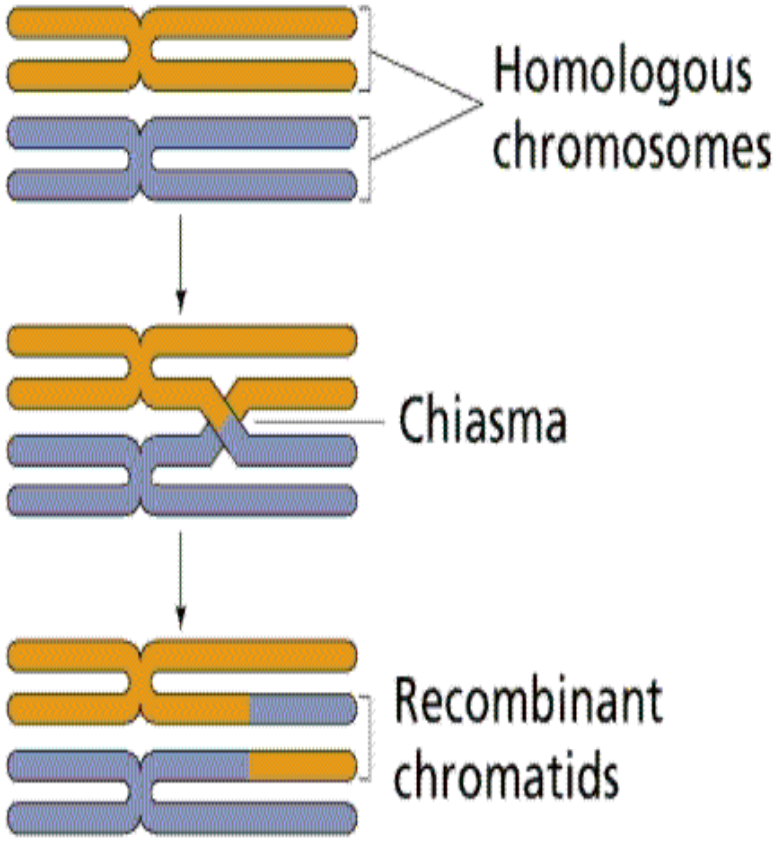


Geometric primitive extraction

- HT is powerful for detecting simple primitives such as lines. However, computational and memory complexity of classical Hough transform increase exponentially along with the number of parameters [Pen99].
- Imperfectness of primitives contributes negatively to the result. Hough transform allievates this problem by including fuzziness. Approach is called fuzzy hough transform, FHT.
- Since parameter number effects running time exponentially, hough transform overcomputes large number of instances of primitives. The solution is adressed by [Kul90] which includes randomness to this process.

Genetic algorithm context

- Genetic algorithm is a technique to find solutions to optimization problems. [Rot93] has shown formally that geometric primitive extraction is, in fact, an optima search problem.
- The information about a primitive is encoded to a chromosome of an individual. Inspirations from evolutionary biology leads to a handful of operations (mutation, inheritance, crossover).



Genetic algorithm context

- Genetic algorithm is algorithmatically implemented as follows :
- Choose the initial population of individuals.
  Evaluate the fitness of each individual in that population
  Repeat on this generation until termination:
  - Select the best-fit individuals for reproduction.
  - Breed new individuals through crossover and mutation operations to give birth to offsprings.
  - Evaluate the individual fitness of new individuals
  - Replace least-fit population with new individuals
- [Pen99] and many others realizes this process by representing a primitive as an individual, and applying operators to chromosomes. All of them reports improvements over HT based methods.

Motivation for a GPU implementation

- For our algorithms to truly scale with our datasets, researchers must consider parallel computation of algorithms.
- In our implementation, we have processed all the information in graphics processing units.
- A naive implementation may not include parallel computation of edge detector since it takes small fraction of time required for current central processing units. But as the size of datasets increase, researchers should adress all problems from parallel computing angle.

Motivation for a GPU implementation

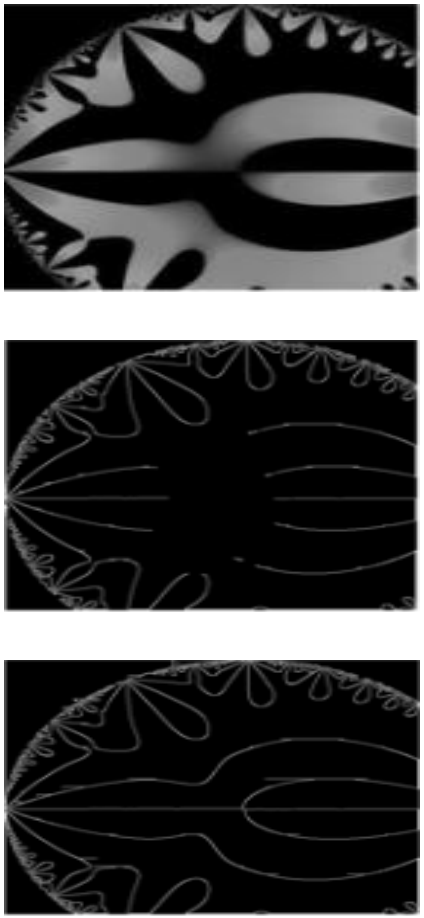
- Current graphics processing units come along with a number of parallel processors providing necessary parallel computation platform for researchers.
- We have chosen Nvidia's Compute Unified Device Architecture, CUDA, as our computing platform due its availabilty and efficiency.
- CUDA features several advantages over general purpose computation on GPUs.
  - It makes possible reading from arbitrary addresses in memory.
  - CUDA compliant GPUs also support on-chip shared memory for fast communication between threads on the same warps.

Edge detection on GPU

- Our GPU has 16 KB shared memory. Since we do not have enough memory for contour images, our expectation from an edge detector is to produce minimal response to a true edge.
- We have chosen Canny's edge detection algorithm aim of which is to produce single response for an edge while maintaining the purpose of detecting all edges in the image.
- Canny's algorithm first smoothens image, computes gradient information, and suppresses non-maximal edges. All of these operations are local to a segment of an image. However last stage of Canny's algorithm to find the connected components requires global information. A pixel might be decided to be true edge because of an edge at the farthest pixel.

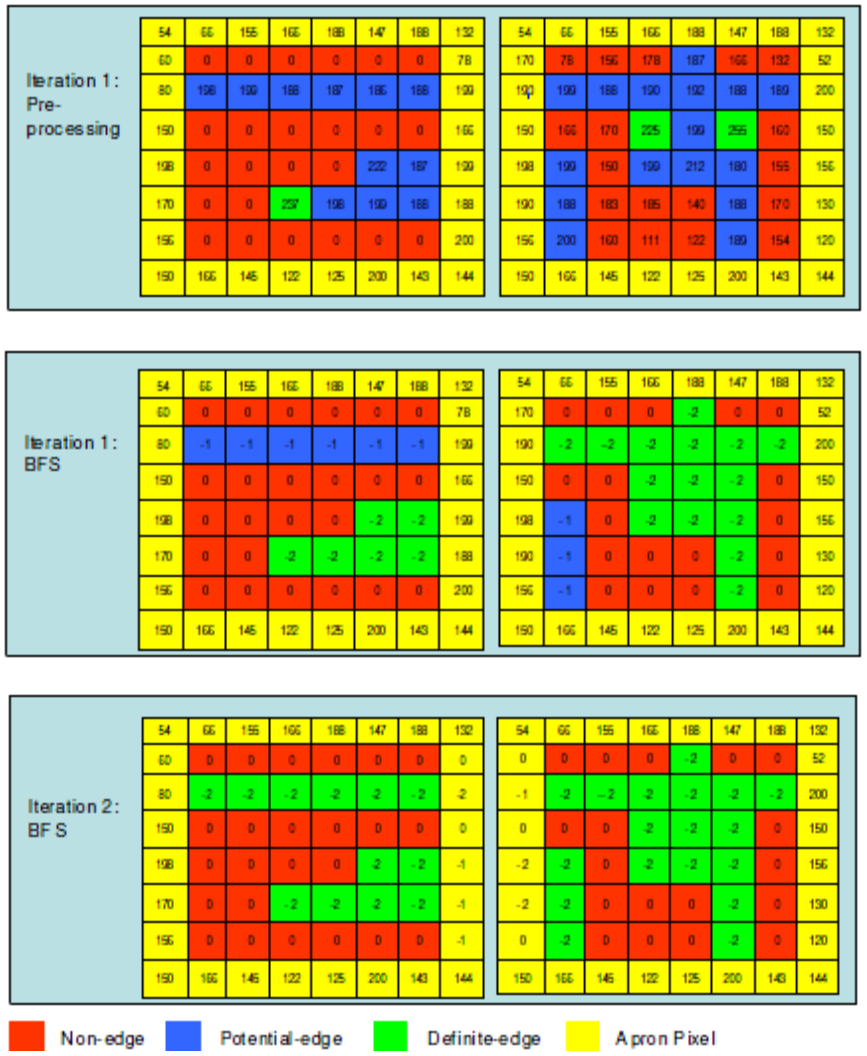
Edge detection on GPU

- Yuancheng et al. have already implemented Canny's algorithm on CUDA [Yua08] . They have released source code of their implementation. However, their implementation is partial since it makes fixed number of iterations to find the connected components at the last stage of Canny's algorithm.
- Figure right shows Yuancheng's Canny edge detector implementation with four and ninety iterations to find connected components.
- As seen in figure right, their algorithm is unable to find connected components path of which involve more than four segments .



Edge detection on GPU

- Yuancheng's implementation uses breadth first search algorithm to find connected components.
- Their kernel call to GPU finishes with BFS of a segment.
- In this implementation, only threads at starting points of BFS is active.
- Figure shows an iteration of two segment image with Yuancheng's implementation.



Edge detection on GPU

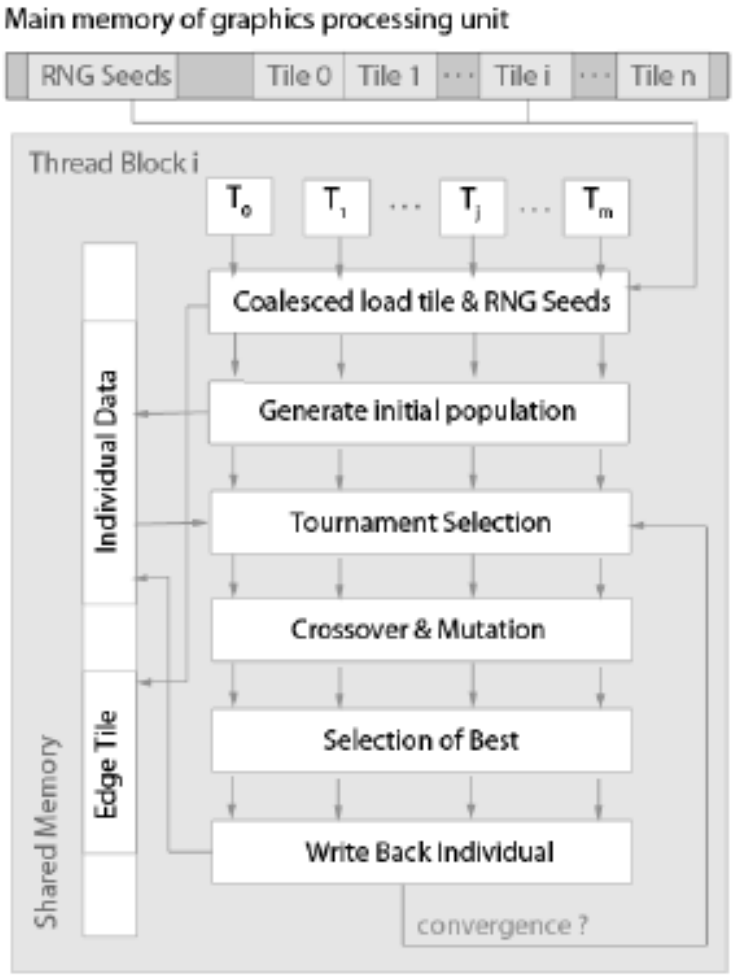
- In our implementation, we have used parallel breadth first search algorithm to detect connected edges, proposed by Pawan et al. [Paw07]. Details of algorithm is shown in figure right.

```
1 : For each definite edge do in parallel
2 :     Set frontier [ edge ] <- True , visited [ edge ] <- False
3 : End For
4 : For each edge do in parallel
5 :     If frontier [ edge ] is True
6 :         Set frontier [ edge ] <- False , visited [ edge ] <- True
7 :         For each neighbour,potential, and not visited edge
8 :             Set frontier [ not visited edge ] <- True
9 :         End For
10 :    End If
11 : End For
```

- Adaptation of Pawan's algorithm leads to increase in efficency and reliability. Our implementation stops with correct and unique BFS tree as opposed to Yuancheng's implementation of Canny edge-detector which makes fix number of iterations to find connected components.

Genetic algorithm on GPU

- Our implementation of geometric primitive extraction starts by detecting edges of image.
- Edge image is then partitioned into tiles on which a copy of our genetic algorithm runs.
- Each block works on its corresponding isolated island.
- Each thread is responsible for an individual on his island.



Genetic algorithm on GPU

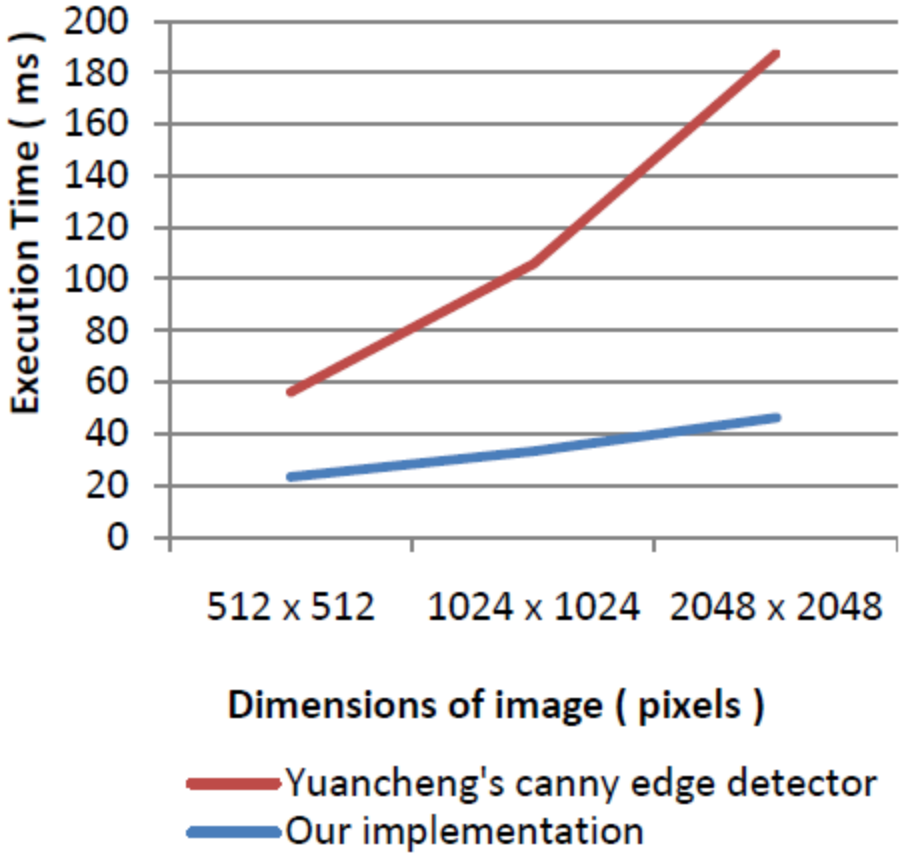
- Individual representation :
  - Minimal number of points that uniquely define geometric primitive.
- Initial population generation :
  - Each thread generating a random chromosome.
- Fitness Evaluation

$$\sum_{x,y}\left(MAX(E(x+i,y+j)-(|i|+|j|)/c)_{\forall\, i,j}\right)$$

  - Where E(x,y) is 1 if there exists an edge pixel on point (x,y), 0 otherwise.
- Crossover
  - In CUDA, every thread selects her own mate to crossover randomly. We have implemented tournament selection on GPU as in [Pos09].
- Mutation
  - Mutation, in our implementation, simply changes a random bit of a chromosome with low probability of occurring.

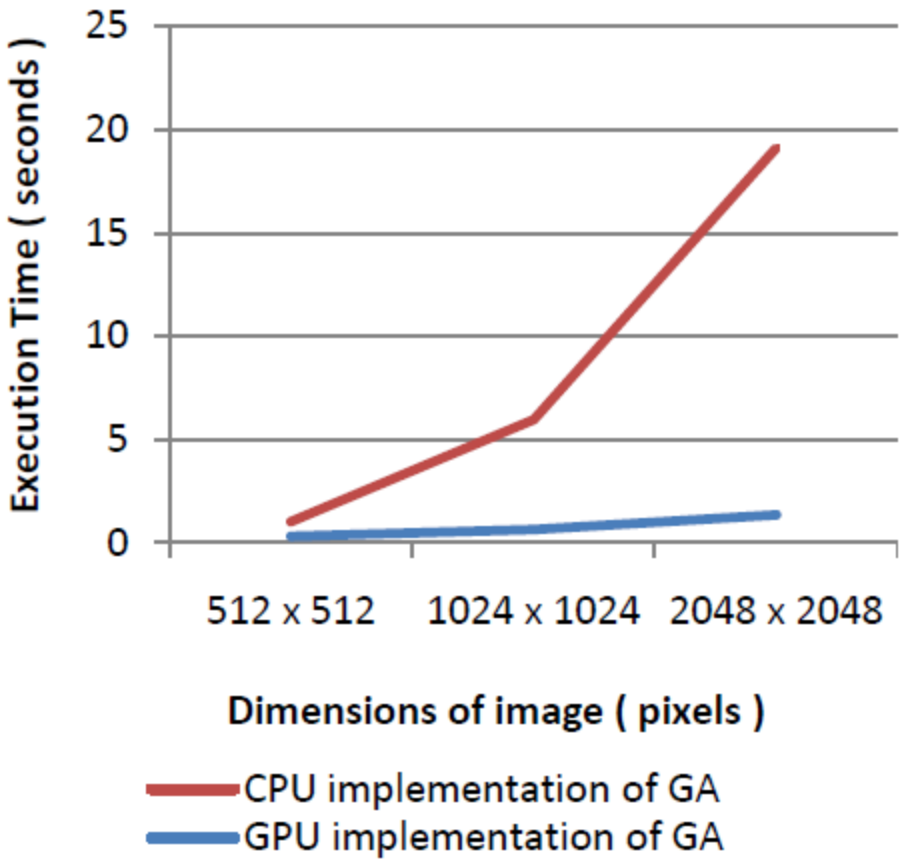
Results & Discussion

- A moderate improvement on canny edge detector has been achieved in our study.
- CUDA implementation of Yuancheng has been improved in terms of reliability and efficiency.



Results & Discussion

- Ellipse detection algorithm has been tested with over 600 images contained in our database.
- CPU implementation has been tested on Intel Core™ i7 at 2.67 GHZ.
- GPU implementation has been tested on Nvidia GeForce GTX 260 graphics card.



References

- [Pen99] Peng-Yeng Yin, A new circle/ellipse detector using genetic algorithms, Pattern Recognition Letters, Volume 20, Pages 731-740, Issue 7, July 1999.
- [Kul90] Kultanen, P.; Xu, L.; Oja, E., "Randomized Hough transform (RHT)," Pattern Recognition, 1990. Proceedings., 10th International Conference on , vol.i, no., pp.631-635 vol.1, 16-21 Jun 1990.
- [Pos09] Pospichal P. , and Jaros J. , GPU-based Acceleration of the Genetic Algorithm, from contest GPUs for Genetic and Evolutionary Computation, 2009.
- [Rot93] G. Roth and M. D. Levine, “Extracting geometric primitives,” Comput. Vision. Graphics Image Processing: Image Understanding, vol. 58, pp. 1-22, 1993.
- [Pen99] Peng-Yeng Yin, A new circle/ellipse detector using genetic algorithms, Pattern Recognition Letters, Volume 20, Pages 731-740, Issue 7, July 1999.
- [Yua08] Yuancheng Luo; Duraiswami, R., Canny edge detection on NVIDIA CUDA, Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on, vol., no., pp.1-8, 23-28 June 2008.
- [Paw07a] Pawan Harish and P. Narayanan. Accelerating large graph algorithms on the gpu using cuda. pages 197–208. 2007.

Thank you for your attention.
Any questions ?