Canny Edge Detector

Institute for Parallel and Distributed Systems, IPVS, University of Stuttgart

Gopika Rajan:	3575765	st179830@stud.uni-stuttgart.de
Swathi Shridhar:	3578034	st179840@stud.uni-stuttgart.de
M.Mehroz Khan:	3523539	st180421@stud.uni-stuttgart.de

1. Objective

Implementation of Canny edge detection algorithm serially using CPU, and parallelizing it with the help of GPU. Analyzing the output image and performance parameters from multiple stages of the algorithm.

2. Overview

Canny edge detection is implemented to detect useful edges from an input image, and thereby giving significant reduction in the data to be processed and stored.

A grayscale image in .pgm format is taken as the Input. As a pre-processing step, histogram equalization is performed on the image to improve contrast and enhance the edges. Now the image is ready for Canny edge detection.

The Canny edge detection algorithm consists of several steps. First, a Gaussian filter is applied on the resulting image to reduce noise. Next, the gradient magnitude, orientation angle and segment of each pixel is obtained using a Sobel filter. The algorithm then performs non-maximum suppression to suppress all gradient values except the pixels corresponding to steep edges. Then the strong edges and weak edges are identified with double threshold application. Finally, the weak edges that are connected to the strong edges are also classified as strong edges, while isolated weak edges are discarded by hysteresis edge tracking [1].

Click on the link to access the project GitHub repository.

3. Tasks implementation and delegation

Task 1: Defining the problem and literature review.
Task 2: Choosing the logic for Gaussian filter and its CPU implementation[2].
Task 3: Choosing the logic for Sobel filter and its CPU implementation[4].
Task 4: Choosing the logic for Non – max suppression and its CPU implementation[2].
Task 5: Choosing the logic for Double Threshold and its CPU implementation[2].
Task 6: Choosing the logic for Hysteresis edge tracking and its CPU implementation[2].
Task 7: GPU implementation of Gaussian filter.
Task 8: GPU implementation of Sobel filter.
Task 9: GPU implementation of Non – max suppression.

Task 10: GPU implementation of Double threshold.
Task 11: GPU implementation of Hysteresis edge tracking.
Task 12: Identifying and implementing (CPU and GPU) a suitable preprocessing method in order that the edge detection be flexible with any image (Histogram equalization)[6].
Task 13: Calculating and the performance parameters for each intermediate CPU and GPU functionalities and the overall performance.
Task 14: Adding relative addressing, function descriptions to the code, repository maintenance.
Task 15: Discussion sessions for synchronizing and debugging.
Task 16: Documentation and report creation.

Task 1 was done in 2 discussion sessions with all the group members. Possible logics for intermediate functions were discussed in these sessions. Task 2 through Task 16 were delegated among the group members. Weekly sessions were conducted to monitor individual progress, synchronizing, and debugging the code. Possibilities on further possible modifications and improvements were also considered during these sessions.

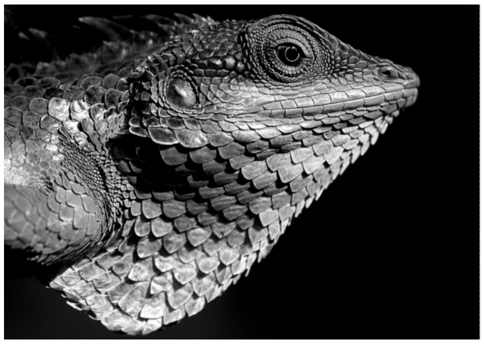
Task delegation:

Team member	**Task number**
Gopika Rajan	1, 2, 3, 4, 10, 11, 13, 14, 15, 16.
Swathi Sridhar	1, 5, 6, 7, 8, 9, 12, 14, 15, 16.
M.Mehroz Khan	1, 5, 6 ,7, 9, 14, 15, 16.

4. Results and Comparisons:

This section shows the input, the intermediate and the final output images of the Canny edge detection algorithm.

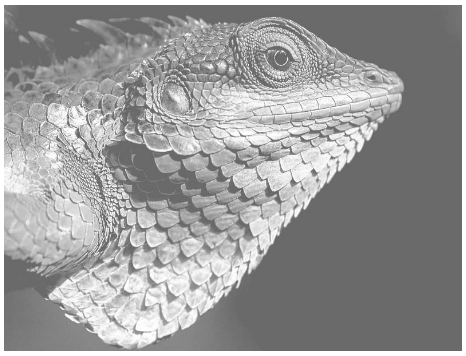
Input Image: Lizard.pgm (800 x 565 Pixels)[5]



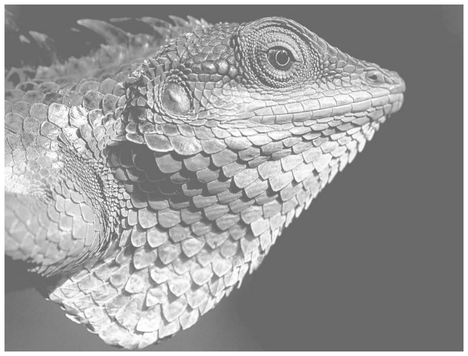
Output Images from CPU: Intel CORE i7 10th GEN and **GPU:** NVIDIA GeForce MX250 (6.1)

CPU outputs

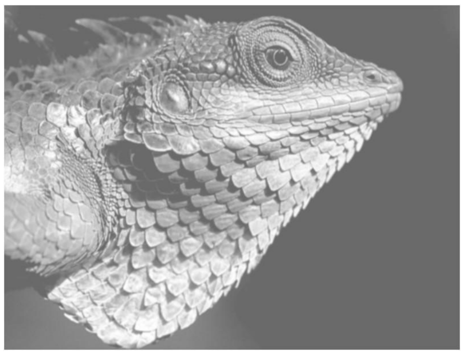
GPU outputs



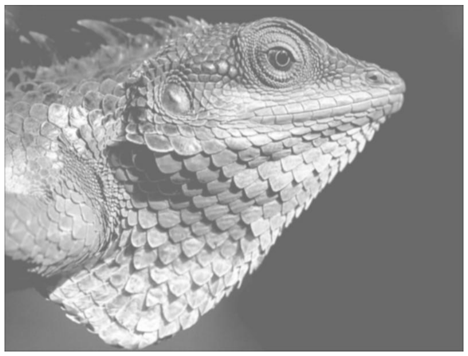
1_Histogram_Equalization_Cpu_Output.pgm



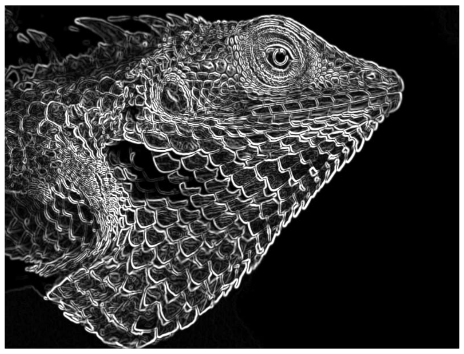
7_Histogram_Equalization_Gpu_Output.pgm



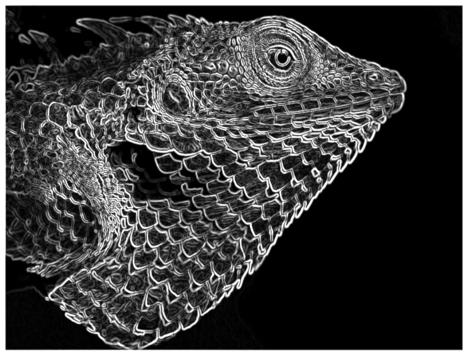
2_Gaussian_Cpu_Output.pgm



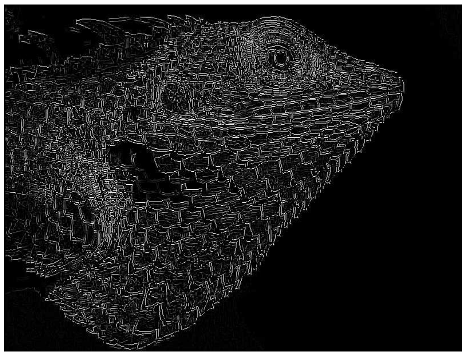
8_Gaussian_Gpu_Output.pgm

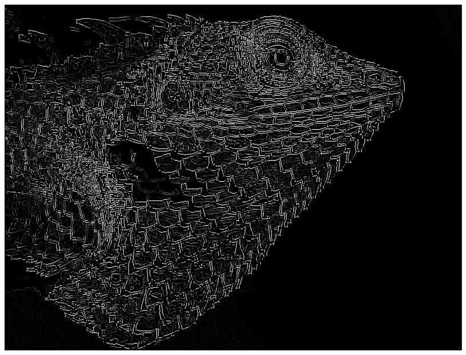


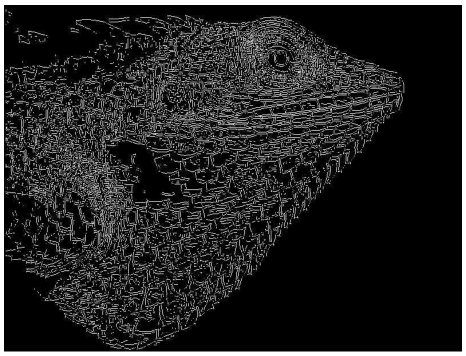
3_Sobel_Cpu_Output.pgm

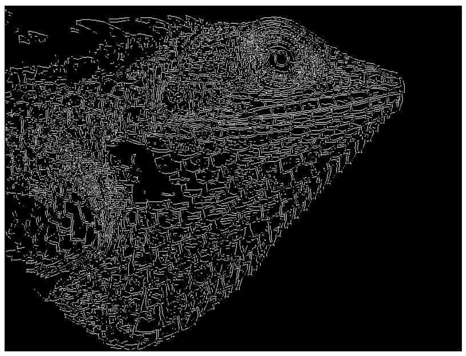


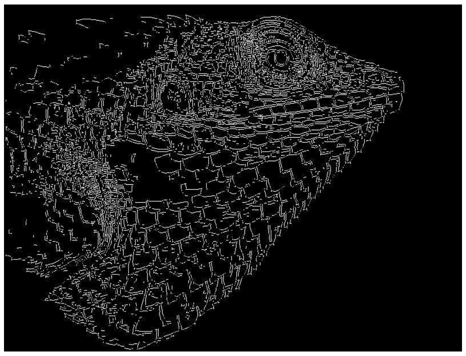
9_Sobel_Gpu_Output.pgm

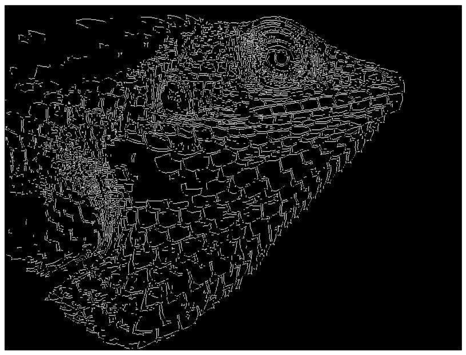

4_NonMaxSupression_Cpu_Output.pgm


10_NonMaxSupression_Gpu_Output.pgm


5_DoubleThreshold_Cpu_Output.pgm


11_DoubleThreshold_Gpu_Output.pgm

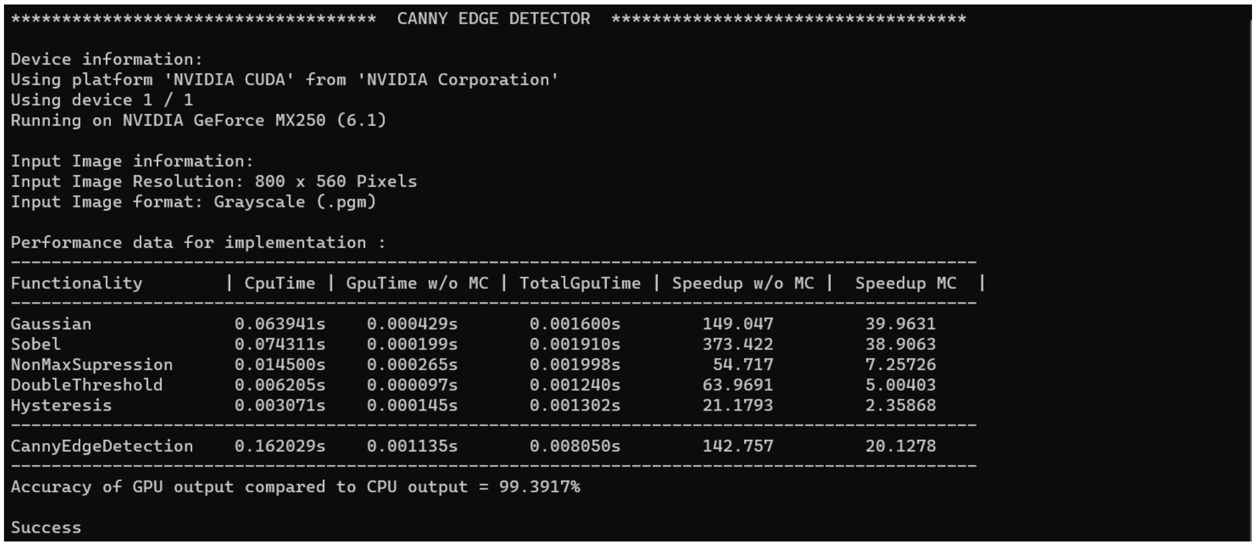

6_CannyEdgeDetection_Cpu_Outpu.pgm


12_CannyEdgeDetection_Gpu_Output.pgm

N. B. : The output images from the GPU functions were slightly varying with GPU's.

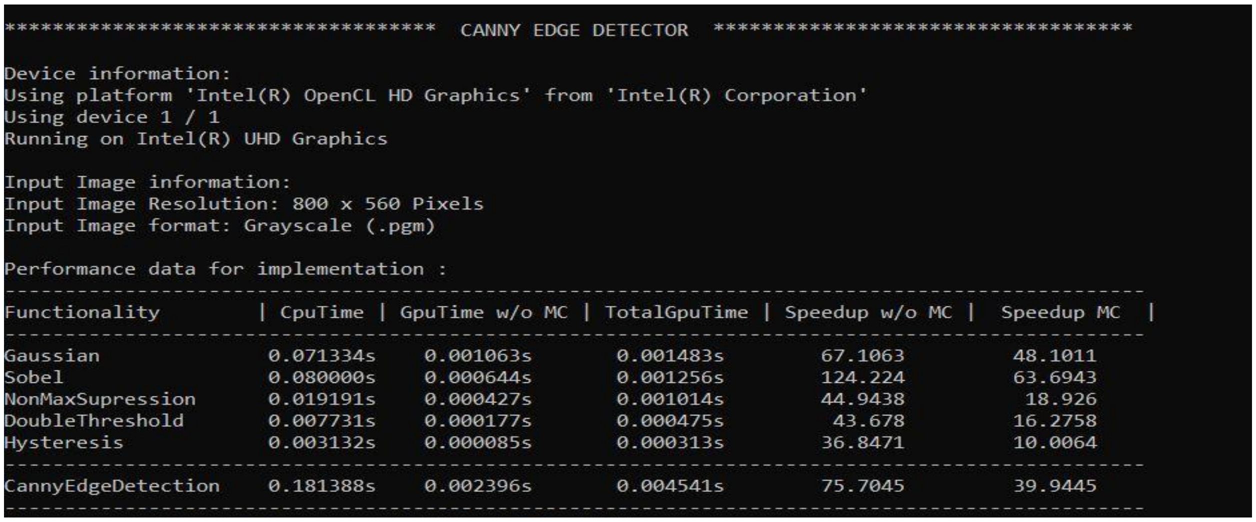
Performance parameters:

The program was tested on two laptops with different CPU and GPU configurations. The performance on the GPU was significantly faster when compared to that of the CPU. The execution time for different functions and the speedups are as shown below figures 1 and 2.



```
************************************  CANNY EDGE DETECTOR  ***********************************

Device information:
Using platform 'NVIDIA CUDA' from 'NVIDIA Corporation'
Using device 1 / 1
Running on NVIDIA GeForce MX250 (6.1)

Input Image information:
Input Image Resolution: 800 x 560 Pixels
Input Image format: Grayscale (.pgm)

Performance data for implementation :
-----------------------------------------------------------------------------------------------
Functionality         | CpuTime  | GpuTime w/o MC | TotalGpuTime | Speedup w/o MC |  Speedup MC  |
-----------------------------------------------------------------------------------------------
Gaussian              0.063941s    0.000429s       0.001600s        149.047         39.9631
Sobel                 0.074311s    0.000199s       0.001910s        373.422         38.9063
NonMaxSupression      0.014500s    0.000265s       0.001998s         54.717         7.25726
DoubleThreshold       0.006205s    0.000097s       0.001240s        63.9691         5.00403
Hysteresis            0.003071s    0.000145s       0.001302s        21.1793         2.35868
-----------------------------------------------------------------------------------------------
CannyEdgeDetection    0.162029s    0.001135s       0.008050s        142.757         20.1278
-----------------------------------------------------------------------------------------------
Accuracy of GPU output compared to CPU output = 99.3917%

Success
```

Figure 1. Performance from, CPU: Intel CORE i7 10th GEN and GPU: NVIDIA GeForce MX250 (6.1)



```
************************************  CANNY EDGE DETECTOR  ***********************************

Device information:
Using platform 'Intel(R) OpenCL HD Graphics' from 'Intel(R) Corporation'
Using device 1 / 1
Running on Intel(R) UHD Graphics

Input Image information:
Input Image Resolution: 800 x 560 Pixels
Input Image format: Grayscale (.pgm)

Performance data for implementation :
-----------------------------------------------------------------------------------------------
Functionality         | CpuTime  | GpuTime w/o MC | TotalGpuTime | Speedup w/o MC |  Speedup MC  |
-----------------------------------------------------------------------------------------------
Gaussian              0.071334s    0.001063s       0.001483s        67.1063         48.1011
Sobel                 0.080000s    0.000644s       0.001256s        124.224         63.6943
NonMaxSupression      0.019191s    0.000427s       0.001014s        44.9438          18.926
DoubleThreshold       0.007731s    0.000177s       0.000475s         43.678         16.2758
Hysteresis            0.003132s    0.000085s       0.000313s        36.8471         10.0064
-----------------------------------------------------------------------------------------------
CannyEdgeDetection    0.181388s    0.002396s       0.004541s        75.7045         39.9445
-----------------------------------------------------------------------------------------------
```

Figure 2. Performance from, CPU: Intel CORE i5 10th GEN and GPU: Intel(R) UHD Graphics

References

[1]Wikipedia Contributors, “Canny edge detector,” Wikipedia, Apr. 14, 2019. https://en.wikipedia.org/wiki/Canny_edge_detector

[2]C. S. Lek, “Parallel Implementation of Canny Edge Detection,” GitHub, Sep. 07, 2023. https://github.com/csl-8bit/canny-edge-parallel/tree/main (accessed Sep. 28, 2023).

[3]The OpenCL Specification Version: 1.1 Document Revision: 44 Khronos OpenCL Working Group

[4]Gasim Mammodov, Sobel filter exercise, High Performance Programming with Graphic Cards Lab Course

[5]Wikimedia.org,2023.https://upload.wikimedia.org/wikipedia/commons/thumb/e/e0/Large_ Scaled_Forest_Lizard.jpg/300px-Large_Scaled_Forest_Lizard.jpg (accessed Sep. 28, 2023).

[6] https://www.geeksforgeeks.org/histogram-equalisation-in-c-image-processing/