Project: Image Blurring with CUDA

Name: Mahaboob Ali Ashraf Mohammad, **Student ID**: 862393694

Project Approval (**Option-2**) Link https://piazza.com/class/ln84yxhb7eo3qk/post/193_f1

Project Overview

The project focuses on accelerating image blurring using CUDA, a parallel computing platform and application programming interface model created by NVIDIA. The goal is to leverage the power of GPUs to enhance the speed of image processing tasks, specifically applying two types of blurs: Box Blur and Gaussian Blur.

How GPU Acceleration is Achieved:

Parallel Algorithm Design/Implementation:

- The blurring algorithms are designed to process pixels in parallel, taking advantage of the massively parallel architecture of GPUs.
- Each pixel's color value is computed independently, making it suitable for parallelization.

Problem Space Partitioning:

- The problem space is partitioned into threads and thread blocks.
- Each thread is responsible for processing a single pixel or a small group of pixels, and thread blocks are organized to efficiently cover the entire image.

Parallelization Stage:

- The blurring operation is parallelized, focusing on the nested loops that iterate over pixels and their neighbors.
- The computation of each pixel's value is independent, allowing for parallel execution.

Libraries Used:

- The CUDA toolkit is used for GPU programming.
- OpenCV is employed for image input/output and essential image processing operations.

Implementation Details:

Box Blur Implementation:

- Applies a simple averaging operation to neighboring pixels.
- Utilizes a square convolution kernel with equal weights.

Gaussian Blur Implementation:

- Applies a weighted average using a Gaussian distribution.
- Weights decrease as pixels get farther from the center, resulting in a smoother blur.

Running the Code:

- Ensure that the CUDA toolkit and OpenCV library are installed.
- Update the path to the input image in the code.
- Compile the code using CMake and make.
- Run the executable to apply both box blur and Gaussian blur.

For TA/Instructors,
run 1. make clean 2. make 3. ./sgemm-tiled

```
bender /home/csgrad/mmohammad/cuda-gpu-project $ make clean
rm -rf *.o sgemm-tiled
bender /home/csgrad/mmohammad/cuda-gpu-project $ make
nvcc -c -o main.o main.cu -O3 --std=c++03 -I/usr/include/opencv
nvcc main.o -o sgemm-tiled -lcudart -lopencv_calib3d -lopencv_contrib -lopenc
v_core -lopencv_features2d -lopencv_flann -lopencv_highgui -lopencv_imgproc -
lopencv_legacy -lopencv_ml -lopencv_objdetect -lopencv_photo -lopencv_stitchi
ng -lopencv_superres -lopencv_ts -lopencv_video -lopencv_videostab
bender /home/csgrad/mmohammad/cuda-gpu-project $ ./sgemm-tiled
bender /home/csgrad/mmohammad/cuda-gpu-project $ ls
```

Expected Results:

- Two blurred images (box-blurred and Gaussian-blurred) will be generated and displayed.
- The output images will be saved as "box_blurred_output.jpg" and "gaussian_blurred_output.jpg."

Evaluation/Results:

- Timing comparisons between CUDA-accelerated code and sequential CPU code.
- Profiling to identify bottlenecks and assess GPU utilization.
- Screenshots of the original and blurred images for visual verification.

Status of the Project:

Feature Complete:

- The project is feature complete, successfully implementing box blur and Gaussian blur using CUDA.

Limitations:

- Limited to square images.
- Image size should be a multiple of the thread block size for optimal performance.

Challenges Encountered:

- Ensuring proper memory management in CUDA.
- Addressing boundary conditions for pixel processing.
- The project provides an efficient GPU-accelerated solution for image blurring, showcasing the benefits of parallel computing in image processing tasks.

Credits:

- Input image is a random image from the web. (https://pixabay.com/vectors/chessboard-chess-board-game-29630/)
- Used Chat GPT when I got stuck on code.