AN INTERMEDIATE LEVEL VISION SYSTEM FOR POPULATED PRINTED CIRCUIT BOARD INSPECTION

Ian Wallace
Department of Information Systems
Kingston Polytechnic

Penrhyn Road, Kingston upon Thames
Surrey. England

ABSTRACT

This paper outlines work on the development of a system for producing descriptions of fully populated Printed Circuit Boards, and describes the hardware and software components which supports the first level of this system. Inspection is defined as a two stage process, description followed by interpretation. Description avoids the use of domain specific information and models. The system is used to test out two hypothesis regarding delaying the use of object models until the interpretation phase.

INTRODUCTION

Existing industrial inspection systems tend to use unsophisticated image processing techniques and dedicated hardware. Despite this however they are increasingly important in industry, and ensure that there will be a market for the 'second generation' inspection systems using AI techniques and specialised architectures. As flexible manufacturing and CIM systems mature there will be a need for more powerful, intelligent, and flexible inspection systems.

The Advanced Cim Inspection Device (ACID) is a three stage intermediate level vision, designed to test the model free hypothesis. The industrial inspection problem is reformulated as a two stage process, production of a general description, followed by domain dependent interpretation. An examination of existing image understanding systems leads to the conclusion that object models are being used too early in processing, resulting in systems which are overly complex. The strong form of the model free hypothesis states that a general scene description can be produced without recourse to domain specific object models, using general mechanisms based on recently discovered physiological mechanisms [1]. A week form of the hypothesis allows that it may be necessary to use domain specific features, but not object models. Domain specific features are those which might be extracted from any image, but which would only be expected to yield useful information for particular image classes.

ACID - SYSTEM DESIGN

ACID's first stage, LAPSYS, is based around a Linear Array Processor (LAP) developed at the National Physical Laboratory (NPL) Teddington England. This system is responsible for low level image processing operations, achieved through algorithms written in Picture Processing Language (PPL). The Second stage, SUNSYS, is based around a Sun 3 workstation. A suite of program modules were developed to implement a simple fourth generation language for intermediate level image processing. This language enabled rapid prototying of different approaches to the conversion of iconic image data to a semi symbolic form. The output from the second stage is a set of Prolog statements, describing the PCB in terms of lines, edges, pins, and writing. The third and final stage, called PROSYS, is a production rule based system. The Prolog statements produced by SUNSYS are woven into a semantic net, where they are manipulated by the production rules. At any time during processing the semantic net will contain the current state of the description. When no more rules are applicable the description is output as text or graphics. The ACID system is not concerned with the subsequent interpretation of this description, it does not take processing further than the intermediate level. It is envisaged that a domain specific interpreter will be developed to bolt on to the top of ACID. Different interpreters would enable the system to operate in different domains, each

one drawing on relevant domain specific information and models.

LAPSYS

Recently it has been noted that the efficiency of bi-dimensional array processors is not as high as might be hoped. Two problems exist. When the image size is greater than the processor array size, there are considerable overheads involved in allocating processors to pixels. There is also a general overhead involved due to the communications between processors. This overhead can be greatly reduced by adopting a one dimensional linear processor configuration in preference to the traditional bi-dimensional configuration. A linear architecture offers a number of advantages :

Cost is lower - £5000 for a 256 element LAP

Lower communications overheads- better utilization of processors

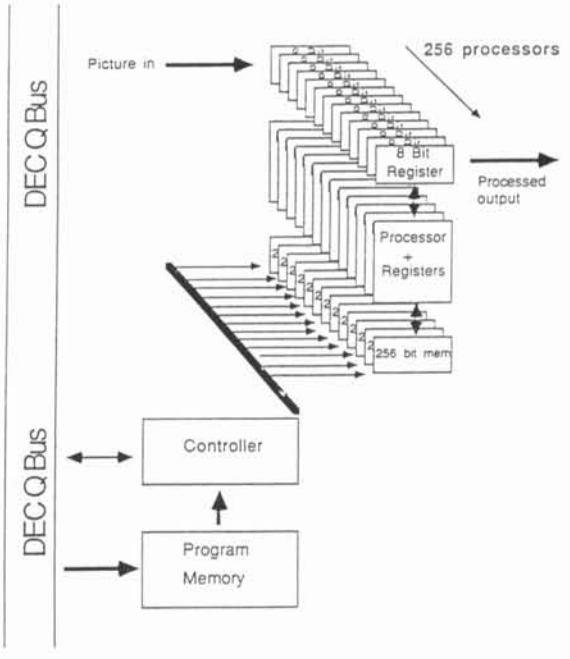
Easier to allocate processors to pixels. A line of pixels can be processed as a number of sections if there are insufficient processors available.

LAPSYS consists of three major hardware components

PDP 11/23 mini computer
LAP array processor
Framestore & video I/O

The LAP was developed at the National Physical Laboratory, Teddington Middlesex, during 1982 by Dr Peirs Plummer. It had been realised that a major obstacle to the widespread practical application of image processing in the fields of industrial inspection and robotics was the lack of any hardware fast enough to perform real time processing and which was inexpensive (less than five thousand pounds) and flexible.
The LAP is able to perform complex image-processing operations on both grey-scale and binary images. It is particularly suited to processing the output from a linear photo diode array camera scanning a conveyor belt in applications such as automated visual inspection. For such applications it is claimed to be about two orders of magnitude faster than a conventional microprocessor.

The LAP is able to process about 128 million instructions per second (mips) and consists of a linear array of processing elements, one for each pixel across the width of the image. Each column contains a fast single-bit Boolean processing element, 16 single bit working registers, and 256 bits of random access memory (Figure 1). In addition each column has 8/16 bit input/output register. Every column has interconnections to its two adjacent columns and thus information can be transferred horizontally between columns as well as vertically within a column. The LAP can be built up of blocks of 64 such columns to give between 256 and 2048 elements. In use all columns operate in parallel under the control of a master control unit. This executes microcode instructions defining the required operations, which are held in a program memory with a capacity of 16,384 instructions, each 24 bits long. The controller sends one instruction at a time to each processor simultaneously. For a particular application the microcode is down-loaded from the PDP11 host processor.



Linear Array Processoror

Figure 1

Rows of pixels are processed in parallel, while each row is loaded and unloaded serially. This is done by chaining the 8 bit input/output registers associated with each column into a single shift register, thus for a 256 elementary array processor the shift register would be 2048

(256x8) bits long. This data transfer occurs at a rate of 250K bytes per second. The loading of one row occurs whilst another row is being processed thus avoiding a potential bottleneck at the serial stage. Most commonly used image processing operators using 3x3 or larger windows can be efficiently implemented for any precision between 1 and 16 bits per pixel. Since processing is bit serial the fewer bits per pixel, the faster the processing.

Picture Processing Language [2]

The basic instruction level of the LAP PE's is a 25 bit microcode instruction. However, to facilitate rapid development of algorithms a compiler has been written to translate from the high level Picture Processing Language (PPL) into microcode. PPL allows convenient expression of picture processing operators. PPL code is entered either from the keyboard or from disk, allowing interactive programme development. When used in interpreter mode the results of an operation can be seen immediately on the video display. PPL has the usual language constructs such as logical and arithmetic operators,loops and conditional tests. What makes PPL useful are its pixel addressing mechanisms. Many image processing routines involve a pixel and its eight immediate neighbors,for example edge detectors can be written using a 3x3 window. PPL allows pixels to be addressed using the following numbering convention :

```
[ 4 ] [ 3 ] [ 2 ]
[ 5 ] [ 0 ] [ 1 ]
[ 6 ] [ 7 ] [ 8 ]
```

Any pixel in this window can be addressed by reference to its number in square brackets, for example the statement

```
apply not [0] end
```

will produce a complement of the current picture. This introduces the main structure in PPL

```
apply.....end
```

Apply evaluates the statement that follows it until it reaches an 'end' and replaces the value of the current pixel [0] by the result of that evaluation. There is no need for the programmer to concern him or her self with the usual loops necessary in other languages to apply an operation to each pixel in turn , apply works on every pixel in the whole image, thus greatly simplifying the code. For example :

```
apply
if [0] > THRESHOLD then white else

black
end
{simple threshold}
```

Feature Extraction

Four image features are extracted from PCBs, two general features and two domain specific features. The general features are 1) outlines of major regions and 2) line detail. These two features are used to test the strong form of the model free hypothesis. In order to test a weaker form of the hypothesis two additional features are extracted. These are 1) passing the image through a very high threshold filter, so only the very brightest points are detected and 2) a texture based segmentation which picks up lettering. The bright points carry information about the location of highly reflective objects such as pins on IC packages. It was realised that the tops of many IC packages can be characterised by texture. This texture is produced by the white writing on the packages which designates manufacturer and part number. The following algorithm was implemented and was successful in extracting information about the locations of IC packages.

Threshold the image to set writing to white and package to

black.

Pass a long thin window over the whole image, and for each position in the image count the number of black to white, and white to black transitions that occur within the window.

If the count is high then there is good evidence for labeling this position as being writing, otherwise it is background.

CONCLUSIONS

The first two stages of ACID have been completed, and the results transferred to PROSYS. The first prototype of this Prolog system has been successful in interpreting the statements written by SUNSYS, and has produced partial descriptions of it in terms of shapes, locations, and relationships such as is_contained_by and is_part_of. This supports the model free hypothesis, ie it is possible to produce general descriptions of PCB images, without the need to use domain specific knowledge or object models at either the iconic or intermediate levels of processing.

ACKNOWLEDGEMENT

I would like to express my gratitude to my supervisor Professor G. Rzevski (Director Kingston CIM Centre) for his comments and continued encouragement.

REFERENCES

[1] Walters D, Selection of image primatives for general-purpose visual processing. CVGIP (37) 261-298. 1987

[2] Plummer P, The Picture Processing language compiler manual. National Physical Laboratory, Teddington. 1982