An Analysis of Strategies to Reduce Computational Complexity and Processing Time in Industrial Optical Data Processing and Analysis

Adrian Trenaman, David Barry, David Vernon[1].
Department of Computer Science, Maynooth College, Co. Kildare, Ireland

Telephone: 708 3847
Fax. 708 3848

trenaman@cs.may.ie, drbarry@cs.may.ie, dvernon@cs.may.ie

ABSTRACT

In this paper we address the evaluation of the efficiency of machine vision systems. We provide a framework for the analysis of such systems, incorporating algorithmic complexity and top-down structured analysis. Using the framework it is possible to compare different solutions working over different machine architectures. An example of such a comparison is presented, where we compare the image acquisition characteristics of a contour following algorithm. A simulation of the algorithm verifies the correctness if the component based framework.

1. Introduction

Very little work has been done in providing a general approach to the analysis of the performance of machine vision systems, despite the potential usefulness of such a tool. Some reasons for this lack of research in such an important area are offered by Forstner [1], who lists among them the complexity of vision systems, the impact of image data on performance, and the number of tuning parameters for a vision system. Performance evaluation is a two fold phenomenon, that is, by "performance" we mean firstly whether the technique is correct, and secondly, how efficient the technique is, in terms of execution time and resource usage. It is this second meaning of performance that we shall adhere to in this paper.

While there has been an increase in activity in the area of performance as correctness, there is very little work done in the area of efficiency of computer vision algorithms. We propose in this paper to address this problem. In describing the *efficiency* performance of vision algorithms, we borrow the notion of algorithmic complexity from the field of algorithmics [2]. From every algorithm can be derived a complexity function, describing how long the algorithm will take to execute over a certain size of input data. We can describe the complete machine vision system in terms of algorithmic processes, from software right down to hardware. We provide a way to link these components together, showing the data-flow between them and describing how the processes alter the size of this data-flow. Effectively, we perform a top down analysis of the complexity of a vision algorithm, and arrive at a concrete estimate for its performance. This methodology embraces both the software algorithms and the underlying hardware of the solution to give concrete estimates of the algorithms performance. It provides a solid framework against which different solutions can be quantitatively compared and optimal solutions found.

In Section 2 we describe this methodology, introducing a graphical notation to capture the execution of the machine vision algorithm. This methodology is an extension of previous research by the authors in this area [3]. In Section 3 an example is given of the framework in use in which we compare a contour-following algorithm working over two different architectures: a conventional frame-grabber working on the CCIR standard, and a Random Access Image Sensor. We show that the latter architecture will reduce image acquisition time by a factor of 4 for this application. A simulation of the random access solution was performed in software, and the number of pixel accesses were exactly as predicted by the component-based analysis. We conclude that this technique forms a solid framework for the *a priori* evaluation and comparison of machine vision solutions.

[1] This work is funded by ESPRIT Project 20557 RAMAP

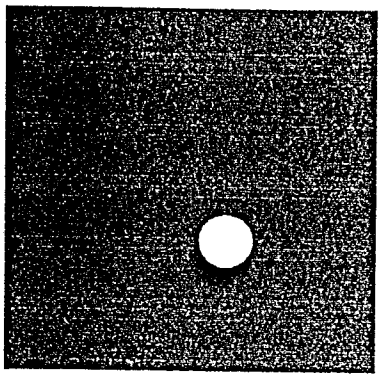


Figure 5: Sample input image.

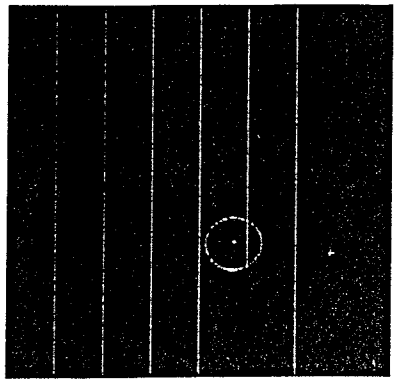


Figure 6: Pixel accesses in running the blob detection algorithm.

The solution we will apply to the problem involves scanning the image to locate the edge of the blob, followed by contour following to produce a binary chain-code representation of the blob. From this representation it is easy to compute the perimeter, area and centroid of the blob [6]. Figure 6 shows the pixels accessed by the algorithm in the course of its execution.

3.1 Frame acquisition using CCIR

Implementing the contour following solution with a frame acquisition architecture using the CCIR signal standard [7] , we identified the following components, as shown in Table 1.

Its clear from the component diagram (Figure 7) that the acquisition stage occurs in the getFrame component, which has a constant time complexity of 40ms.

Component	Time Complexity, t()	Data Reduction, r()
getFrame	40ms	262144
scan	3584 * $T_{3x3Filter}$	1
followContour	240 * $T_{decision}$	240

Table 1: Components for a frame-acquisition based solution.

3.2 Random Access sensor.

This implementation shares the algorithmic components `scan` and `followContour` but use a different image access technique. Since we are interested in evaluating the pixel accesses of the algorithm and since both the scan and followContour processes make such accesses, the component diagram is a little more complex. Pixel access can be either slow or fast, due to the asymmetry of the random access chip. Further, in-memory caching is used to ensure that pixels are not fetched more than once. We identify the following new components in Table 2, and the algorithm is shown in Figure 8.

Component	Time Complexity, t(n)	Data Reduction, r(n)
getFilter	Tslow + 2 * Tfast	9
getPixel	Tslow * 0.66 + Tfast * 0.33	1
slowAccess	Tslow = 2μs	1
fastAccess	Tfast = 0.2μs	1

Table 2: Components for the pixel-acquisition based solution.

Using the component analysis technique, the total acquisition time for the random access approach is estimated as:

$$T_{scan} + T_{follow} = 10267.2\ \mu s$$

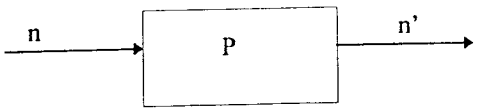
So, a worst case estimate for the acquisition time is 10.2 ms, for this implementation of the algorithm.

2. The Framework.

We analyse our machine vision system in terms of components. A component is an algorithmic process working in hardware or software, and as such has a number of attributes: Time complexity, Data Complexity and a Data Reduction Function. An algorithm working on a data-set of size *n*, has a *time complexity* function, *t(n)*, which describes the time required to successfully run the algorithm. Usually *t()* gives a worst-case estimate of the running cost of the algorithm. In the same way that *t(n)* describes the amount of time taken to execute an algorithm working on n items of data, *d(n)*, the *data complexity* function, describes the amount of space in memory that an algorithm requires to run. In general, there is a trade-off between time complexity and time complexity. *The data reduction function, r(n)* describes the amount of data that an algorithm turns out. For example, an algorithm to shrink the size of an image by half, that is, reduce the height and the width of an image by a factor of ½, will have a data complexity of $r(n) = n/4$. A component's time complexity, data complexity and data reduction function fully describe how an algorithm will work in time - that is, how fast it will perform and how much information it will produce.

Components are represented diagramatically as a rectangular box, with arrows indicating input and output. This graphical notation for a component is shown in Figure 1.

Complexity analysis solutions are given in terms of fundamental units of operation, that is, operations that take a fixed amount of time to complete. In machine vision algorithms, the fundamental units of operation may in fact be complex vision operations themselves, and so simple time complexity analysis breaks down. We attack this problem by building hierarchies of components, where each component represents a complex machine vision operation, performed in hardware or software.



n = number of input elements
n' = r(n) = number of output elements
$T_P = t_P(n)$ = Time taken for P to execute.

Figure 1: A component

Algorithms are often designed in a top-down fashion, and the hierarchical component-analysis approach reflects this. Hierarchical component structures are built using composition, iteration and alternation. We introduce these structures with a graphical notation and show how their performance can be analysed. The graphic notation borrows heavily from that of the Jackson Structured Programming technique [4] [5].

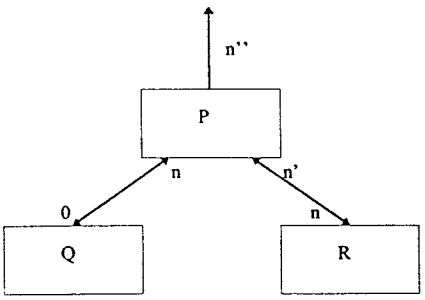


Figure 2: Composition: Component P calls Component Q and R

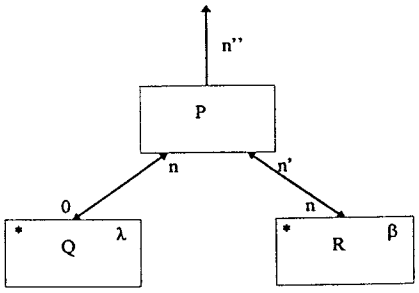


Figure 3: Component P calls Q and R a number of times.

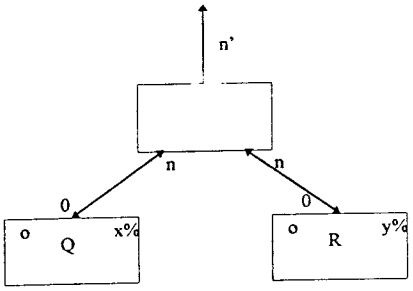
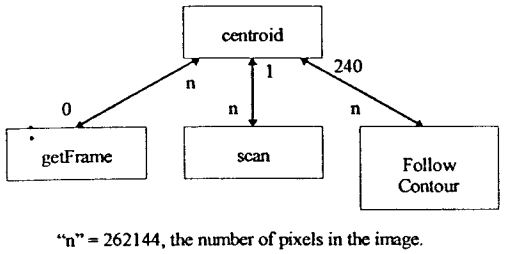


Figure 4: Alternation - P will call either Q or R.

3. Example - Frame versus Random Access.

The notation presented allows graphical description of the way components work together, using composition, iteration and alternation. We now show the notation at work, using it to describe a contour following algorithm, to locate an arbitrary blob in a scene and calculate its centroid. Our analysis shall focus on the image acquisition bottleneck of the vision system, that is, our intent is to estimate the number of pixel acquisitions that will be made and the amount of time spent acquiring them. We will purposefully omit an analysis or comparison of the image processing and analysis time for the two solutions, as we assume that the processing hardware shall be the same for both solutions.

The problem is find the centroid of a bright object against a dark background - a sample image is shown below. The image dimensions are 512 x 512 pixels, that is, the image is a data-set of 262,144 elements. The blob is roughly circular in shape, with a diameter of about 75 pixels. A sample input image is shown in Figure 5



"n" = 262144, the number of pixels in the image.

Figure 7: Component diagram for frame-acquisition based solution.

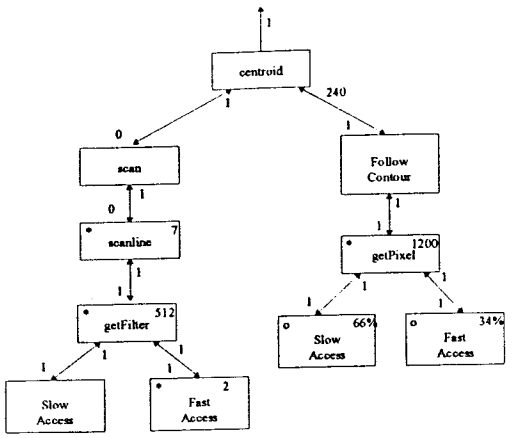


Figure 8: Component diagram for pixel-acquisition based solution.

3.3 Comparison of the two solutions

Clearly we have shown that using this algorithm with a random access sensor reduces the total acquisition time by a factor of 4, that is, from 40ms to 10ms. A simulation of the algorithm was performed, where accesses to image data where monitored and logged as if made to a random access sensor. As can be seen from the table below, the figures found in the simulation of the algorithm closely match those found in our simulation.

Thus, the rigorous component based analysis of the algorithm provides accurate results. More importantly, it places the algorithm in a framework where it can be evaluated over any number of different architectures.

Value	Predicted	Found by simulation
Accesses in Scan	10652	8760
- Fast	66.6%	66.3% (5810)
- Slow	33.3%	33.6% (2950)
Accesses in Follow Contour	1200	1142
- Fast	33.3%	34% (387)
- Slow	66.6%	66% (755)
Size of Binary Chain Code	240	207
Total Acquisition Time	10.2ms	8.6ms

Table 3: Comparison of theoretical vs. actual results gathered from a simulation of the algorithm.

4. Conclusion

We have presented a top-down technique for analysing and estimating the performance of machine vision algorithms, and shown it to give excellent results in comparison with figures obtained from experience with real data. The technique is of great import to the field of real-time industrial machine vision, where the ultimate aim is to reduce acquisition time. The framework allows us to identify solutions built with components of low computational complexity working together using minimum pixel acquisitions.

References

1 W. Forstner. 10 Pros and Cons Against Performance Characterisation of Vision Algorithms. In H. I. Christensen, W. Forstner & C. B. Madsen, *Proceedings for the Workshop on Performance Characteristics of Vision Algorithms 1996*, Cambridge, UK.

2 David Harel, *Algorithmics, The Spirit of Computing* Second Edition, Addison Wesley, 1992.

3 A. Trenaman, D. Barry, D. Vernon, *An Analysis of Strategies to Reduce Computational Complexity and Processing Time in Industrial Optical Data Processing and Analysis*. Presented at OEPE '96, Optical Engineering Society of Ireland.

4 Alistair Sutcliffe, *Jackson System Development*, Prentice Hall, UK, 1988.

5 M. J. King & J. P. Pardoe, *Program Design Using JSP - A practical introduction*. Basingstoke: Macmillan, 1992.

6 J.M. Wilf, Chain Code, *Robotics Age*, Vol. 3, No. 2.

7 D. Vernon, 1991. *Machine Vision - Automated Visual Inspection and Robot Vision*, Cambridge University Press, pp. 22-23.