

STUDY ON SUITABLE HIGH PERFORMANCE COMPUTING
SYSTEM FOR HD IMAGES PROCESSING

CHENG KAH LOON

This Report Is Submitted In Partial Fulfilment Of Requirements For The Bachelor
Degree of Electronic Engineering (Computer Engineering)

Fakulti Kejuruteraan Elektronik dan Kejuruteraan Komputer
Universiti Teknikal Malaysia Melaka

June 2014


UTeM

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

 UNIVERSITI TEKNIKAL MALAYSIA MELAKA
 FAKULTI KEJURUTERAAN ELEKTRONIK DAN KEJURUTERAAN KOMPUTER

**BORANG PENGESAHAN STATUS LAPORAN
 PROJEK SARJANA MUDA II**

 Tajuk Projek : STUDY ON SUITABLE HIGH PERFORMANCE COMPUTING SYSTEM
 FOR HD IMAGES PROCESSING

 Sesi Pengajian :

1	3	/	1	4
---	---	---	---	---

 Saya **CHENG KAH LOON**

(HURUF BESAR)

mengaku membenarkan Laporan Projek Sarjana Muda ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Laporan adalah hakmilik Universiti Teknikal Malaysia Melaka.
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan laporan ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. Sila tandakan (✓) :

SULIT*

*(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)

TERHAD**

**(Mengandungi maklumat terhad yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)

TIDAK TERHAD


(TANDATANGAN PENULIS)

 Disahkan oleh:
DR. LIM KIM CHUAN
 Pensyarah Kanan
 Fakulti Kejuruteraan Elektronik Dan Kejuruteraan Komputer
 Universiti Teknikal Malaysia Melaka (UTeM)
 Hang Tuah Jaya, 76100 Durian Tunggal,
 Melaka.
 (COP DAN TANDATANGAN PENYERAH)

Tarikh:

 Tarikh: 17/7/14

“Saya akui laporan ini adalah hasil kerja saya sendiri kecuali ringkasan dan petikan yang tiap-tiap satunya telah saya jelaskan sumbernya.”

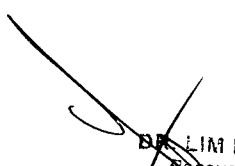
Tandatangan : 

Nama Penulis : CHENG KAH LOON

Tarikh : 2ND JUNE 2014

“Saya/kami akui bahawa saya telah membaca karya ini pada pandangan saya/kami karya ini adalah memadai dari skop dan kualiti untuk tujuan penganugerahan Ijazah Sarjana Muda Kejuruteraan Elektronik (Kejuruteraan Komputer).”

Tandatangan


DR. LIM KIM CHUAN
Pensyarah Kanan,
Fakulti Kejuruteraan Elektronik Dan Kejuruteraan Komputer
Universiti Teknikal Malaysia Melaka (UTeM)
Hang Tuah Jaya, 76100 Durian Tunggal,
Melaka.

Nama Penyelia

.....

Tarikh

..... 1/7/2014

For my beloved, father and mother

ABSTRACT

Under the International Road Assessment Programme (iRAP, sponsorship of UK charity organization FIA Foundation for the Automobile and Society), led by AAM, JKR, JKJR, and Malaysia Institute of Road Safety Research (MIROS), a series of road safety assessment activity over 3600 km of road in Malaysia has been carried out. The current system being used to carry out the road safety assessment was found not adequate by MIROS. A cost effective, reliable and robust road survey data collection and analysis system is required by MIROS for this road rating assessment. As a sub-system of this required system, a computer system which is capable of real-time processing of multiple high definitions (HD) image streams is utterly needed to be designed and developed. Manager-worker pattern is an effective concurrent software programming technique that could utilize the available central processing unit (CPU) cores in a computer system. By implementing this programming pattern, the HD image capture, display and record process from multiple cameras streams can be processed concurrently and thus a result of better performance and lesser latency can be obtained. The system is then further developed with the enhancement of swipe gesture enabled graphical user interface (GUI). The CPU usage after running the GUI-based software for capturing, displaying and recording video from three cameras when using Intel i7-4700MQ with camera setting of 1080p resolution, 4096kbps bit rate , and 15 frames per second is 61%,

ABSTRAK

Di bawah Program Penilaian Jalan Antarabangsa (iRAP, tajaan UK organisasi kebajikan FIA Yayasan Automobil dan Masyarakat), yang diketuai oleh AAM, JKR, JKJR dan Malaysia Institut Penyelidikan Keselamatan Jalan Raya (MIROS), satu siri aktiviti penilaian keselamatan jalan raya yang meliputi jalan raya sepanjang 3600 km di Malaysia telah dijalankan. Sistem semasa yang digunakan untuk menjalankan penilaian keselamatan jalan raya tersebut didapati tidak memenuhi keperluan MIROS. Satu sistem pengumpulan data dan analisis kajian jalan yang berkos efektif, berkebolehpercayaan dan mantap amat diperlukan oleh MIROS untuk menjalankan aktiviti penilaian jalan tersebut. Sebagai sub-sistem daripada system tersebut, satu sistem komputer yang mampu memproseskan beberapa aliran imej yang berdefinisi tinggi (HD) dengan masa nyata perlu direkai dan dibangunkan. Corak Pengurus-Pekerja merupakan satu teknik pengaturcaraan serentak yang berkesan dan ia menggunakan unit pemprosesan pusat (CPU) yang sedia ada dalam suatu sistem komputer. Dengan melaksanakan corak ini, tangkapan, paparan dan proses rekod imej HD daripada pelbagai aliran kamera dapat diproses bersama dan dengan itu hasil video yang berprestasi lebih baik dan kependaman yang lebih kurang boleh diperolehi. Sistem ini seterusnya dibangunkan dengan peningkatan antara muka grafik pengguna (GUI). Penggunaan CPU selepas menjalankan perisian berasaskan GUI untuk menangkap, memapar dan merakam video dari tiga kamera dengan menggunakan Intel I7-4700MQ yang cameranya beresolusi 1080p, kadar bit sebanyak 4096kbps, serta 15 bingkai sesaat adalah 61%.

TABLE OF CONTENT

CHAPTER	TITLE	PAGE
	PROJECT TITLE	i
	REPORT STATUS CONFIRMATION FORM	ii
	DECLARATION	iii
	VERIFICATION OF SUPERVISOR	iv
	DEDICATION	v
	ACKNOWLEDGMENT	vi
	ABSTRACT	vii
	ABSTRAK	viii
	TABLE OF CONTENTS	ix
	LIST OF TABLES	xiii
	LIST OF FIGURES	xiv
I	INTRODUCTION	1
	1.1 BACKGROUND	1
	1.2 PROBLEM STATEMENT	2
	1.3 OBJECTIVES	2
	1.4 SCOPE OF PROJECT	2
	1.5 STRUCTURE OF THESIS	2
II	LITERATURE REVIEW	4
	2.1 PARALLEL COMPUTING	4

2.2	SOFTWARE PATTERN	5
2.3	PARALLEL SOFTWARE DESIGN	5
2.4	PRODUCER-CONSUMER PATTERN	6
2.5	MANAGER-WORKER PATTERN	7
2.6	MULTITHREADING	8
2.7	INTER-PROCESS COMMUNICATION	10
2.8	QUEUE	10
2.9	LOCK	11
2.10	SEMAPHORE	11
2.11	MUTEX	12
2.12	INTERNET PROTOCOL CAMERA	12
2.13	OPENCV	12
2.14	EMGUCV	13
2.15	.NET FRAMEWORK	13
2.16	CAMERA PARAMETER	13
	2.16.1 Image Resolution	14
	2.16.2 Frame Rate	14
	2.16.2 Bit Rate	15
2.17	INTEL CORES	15
	2.17.1 Number Of Cores	15
	2.17.2 Intel Turbo Boost	15
	2.17.3 Cache Size	16
	2.17.4 Hyper-Threading	16
2.18	GPU ACCELERATED IMAGE PROCESSING	16
III	METHODOLOGY	18
3.1	CAMERA PARAMETERS CONFIGURATION	18
3.2	HARDWARE DESIGN	22
3.3	SOFTWARE DEVELOPMENT WITH SERIALISE PROGRAMMING	22
3.4	SOFTWARE DEVELOPMENT WITH PARALLEL	23

	PROGRAMMING	
	3.4.1 Capture Thread	25
	3.4.2 Graphic Canvas Thread	25
	3.4.3 Display Thread	25
	3.4.4 Record Thread	25
	3.5 SOFTWARE DEVELOPMENT WITH GUI	25
IV	RESULT AND DISCUSSION	26
	4.1 CAMERA PARAMETERS	27
	4.1.1 Video Resolution	27
	4.1.1.1 1080p	27
	4.1.1.2 720p	28
	4.1.2 Frame Rate	30
	4.1.2.1 25FPS	30
	4.1.2.2 15FPS	31
	4.1.2.3 5FPS	33
	4.1.3 Bit Rate	34
	4.1.3.1 6144 kbps	34
	4.1.3.2 4096 kbps	36
	4.1.3.3 1024 kbps	37
	4.1.4 Data Tabulation	39
	4.2 SERIALISED PROGRAMMING	41
	4.3 SOFTWARE WITHOUT GUI	44
	4.3.1 Running in i3-2310M	44
	4.3.2 Running in i7-4700MQ	50
	4.4 SOFTWARE WITH GUI	55
	4.4.1 Running in i7-4700MQ	55
	4.5 PERFORMANCE COMPARISON AND DISCUSSION	59
V	CONCLUSION AND FUTURE WORK	61
	5.1 CONCLUSION	61

5.2 FUTURE WORK

62

REFERENCES

63

LIST OF TABLES

NO	TITLE	PAGE
4.1	Results obtained for two different video resolutions.	39
4.2	Results obtained for three different frame rates	39
4.3	Results obtained from three different bit rates	40
4.4	Comparison of CPU utilisation for i3 and i7 (without GUI)	59
4.5	CPU utilisation of GUI-based software tested with i7	59
4.6	Comparison in CPU utilisation of the software with and without GUI	60

LIST OF FIGURES

NO	TITLE	PAGE
2.1	A Producer-Consumer pattern	7
2.2	A Manager-Workers organisation block diagram	8
2.3	Sequence Diagram for Manager-Workers Pattern	9
2.4	The operation of a FIFO queue	11
2.5	The difference between 60 fps and 24 fps	14
3.1	Network configuration page of the IndigoVision IP camera	19
3.2	Camera exposure setting	20
3.3	Configuration of frame rate and bit rate	21
3.4	Configuration of video resolutions	21
3.5	Block diagram of the proposed system	22
3.6	Timing diagram of serialised software to handle display and encode operation of three cameras	23
3.7	Sequential diagram for video image capturing, displaying and recording	24
3.8	Timing diagram of parallelise software to handle display and encode operations of three cameras	24
4.1	CPU usage is around 83% when using Intel i3 CPU to capture, display, and record single camera stream (camera setting: 1080p, 6144kbps, 25FPS)	27
4.2	Calculated FPS (8.97666) output by the developed C++ program for capture, display and record single camera stream with Intel i3 (camera setting: 1080p, 6144kbps, 25FPS)	28
4.3	CPU usage is around 30% when using Intel i3 CPU to capture, display, and record single camera stream (camera setting: 720p,	29

	6144kbps, 25FPS)	
4.4	Calculated FPS (24.98) output by the developed C++ program for capture, display and record single camera stream with Intel i3 (camera setting: 720p, 6144kbps, 25FPS)	29
4.5	CPU usage is around 31% when using Intel i3 CPU to capture, display, and record single camera stream (camera setting: 720p, 6144kbps, 25FPS)	30
4.6	Calculated FPS (25.01) output by the developed C++ program for capture, display and record single camera stream with Intel i3 (camera setting: 720p, 6144kbps, 25FPS)	31
4.7	CPU usage is around 14% when using Intel i3 CPU to capture, display, and record single camera stream (camera setting: 720p, 6144kbps, 15FPS)	32
4.8	Calculated FPS (15.05) output by the developed C++ program for capture, display and record single camera stream with Intel i3 (camera setting: 720p, 6144kbps, 15FPS)	32
4.9	CPU usage is around 7% when using Intel i3 CPU to capture, display, and record single camera stream (camera setting: 720p, 6144kbps, 5FPS)	33
4.10	Calculated FPS (4.998) output by the developed C++ program for capture, display and record single camera stream with Intel i3 (camera setting: 720p, 6144kbps, 5FPS)	34
4.11	CPU usage is around 33% when using Intel i3 CPU to capture, display, and record single camera stream (camera setting: 720p, 6144kbps, 25FPS)	35
4.12	Calculated FPS (25.05) output by the developed C++ program for capture, display and record single camera stream with Intel i3 (camera setting: 720p, 6144kbps, 25FPS)	35
4.13	CPU usage is around 24% when using Intel i3 CPU to capture, display, and record single camera stream (camera setting: 720p, 4096kbps, 25FPS)	36
4.14	Calculated FPS (25.02) output by the developed C++ program for capture, display and record single camera stream with Intel i3	37

	(camera setting: 720p, 4096kbps, 25FPS)	
4.15	CPU usage is around 18% when using Intel i3 CPU to capture, display, and record single camera stream (camera setting: 720p, 1024kbps, 25FPS)	38
4.16	Calculated FPS (25.03) output by the developed C++ program for capture, display and record single camera stream with Intel i3 (camera setting: 720p, 1024kbps, 25FPS)	38
4.17	Graph of results obtained for three different frame rates	40
4.18	Graph of results obtained from three different bit rates	41
4.19	A typical serialised programming for capturing, displaying, and recording three cameras	42
4.20	CPU usage of serialised software for three cameras when using Intel i3 (camera setting of the three cameras: 1080p, 4096kbps, 15FPS)	43
4.21	Quality of the images from three cameras when using Intel i3 (camera setting of the three cameras: 1080p, 4096kbps, 15FPS)	43
4.22	The CPU utilisation of an Intel i3 before running the software	44
4.23	The CPU utilisation after running the software for capturing, displaying and recording video from one camera when using Intel i3-2310M (camera setting of camera #1: 1080p, 4096kbps, 15FPS)	45
4.24	Calculated FPS (15.09) output by the developed parallelised software for capture, display and record single camera stream with Intel i3-2310M (camera setting of camera #1: 1080p, 4096kbps, 15FPS)	46
4.25	The CPU utilisation after running the software for capturing, displaying and recording video from two cameras when using Intel i3-2310M (camera setting of camera #1 & #2: 1080p, 4096kbps, 15FPS)	47
4.26	Calculated FPS (camera #1 14.79FPS; #2 15.02FPS) output by the developed parallelised software for capture, display and record single camera stream with Intel i3-2310M (camera setting of camera #1 & #2: 1080p, 4096kbps, 15FPS)	47
4.27	The CPU utilisation after running the software for capturing,	48

	displaying and recording video from three cameras when using Intel i3-2310M (camera setting of camera #1, #2 & #3: 1080p, 4096kbps, 15FPS)	
4.28	Calculated FPS (camera #1 10.15FPS) output by the developed parallelised software for capture, display and record single camera stream with Intel i3-2310M (camera setting of camera #1 & #2: 1080p, 4096kbps, 15FPS)	49
4.29	Quality of videos from three cameras after the implementation of Manager-Workers pattern	49
4.30	The CPU utilisation of an Intel i7 before running the software	50
4.31	The CPU utilisation after running the software for capturing, displaying and recording video from one camera when using Intel i7-4700MQ (camera setting of camera #1: 1080p, 4096kbps, 15FPS)	51
4.32	Calculated FPS (15.07) output by the developed parallelised software for capture, display and record single camera stream with Intel i3-2310M (camera setting of camera #1: 1080p, 4096kbps, 15FPS)	51
4.33	The CPU utilisation after running the software for capturing, displaying and recording video from two cameras when using Intel i7-4700MQ (camera setting of camera #1 & #2: 1080p, 4096kbps, 15FPS)	52
4.34	Calculated FPS (camera #1 14.79FPS; #2 15.02FPS) output by the developed parallelised software for capture, display and record single camera stream with Intel i3-2310M (camera setting of camera #1 & #2: 1080p, 4096kbps, 15FPS)	53
4.35	The CPU utilisation after running the software for capturing, displaying and recording video from three cameras when using Intel i7-4700MQ (camera setting of camera #1, #2 & #3: 1080p, 4096kbps, 15FPS)	54
4.36	Calculated FPS (camera #1 14.70; #2 14.72FPS; #3 14.72) output by the developed parallelised software for capture, display and record single camera stream with Intel i3-2310M (camera setting of	54

	camera #1 & #2: 1080p, 4096kbps, 15FPS)	
4.37	The CPU utilisation after running the GUI-based software for capturing, displaying and recording video from one camera when using Intel i7-4700MQ (camera setting of camera #1: 1080p, 4096kbps, 15FPS)	55
4.38	The CPU utilisation after running the GUI-based software for capturing, displaying and recording video from one camera when using Intel i7-4700MQ (camera setting of camera #1 & #2: 1080p, 4096kbps, 15FPS)	56
4.39	Left camera display	57
4.40	Middle camera display	57
4.41	Right camera display	58
4.42	The CPU utilisation after running the GUI-based software for capturing, displaying and recording video from three cameras when using Intel i7-4700MQ (camera setting of camera #1, #2 & #3: 1080p, 4096kbps, 15FPS)	58
4.43	Graph of performance comparison	60

CHAPTER 1

INTRODUCTION

1.1 Background

At year 2000, the Road Assessment Programmes (RAPs) was initiated by the Automobile Association in Europe to upgrade the road safety in low and middle income countries. An umbrella organization, the International RAP (iRAP) is setup to promote and overlook the consistency of the globally implemented RAPs. With the strong will and commitment demonstrated by the government of Malaysia for road safety improvement, Malaysia was selected as pilot country for the iRAP in the Asia region. Under the sponsorship of UK charity organization FIA Foundation for the Automobile and Society, led by AAM, JKR, JKJR, and Malaysia Institute of Road Safety Research (MIROS), technology and system were provided by the member of RAPs and up to 3600km of road has been inspected at year 2007. Several countermeasures have been proposed to improve the road safety through the analysis of the recorded video images. This project focuses in the design and development of the high performance video image processing system that could effectively make use of the available computing power to handle multiple incoming video image streams.

1.2 Problem Statement

The allocation of system resource for high definitions images processing is the main concern of this study. The effect of various camera parameters setting with respect to the system resource consumption is yet to be determined.

The suitable system design to effectively make use of the available system resources of the computing device is yet to be determined.

1.3 Objectives

The following objectives will be achieved throughout this project:

1. To analysis the effect of various camera parameters setting with respect to the performance of the CPU for image processing related task.
2. To study the underlying programming technique for optimal system performance.
3. To design a touch screen based Graphical User Interface (GUI) to perform the video displaying and recording.

1.4 Scope of Project

Since the main constraint of the project is to design the system in a cost effective way, this study will not take graphic processing unit (GPU) into consideration but only utilises the available central processing unit (CPU) of a computing device.

1.5 Structure of Thesis

The related components of the high performance video image processing, both the hardware and software, will be firstly reviewed in the chapter 2. The needed components and the design of the system will be explained in detail in chapter 3. The results obtained with the defined methods will then be discussed in chapter 4. The first part of results discusses about the effect of different camera parameters to the

CPU utilisation and the video frame rate. The next part demonstrates the feasibility of the high performance parallel programming by implementing the Manager-Worker pattern without GUI and subsequently followed with the implementation of GUI. Last but not least, the achievement of this project is summarized at the end of this chapter.

CHAPTER 2

LITERATURE REVIEW

Nowadays, the concept of “using more than one computation resource for solving certain time consuming problems” is getting more interesting and valid. However, coordination among multiple computation resources and intelligent work distribution among them is a major challenge for system designer. Naazish, et al. (2013) described parallel computing as the simultaneous execution of the same task on multiple processors in order to obtain faster results. Parallel computing has traditionally been associated with ‘high performance computing’, which uses high-end computer resources to solve ‘grand challenge’ computational problems (Jorge 2010). The expected result is a faster computation compared to execution on a single-processor/core system. With the advent of commodity-market multi-core processors (AMD 2008; Intel 2008) and cluster blade computers or low-cost servers, parallel computing is now available to many application developers. It is always desirable to have the results available as soon as possible, and for many applications, such as real-time imaging system, late results often imply useless results. A high performance parallel video image computing should consist of both software and hardware implementations to process the video images produced by the Internet Protocol Enabled Camera. Suitable design of software pattern is necessary by

CHAPTER 2

LITERATURE REVIEW

Nowadays, the concept of “using more than one computation resource for solving certain time consuming problems” is getting more interesting and valid. However, coordination among multiple computation resources and intelligent work distribution among them is a major challenge for system designer. Naazish, et al. (2013) described parallel computing as the simultaneous execution of the same task on multiple processors in order to obtain faster results. Parallel computing has traditionally been associated with ‘high performance computing’, which uses high-end computer resources to solve ‘grand challenge’ computational problems (Jorge 2010). The expected result is a faster computation compared to execution on a single-processor/core system. With the advent of commodity-market multi-core processors (AMD 2008; Intel 2008) and cluster blade computers or low-cost servers, parallel computing is now available to many application developers. It is always desirable to have the results available as soon as possible, and for many applications, such as real-time imaging system, late results often imply useless results. A high performance parallel video image computing should consist of both software and hardware implementations to process the video images produced by the Internet Protocol Enabled Camera. Suitable design of software pattern is necessary by

utilising the operating system features. The performance of the hardware of the computing device such as CPU cores should also be taken care throughout this study.

2.1 Parallel Software Pattern and Design

A pattern is '*a recurring solution to a standard problem*' (Coplien 1994; Gabriel 1996). Jorge (2010) considered a software pattern as a function-form relation that occurs in a context, where the function is described in problem domain terms as a group of unresolved trade-offs or forces, and the form is a structure described in solution domain terms that achieves a good and acceptable equilibrium among those forces.

In general, the concept of software patterns is not confined to a particular software domain. As software patterns express recurring designs, they can be used to document design decisions at any level in any software domain. This generality is particularly important for parallel software design: software patterns are useful in documenting the design decisions in any aspects of a complete parallel system: for example: to document hardware systems or subsystems, communication and synchronization mechanisms, partitioning and mapping policies and so on.

Parallel software design is critical to effective parallel programming, since it is significantly harder than programming sequential programs on single-processor computers. Parallel software design begins when a need for high performance is identified, and software designer starts creating a parallel software system. Often the hardware and software resources are given. For example, a parallel program might have to be designed using fine other important elements, such as operating system or middleware. The problem of parallelization is normally described in terms of a data set and an algorithm that performs operations in it. This algorithm can be a sequential algorithm or a parallelized algorithm. The main performance goal is usually optimizing execution time (Pancake & Bergmark 1990; Pancake 1996).

Parallel programming relies on the coordination of computing resources so that they work simultaneously and efficiently towards a common objective. Achieving such objective solely depend upon significant effort from software designers due to the complexity involved. As parallel programming is intended to

improve performance, software designers also need to consider cost-effective techniques for performance measurement and analysis. Most programming problems have several possible parallel solutions, so parallel software design cannot easily be reduced to recipes. At best, the designer has access to several parallel organization structures, and needs to decide which to use as a basis, selection is performed commonly based only on the information available at this stage and the intuition of the software designers.

2.1.1 Producer-Consumer Pattern

Producer-Consumer pattern is a classic concurrent programming design pattern, where the processes are classified as either producers or consumers. The producer is the process which generates some data and put it into a shared data structure while the consumer is responsible for the data removing from that structure. This pattern describes the multi-process synchronisation problem in which the producer is not going to increment with any data when the shared data structure is full while the consumer will not attempt to remove any data from an empty structure. The proposed solution is that either the producer goes to sleep or discard the data if the shared data structure, such as a queue, is full; until the consumer consumes an item from the queue and notifies the producer to continue incrementing the queue. Likewise, the solution could be the consumer goes to sleep if it finds the queue to be empty till the producer generates and puts the data into the queue and wakes the consumer up. Figure 2.1 illustrates the Producer-Consumer pattern.