Optics and Lasers in Engineering 47 (2009) 1286–1292

Contents lists available at ScienceDirect

Optics and Lasers in Engineering

journal homepage: www.elsevier.com/locate/optlaseng

Parallel computing for fringe pattern processing: A multicore CPU approach in MATLAB® environment

Wenjing Gao, Qian Kemao *, Haixia Wang, Feng Lin, Hock Soon Seah

School of Computer Engineering, Nanyang Technological University, Singapore 639798, Singapore

ARTICLE INFO

Article history:
Received 5 March 2009
Received in revised form
25 April 2009
Accepted 27 April 2009
Available online 6 June 2009

Keywords:
Parallel computing
Fringe pattern analysis
MATLAB® parallel computing toolbox
Windowed Fourier transform

ABSTRACT

In the process of measurements such as optical interferometry and fringe projection, an important stage is fringe pattern analysis. Many advanced fringe analysis algorithms have been proposed including regularized phase tracking (RPT), partial differential equation based methods, wavelet transform, Wigner–Ville distribution, and windowed Fourier transform. However, most of those algorithms are computationally expensive. MATLAB® is a general algorithm development environment with powerful image processing and other supporting toolboxes. It is also commonly used in photomechanical data analysis. With rapid development of multicore CPU technique, using multicore computer and MATLAB® is an intuitive and simple way to speed up the algorithms for fringe pattern analysis. The paper introduces two acceleration approaches for fringe pattern processing. The first approach is task parallelism using multicore computer and MATLAB® parallel computing toolbox. Since some algorithms are embarrassing problems, our first approach makes use of this characteristic to parallelize these algorithms. For this approach, parallelized windowed Fourier filtering (WFF) algorithm serves as an example to show how parallel computing toolbox accelerates the algorithm. Second, data parallelism using multicore computer and MATLAB® parallel computing toolbox is proposed. A high level parallel wrapping structure is designed, which can be used for speeding up any local processing algorithms. WFF, windowed Fourier ridges (WFR), and median filter are used as examples to illustrate the speedup. At last, the results show that the parallel versions of former sequential algorithm with simple modifications achieve the speedup up to 6.6 times.

1. Introduction

In the process of measurements such as optical interferometry and fringe projection, an important stage is fringe pattern analysis. Phase-shifting algorithms and Fourier transform are traditional fringe analysis algorithms [1,2]. Recently many advanced fringe processing algorithms emerge including regularized phase tracking (RPT) [3], partial differential equation (PDE)-based methods [4], wavelet transform [5,6], Wigner–Ville distribution [7], and windowed Fourier transform (WFT) [8], to name a few. These algorithms can suppress noise presented in fringe patterns or extract the phase information from intensity fringe patterns or do both. However, most of these algorithms are computationally expensive. For instance, to process a 256×256 gray level fringe pattern, RPT method using 1.5 GHz Intel Pentium-M personal computer (PC) takes 1–2 min [3]; PDE-based methods by Pentium-IV 3.6, 3.59 GHz 1 G RAM CPU will generally take 1–5 min [4]; and WFT-based methods by Pentium-IV 3.2 GHz desktop costs 45 s–3 min [8]. These advanced algorithms should be accelerated in order to gain wider applications.

Parallel computing is a form of computation that uses multiple computer resources simultaneously to accelerate processing the problem [9]. It is now widely used in areas such as medical image processing or weather forecasting, which are computationally time consuming. In fringe pattern processing, parallel computing technique has also been applied utilizing cluster computers to accelerate temporal fringe pattern processing with speedup of 1.6 times [10].

Since IBM released the first dual-core processor in 2001, multicore computer has been the mainstream computing model in computer market nowadays. Recently, Intel even released 80 cores processor called Teraflops Research Chip [11], which can overwhelm a supercomputer in the aspect of floating point operations per second. With the support of corresponding software, multicore CPU computer will challenge the current popular acceleration scheme using graphics processing unit (GPU) or field programmable gate array (FPGA) [12]. MATLAB® [13] is among the first batch of software that supports programming multicore computer. It is a general algorithm development environment with powerful image processing and other supporting toolboxes.

* Corresponding author.
E-mail address: mkmqian@ntu.edu.sg (Q. Kemao).

0143-8166/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.optlaseng.2009.04.018

As a high level programming language, the code of MATLAB® is concise, portable and easy to maintain. Moreover, MATLAB® is also commonly used in photomechanical data analysis [14]. Using parallel computing toolbox [15] of MATLAB® based on multicore CPU is an intuitive and simple way to speed up the fringe pattern analysis. Based on the above reasons, this paper presents two approaches for accelerating fringe pattern processing. The first approach is task parallelism using multicore computer and MATLAB® parallel computing toolbox. Since some algorithms are embarrassing problems, our first approach makes use of this characteristic to parallelize these algorithms. For this approach, parallel windowed Fourier filtering (WFF) algorithm serves as an example to show how parallel computing toolbox accelerates the algorithm. Second, data parallelism using multicore computer and MATLAB® parallel computing toolbox is proposed. A high level parallel wrapping structure is designed, which can be used for speeding up any local processing algorithms. WFF, windowed Fourier ridges (WFR), and median filter [16] are used as examples to illustrate the speedup. The rest of the paper is organized as follows. Basic principles about parallel computing and MATLAB® parallel computing toolbox are introduced in Section 2; Using MATLAB® parallel computing toolbox and multicore CPU technique to accelerate algorithms is presented in Section 3; The results show that the parallel versions of former sequential algorithms with simple modifications achieve the speedup up to 6.6 times, which is given in Section 4. The paper is concluded in Section 5. It is noted that 'multicore computer' in the paper is used to refer to multicore CPU-based personal computer.

2. Parallel computing and MATLAB® parallel computing toolbox

In this section, we introduce the principles of parallel computing in Section 2.1, and MATLAB® parallel computing toolbox in Section 2.2.

2.1. Parallel computing

Parallel computing is a form of computation that solves a problem by simultaneously using multiple computer resources. Hardware supporting parallel computing consists of multicore computer [17], symmetric multiprocessor [18], distributed computer such as cluster workstations [9], and specialized parallel processors such as FPGA [12], GPU [12], and application-specific integrated circuit (AISC) [19]. With the development of the supporting parallel hardware, especially the development of multicore computer, parallel programming architecture becomes more important than before.

Most common ways for parallelism include task parallelism, pipeline parallelism, and data parallelism [20,21]. Task parallelism, also known as functional parallelism, is a parallel development structure that independent calculation parts of a method can be performed in different processors simultaneously. In the case of pipeline parallelism, the problem is divided into a series of tasks. Every task will be executed by a separate process or processor. Each parallel process is usually referred to as a pipeline stage [9]. The output of one pipeline stage serves as the input of another so that at any given time each pipeline stage is working on a different dataset. Data parallelism mainly focuses on the same process being applied to different parts of a dataset simultaneously. That is to say, similar operation sequences or functions are executed in parallel on elements of a large data structure. In addition, given enough parallel resources, the computation time of data parallelism structure is usually independent of the problem size. One of previously mentioned parallelism methods or a combination of them may be used in parallelism applications.

In the aspect of parallel degree, an application can be fine-grained, coarse-grained, and embarrassing parallelism. Fine-grained applications are those whose subtasks are relatively small and communicate frequently. Parallel applications exhibit coarse-grained parallelism if their subtasks communicate infrequently after larger amounts of computation. Embarrassing parallelism applications can be naturally divided into completely independent parts with no or small amount of communication. WFF is such embarrassing problem because processing an image with a specific frequency is unrelated to that with another frequency, which will be further discussed in Section 3.

The only factor that affects parallelizing performance from linear speedup [22] is overhead. The overhead covers the following three aspects: (i) communication time for interactions between processors; (ii) extra computations in the parallelized algorithm but not required in counterpart sequential version; and (iii) some parts of algorithm cannot be parallelized, causing the load imbalance among processors.

2.2. Introduction to MATLAB® parallel computing toolbox

Parallel computing toolbox provides a high-level structure for solving computationally expensive problems over MATLAB® and Simulink®. The toolbox makes use of multicore or bus connected computers to share the computation load by parallel processing. First released in September 2006 as a part of MATLAB® 2006b, parallel computing toolbox has been developed to version 4.1 supporting up to eight workers or worker sessions [23] locally on a multicore desktop. Furthermore, together with MATLAB® distributed computing server, parallel toolbox also supports cluster-based applications that use any scheduler or any number of workers on a computer cluster. The algorithms can be parallelized in MATLAB® without additional coding for specific hardware and network architectures. Consequently, converting former sequential MATLAB® code to parallel applications only needs a few modifications of code and no programming in a low-level communication driver. These features of parallel computing toolbox enable scientists and engineers to focus on their algorithm design while enjoy the benefits of speedup using multicore computer or a cluster of workstations.

Parallel computing toolbox usually supports task-parallel and data-parallel application development. In general, task-parallel in parallel computing toolbox is restricted to embarrassing problems in which 'iterations' of a parallel loop have no dependency or communication among them. By simply replacing 'for' in former MATLAB® code by 'parfor', parallel computing toolbox automatically manages data and code transfer between the MATLAB® client session and the worker sessions. If parallel computing toolbox fails to detect the presence of the worker sessions, it automatically reverts to sequential execution. Data-parallel application development in MATLAB® parallel computing toolbox mainly targets at the algorithms that require large amount of dataset to be processed. Parallel computing toolbox provides distributed arrays, parallel functions and uses 'spmd' keyword for parallel execution on several worker sessions. When the dataset is divided into smaller pieces and sent to different cores, some non-edge pixels in the original dataset become edges pixels of these smaller datasets. Consequently, edge effects will be introduced. As the data-parallel commands provided in MATLAB® parallel computing toolbox do not provide edge consistency function, it will not be used in this paper.

Fig. 1 shows the structure of parallel computing toolbox as a superstructure over other toolboxes and MATLAB® environment.