Sobel Filter Using Cuda

### A fundamental task in the field of Computer Vision is Edge Detection. A sobel filter, a widely used image processing technique, is used for just this purpose. It helps in identifying edges or boundaries of objects within an image by highlighting the areas with rapid changes in intensity.

# CUDA Acceleration

CUDA is a parallel computing platform and API developed by NVIDIA that lets us leverage the power of NVIDIA GPUs for high-performance computing tasks. We can use CUDA to significantly speed up the Sobel filter operation on large images.

# Implementation

import numpy as np

from numba import cuda

@cuda.jit

def sobel\_filter(input\_image, output\_image):

x, y = cuda.grid(2)

if x < input\_image.shape[0] - 2 and y < input\_image.shape[1] - 2:

gx = (input\_image[x, y] - input\_image[x + 2, y] +

2 \* input\_image[x, y + 1] - 2 \* input\_image[x + 2, y + 1] +

input\_image[x, y + 2] - input\_image[x + 2, y + 2])

gy = (input\_image[x, y] - input\_image[x, y + 2] +

2 \* input\_image[x + 1, y] - 2 \* input\_image[x + 1, y + 2] +

input\_image[x + 2, y] - input\_image[x + 2, y + 2])

output\_image[x + 1, y + 1] = np.sqrt(gx\*\*2 + gy\*\*2)

# Load an example image (replace with your image)

image = np.array([[1, 2, 1],

[0, 0, 0],

[-1, -2, -1]])

# Create CUDA device array for the image

d\_image = cuda.to\_device(image)

# Allocate memory for the output image

output\_image = np.zeros\_like(image)

# Define block and grid dimensions

threadsperblock = (16, 16)

blockspergrid\_x = (image.shape[0] + threadsperblock[0] - 1) // threadsperblock[0]

blockspergrid\_y = (image.shape[1] + threadsperblock[1] - 1) // threadsperblock[1]

blockspergrid = (blockspergrid\_x, blockspergrid\_y)

# Apply Sobel filter using CUDA

sobel\_filter[blockspergrid, threadsperblock](d\_image, output\_image)

# Copy the result back to the CPU

output\_image = d\_image.copy\_to\_host()

print("Sobel Filter Result:")

print(output\_image)