IMPLEMENTATION OF UNMANNED AERIAL VEHICLE MOVING OBJECT DETECTION ALGORITHM ON INTEL ATOM EMBEDDED SYSTEM

CHEONG WEI WEI

UNIVERSITI TEKNOLOGI MALAYSIA

IMPLEMENTATION OF UNMANNED AERIAL VEHICLE MOVING OBJECT DETECTION ALGORITHM ON INTEL ATOM EMBEDDED SYSTEM

CHEONG WEI WEI

A project report submitted in partial fulfilment of the
requirements for the award of the degree of
Master of Engineering
(Electrical - Computer and Microelectronic System)

Faculty of Electrical Engineering
Universiti Teknologi Malaysia

JUNE 2015

Specially dedicated to my beloved family, lecturers and friends
For the guidance, encouragement and inspiration
Throughout my journey of education

ACKNOWLEDGEMENT

First of all, I would like to take this opportunity to express my highest gratitude to my project supervisor, Dr. Usman Ullah Sheikh for his valuable guidance and support. Without his highly knowledgeable suggestions, I would not be able to complete this research.

Secondly, I would like to express my sincere thanks to my manager, colleagues and friends that have lent their helping hands when I was in trouble. I am extremely thankful and indebted to them.

Last but not least, I am grateful to have my family's and partner's support along the journey for this research. Their unceasing encouragement and attention have motivated me whenever I wanted to give up.

ABSTRACT

Unmanned Aerial Vehicles (UAV), which are commonly known as drones, are aircrafts that have no human pilot on board. A UAV always implements plenty sophisticated functions such as for military surveillance. Among all these complicated functions, one common task is normally implemented on a UAV which is the moving object detection algorithm. There are mainly two ways to implement this algorithm on a UAV, one is to use ground control station and another way uses on board processing method. Implementation of an on board processing unit on a UAV can eliminate the need of video streaming and can implement the computer vision algorithms on board. On board processing unit has mainly two options in implementing the moving object detection algorithm: hardware implementation and software implementation. Hardware implementation means the entire algorithm is designed and transformed into a hardware circuitry. Hardware implementation usually provides promising processing speed. However, whenever there is a slight change in the algorithm, there will be a huge redesign effort and development costs involved. Therefore, this research wants to prove that software implementation of the same algorithm by using a low power general purpose processor can achieve the same processing speed as the hardware implementation. Moreover, the processing speed of the algorithm can be further improved by applying appropriate code optimization techniques on the software program of the algorithm. Experimental results show that software implementation of the UAV moving object detection algorithm by using low power Intel Atom processor D2700 can achieve 30 frame per second processing speed. This research also proved that code optimizing techniques such as loop unrolling and Single Instruction Multiple Data (SIMD) can improve the processing speed of the algorithm up to 25 percent. In conclusion, software implementation of the UAV moving object detection algorithm, which requires low redesign effort and development cost, is capable of achieving the same processing speed provided by the hardware implementation.

ABSTRAK

Kenderaan tanpa pemandu udara (UAV), yang dikenali sebagai pesawat robot, adalah pesawat yang tidak mempunyai juruterbang manusia di atas kapal. UAV sentiasa melaksanakan banyak fungsi yang canggih seperti pengawasan tentera. Di antara semua fungsi rumit ini, satu tugas yang biasa dilaksanakan oleh UAV adalah algoritma pengesanan objek bergerak. Terdapat dua cara utama untuk melaksanakan algoritma ini pada UAV, salah satu adalah dengan menggunakan stesen kawalan tanah dan cara kedua adalah menggunakan kaedah pemprosesan atas kapal. Pelaksanaan unit pemprosesan atas kapal pada UAV boleh menghapuskan keperluan video aliran dan boleh melaksanakan algoritma penglihatan komputer di atas kapal. Unit pemprosesan atas kapal boleh dilaksanakan dengan dua pilihan algoritma pengesanan objek bergerak: pelaksanaan perkakasan dan pelaksanaan perisian. Pelaksanaan perkakasan bermakna keseluruhan algoritma direka dan diubah menjadi litar perkakasan. Pelaksanaan perkakasan biasanya memberikan kelajuan kepada pemprosesan. Walau bagaimanapun, apabila terdapat sedikit perubahan dalam algoritma, usaha besar reka bentuk semula dan kos pembangunan diperlukan. Oleh itu, kajian ini ingin membuktikan bahawa pelaksanaan perisian algoritma yang sama dengan menggunakan pemproses kegunaan am kuasa rendah, boleh mencapai kelajuan pemprosesan yang sama seperti pelaksanaan perkakasan. Selain itu, kelajuan pemprosesan algoritma boleh dipertingkatkan lagi dengan menggunakan teknik pengoptimuman kod yang sesuai kepada algoritma program perisian. Keputusan eksperimen menunjukkan bahawa pelaksanaan perisian daripada algoritma pengesanan objek UAV bergerak dengan menggunakan kuasa rendah Intel Atom pemproses D2700 boleh mencapai kelajuan memproses 30 kerangka per saat. Kajian ini juga membuktikan bahawa teknik kod mengoptimuman seperti “loop unrolling” dan “Satu Arahan Pelbagai Data (SIMD)” boleh meningkatkan kelajuan pemprosesan algoritma sehingga 25 peratus. Kesimpulannya, pelaksanaan perisian daripada algoritma pengesanan objek UAV bergerak, yang memerlukan usaha reka bentuk semula dan kos pembangunan yang rendah, mampu mencapai kelajuan pemprosesan yang sama disediakan oleh pelaksanaan perkakasan.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVATIONS

ALU	-	Arithmetic Logic Unit
API	-	Application Programming Interface
ASIC	-	Application Specific Integrated Circuit
CCL	-	Connected Component Labelling
DARPA	-	Defense Advanced Research Project Agency
DRAM	-	Dynamic Random Access Memory
FPGA	-	Field Programmable Gate Array
GCC	-	GNU Compiler Collection
GPP	-	General Purpose Processor
ICC	-	Intel C++ Compiler
IR	-	Instruction Register
OpenMP	-	Open Multi Processing
PC	-	Program Counter
PCI	-	Peripheral Component Interconnect
RANSAC	-	Random Sample Consensus
RISC	-	Reduced Instruction Set Computer
SIMD	-	Single Instruction Multiple Data
UAS	-	Unmanned Aerial System
UAV	-	Unmanned Aerial Vehicle
USB	-	Universal Serial Bus

CHAPTER 1

INTRODUCTION

1.1 Introduction

Unmanned aerial vehicle, which normally known as UAV, can generally be defined as an aircraft that has no on-board pilot and it can be either piloted remotely or flown autonomously. They are often large enough to accommodate sensors, cameras or other information gathering equipment that can provide low-cost aerial information such as large scale low altitude imaging [3]. Over the past five years, UAV is widely used for diverse operations such as military reconnaissance, intelligence-gathering, public security, law enforcement, border patrol, emergency services and video surveillance system in this technology advanced world [3] [4]. With its small and light size, UAV can be flown at a very low altitude range in hazardous area to acquire high resolution images for rescue or other useful purposes.

Among most of the UAV operations, one common basic task, which is moving object detection process, is often used to locate and track a moving object of interest in a series of visual images from a bird's eyes view captured by the camera on UAV. UAV control unit will then utilize the processed video frames/images to determine the next operation to be carried out by the UAV. Therefore, the implementation of vision processing algorithm such as moving object detection process on UAV has become a preeminent task [1]. Figure 1.1 shows an example of object tracking snapshot from camera on UAV.

Figure 1.1 Moving object detection and locating from video captured by camera on UAV [1]

1.2 Background of the Problem

Although the UAV technology holds considerable advantages in the video surveillance system, there are still some limitations faced by this technology. Conventionally, an Unmanned Aerial System (UAS) consists of a UAV, a ground control station and a communications data link for UAV command and control sent from ground control station [4]. Computer vision processing applications is usually compute expensive process that requires large amount of computing power. Previously, this high processing power requirement could be achieved on bulky processing units which can be carried only by large and high cost UAVs.

A typical attempt to enable vision processing unit is to include a ground control station whereby the video frames/images captured by the on board camera

are fed back to station and the processed results and control instructions are sent back to the UAV [5]. However this approach is not practical and with limited success due to the delays and interferences imposed to the system by the imperfection of the communications data link between the ground control station and the UAV. Blurred images and videos with degraded quality received will result in inaccurate object detection and tracking.

Rather than using a ground control station as the processing and control unit for the UAV system, another approach will be implementing an on-board processing unit on the UAV so that UAV can perform computer vision processing itself and eliminate the need to stream the video captured through communication link. However, driven by the intense development of actuators and sensors, unmanned aerial vehicle are getting lighter, smaller and less expensive but with more sophisticated capabilities. This factor introduces several constraints such as weight, size and power constraints in selecting the on-board embedded system that has processing capabilities required by the computer vision applications. In short, a low power consumption, light weight, small size and high performance embedded system would be the best solution for the on-board processing unit on the UAV [1] [5] [6].

1.3 Statement of the Problem

Realization of the on-board computer vision processing algorithm on UAV can be achieved with two different methods which are the hardware implementation on programmable hardware components such as Field Programmable Gate Array (FPGA) or Application Specific Integrated Circuit (ASIC) and the software implementation on the programmable platform such as a General Purpose Processor (GPP) [9]. Due to the high cost of developing ASICs, FPGA circuits are now considered as appealing devices for hardware implementation. The promising performance and power efficiency make FPGA an attractive platform for implementing computer vision applications. However, the design and redesign (maintenance) effort for FPGA remains a significant barrier. FPGA-based design

often requires developing time an order of magnitude more than pure software implementation [8]. Furthermore, hardware design expertise is rare as compared with software design expertise is also one of the drawbacks in FPGA design.

Although software implementations require less development infrastructure, skill level and design time, it has the performance gap and power efficiency issues to cater on. This study believes that appropriate code restructuring and optimization can close the performance gap and achieve better processing speed with software implementation. Moreover, with intense improvement achieved by the general purpose processors recently, low power consumption processor with high processing speed, light weight and small size are highly available in the market nowadays with reasonable pricing. Therefore, now is the right timing to evaluate the software implementation on the on-board vision processing algorithm.

1.4 Objectives of the Study

The objectives of this study are listed as the following:

1. To implement a UAV moving object detection algorithm on a low power processor. The study is focused on implementing the algorithm mentioned on embedded system with Intel Atom Processor as of now.
2. To perform profiling on the UAV moving object detection algorithm.
3. To perform benchmarking on the Intel Atom processor.
4. To perform code optimization on the UAV moving object detection algorithm.
5. The ultimate goal of this study is to implement a real time embedded system with the Intel Atom processor that can process 30 frames per second processing speed using the UAV moving object detection algorithm.

1.5 Scope of Work

The scopes of this study are as follows:

1. The UAV moving object detection algorithm used in this study is an existing algorithm based on [1].
2. The only embedded programming language used in this study is C Programming.
3. The embedded system to be used in this study is targeted on embedded system with Intel x86 architecture processor. The study is focused on embedded system with Intel Atom Processor as of now.
4. Pre-capture video files and existing image database are used in verifying the performance of the algorithm and embedded system. The current database selected is Video Verification of Identity (VIVID) database [7] provided by Defense Advanced Research Project Agency (DARPA).

1.6 Research Contributions

This research contributes in several areas as follows:

1. A software implementation of low cost, high performance moving object detection algorithm on a low power embedded system.
2. Several code optimization techniques that are capable to improve the computer vision processing algorithm.

1.7 Thesis Organization

This thesis is subdivided into 5 chapters:

1. Chapter 1 : Introduction

This chapter outlines the basic understanding on the problem background, objectives, scope and contributions of this research.

2. Chapter 2 : Literature Review

   This chapter includes in-depth literature review on all the related fields such as the algorithm of UAV moving object detection, processor architectures, code profiling methods, processor benchmarking techniques and code optimization techniques.

3. Chapter 3 : Research Methodology

   This chapter describes the flow and methodology used in conducting the entire research. Information such as processor board, benchmarking codes, video database and profiling tool being used can be found in this chapter.

4. Chapter 4 : Experimental Results and Discussion

   This chapters consists of all the results, graphs of data and corresponding discussion on the results for all the experiments conducted in this research.

5. Chapter 5 : Conclusion and Future Work

   This chapter concludes all the findings and achievements obtained in this research and suggests several enhancements can be done in future.

REFERENCES

1. J. W. Tang, "Moving Object Detection For Unmanned Aerial Vehicle Using Field Programmable Gate Array," *Masters Thesis,* 2014.
2. A. Price, J. Pyke, D. Ashiri, and T. Cornall, "Real Time Object Detection for an Unmanned Aerial Vehicle using an FPGA based Vision System," 2006 IEEE International Conference on Robotics and Automation, p. 6, May 2006.
3. McCormack, Edward D., "The Use of Small Unmanned Aircraft by the Washington State Department of Transportation," University of Washington, Washington, June 2008.
4. Colomina, P. Molina, "Unmanned aerial systems for photogrammetry and remote sensing: A review," ISPRS Journal of Photogrammetry and Remote Sensing, p. 19, 2014.
5. S. K. Phang, J. J. Ong, Ronald T. C. Yeo, Ben M. Chen, and T. H. Lee, "Autonomous Mini-UAV for Indoor Flight with Embedded On-board Vision Processing as Navigation System," Computational Technologies in Electrical and Electronics Engineering (SIBIRCON), p. 6, 2010.
6. Fowers, S.G., Lee, Dah-Jye, Ventura, D.A. and Archibald, J.K., "The Nature-Inspired BASIS Feature Descriptor for UAV Imagery and Its Hardware Implementation," Circuits and Systems for Video Technology, p. 13, 2013.
7. R. T. Collins, "VIVID Tracking Evaluation Web Site," School of Computer Science, Carnegie Mellon University., [Online]. Available: http://vision.cse.psu.edu/data/vividEval/datasets/datasets.html. [Accessed 12 2014].
8. J. Bodily, B. Nelson, ZY. Wei, DJ. Lee, and J. Chase, "A Comparison

Study on Implementing Optical Flow and Digital Communications on FPGAs and GPUs," ACM Transactions on Reconfigurable Technology and Systems, p. 22, 2009.

9. N. Lukić, I. Papp, Z. Marčeta and M. Temerinac, "Software Based Video Improvement Implementation," Engineering of Computer Based Systems, 2009. ECBS-EERC '09, p. 6, 2009.
10. John L. Hennessy, David A. Patterson, Computer Architecture: A Quantitative Approach, Waltham, 2012.
11. J. Tong and M. Khalid, "Profiling CAD tools: A proposed classification," Microelectronics, 2007, p. 4, 2007.
12. E. Moorits and G. Jervan, "Profiling in deeply embedded systems," Electronics Conference (BEC), 2012 13th Biennial Baltic, p. 4, 2012.
13. "Details about Intel Vtune Amplifier 2015," INTEL, 2014. [Online]. Available: https://software.intel.com/en-us/intel-vtune-amplifier-xe. [Accessed Dec 2014].
14. Weicker, Reinhold P., "DHRYSTONE: A SYNTHETIC SYSTEMS PROGRAMMING BENCHMARK," Communications of the ACM, vol. 27, no. 10, p. 18, 1984.
15. Z. Wang, T. Wild, S. R and B. Lippmann, "Benchmarking Domain Specific Processors: A Case Study of Evaluating A Smart Card Processor Design," Symposium on VLSI, p. 6, 2008.
16. Weiss, and Alan R., "Dhrystone Benchmark," The EEMBC Certification Laboratories, LLC (ECL), 2002.
17. H. J. Curnow and B. A. Wichmann, "A synthetic benchmark," Computer Journal, vol. 19, p. 16, 1976.
18. A. González, F. Latorre and G. Magklis, Processor Microarchitecture: An Implementation Perspective, Morgan & Claypool, 2011.
19. INTEL, "Intel® Atom™ Processor N2000/D2000 Series, NM10 Chipset," INTEL, [Online]. Available: http://www.intel.com/content/www/us/en/intelligent-systems/cedar-trail/atom-n2000-d2000-ibd.html.

20. J. Turley, "Introduction to Intel® Architecture," INTEL.

21. INTEL, Intel® 64 and IA-32 Architectures Optimization Reference Manual, 2014.

22. INTEL, Intel® 64 and IA-32 Architectures Software Developer's Manual, 2014.

23. J. Buss, "VINTAGE DEC PAGES," [Online]. Available: http://www.xanthos.se/~joachim/dhrystone-src.tar.gz. [Accessed December 2014].

24. "Netlib Repository: Whetstone benchmark," [Online]. Available: http://netlib.org/benchmark/whetstone.c. [Accessed December 2014].

25. "Netlib Repository: Linpack Benchmark," [Online]. Available: http://www.netlib.org/benchmark/linpackc.new. [Accessed December 2014].

26. INTEL, "ARK | Your Source for Intel® Product Information," INTEL, [Online]. Available: http://ark.intel.com/. [Accessed December 2014].

27. F. Vahid, and Tony D. Givargis, Embedded System Design: A Unified Hardware/Software Introduction, Wiley, 2001.

28. Johnson, Neil E., "Code size optimization for embedded processors," University of Cambridge, 2004.

29. R. Leupers, Code Optimization Techniques for Embedded Processors, 2002.

30. A. Kejariwal, A. Veidenbaum, A. Nicolau, X. Tian, M. Girkar, H. Saito and U. Banerjee, "Comparative architectural characterization of SPEC CPU2000 and CPU2006 benchmarks on the intel® Core™ 2 Duo processor," Embedded Computer Systems: Architectures, Modeling, and Simulation, 2008, p. 10, 2008.

31. S. Liao, S. Devadas, K. Keutzer, S. Tjiang and A. Wang, "Code Optimization Techniques for Embedded DSP Microprocessors," Design Automation, 1995, p. 6, 1995.

32. N. Zingirian and M. Maresca, "On the efficiency of image and video processing programs on instruction level parallel processors," IEEE, vol.

90, no. 7, p. 14, 2002.

33. ARM, "Cortex-A Series - ARM," ARM, [Online]. Available: http://www.arm.com/products/processors/cortex-a/index.php. [Accessed December 2014].

34. ARM, "Cortex-R Series - ARM," ARM, [Online]. Available: http://www.arm.com/products/processors/cortex-r/index.php. [Accessed December 2014].

35. ARM, "Cortex-M Series - ARM," ARM, [Online]. Available: http://www.arm.com/products/processors/cortex-m/index.php. [Accessed December 2014].

36. ARM, ARM Architecture Reference Manual, ARM, 2005.

37. M. Booshehri, A. Malekpour, and P. Luksch, "An improving method for loop unrolling," vol. 11, 2013.

38. Huang, J.C., and Leng, T., "Generalized loop-unrolling: a method for program speedup," pp. 244 - 248.

39. L. Baumstark, Jr. and L. Wills, "Exposing Data-Level Parallelism in Sequential Image Processing Algorithms," 2002.

40. Kadidlo, J., and Strey, A., "Exploiting Data- and Thread-Level Parallelism for Image Correlation," pp. 407 - 413, 2008.

41. J. Reinders, Intel Threading Building Blocks, O'Reilly Media, Inc., 2007.

42. G. Slabaugh, R. Boyes, and XY. Yang, "Multicore Image Processing with OpenMP [Applications Corner]," vol. 27, no. 2, pp. 134 - 138, 2010.