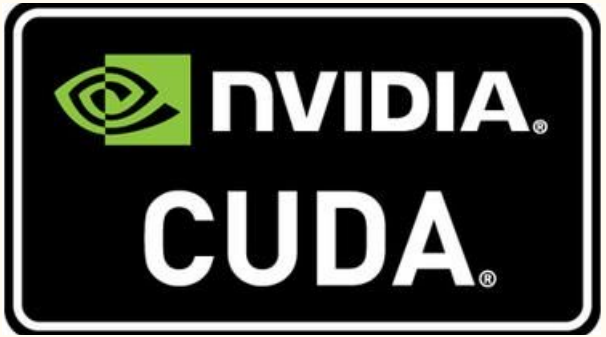
ECE 285 FA19 FINAL PROJECT

CUDA ENABLED CANNY EDGE DETECTION

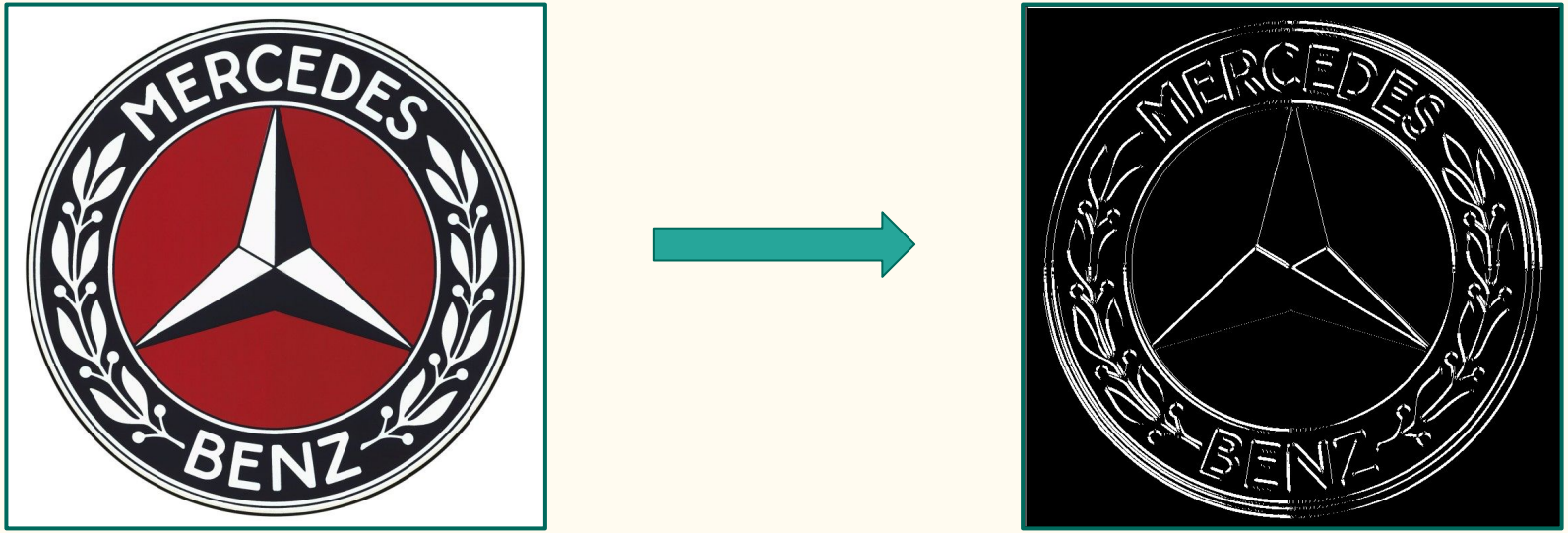
Group G Members:
Rohit Gupta
Sumiran Shubhi



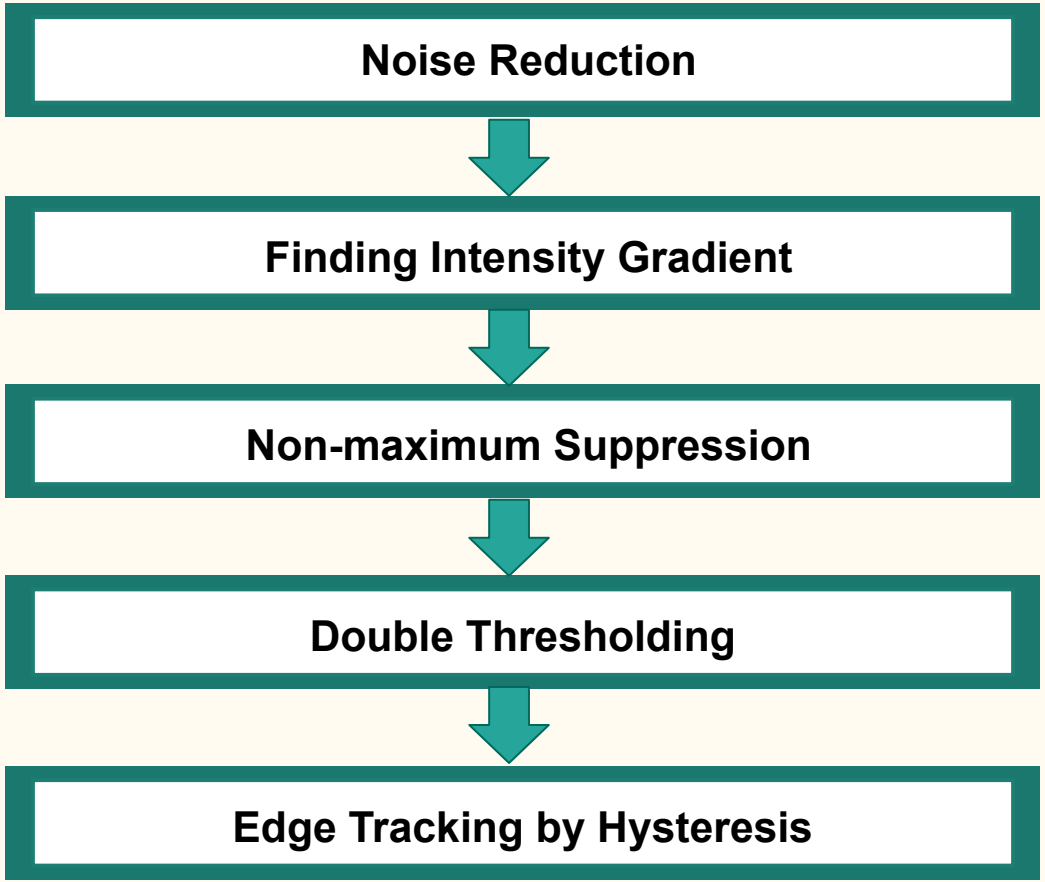
What is Canny Edge Detection?

An algorithm to sketch the edges of the objects in the image.

- Developed by John F. Canny in 1986.
- Works well with noisy images.
- Reduces the amount of data to be processed in computer vision operations.



Canny Edge Detection Algorithm



Step 1: Noise Reduction

- Gaussian Filter
- Kernel Size : 3x3

$$H_{ij} = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(i-(k+1))^2 + (j-(k+1))^2}{2\sigma^2}\right); 1 \leq i, j \leq (2k+1)$$

Step 2: Gradient Calculation

- Sobel Filter - x & y direction
- Intensity of gradient
- Direction of gradient

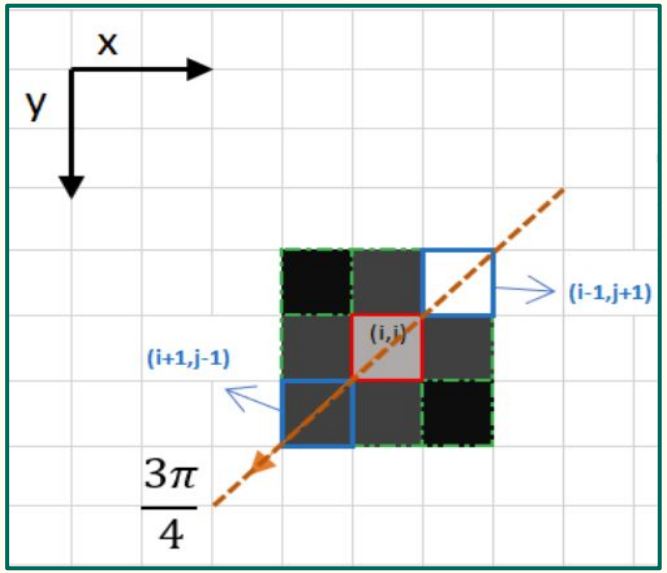
$$|G| = \sqrt{I_x^2 + I_y^2},$$
$$\theta(x, y) = \arctan\left(\frac{I_y}{I_x}\right)$$

x-direction

y-direction

Step 3: Non-Maximum Suppression

- Edges need to be sharp.
- Uses intensity and direction of gradient.



Step 4: Double Threshold

- High Threshold, Low Threshold values.
- Detects :
  - ➔ Strong edges
  - ➔ Weak edges
  - ➔ Irrelevant pixels.

Step 5: Edge Tracking by Hysteresis

- Convert weak edge to strong if connected
- Discard remaining weak edges
- Shows the final detected edges

Implementation on CPU & GPU

CPU

apply_gaussian_kernel()
compute_pixel_thresholds()
apply_sobel_filter_x()
apply_sobel_filter_y()
calculate_sobel_magnitude()
calculate_sobel_direction()
apply_non_max_suppression()
apply_double_thresholds()
apply_hysteresis_edge_tracking()

GPU

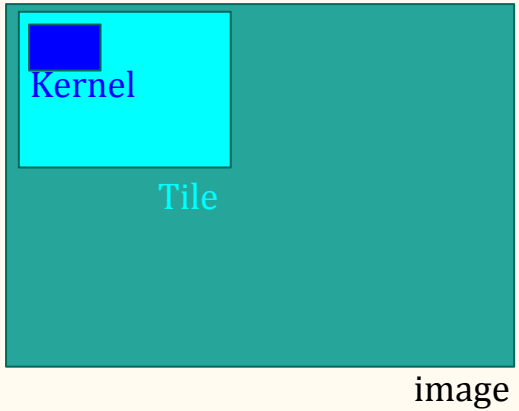
apply_gaussian_kernel()
compute_pixel_thresholds()
streamed_apply_sobel_filter_x_y()

streamed_calculate_sobel_magnitude_direction()

apply_non_max_suppression()
apply_double_thresholds()
apply_hysteresis_edge_tracking()

CUDA Optimizations

- **Shared Memory & Constant Memory (Convolution)**




```
__global__ void kernel(float*Y, float* X, float*W) {
    float* __shared__ W_smem[R*S];
    float* __shared__ X_smem[];
     float acc=0.0;
     thx = threadIdx.x; thy = threadIdx.y;
     y=blockIdx.z*blockDim.y+thy;
     x=blockIdx.y*blockDim.x+thx;
     k = blockIdx.x;

     for (int c=0; c < C; c++){
     if ((thx < S) && (thy < R))
          W_smem[thy, thx] = W[c,thy,thx];
     __syncthreads();
     X_smem[thy,thx] = X[c,y,x];
     __syncthreads();

     for (int i=0; i<R; i++)
     for (int j=0; j<S; j++)
       acc += X_smem[c,y+I, x+j]*W_smem[I,j];
     __syncthreads();
     }
     Y[k,y,x] = acc;
}
```

Reduce duplicate access with shared memory

Slides “Case study: Convolution and CuDNN”

CUDA Optimizations

- **Streamed Execution**

Applying Sobel Filters in X direction

Applying Sobel Filters in Y direction

26% reduction 

Gradient Magnitude

Gradient Direction

10% reduction 

*** for Red Mazda(2528 x 1368) image**

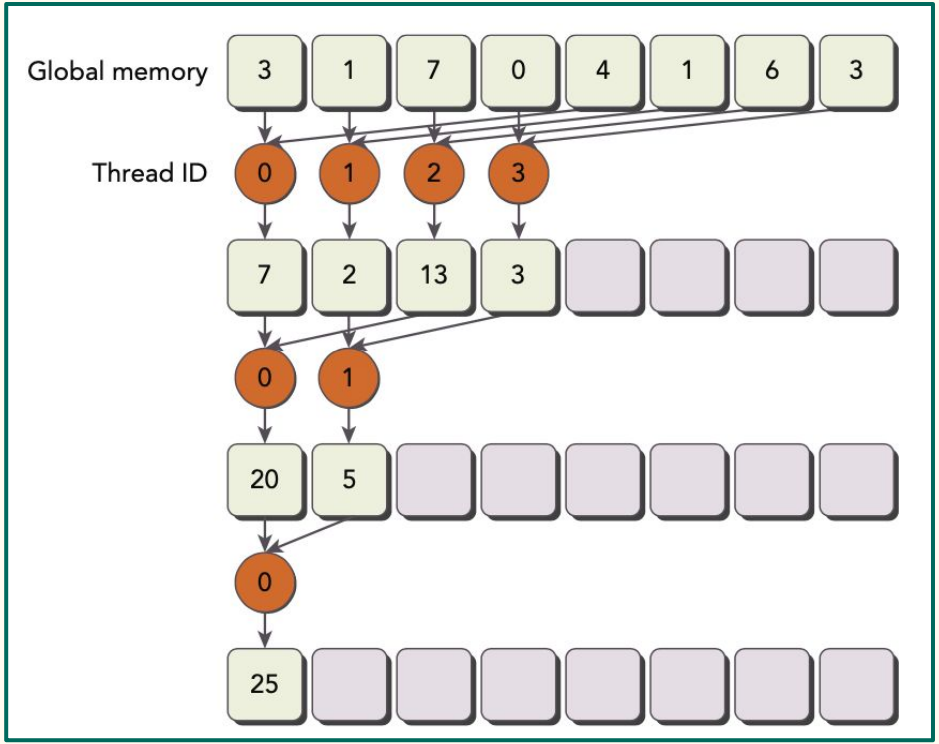
CUDA Optimizations

- **Reduction (sum of elements)**

Interleaved pair implementation

=> Less divergence in a warp!

Block view



John Cheng, Max Grossman, Ty McKercher, "Professional CUDA C Programming", Wiley, 2014.

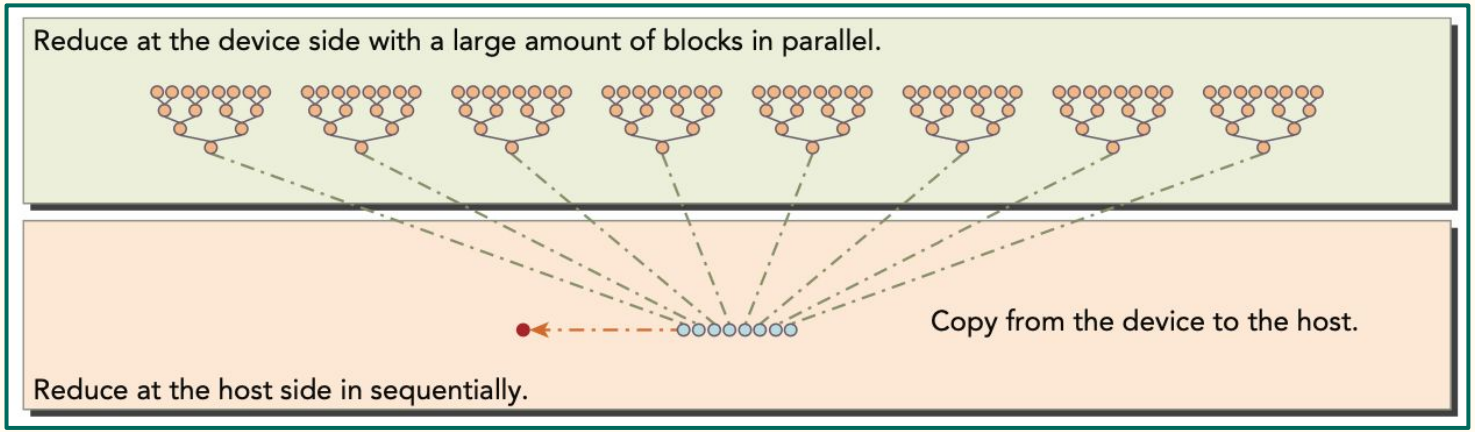
CUDA Optimizations

- **Reduction (sum of elements)**

Interleaved pair implementation

=> Less divergence in a warp!

Grid view



John Cheng, Max Grossman, Ty McKercher, "Professional CUDA C Programming", Wiley, 2014.

CUDA Optimizations

- **Shared Memory & Constant Memory**
- **Streamed Execution**
- **Reduction**

- **Restricting warps/threadblock to 8**
- **Reduced Memory Access**
- **Issue memory accesses earlier**

Results

Image Dimensions (2528 x 1368)

Results

Image Dimensions (2528 x 1368)

- **Gaussian Filtering**

CPU	GPU
99.57 ms	0.82928 ms

Results

Image Dimensions (2528 x 1368)

- **Computing thresholds**

CPU	GPU
99.57 ms	0.82928 ms
3.99 ms	0.83062 ms

Results

Image Dimensions (2528 x 1368)

- **Applying Sobel Filters in X and Y direction**

X direction

CPU	GPU
99.57 ms	0.82928 ms
3.99 ms	0.83062 ms
196.63 ms	2.38454 ms
191.88 ms	

Results

Image Dimensions (2528 x 1368)

- **Applying Sobel Filters in X and Y direction**

CPU	GPU
99.57 ms	0.82928 ms
3.99 ms	0.83062 ms
196.63 ms	2.38454 ms
191.88 ms	

Y direction

Results

Image Dimensions (2528 x 1368)

- **Gradient Magnitude and Direction Computation**

CPU	GPU
99.57 ms	0.82928 ms
3.99 ms	0.83062 ms
196.63 ms	2.38454 ms
191.88 ms	
15.83 ms	1.01987 ms
33.42 ms	

Results

Image Dimensions (2528 x 1368)

- **Non-Maximal Suppression (NMS)**

CPU	GPU
99.57 ms	0.82928 ms
3.99 ms	0.83062 ms
196.63 ms	2.38454 ms
191.88 ms	
15.83 ms	1.01987 ms
33.42 ms	
21.54 ms	0.33978 ms

Results

Image Dimensions (2528 x 1368)

- **Double Thresholds**

CPU	GPU
99.57 ms	0.82928 ms
3.99 ms	0.83062 ms
196.63 ms	2.38454 ms
191.88 ms	
15.83 ms	1.01987 ms
33.42 ms	
21.54 ms	0.33978 ms
10.11 ms	0.24208 ms

Results

Image Dimensions (2528 x 1368)

- **Edge Tracking**

CPU	GPU
99.57 ms	0.82928 ms
3.99 ms	0.83062 ms
196.63 ms	2.38454 ms
191.88 ms	
15.83 ms	1.01987 ms
33.42 ms	
21.54 ms	0.33978 ms
10.11 ms	0.24208 ms
10.61 ms	0.51139 ms

Results

Image Dimensions (2528 x 1368)

- **Overall**

CPU	GPU
583.58 ms	6.15756 ms

95x

Results

Platform	Daimler (800 x 777)	Range Rover (1920 x 1080)	Horse (9192 x 6012)
CPU	107.40 ms	359.16 ms	9,626.51 ms
GPU	1.39 ms	4.01 ms	104.50 ms
	77x ↑	90x ↑	93x ↑

Scalable!

Conclusion

Canny Edge Detection

- **CPU**
- **CUDA-GPU**

93x

Efficient and Scalable!

References

- http://pngtransparent.com/image/69658-red-mazda-png.html
- https://mercedesblog.com/90-years-of-mercedes-benz-logo/
- https://wallpaperscraft.com/download/range_rover_2014_startech_white_side_view_102206/1920x1080
- https://www.wallpaperup.com/620722/Horses_Run_Animals_horse.html
- https://favpng.com/png_view/white-bmw-m2-coupe-front-view-car-car-bmw-mercedes-benz-clip-art-png/3fJcQkny
- https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123


Thank you!