Introduction to Cuda Parallel Programming Term Project

A CUDA acclerated Harris Corner Detector

r08944022 蔡仲閔

vtsai01@cmlab.csie.ntu.edu.tw

()

Introduction

This project is to implement a GPU-accelerated module for Harris Corner Detector (https://en.wikipedia.org/wiki/Harris_Corner_Detector) using CUDA. Harris Corner Detector is a detection operator used in computer vision to detect the feature from image. In my experiment, a image of size *3648x5472* takes almost *20 seconds* to perform the detector without any parallel. In my oberservation there are lots of way to optimize. There is also some research about it "Optimizing Harris Corner Detection on GPGPUs Using CUDA Article · March 2015" (https://www.researchgate.net/publication/304220484_Optimizing_Harris_Corner_Detection_on_GPGPUs_Using_CUDA).

figure 1: Response of Harris Corner Detector

Algorithm of Harris Corner Detector

Commonly, Harris corner detector algorithm can be divided into five steps.

- Color to grayscale
- Spatial derivative calculation
- Structure tensor setup
- Harris response calculation
- Non-maximum suppression

In the following paragraphs, we will focus on step 2 to 4 and leave the step 5.

Spatial Derivative Calculation

Compute the x and y derivatives of image,

$$I_x = G_\sigma^x * I$$
$$I_y = G_\sigma^y * I$$

Structure Tensor Setup

Construct the tensor M,

$$M = \sum_{(x,y)\in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \sum_{(x,y)\in W} I_x^2 & \sum_{(x,y)\in W} I_x I_y \\ \sum_{(x,y)\in W} I_x I_y & \sum_{(x,y)\in W} I_y^2 \end{bmatrix}$$

Harris Response Calculation

according to the classification of image points using eigenvalues of M shown in figure 1.
the commonly used Harris response calculation is shown as below,
$R = \lambda_1 \lambda_2 - k \cdot (\lambda_1 + \lambda_2)^2 = \det(M) - k \cdot \mathrm{tr}(M)^2$
where k is an empirically determined constant;

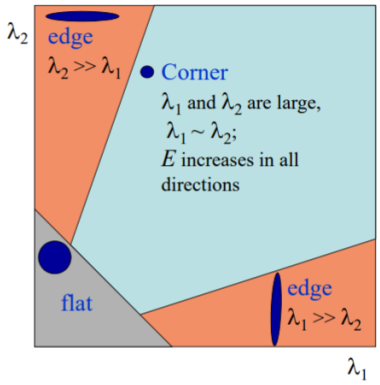

figure 1: classification of image points using eigenvalues of ***M***

Implementation

We implemented the project using C/C++ with ***i7-7700*** and ***RTX 2080Ti***. Here is the directory tree structure for the project.

```
├── input
│   └── image.ppm           // input images(should be ppm format)
├── Makefile
├── output
│   └── ...                 // output images
├── README.md
└── source
    ├── CornerDetector.cu   // main part of the project
    ├── GaussFilter.cuh     // CUDA header file of gaussian filter
    ├── Gauss.h             // header file of gaussian filter
    ├── Matrix.h            // header file of class Matrix
    ├── PPM.h               // header file to deal with portable pixmap format
    ├── SobelFilter.cuh     // CUDA header file of sobel filter
    ├── Sobel.h             // header file of sobel filter
    ├── utils.h             // other utilities
    └── VectorOperation.cuh // CUDA header file of vector operation
```

We also implemented the shard memory and dynamic shared memory version of sobel filter.

Usage

There are python file and ipython notebook for you to choose.

Prepare Images and Meta Data

Put your images into the `./input` folder. Please note that the image must be `.ppm` format. With linux you can convert the image format using

```
convert image.jpg image.ppm
```

Start

here is an example to run the code.

Compilation

you can just use `make` command, or

```
nvcc CornerDetector.cu -o CornerDetector
```

Execution

There are four parameter to assign.

```
./CornerDetector <imagePath> <gaussMask=size> <tpb=threadsPerBlock> <sigma=doub
```

example

```
./CornerDetector ./input/IMG_0125.ppm
```

Result

To compare the perfomance between CPU and GPU, we perform our Harris Corner Detector on a image of size *3648x5472*, and the comparison results shown in *Table 1*. We could see that the GPU version is much faster the CPU version.

	CPU	GPU-Shared-Memory	GPU-Dynamic-Shared-Memory
Running Time	17.378s	0.246s	0.247s
Difference	-	0	0

The output image shown in *figure 1* and *figure 2*. The red spot are Harris Corner Response.

figure 2: Response of Harris Corner Detector

Reference

- Digital Visual Effects, Yung-Yu Chuang
  (https://www.csie.ntu.edu.tw/~cyy/courses/vfx/20spring/assignments/proj1/)

- Introduction to Cuda Parallel Programming, Ting-Wai Chiu (https://nol2.aca.ntu.edu.tw/nol/coursesearch/print_table.php?course_id=222%20D3160&class=&dpt_code=2220&ser_no=35335&semester=106-2&lang=CH)
- Corner-Detector, jariasf (https://github.com/jariasf/Corner-Detector)

tags: NTU Homework cuda