Multi-core Programming
Final Project Report

Implementation and Optimization of Image Template Matching in CUDA

Submitted in partial fulfillment of the degree

Bachelor of Science
in
Software Engineering

Submitted by
Ali Gholami

Under the guidance of
Prof. Mahmoud Momtazopur

Department of Computer Engineering and Information Technology
Amirkabir University of Technology

Spring Semester 2018

Abstract

In most computer vision and image analysis problems, it is necessary to define a similarity measure between two or more different objects or images. Template matching is a classic and fundamental method used to score similarities between objects using certain mathematical algorithms[1]. In this project, we'll propose two implementations of image template matching in CUDA. Our first method is based on the *naive* version of the template matching and the second method is based on the *Fast Fourier Transform* algorithm.

Contents

List of Figures

Chapter 1

Introduction

In this section we'll take a brief look at the recent researches on *Template Matching* task, its variants and the reliability of the models proposed in this task.

1.1 Background and Recent Research

1.1.1 Feature Based Approach

A featured-based approach is appropriate when both reference and template images contain more correspondence with respect to features and control points. In this case, features include points, curves, or a surface model to perform template matching. In this category, the final goal is to locate the pair-wise connections between the target or so-called reference and the template image using spatial relations or features. In this approach, spatial relations, invariant descriptors, pyramids, wavelets and relaxation methods play an important role in extracting matching measures[1].

1.1.2 Area Based Approach

Area-based methods, which are usually known as correlation methods or template matching, were developed for the first time by Fonseca et al. [6] and are based on a combined algorithm of feature detection and feature matching. This method functions very well when the templates have no strong features with an image, since they operate directly on the pixel values. Matches are measured using the intensity values of both image and template. The matching scores are extracted by calculating squared differences in fixed intensities, correction-based methods, optimization methods and mutual information[2].

1.1.3 Naive Template Matching

Nave template matching is one of the basic methods of extracting a given which is identical to the template from the image target. In this approach, with or without scaling (usually without scaling), the target image is scanned by the template, and the similarity measures are calculated. Finally, the positions with the strongest similarities are identified as potential pattern positions[1].

1.1.4 Image Correlation Matching

In this classic template matching method, the similarity metric between the target and the template is measured. Unlike the naive template matching algorithm, the target and the template might have different image intensities or noise levels. However, those images must be aligned. The similarity metric used in this approach is based on the correlation between the target and the template[3].

1.1.5 Sequential Similarity Detection Algorithms

Sequential similarity detection algorithms (SSDAs) are a more efficient alternative to correlation-based methods, including matched filters for translational registration. The measure of match is indirectly calculated based on an error for corresponding pixels in f and g in the images under comparison at any stage of the registration process[4].

1.2 Motivation

There are dozen optimized implementation of *Template Matching* techniques, but the main motivation for me to do so, was the ability to model and code a real-life problem from scratch. I've also improved my CUDA and C++ programming skills with this project. Also, there are multiple math concepts covered in the *Fast Fourier Transform* section which understanding them can bring an enhanced point of view in image and signal processing tasks.

Chapter 2

Problem Definition

The emerging need of image processing techniques in everyday life is inevitable. There have been lots of image processing algorithms proposed to increase the overall availability of tools in image understanding and using these tools to improve the human life. *Template Matching* is a task that its applications are obvious in everyday life. Computer vision tasks such as *object detection*, *object recognition* and other tasks based on these two main questions; Is there a certain object in the image? And if yes, where is it in the image?, can be classified as subproblems of *template matching* task. This task can be extended to counting the number of occurrences of an object in a given image. Given a main image and a template image, we want to find occurrences of the template image in main image. This problem can be solved in many ways. In the next chapter, we'll analyze the procedure of *naive* template matching to solve this problem. In further chapters, we'll provide the procedure of using *Fast Fourier Transform* to convert the images into frequency domain. We'll show that finding the maximum value of the result of the convolution of two images is where the template has occurred in main image.

2.1 Naive Template Matching

The main task of naive template matching was clearly explained in the previous section. There are various similarity measures in naive template matching. The one we use here is *SAD* or *Sum of Absolute Deviations* which is formally described further. Let f and t be the main image and the template image respectively and (x, y) represent the column and the row of each image. The *Sum of Absolute Deviations* error metric for two images can be written as:

$$SAD = \sum_{x,y} f(x,y) - t(x-u, y-v) \tag{2.1}$$

Note that there is also a more precise error metric called *Sum of Squared Errors* which is represented as:

$$SSD = \sum_{x,y} [f(x,y) - t(x-u, y-v)]^2 \tag{2.2}$$

In this project, we'll use the first error metric because of the processing power limits.

2.2 FFT-Based Template Matching

2.2.1 One-Dimensional Discrete Fourier Transform

The Fourier transform of a discrete function of one variable, $f(x)$, $x = 0, 1, 2, ..., M-1$, is given by the equation

$$F(u) = \frac{1}{M} \sum_{x=0}^{M-1} f(x) e^{-j2\pi ux/M} \qquad u = 0, 1, 2, ..., M-1 \tag{2.3}$$

Similarly, given $F(u)$, we can obtain the original function back using the inverse DFT:

$$f(x) = \sum_{u=0}^{M-1} F(u) e^{j2\pi ux/M} \qquad x = 0, 1, 2, ..., M-1 \tag{2.4}$$

2.2.2 Two-Dimensional Discrete Fourier Transform

Extension of the one-dimensional discrete Fourier transform and its inverse to two dimensions is straightforward. The discrete Fourier transform of a function (image) $f(x,y)$ of size $M * N$ is given by the equation

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi(ux/M + ry/N)} \tag{2.5}$$

As in the 1-D case, this expression must be computed for values of $u = 0, 1, 2, ..., M-1$, and also for $v = 1, 2, ..., N-1$. Similarly, given $F(u,v)$, we obtain $f(x,y)$ via the inverse Fourier transform, given by the expression

$$f(x,y) = \sum_{u=0}^{M-1} \sum_{r=0}^{N-1} F(u,v) e^{j2\pi(ux/M + ry/N)} \tag{2.6}$$

2.2.3 Convolution and Correlation Theorems

The discrete convolution of two functions $f(x, y)$ and $h(x, y)$ of size $M * N$ is denoted by $f(x, y) \star h(x, y)$ and is defined by the expression

$$f(x,y) \star h(x,y) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n)h(x-m, y-m) \tag{2.7}$$

we also know that the convolution theorem consists of the following relationships between the two functions and their Fourier transforms:

$$f(x,y) \star h(x,y) \leftrightarrow F(u,v)H(u,v) \tag{2.8}$$

and

$$f(x,y)h(x,y) \leftrightarrow F(u,v) \star H(u,v) \tag{2.9}$$

The correlation of two functions $f(x, y)$ and $h(x, y)$ is defined as

$$f(x,y) \circ h(x,y) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f^*(m,n)h(x+m, y+n) \tag{2.10}$$

where f^* denotes the complex conjugate of f. Convolution is the tie between filtering in the spatial and frequency domains. The principal use of correlation is for matching. In matching, $f(x, y)$ is an image containing objects or regions. If we want to determine whether f contains a particular object or region in which we are interested, we let $h(x, y)$ be that object or region (we call this image a template). Then, if there is a match, the correlation of two functions will be maximum at the location where h finds a correspondence in f. An example of this phenomenon is provided in figure 2.1.

2.3 Fast Fourier Transform

One of the main reasons that the DFT has became an essential tool in signal processing was the development of the fast Fourier transform. Computing the 1-D Fourier transform of M points requires on the order of M^2 multiplication/addition operations. The FFT accomplishes the same task on the order of $M \log_2 M$ operations. If, for example $M = 1024$, the brute-force method will require approximately 10^6 operations. While the FFT will require approximately 10^4 operations. This is a computational advantage of 100 to 1. The decrease in computational complexity significantly impacts the time needed for processing the images. The FFT algorithm is based on the so-called *successive doubling method.* Let's define the equation

$$F(u) = \frac{1}{M} \sum_{x=0}^{M-1} f(x) W_M^{ux} \tag{2.11}$$

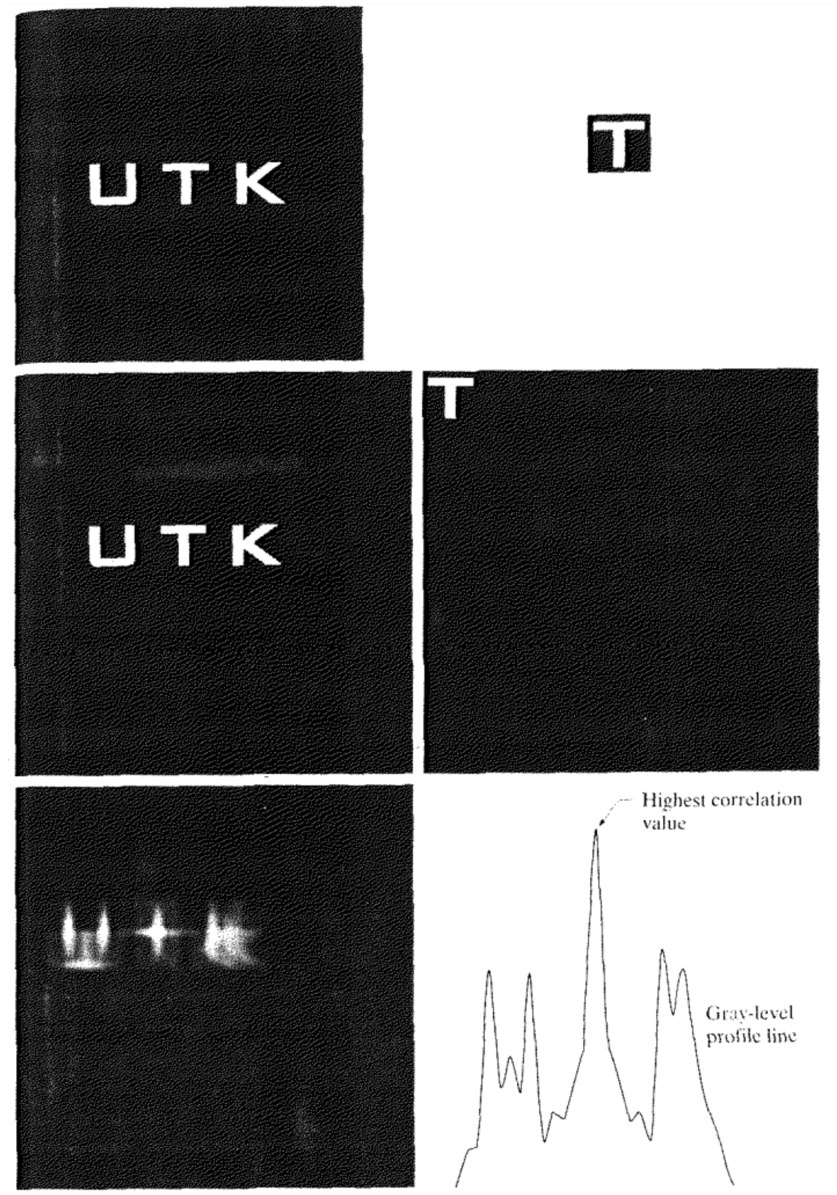

Figure 2.1: Illustration of image padding and correlation. The highest value of the correlation function occurs at the point where template is exactly on top of the T in the image.

where

$$W_M = e^{-j2\pi/M} \tag{2.12}$$

and M is assumed to be of the form

$$M = 2^n \tag{2.13}$$

with n being a positive integer. Hence, M can be expressed as

$$M = 2K \tag{2.14}$$

Substitution of (2.14) into (2.11) yields

$$F(u) = \frac{1}{2}[\frac{1}{K}\sum_{x=0}^{K-1} f(2x)W_{2K}^{u(2x)} + \frac{1}{K}\sum_{x=0}^{K-1} f(2x+1)W_{2k}^{u(2x+1)}] \tag{2.15}$$

The number of multiplications and additions required to implement FFT:

$$m(n) = 2m(n-1) + 2^{n-1} \quad n \geq 1 = \frac{1}{2}M \log_2 M \tag{2.16}$$

and

$$a(n) = 2a(n-1) + 2^{n} \quad n \geq 1 = M \log_2 M \tag{2.17}$$

Chapter 3

Implementation and Performance Analysis

3.1 Analysis of Naive Template Matching(NTM)

In this section, we'll analyze the implementation and the performance of the *Naive Template Matching* algorithm. First of all, let's consider the serial implementation case.

3.1.1 Bitmap Images

For this project, we've used the *bitmap* image library by *Arash Partow* which is accessible here.

3.1.2 Serial Implementation of NTM

The serial implementation of this algorithm consists of 2 main loops to iterate over image pixels. There is also a need for each pixel we are iterating through, to calculate the *Sum of Absolute Deviations* which is the similarity measure between the main image pixels and the template image. Please refer to the *Naive Template Matching* algorithm to see the code. Since the task is mainly focused on counting the number of occurance inside the main images, we can obtain this by finding the minimum value of *SAD* while iterating through the main image pixels and counting the number of occurrences of the found minimum inside the main image matrix.

3.1.3 Timing the Serial NTM

We can obtain the elapsed time for the serial iteration using the *chrono* library. The code for this section is provided here. Please refer to the serial implementation code for the details.

3.1.4 CUDA Implementation of NTM

In this section we'll walk through the kernel codes provided in the *kernel.cu*.

Compute Sad Array Kernel

The image data is being stored in an 1-D array. Thus, the best choice for CUDA kernel launch parameters would be to set grid dimensions as $(image\_width/block\_size\_x, image\_height/block\_size\_y, 1)$. The main purpose of this choice is not the way we have stored the data, it actually depends on the input data. Since we are dealing with images it is more suitable to have 2-D blocks of threads. Further explanations focus on the kernel implementation. Initially, each thread finds its own global 2-D identification as rows and columns. Then, each thread virtually sees a kernel (template) around it self. This is the core of the parallelization section. All threads can see all of the iterations in one place. We then compute the SAD of each pixel and load it inside an SAD array. The next section describes the reason to do so.

Find Minimum in Array Kernel

We can obtain the best possible match by finding the minimum SAD in SAD array. We'll do so using shared memory to speed up the process. Each thread helps loading the data inside a block into the shared memory of that block. We then use *fminf* to find the minimum inside each block. Since we are using the shared memory, we need to do a *reduction* to find the minimum across different blocks. We can obtain this reduction task defining a *mutex* to control the global memory write by first thread of each block.

Find Number of Occurrences

Since the main task is to find the number of occurrences of template image in main image, we should count the number of occurrences of minimum SAD in the SAD array. To speed up the process we've used shared memory again. The intuition of using shared memory is exactly the same as the previous section.

Chapter 4

Future Work

¡Future work here¿

Chapter 5

Conclusion

¡Conclusion here¿

Acknowledgments

¡Acknowledgements here¿

¡Name here¿

¡Month and Year here¿
National Institute of Technology Calicut

References

[1] Hashemi, Nazanin Sadat, et al. "Template Matching Advances and Applications in Image Analysis." arXiv preprint arXiv:1610.07231 (2016).

[2] Mahalakshmi, T., R. Muthaiah, and P. Swaminathan. "An overview of template matching technique in image processing." Research Journal of Applied Sciences, Engineering and Technology 4.24 (2012): 5469-5473.

[3] Perveen, Nazil, Darshan Kumar, and Ishan Bhardwaj. "An overview on template matching methodologies and its applications." IJRCCT 2.10 (2013): 988-995.

[4] Cox, Greg S. "Template matching and measures of match in image processing." University of Cape Town, South Africa (1995).