Title:- Blur Detection with CUDA

GPU SPECIALIZATION CAPSTONE PROJECT

By Anya H.

Dependencies

- CUDA Toolkit 11.x or newer version
- OpenCV 4.1.2+ Windows Build
- Microsoft Visual Studio 2022
(C x64 Native Tools command prompt)

SHORT-WALKTHROUGH

OVERVIEW

PROBLEM STMT :-
- CPU is too slow for large image batches
- Blur Detection → Edge loss → Laplacian Method
- Goal :- GPU-Parallel version

WHAT'S BLUR DETECTION ?

Blurriness means loss of high frequency components.
A standard method to measure this is Laplacian Variance.

↓

Laplacian filter → measure edges
Low Variance = Blurred Image
High Variance = Sharp Image

WHAT DOES MY CODE DO ??

Here's, how I've used this method :

1) Load an image using OpenCV
2) Convert it to a grayscale (as it's easier to work with)
3) Apply the Laplacian Operator on the image using a CUDA kernel (for gpu acceleration)
4) Compute the variance of the Laplacian result.
5) Based on the variance, label image as blurry or not.
6) Save and display result in a CSV.

BUT WHAT'S THIS ??

↓

It's a method to detect edges in a image.

↓

HOW DOES IT WORK ??

Laplacian is a mathematical operator.
It applies a formula (called a convolution) on each pixel of the image to check how much the brightness changes around it.

↓

IF brightness changes a lot → pixel part of an edge
if not → it's probably a flat/smooth part

↓

THE MATH ??

Boring :-

DRAW :-
- 3x3 Laplacian Kernel
- Blur Vs. Sharp Img Vis
- Formula :- Laplacian = $d^2I/dx^2 + d^2I/dy$
  Variance = mean((Laplacian − μ)^2)

↓

In This Project, how it' applied ??
- Load an image
- Apply the Laplacian filter to detect edges
- Output the result

↓

What's the vision then ?? (My Learnings + Challenges)

On CPU :- each pixel is processed one by one = SLOW

On GPU :- we run thousands of threads to process many pixels in parallel

↓

This is a huge boost !
- Can do batch-processing i.e 100s/1000s of images at once
- Real time blur detection in few seconds !

↓

In many case, I used 25 images
so, runtime was less than 3 seconds !