2nd International Conference on Autonomous Robots and Agents
December 13-15, 2004 Palmerston North, New Zealand

An Analytic Model for Embedded Machine Vision: Architecture and Performance Exploration

Chan Kit Wai, Prahlad Vadakkepat, Tan Kok Kiong
Department of Electrical and Computer Engineering,
4 Engineering Drive 3,
National University of Singapore,
Singapore 117576
{g0203549, prahlad}@nus.edu.sg

Abstract

This paper represents a model to analyze the performance of an embedded machine vision system. Such an application is computationally expensive and often is a challenging problem in the field of embedded systems. A model is derived to analyze the performance. It predicts system performance with minimal efforts and costs over a wide range of processor architectures. Specifically, image segmentation is evaluated with the microprocessor technology. In the same way, a custom logic is designed with the help of the model.

Keywords: Analytic Model, Embedded Machine vision, Custom vision architecture

1 Introduction

Image processing is often associated with huge amount of data processing [**?**]. Such computational requirements often pose a challenging problem in the field of embedded systems.

Most of the embedded systems have limitations of computational power, low data transfer rate, very tight memory budget, size constraints and most importantly cost. Additionally, low energy requirement has become an important factor in recent years [**?**] [**?**].

For battery operated devices, it is preferable to have low power consumption. Such concern is important in order to reduce the recharging frequency.

The memory constraint in embedded technology also plays an major obstacle. Vision algorithms demand high memory requirements compared to other applications, on the contrary, most embedded systems have tight memory budget. Such limitations forbid the implementation of many vision applications.

Lastly, the speed of executing an image processing task must be comparable to the speed of acquiring the image (i.e. frame image rate) [**?**]. As such, the limitation of computing power in embedded systems inhibit the task to be completed on time.

Nevertheless, many of the vision systems have carefully tackled to meet the computational time requirements. Perhaps reducing the image resolution is one the the easiest methods to handle the real time constraints. Examples of such systems are found in eyebot [**?**], CMUCam [**?**] and Khepera vision turret [**?**]. Other methods include running the algorithms on a high speed computer, exploring computationally efficient algorithms and exploring various hardware architectures.

While several articles in the literature discuss about complex vision algorithms, this paper focuses on the importance of the vision system's performance. However it does not discuss the compiler techniques for memory and computational optimization. Rather, this paper introduces a model to estimate the performance requirements for a given task and specifications.

In literature, there are three basic techniques for performance evaluation; namely measurement, simulation and analytic modelling [**?**] [**?**].

An analytic model is often used to predict performance. It can evaluate the performance with minimal efforts and costs over a wide range of choices in the system parameters and configurations [**?**]. For various processor architectures, the key resources and workload requirements can be analytically modelled with sufficient realism to provide insight into the bottlenecks and key parameters affecting the system performance [**?**]. Nevertheless, the model fails if it is required to model the vision system in great details [**?**].

This paper represents a model to analyze the performance of machine vision in embedded environments. It also discusses about the limitations of the general purpose processors and the possibilities of alternative processors.

This paper is organized as follows: Section 2 describes the analytic model for image processing. Section 3 discusses image segmentation using microprocessor. Cus-

tom architecture is provided in Section 4 and Section 5 concludes the work.

2 Analytic Model for Image Processing

The data processing is represented using a pyramid architecture as shown in Figure **??**. The bottom level of the pyramid represents the data volume to be processed. The top level of the pyramid represents abstract information derived from the image. The bottom level comprises of raw pixels acquired from source image. Intermediate levels are typically pre-processing, segmentation, feature extraction and classification. The top level produces the abstract data to the rest of the system.

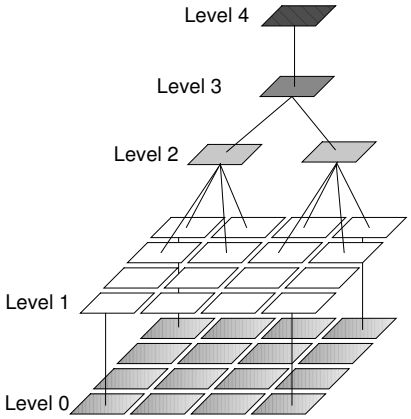


Figure 1: Data processing at different levels

The lowest level vision task often consumes the most computing resources. Low-level task consists of pixel-based transformations such as filtering, edge detection and segmentation processes. These tasks are characterized by a large amount of data (pixels), small neighbourhood operators and relatively simple structure operations (e.g multiply, add or compare) [**?**].

On the other hand, high level tasks are more dynamic in nature. They are more decision oriented and do not have a repetitive execution of a set of algorithm. Hence, the model discussed in this paper is applicable specifically for low-level repetitive tasks.

An analytic model is derived to provide a mathematical description of the vision system. Such approach is considered being far less time-consuming and more flexible compared to simulation based methods. The model can be used to describe a different level of abstraction. The model described in Section 2.2 and 3.2 is used to estimate the processor clock speed requirements as well as exploring custom architecture for such application. The mathematical model helps to represent computing resources and the speed of the hardware. Such model allows a simple and fast calculation of different processor's response time.

2.1 Performance Parameters

The specifications generally refer to the response time, frame rate, image resolution, pixel transfer rate and the computational task. The performance requirements of a system is referred to the number of processing units, the memory size and clock speed to meet the specifications provided.

The definitions used in the model to evaluate the system performance parameters are as follows:

$\lambda = Arrival\ rate\ of\ pixels$
$r = response\ time$
$w = wait\ time$
$s = service\ time$
$s_p = service\ time per pixel$
$n = number\ of\ jobs\ in\ system$
$n_q = number\ of\ jobs\ in\ queue$
$n_s = number\ of\ jobs\ in\ service$
$i \times j = population\ size(in pixels)$
$CPI = cycles\ per\ instruction$

Queueing theory plays a very important role in analytical modelling [**?**]. From the Little's Law in queueing theory [**?**], equation (**??**) to (**??**) are used as the fundamental formulae to calculate the required vision system requirements. These requirements include the number of processors, processor's computing speed, instruction sets, arithmetic units (ALU) and memory size of the system.

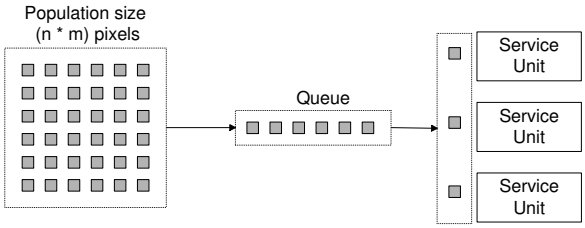


Figure 2: Queue model of vision system

2.2 Queuing Theory

The system can be modelled using the queueing analysis as illustrated in figure **??**. The population size, queue size and service centers refer respectively to the number of pixels, memory size, processing units in a vision system.

$$n = \lambda r\ , \tag{1}$$

$$n_q = \lambda w\ , \tag{2}$$

$$n_s = \lambda s\ , \tag{3}$$

where

$$\lambda = \frac{a}{T}\ , \tag{4}$$

$$r = w + s \ , \\ n = n_q + n_s \ , \tag{5}$$

where a is the number of jobs arrived, in time T. The basic approach is to identify whether the system is able to handle a given load, in another words, the stability condition is satified (**??**).

$$\lambda < m\mu \ , \tag{6}$$

where m is the number of service units and $\mu = 1/s$ is the service rate of each service unit. If the number of jobs in a system grows continuously and becomes infinite, the system is said to be unstable. For stability, λ (mean arrival rate) should be less than $m\mu$ (mean service rate) [**?**].

To find the stability condition, λ and s are required. As a result, s (service time for the image) and s_p (service time per pixel) can be calculated as shown in equations (**??**) and (**??**) respectively.

$$s = s_p(i * j) \ , \tag{7}$$

$$s_p = n_{ipp} * n_{cpi} * t_{clk} \ ,$$

$$s_p = \sum_{g=1}^{g=g_{max}} [n_{ipp}(g) * n_{cpi}(g)] * t_{clk} \ , \tag{8}$$

where n_{ipp} is the number of instructions per pixel and n_{cpi} is the number of cycles per instruction. Both these parameters are a function of g, a group of instructions set with the same CPI. An example of such grouping is shown in Table 1.

Table 1: Instruction grouping

g	instruction set	CPI
1	MOVF	1
	SUBWF	1
2	BTFSC	2
	GOTO	2

The model decomposes the system into algorithms, instruction sets and instruction cycles. Instructions with different number of CPIs are grouped together. Thus, the total number of cycles in the program can be easily calculated. Finally, with all the input parameters, it is possible to estimate the required processing power, in particular, the processing frequency.

However, this model does not consider several computational overheads, like initialization routine, loops and error checking, etc.

The following sections discuss the usage of such model for image segmentation in two (2) different types of processing architectures; namely microprocessor and custom architecture.

3 Image Segmentation using Microprocessor

The segmentation process partitions an image into meaningful regions [**?**]. The nature of this process involves scanning of each pixels.

A simple experimental setup for color segmentation is tested in Visual C/C++ environment. The program reads in the raw image data from a Portable Pixel Map (PPM) file, converts the Red, Green, Blue (RGB) primaries to the Hue, Saturation, Value (HSV) color space, performs color segmentation and converts back to RGB format. Finally, it writes the image into a destination PPM file for viewing. Figure **??** illustrates the process.

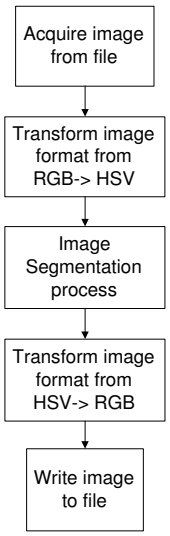


Figure 3: Experimental setup

3.1 An Algorithmic Approach

For instance, a real-time vision system with a resolution of 640x480 is required to be processed at a rate of 30 fps (frames per second). The analytical model is used to estimate the performance of the color segmentation process. A reference color is selected and the distance between the two color points $p_i = [H_i, S_i, V_i]$ and $p_{ref} = [H_{ref}, S_{ref}, V_{ref}]$ is given as,

$$d_H = \begin{cases} |H_i - H_{ref}| & if\ |H_i - H_{ref}| < 2\pi \\ 2\pi - |H_i - H_{ref}| & otherwise \end{cases} \ ,$$

$$d_S = |S_i - S_{ref}| \ ,$$

$$d_V = |V_i - V_{ref}| \ . \tag{9}$$

The segmentation process is as follows,

$$I(i,j) = \begin{cases} 1 & if\ (d_H < d_{H_{ref}}\ and\ d_S < d_{S_{ref}} \\ & \quad and\ d_V < d_{V_{ref}}) \\ 0 & otherwise \end{cases} , \quad (10)$$

where $d_{H_{ref}}$, $d_{S_{ref}}$ and $d_{V_{ref}}$ are the acceptable sensitivity distances. From Equation (**??**), the image is segmented into a binary image of '1's and '0's.

The microprocessor technology is often used for image segmentation process. It requires several load, store and branch operation to perform data manipulation. Figure **??** shows the implementations in C language and the equivalent assembly instruction code. The results obtained are shown in Figure **??**.

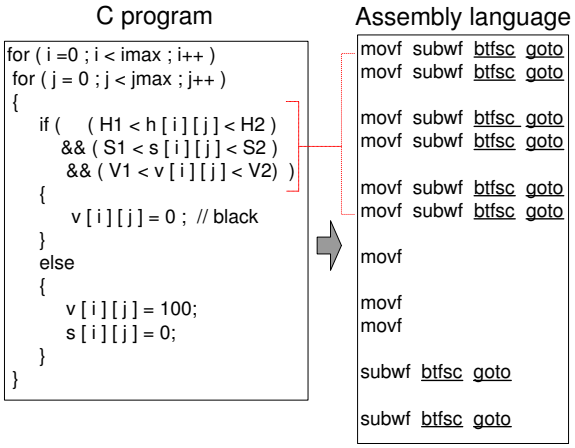
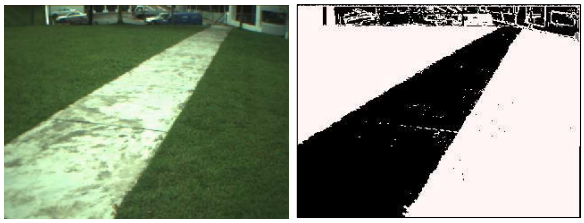


Figure 4: Assembly code representation of C program



(a) (b)

Figure 5: (a) original image (b) segmented image using HSV color space

From equation (**??**), the processing speed for performing the segmentation process is estimated.

$$t_{clk} = \frac{s_p}{\sum_{g=1}^{g=g_{max}} [n_{ipp}(g) * n_{cpi}(g)]} , \quad (11)$$

where ,

$$s_p = \frac{s}{i * j} , \quad (12)$$

$$s = r - w , \quad (13)$$

$$w = n_q * s_p . \quad (14)$$

Substituting equations (**??**) and (**??**) into equation (**??**) gives,

$$s_p = \frac{r - n_q s_p}{i * j} , \qquad s_p = \frac{r}{i * j + n_q} . \quad (15)$$

3.2 Estimating the Required Processing Speed

In this example, it is desired to estimate the required processing speed. The desired response time is given to be $1/30 fps$ = 33.33 ms and the image resolution is 640x480 pixels. Equations (**??**) to (**??**) are used as the model for calculations. As such, from Figure **??**, $n_{ipp}(g = 1)$ is found to be 17 with one CPI, and $n_{ipp}(g = 2)$ is 8, taking the assumption of branch-always-taken with a CPI of two. Subsequently, two extreme cases of n_{q1}=0 (0 pixels in queues) and n_{q2}=640x480 (max no. of pixels in queue) are used to determine the t_{clk}. For both cases, $t_{clk1}(n_{q1})$ is 3.28ns (304.8 Mhz) and $t_{clk2}(n_{q2})$ is 1.64ns (608.25 Mhz).

From the calculations, it is noted that, a simple image segmentation operation requires a very high clock frequency. The performance of such a system greatly depends on the computing speed of the processor.

The segmentation algorithm does not actually map to appropriate hardware resource to achieve efficiency. Most of computing time is spent on "overheads" instructions rather than for the actual processing of data. Hence, it is reported that most computationally complex applications spend 90% of their execution time on only 10% of their code [**?**].

4 Customized Architecture for Image Segmentation

The mathematical model helps to increase response time by choosing the right hardware configurations. The parameters corresponding to the hardware resources are identified to improve system performance.

For instance, from equation (**??**), m, the number of service units represents the number of processors servicing the jobs. If the system does not meet the condition specified in (**??**), m is increased to bring the system from an unstable condition to a stable condition. This reflects in the actual implementation of increasing the number of processors in the system. Another example is observed from equation (**??**). The parameters n_{ipp} and n_{cpi} can be reduced by selecting an appropriate instruction-set-architecture (ISA). A suitable processor ISA should contain an instruction to compare two registers and branch to a specific location within a single cycle. Consequently, a lower processor clock speed can be used as a result of reducing n_{ipp}.

4.1 Achieving One Pixel per Cycle

To achieve a very efficient system, the software algorithm and hardware architecture must match well. This can be achieved for algorithms with high degree of regularity that are identified to exploit parallelism. With the model as a reference, these operations are mapped into custom functional units to achieve a higher performance compared to the fixed instruction set architecture.

In ideal case, one can set $\sum_{g=1}^{g=g_{max}}[n_{ipp}(g) * n_{cpi}(g)] = 1$. Effectively, from (**??**), $t_{clk1}(n_{q1})$=108ns; which shows the reduction of clock frequency by 96.95%.

Such performance with low clock frequency is achieved by customizing a dedicated hardware resource. The customized architecture allows direct computation instead of conventional load store operations. Such structure is able to compute a pixel in a single instruction within a single clock cycle.

To explore further, parallel processing of multiple pixels is also possible, considering that the pixels are transferred parallelly. Otherwise, queueing of pixels is necessary, which leads to undesirable idle time. Hence, sequential processing of a single pixel provides a better solution.

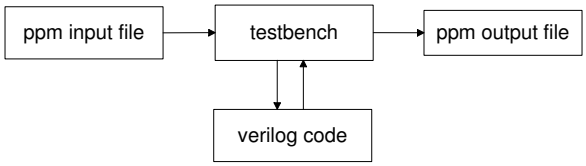


Figure 6: Simulation configurations

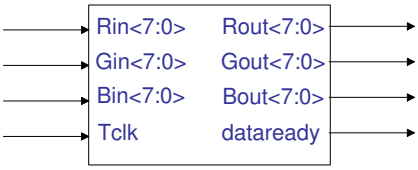


Figure 7: Block diagram of image segmentation processor

The customized hardware architecture is described using the Hardware Description Language (HDL); specifically verilog HDL. The simulation configurations are shown in Figure **??**. Model Sim (a verilog code simulator program) is used together with the test bench, the verilog code, and the ppm image files to simulate the vision system.

The block diagram and the RTL schematic of the design are shown in Figures **??** and **??** respectively. It is noted that the RGB color space domain is used instead of the HSV for reasons of simplicity.

The simulation results are obtained from the waveform viewer in ModelSim. Figure **??** shows the timing diagram of a pixel being received at t_{clk}'s rising edge and

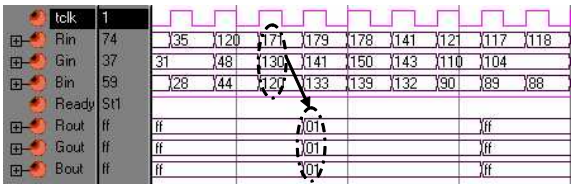


Figure 8: Timing diagram of segmentation process

an output pixel being generated at the following rising edge.

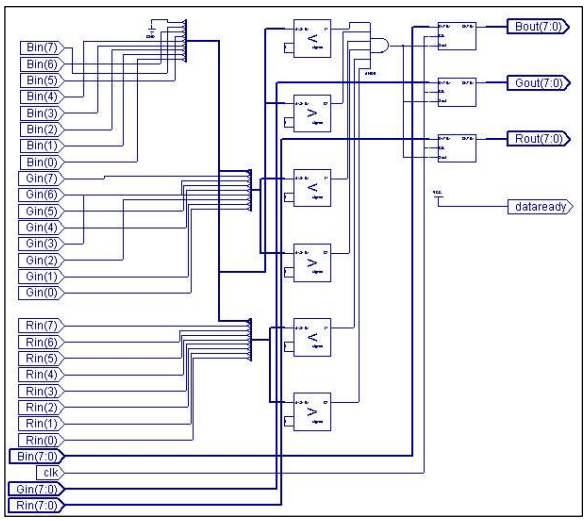


Figure 9: RTL schematic for segmentation

The verilog code is synthesized and the RTL schematic is shown in Figure **??**. The resulted schematic is simple, consisting of just a few comparators and flip-flops. It is noted that the architecture does not really need complex logic circuits to achieve the desired performance.

(a) (b)

Figure 10: (a) original image (b) segmented image using RGB color space

The image is segmented into binary image, where logic '1' is represented by RGB = (0xff, 0xff, 0xff) and logic '0' is represented by RGB = (0x01, 0x01, 0x01). The RGB values are written into a ppm image file and the results are shown in Figure **??**.

There are several ways to realize the implementation of custom logic. However, the most common technolo-

gies are to fabricate it in ASIC (Application Specific IC) or FPGAs (Field Programmable Gate Array). Particularly, FPGAs offer certain advantages over ASIC. It provides the benefits of customized hardware architecture and at the same time allowing dynamic reprogrammability. It is an important characteristic which allows to meet the changing requirements.

5 Conclusion

The image processing analytic model is derived to estimate the various performance parameters associated with the vision system. Such an approach is considered being far less time-consuming and more flexible compared to simulation based approaches.

The derived model allows system designers to estimate the system responses of given specifications. It helps to compare different processor architectures without actual implementation. For instance, the estimation of the clock frequency, response time and delay time is easier. It can be extended to calculate the amount of energy consumed.

The model provides the design space exploration to achieve the desired performance. To reduce the clock frequency, it is needed to reduce the number of instructions. Such hints assist in the design of the custom architecture. The results show that the custom architecture is able to achieve a reduction of 96.95% in clock frequency with respect to the fixed instruction set architecture. In conclusion, the possibilities of such an architecture can be realized in a reconfigurable fabric of FPGA.

References

[1] D. Crookes. Architectures for high performance image processing: The future. In *Journal of Systems Architecture*, volume 45, pages 739–748, 1999.

[2] G.Stitt, B.Grattan, J.Villarreal, and Frank Vahid. Using on-chip configurable logic to reduce embedded system software energy. In *IEEE Symposium on Field-Programmable Custom Computing Machines*, volume 10, pages 143–151, 2002.

[3] T.Simunic, L.Benini, and G.D.Micheli. Energy-efficient design of battery-powered embedded systems. In *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, volume 9, 2001.

[4] R. T. Chin and C. R. Dyer. Model-based recognition in robot vision. In *ACM COMPUTING SURVEYS*, volume 18, pages 67–108, 1986.

[5] Thomas Braunl. Improv amd eyebot real-time vision on-board mobile robots. In *IEEE, Mechatronics and Machine Vision in Practice*, volume 4, pages 131–135, 1997.

[6] A. Rowe, C. Rosenberg, and I. Nourbakhsh. A low cost embedded color vision system. In *IROS 2002 conference*, 2002.

[7] Khepera vision turret. Web page. http://www.k-team.com/robots/khepera/k6300.html, visited on June 2004.

[8] K.Kant. *Introduction to Computer System Performance Evaluation*. Mc Graw-Hill, Singapore, 1992.

[9] David J. Lilja. *Measuring Computer Performance: A practitioner's guide*. Cambridge University Press, United Kingdom, 2000.

[10] Hisashi Kobayashi. *Modeling and Analysis: An introduction to System Performance Evaluation Methodology*. Addison-Wesley, Philippines, 1978.

[11] P.Heidelberger and S.S.Lavenberg. Computer performance evaluation methodology. In *IEEE Transactions on Computers*, volume C-33, pages 1195–1220, 1984.

[12] N. K. Ratha and A. K. Jain. Computer vision algorithms on reconfigurable logic arrays. In *IEEE Transactions, Parallel and Distributed Systems*, volume 10, pages 29–43, 1999.

[13] Raj Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley- Interscience, New York, 1991.

[14] G. J. Awcock and R. Thomas. *Applied Image Processing Book*. McGraw-Hill, USA, 1996.

[15] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A quantitative approach*. Morgan Kaufmann, Calif., 1990.