Concurrent Viola Jones Classifiers on a Portable Beowulf Cluster

A thesis presented in partial
fulfilment of the requirements

for the degree

of Master of Engineering

in Mechatronics at
Massey University

Ravi Kiran Chemudugunta
2008

Abstract

Real-time Computer Vision is an interesting application for supercomputing, real-time applications (vision processing in particular) employ special purpose hardware such as DSPs to achieve high performance. This thesis explores parallel computers particularly commodity general purpose hardware. We also build a prototype to better understand the economics of supercomputing, specifically related to mobile computing - low power, rugged design by building a mobile computer. A new communication layer is built, where by the nature of the locality of the nodes allows one to optimise the protocols to reduce the latency comparably. Finally a study and in depth results of the algorithm, the Viola Jones Object detector in parallel are presented followed by reflection and future work based on the current results and platform.

Acknowledgements

First and foremost I would like to thank my supervisor Dr. Andre Barczak. Without his advise and motivation I wouldn't have been able to succeed. I thank him for listening to my sometimes strange ideas and steering me in the right direction throughout this work. I would also like to acknowledge my co-supervisor Dr. Johan Potgieter for input and advise on the project.

I extend my thanks to Dr. Napoleon Reyes for being encouraging and Dr. Chris Messom for his valuable insights and input. I would like to acknowledge everyone from the Computer Sciences Department, Guy Kloss, Rushad Irani, Anton Gerdelan and Daniel Playne for their encouragement and friendship through the years.

I would like to thank Thomas Felton for counselling me through my life, his support has been integral for me to be able to finish this work. Finally, I would like to thank my family, who supported me in everyway during this work, without them I would not be here today.

Contents

Acknowledgements iii

1 Introduction 1

2 Literature Review 3

2.1 Object Detection . . . 3

2.1.1 Training . . . 5

2.1.2 A Cascade of Detectors . . . 7

2.2 Parallel Computing on Clusters . . . 9

2.3 Motivation and Base Work . . . 9

2.4 Hardware Infrastructure . . . 10

2.5 Software Infrastructure . . . 10

2.5.1 Interconnect Software Protocols . . . 11

2.6 Conclusions of the Literature Review . . . 12

2.6.1 Implementation Details . . . 13

2.6.2 On UDP . . . 13

3 Method / Material 14

3.1 Hardware . . . 14

3.2 Physical Construction . . . 14

3.3 Parallel Knoppix . . . 16

3.4 Testing Tools . . . 17

3.4.1 Qemu . . . 17

3.4.2 VDE Ethernet . . . 17

3.5 Distribution . . . 17

3.5.1 TTY Linux build system . . . 18

3.5.2 Methods to Reduce Packages Sizes . . . 18

3.6 Moose Linux . . . 19

3.6.1 Build System Overview 19
3.6.2 Distribution 19
3.6.3 Packaging 20
3.6.4 Testing 20
3.7 Measuring Time 20

4 Implementation **21**
4.1 Synchronisation by Broadcast Messages 21
4.1.1 Naive Approach 22
4.1.2 Knowledge Rows 22
4.1.3 Results 22
4.2 Broadcasting Protocol 23
4.3 LSPIP: Loosely Synchronised Parallel Image Processing Library 24
4.4 Parallelising Detectors 27
4.5 Single Classifiers 27
4.6 Multiple Classifiers 28
4.6.1 Distribution of Classifiers among Nodes 29
4.7 Dealing with Results 29
4.8 Measuring System Efficiency 30
4.8.1 Testing Method, Image Sequences 32
4.8.2 Testing Method, USB Camera 32
4.9 Summary 32

5 Results and Discussion **33**
5.1 Network Performance 33
5.2 CPU Performance 33
5.2.1 USB Camera 35
5.3 Experimental Conditions 36
5.4 Average System Performance 36
5.5 Individual Node Performance 37
5.6 Scalability 39
5.6.1 Resolution Scaling 41
5.7 Performance Analysis 41

6 Conclusion and Future Work **45**
6.1 Conclusions from Results 45
6.2 Perspectives 45
6.3 Future Work 46
6.3.1 Parallelising Single Classifiers 46
6.3.2 Strategies of Applying Multiple Classifiers 46
6.3.3 Moments Based Detection Algorithms 46
6.3.4 Other Hardware Platforms 47
DSP Hardware 47
GPGPU Hardware 47

A UDP Image Broadcast Header File **48**

B UDP Image Result Collection Header File **50**

C LSPIP: 'Source' Program **52**

D LSPIP: 'Process' Program **55**

E LSPIP: 'Sink' Program **58**

F Mooselinux Build System: 'Get' **61**

G Mooselinux Build System: 'Update' **64**

H Mooselinux Build System: 'Build' **66**

I Timing Routines for Benchmarking **68**
I.1 Header File 68
I.2 Implementation 69

Bibliography **70**

List of Figures

2.1 Samples of Hand Detection, source [Barczak and Chemudugunta, 2006] . . . 4
2.2 Generator Function . . . 6
2.3 Flow Diagram . . . 6
2.4 Lena . . . 7
2.5 Composition of a Strong Haar Classifier . . . 7
2.6 Parallel Classifiers, source [Barczak and Chemudugunta, 2006] . . . 8
2.7 Haar Features, source [Intel, 2007] . . . 8
2.8 Summed Area Table, adapted from [Viola and Jones, 2002] . . . 8
2.9 Cascade of Boosted Classifiers, adapted from [Lienhart and Maydt, 2002] . . . 8
2.10 The OSI Model, source [CiscoSystems, 2006] . . . 9

3.1 Hardware platform, source [Barczak and Chemudugunta, 2006] . . . 15
3.2 Real-time Constraints, source [Barczak and Chemudugunta, 2006] . . . 15
3.3 Prototype . . . 16
3.4 Moose Build System . . . 19

4.1 Broadcast Synchronisation . . . 21
4.2 Enabling Broadcasting using BSD Sockets API . . . 25
4.3 Packet Structure . . . 25
4.4 Network Layout, source [Barczak and Chemudugunta, 2006] . . . 27
4.5 Subwindow List . . . 28
4.6 Parallel Classifiers, source [Barczak and Chemudugunta, 2006] . . . 30
4.7 Communication Pattern 1, source [Barczak and Chemudugunta, 2006] . . . 30
4.8 Communication Pattern 2, source [Barczak and Chemudugunta, 2006] . . . 31

5.1 Broadcast Performance of MPI vs UDP Protocol . . . 34
5.2 Classifier Performance on a Single Node . . . 34
5.3 CPU vs Send FPS (640x480) . . . 35
5.4 Test Sequence . . . 36

5.5 Average System Performance 37
5.6 Individual Node Performance for 160x120 38
5.7 Individual Node Performance for 320x240 38
5.8 Individual Node Performance for 640x480 39
5.9 Performance Scalability for 160x120 40
5.10 Performance Scalability for 320x240 40
5.11 Performance Scalability for 640x480 41
5.12 Resolution Scaling for 1 Node 42
5.13 Resolution Scaling for 2 Nodes 42
5.14 Resolution Scaling for 3 Nodes 43
5.15 Resolution Scaling for 4 Nodes 43
5.16 Integral Image Computation Times 44

List of Tables

3.1 Cluster Specifications 14

5.1 Cluster Specifications relevant to performance 33
5.2 II Computation Time and Sub Window Load for each of the Resolutions . . . 41

Chapter 1

Introduction

The motivation for this work is to realise a concept of real-time video processing on a parallel computer. Image Processing promises intelligent systems that are able to communicate with humans better. However it is a very compute intensive process and requires specialised hardware to make implementation viable. Image Processing application exhibit inherent parallelism and so can be parallelised using multiple processors each doing some sub-task in a larger task thereby speeding up the operation. Using the analogy of a hole being dug by many workers - the more workers the faster it is completed, however one cannot have an infinite number of workers digging the same hole as in reality there are space restrictions around how infinite workers can be placed in a physical space and issues of how they collaborate with each other so they do not run into each other all the time. The same is true for parallel computing, the more computers there are on some task, the faster it can be performed, but like the workers it depends on how much interprocess communication exists between them, how often they have to synchronise and physical restrictions such as bandwidth and memory.

The objectives for this work are to build a system that demonstrates the use of multiple processors on an vision detection algorithm, and optimising where necessary algorithms and communication protocols to make the system as efficient as possible with the given resources. The vision is to have a reusable framework for medium sized mobile robotics such as humanoid size robot and autonomous vehicles to enable high speed image processing so to make them more aware of their surroundings (for e.g. localisation) and intelligent in they way they interact with humans (for e.g. gesture recognition).

The scope of the work includes using a particular form of vision detection algorithm, the Viola Jones method. Extensive research has been conducted on this topic and many resources are available from which to start from, these include work by Andre L. Barczak (supervisor of this thesis) for e.g. in [Barczak et al., 2005a] that provides a framework for hands detection (see chap. 2). Many types of architectures exist for computing, these include embedded microprocessors, desktop computing processors and special purpose hardware like FPGA and DSP processors. This work will focus on using commodity off the shelf hardware that is able to run linux out of the box and provide common interfaces such as USB for easy interfacing with easily available web camera hardware.

Chapter two reviews relevant literature and how it has shaped this work. The literature review is split into three categories, fundamental theory, software and hardware to de-lineate the different aspects of the project. First fundamental theory explains the workings of object

detection method used for this thesis and links some of the aspects of the algorithms to how it may function on parallel machines.

Past work is reviewed, including designs and proposals of architectures and systems for parallel image processing. This includes review of literature on parallel machines based on the Beowulf formula for achieving faster computing, their operation and construction.

A particular programming problem can be solved in many ways; either leveraging on already available application programming libraries or starting from scratch. The section on software infrastructure surveys available options such as Message Passing Interface messaging platform for communication.

The literature review concludes with a summary of all of the literature reviewed and what things were taken into consideration and how they effected this work.

Chapter three discusses the materials and methods used during for the development. The specifications of hardware chosen, why it was chosen and its general setup are outlined. The second part of the chapter details the software infrastructure used to develop the system. The distribution and its build system used to build the software and deploy the system is shown in detail. Testing tools allow one to improve the ability to check the functionality of some feature without having to actually deploy on the hardware. The various testing tools and how they were used in this work are discussed. Methods of benchmarking are shown and their results verified to show that all results are accurate are outlined towards the end of the chapter.

In chapter four a detailed description of the software, including the protocol and framework. At the lowest level the algorithms and packet structures are discussed. Synchronisation is an important aspect of any parallel programming, especially with communication protocols. Synchronisation over broadcast channels are explored, following by some conclusions on the approach. Finally the chapter concludes with a detailed description of the implementation of protocols designed around UDP, reusable framework and application programming interface that is exposed for the implementation of the system.

In chapter five results are presented. The chapter provides details of the setup of the system used for benchmarking the system. First isolated benchmarks of the protocols is presented conducted on a larger cluster to examine the effect of increased number of nodes. Following this, an examination of the hardware used for the embedded system is presented specifically related to rates at which it is able to compute algorithm related tasks.

Following isolated benchmarks, the results of the benchmarks on the completed system with all of the protocols and algorithms are shown with rigorous analysis and reflection of results.

Chapter six concludes the thesis by reviewing the results from chapter 5. A review of the work conducted and some perspectives on the various aspects are provided for anyone wishing to repeat a similar project. Finally the chapter concludes with future work and the various directions that can be taken from this point such as other object detection algorithms that can be performed on the system or modifications to the hardware to improve on the performance already gained.