Sean Fauteaux
12/7/2021
CSCD 445

Final Project Report – Canny Edge Detection

My original goal with this project was to implement the Canny Edge Detection algorithm using CUDA and parallel computing. The algorithm has 5 main parts:

1. Applying a Gaussian filter to the image
2. Finding the intensity gradients in the x and y directions
3. Applying a threshold to the magnitude of the gradients
4. Double threshold to find edges.
5. Edge tracking by hysteresis

I was unable to get past section 3, applying the gradient threshold. I found many different implementations of this online, but I did not fully understand how they worked or how to implement it on my own. My program creates a Gaussian filter and applies it to the image, then finds the gradients + gradient magnitudes of the image. Although my original goal wasn't satisfied, I still learned a lot about image processing during this project.

The program is run by typing "./cannyEdge filename.pgm". The only argument passed to the program is the input file name. All of the output files will have their respective header appended to the front of the filename (i.e. the CUDA solution of the Gaussian blur has "cudablur_" in front of the original filename.)

**Before running the program, whichever .PGM file you want to process should be in the same directory as the source code.

Sean Fauteaux
12/7/2021
CSCD 445

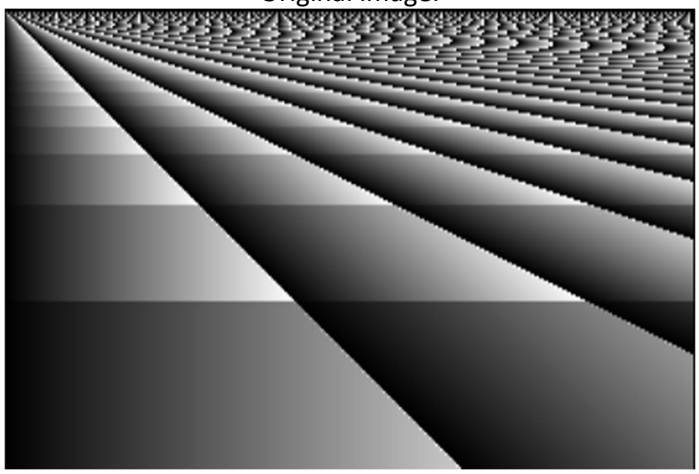
Sample runs -

Tracks:
Dimensions 300 x 200 (60,000 elements)
CPU runtime in ms = 11.167000
CUDA runtime in ms = 0.154240

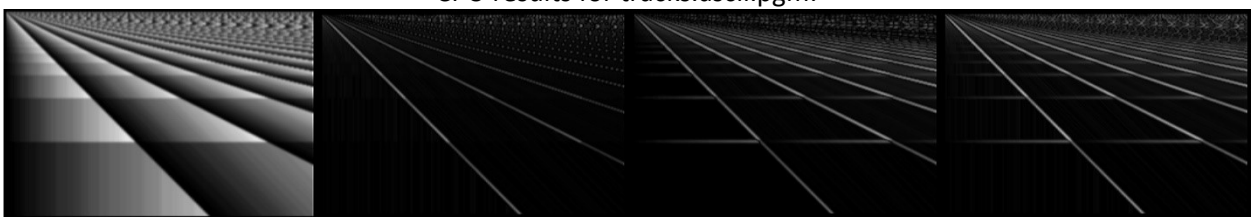
```
[ewu445st2@ip-172-31-26-254 final]$ ./cannyEdge tracks.ascii.pgm

CPU runtime in ms = 11.167000
CUDA runtime in ms = 0.154240
[ewu445st2@ip-172-31-26-254 final]$
```

Original image:

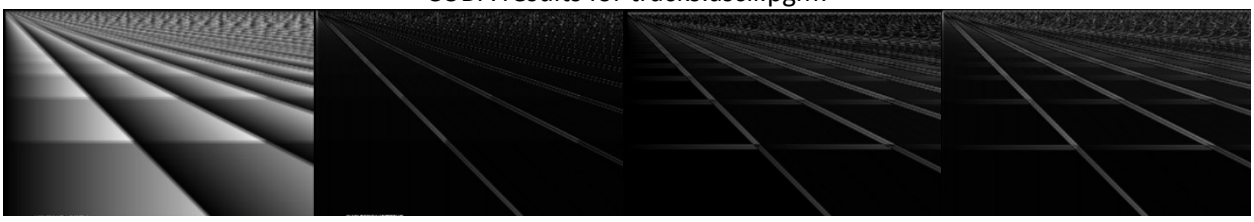


CPU results for tracks.ascii.pgm:



Gaussian Filter -> Gradient (x) -> Gradient (y) -> Magnitude of Gradients

CUDA results for tracks.ascii.pgm:



Sean Fauteaux
12/7/2021
CSCD 445

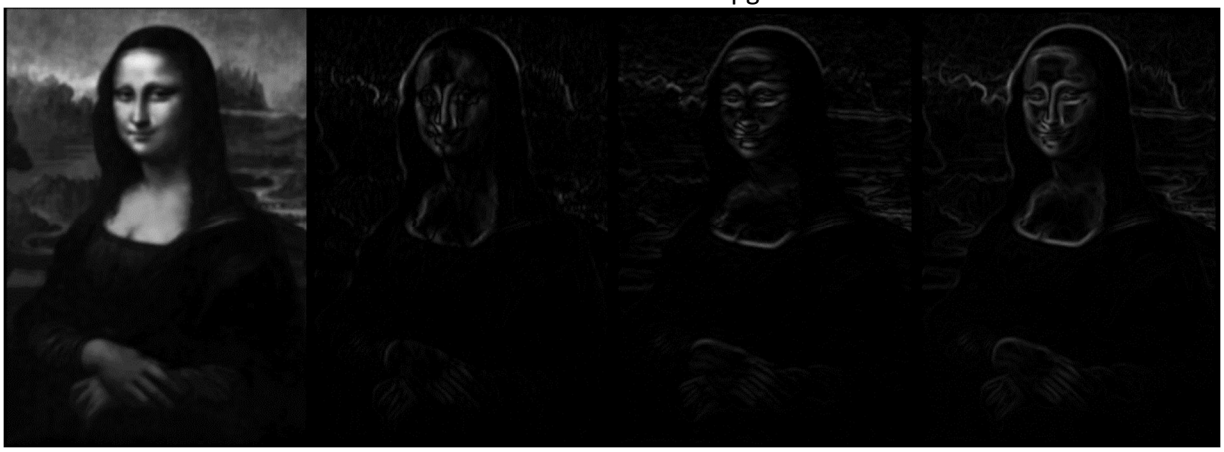
Mona Lisa:
Dimensions 250 x 360 (90,000 elements)
CPU runtime in ms = 17.015999
CUDA runtime in ms = 0.163712

```
[ewu445st2@ip-172-31-26-254 final]$ ./cannyEdge monalisa.pgm

CPU runtime in ms = 17.015999
CUDA runtime in ms = 0.163712
```

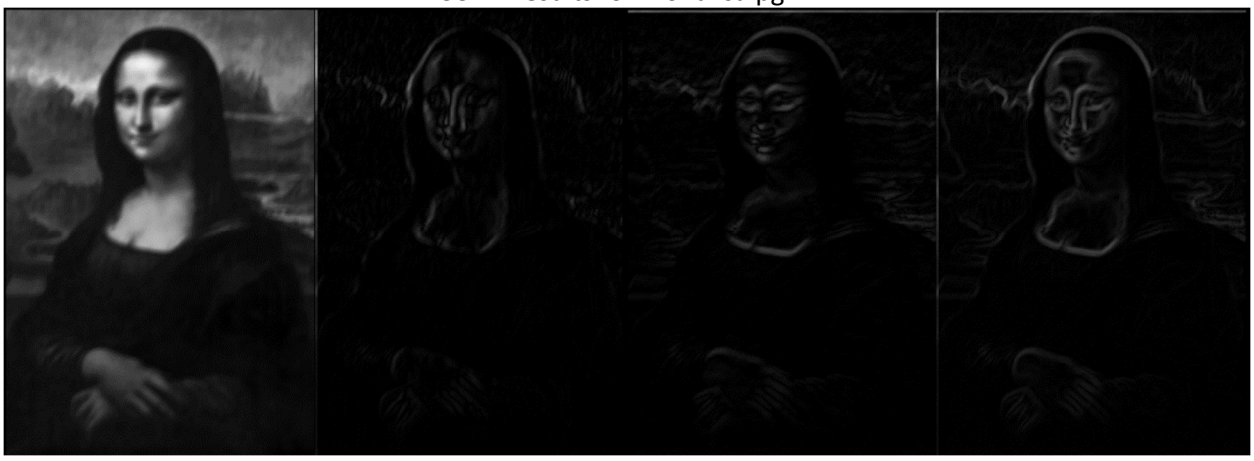
Original Image:

CPU results for monalisa.pgm:



Gaussian Filter -> Gradient (x) -> Gradient (y) -> Magnitude of Gradients

CUDA results for monalisa.pgm:



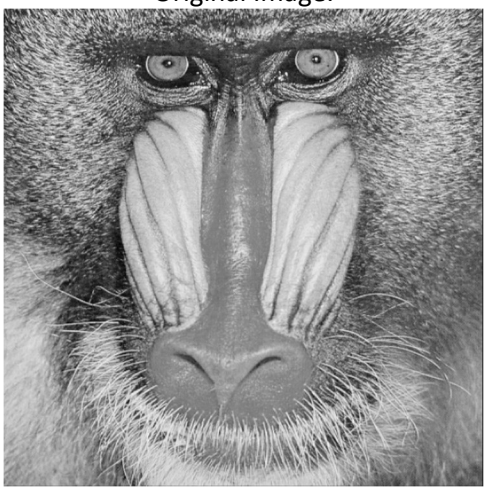
Sean Fauteaux
12/7/2021
CSCD 445

Monkey:
Dimensions 512 x 512 (262,144 elements)
CPU runtime in ms = 59.170002
CUDA runtime in ms = 0.467968

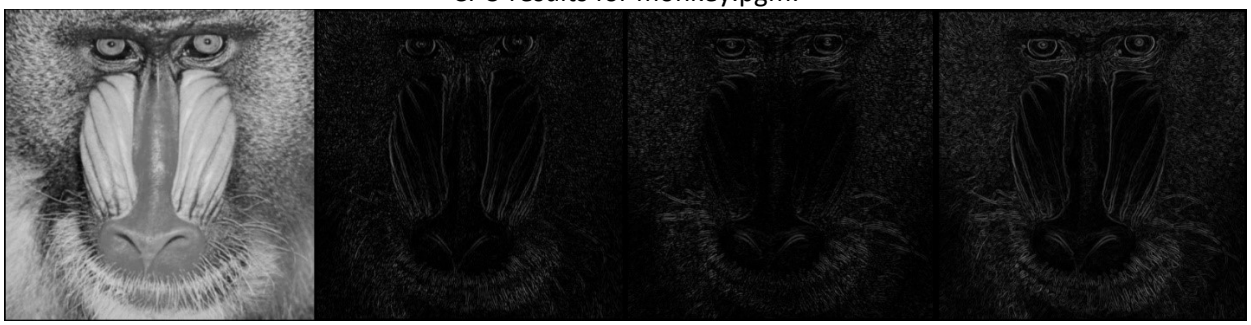
```
[ewu445st2@ip-172-31-26-254 final]$ ./cannyEdge monkey.pgm

CPU runtime in ms = 59.170002
CUDA runtime in ms = 0.467968
[ewu445st2@ip-172-31-26-254 final]$
```

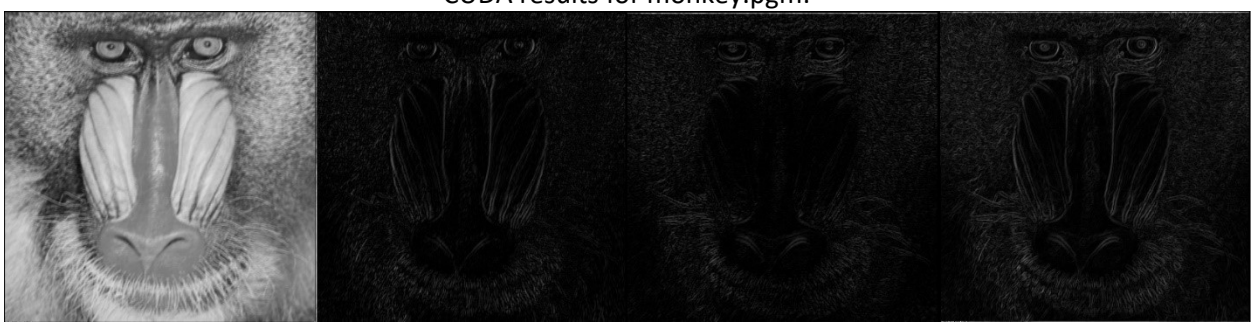
Original Image:



CPU results for monkey.pgm:



CUDA results for monkey.pgm:



Sean Fauteaux
12/7/2021
CSCD 445

Balloons:
Dimensions 640 x 480 (307,200 elements)
CPU runtime in ms = 69.638000
CUDA runtime in ms = 0.545216

```
[ewu445st2@ip-172-31-26-254 final]$ ./cannyEdge balloons.pgm

CPU runtime in ms = 69.638000
CUDA runtime in ms = 0.545216
[ewu445st2@ip-172-31-26-254 final]$
```

CPU results for balloons.pgm:

Sean Fauteaux
12/7/2021
CSCD 445

CUDA results for balloons.pgm:

a) GPU speedup compared to CPU

Input Size	**60,000**	**90,000**	**262,144**	**307,200**
CPU Time (ms)	11.167	17.015999	59.170002	69.638000
GPU Time (ms)	0.154240	0.163712	0.467968	0.545216
Speedup	72.4x	103.94x	126.44x	127.72x

As the dataset increased in size, the speedup also increased. Because of the amount of operations done at each pixel and the amount of times that the image has to be iterated through, this problem is perfect for parallel computing.