'A HARDWARE IMPLEMENTATION OF A RELAXATION ALGORITHM TO SEGMENT IMAGES'

by

ANTONIO G. LODA' , HEGGERE S. RANGANATH
2003 Fulton Dr., Huntsville , Computer Science Department
University of Alabama in
Huntsville

ABSTRACT

Relaxation labelling is a mathematical technique frequently applied in image processing algorithms. In particular, it is extensively used for the purpose of segmenting images. The paper presents a hardware implementation of a segmentation algorithm, for images that consist of two regions, based on relaxation labelling. The algorithm determines, for each pixel, the probability that it should be labelled as belonging to a particular region, for all regions in the image. The label probabilities ('labellings') of every pixel are iteratively updated, based on those of the pixel's neighbors, until they converge. The pixel is then assigned to the region correspondent to the maximum label probability. The system consists of a control unit and of a pipeline of segmentation stages. Each segmentation stage emulates in the hardware an iteration of the relaxation algorithm. The design of the segmentation stage is based on commercially available digital signal processing integrated circuits. Multiple iterations are accomplished by stringing stages together or by looping the output of a stage, or string of stages, to its input. The system interfaces with a generic host computer. Given the modularity of the architecture, performance can be enhanced by merely adding segmentation stages. The processing speed is near real time.

I. Introduction.

Image analysis is concerned with the description of images and the recognition of objects. Traditionally, image analysis has been applied extensively in the space exploration field, for example in

the analysis of pictures taken from spacecrafts or satellites. The first and foremost step in an image analysis algorith is segmentation. Segmentation consists of processing an image into meaningful regions. Therefore the success of image analysis depends largely on the accuracy of the segmentation algorithm. A widely accepted segmentation technique is called 'relaxation labelling'. A full description of this algorithm is deferred to section II. Among its virtues, relaxation lends itself very well to hardware implementation.

This paper describes the architecture of a blob detector, a system that segments images characterized by two regions. The system is a hardware implementation of the relaxation labelling algorithm , in its classical probabilistic form. In section II, the relaxation algorithm is described in detail. In section III, the architecture of the system is defined. The conclusions are drawn in section IV.

II . Classical probabilistic relaxation labelling.

The following discussion is to familiarize the reader with a segmentation algorithm called 'classic probabilistic relaxation labelling' [1]. This segmentation technique is here described in a step by step fashion. The discussion sets the background for the definition of the hardware implementation of the algorithm, in section III.

II.1. The algorithm.

Let $D=\{px(i,j),i=1,..,n1,j=1,..,n2\}$ be a digital image consisting of m regions $c_1,..,c_m$. Let g_k be the average gray level for region k. What follows is a step by step procedure to segment the image:

STEP 1: For every pixel px(i,j) a set of probabilities $P^{(0)}(i,j,1),..,P^{(0)}(i,j,m)$, or 'probability function' $P^{(0)}$, is computed as follows:

$$P^{(0)}(i,j,k)=(1/abs(px(i,j)-g_k)+\varepsilon)/\sum_{k=1}^{m}(1/abs(px(i,j)-g_k)+\varepsilon) \qquad k=1,..m \quad \{2.1.1\}$$

where $P^{(0)}(i,j,k)$ represents the probability that pixel px(i,j) belongs to region c_k, ε is non zero constant and

$$\sum_{k=1}^{m} P^{(0)}(i,j,k)=1 \qquad \{2.1.2\}$$

The following steps are iterated. The formulas are generalized for iteration r.

STEP 2 : For every pixel px(i,j), a set of 'compatibility coefficients' $c^{(r)}(i,j,k,i1,j1,k1)$, or 'compatibility function' $c^{(r)}$, is defined, where i1=1,..,n1, j1=1,..,n2, k,k1=1,..,m, and (i1,j1)=(i,j). $c^{(r)}(i,j,k,i1,j1,k1)$ represents the compatibility between the assignment of px(i,j) to region c_k and that of p(i1,j1) to c_{k1}. The compatibility function is basically a heuristic evaluation of the validity of a pixel's region assignment (labelling), on the basis of the labellings of the rest of the image pixels.

Two commonly used assumptions in the formulation of the compatibility function are as follows:

1. Only the neighborhood pixels are relevant to the classification of the pixel under scrutiny. Therefore

$$c^{(r)}(i,j,l,i1,j1,k1) = 0 \quad \text{if } i-1 < i1 < i+1 \text{ and } j-1 < j1 < j+1$$
$$= 0 \quad \text{otherwise} \qquad \{2.1.3\}$$

2. The compatibility function is 'space invariant' that is, for every integer ii, jj such that px(i+ii,j+jj) and px(i1+ii,j1+jj) belong to D,

$$c^{(r)}(i,j,k,i1,j1,k1) = c^{(r)}(i+ii,j+jj,k,i1+ii,j1+jj,k1)$$
$$i=1,..,n1;\ j=1,..,n2;$$
$$i-1 < i1 < i+1, \text{ and } j-1 < j1 < j+1 \qquad \{2.1.4\}$$

Several definitions have been introduced for the compatibility function, one is given below. Let

$$C^{(r)}(k,ii,jj,k1) = c^{(r)}(i,j,k,i+ii,j+jj,k1)$$
$$i=1,..,n1,\ j=1,..,n2,\ -1 < ii < 1,\ -1 < jj < 1 \qquad \{2.1.5\}$$

Let $p1^{(r)}(k)$ represents the a priori probability of an image pixel belonging to region c_k. Let, also, $jp^{(r)}(k,ii,jj,k1)$ be the joint probability that an image pixel belongs to region c_k and its neighbor, at the orientation specified by (ii,jj), belongs to region c_{k1}.

We define

$$C^{(r)}(k,ii,jj,k1)=[\log R^{(r)}(k,ii,jj,k1)], \qquad \{2.1.6\}$$

where

$$R^{(r)}(k,ii,jj,k1)= [jp^{(r)}(k,ii,jj,k1)/(p1^{(r)}(k)*p1^{(r)}(k1))] \qquad \{2.1.7\}$$

For practical purposes, the values of the compatibility function are truncated to the interval [-1,1].

STEP 3: A set of supporting coefficients $Q^{(r)}(i,j,1),..,Q^{(r)}(i,j,m)$, or 'support function' $Q^{(r)}$, is computed as follows:

$$Q^{(r)}(i,j,k)= (1/8) \sum_{i1=i-1}^{i+1} \sum_{j1=j-1}^{j+1} \sum_{k1=1}^{m} C^{(r)}(k,i1-i,j1-j,k1)P^{(r)}(i1,j1,k1) \qquad \{2.1.8\}$$

$Q^{(r)}$ represents the contribution of the total relevant environment of pixel px(i,j) to $P^{(r)}(i,j,k)$.

STEP 4: $P^{(r)}(i,j,k)$ is updated as follows:

$$P^{(r+1)}(i,j,k) = [P^{(r)}(i,j,k)[1+Q^{(r)}(i,j,k)+\varepsilon]]/ \sum_{k=1}^{m} P^{(r)}(i,j,k)[1+Q^{(r)}(i,j,k)+\varepsilon] \qquad \{2.1.9\}$$

Each pixel px(i,j) is then assigned to the region $c_{K(r)}$, where $K^{(r)}$ is such that the probability $P^{(r)}(i,j,k)$ is maximum for $k = K^{(r)}$.

The iteration is repeated until the labellings converge. Alternatively, one can stop the algorithm after a fixed number of iterations has been executed, or according to some other termination scheme.

III. The blob detector.

The algorithm presented in the previous section is the centerpiece in the design of the blob detector. The system is intended for scientific and industrial applications. The speed required in these applications cannot be normally accomplished by general purpose computers. Computers based on special purpose architectures, such as array processors or systolic arrays, are better suited but also result in costs often not justifiable in the context of simple applications.

The system defined in this paper derives its speed from its dedicated architecture. By optimizing the design for a specific algorithm the system complexity is reduced as well. Also, the architecture is pipelined, since the promptness of the result is not as important as the system's throughput. Finally, the design achieves expandability through modularity and is intended as a peripheral to a commercial general purpose personal computer, to facilitate its use.

As a result of these design choices the blob detector is a high speed, low-cost, low-complexity system, configured as a peripheral to a personal computer.

III.1. General system architecture

The system consists of a microcontroller (MC) and of the relaxation engine (RE) (Figure 1). The microcontroller controls the synchronization of all system operations, through the system control (SCB) and i/o (SIOB) buses. It communicates with the host computer

through the host interface(HI). The latter allows the host to upload the image data, request the execution of the segmentation procedure and download the processed image. The segmentation engine is responsible for the execution of the segmentation algorithm. The input and the segmented images are received and transmitted over the system i/o bus .

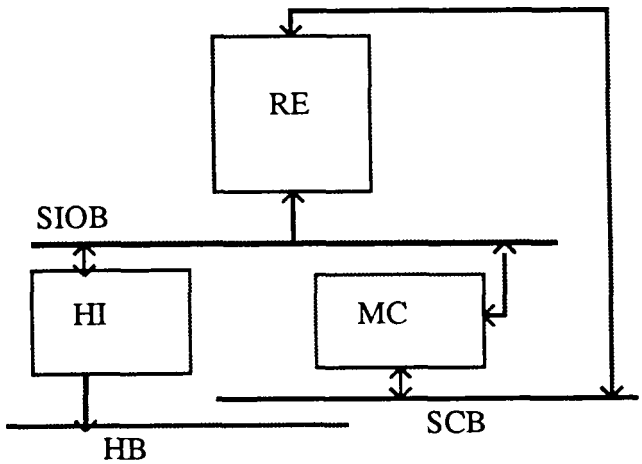

Figure 1. The system architecture.

The function of the microcontroller is to coordinate the overall system operation and the communication with the host computer. Furthermore, it is designed to have enough processing power to perform some post-segmentation simple image processing tasks, should this be required.

The architecture of the microcontroller is based on a commercial 32-bit microprocessor (MP), coupled with a math coprocessor (MCP) (Figure 2). A memory management unit (MMU) oversees the microcontroller's accesses to the memory system (MS). The memory system consists of both RAM and ROM type memories. The ROM memory is necessary for system initialization and for storing the system algorithms. The RAM memory is used for storing the segmented image and user specific algorithms dowloaded from the host

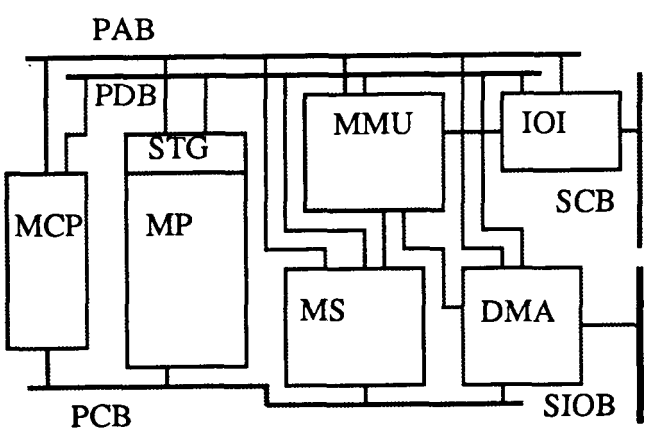

Figure 2. The microcontroller.

system. The timing of the microcontroller as well as of the entire system is generated by the system timing generator (STG).

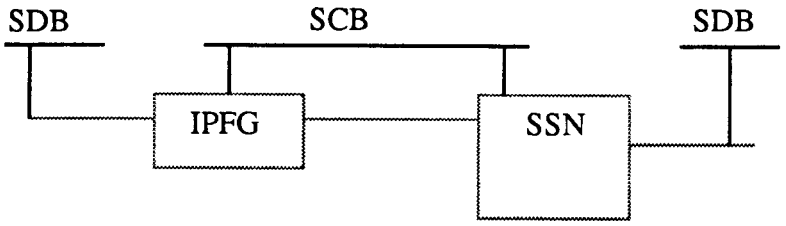

Figure 3. The relaxation engine.

The host interfce consists solely of the circuitry necessary to insure the electrical continuity between the host interface and the system and to handle the handshake protocols.

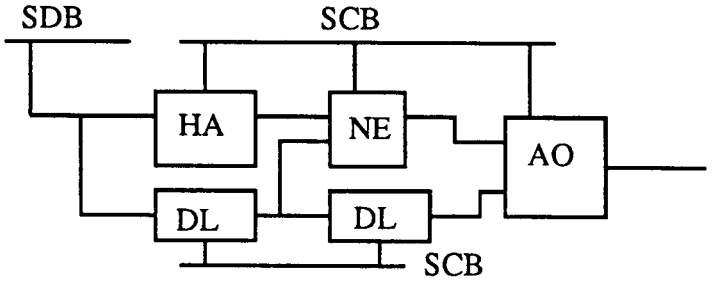

Figure 4. The initial probability function generator.

III.2. The relaxation engine

The algorithm described in section I is executed by the relaxation engine (Figure 3). The image data, arriving from the host system via the microcontroller, is initially analized to derive the initial probability function. This task is performed by the initial probability function generator (IPFG) (Figure 4). This circuitry determines the function $P(r)$ as defined in {2.1.1}. The image data is first processed by the histogram analizer (HA), which determines the average gray level values for the two regions that are to be segmented in the picture.

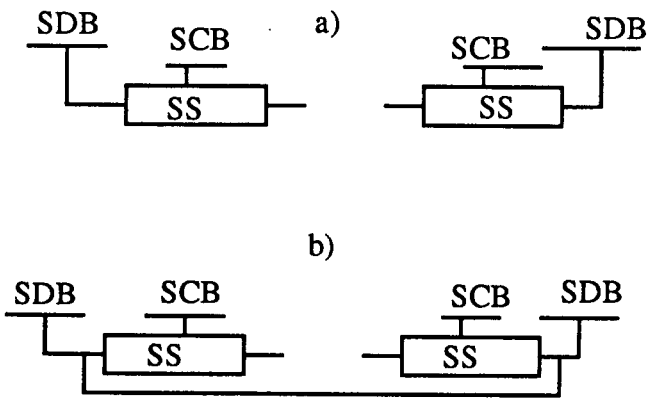

Figure 5. Segmentation stage networks:
a) chain configuration b) loop configuration.

The image data, delayed by the histogram

analyzer processing time is then passed through the neighborhood extractor (NE), a circuit that latches, for each pixel, its eight neighbors. These pixel values, together with the region average gray level values, are then fed through a series of arithmetic operators (AO) arranged to implement {2.1.1}.

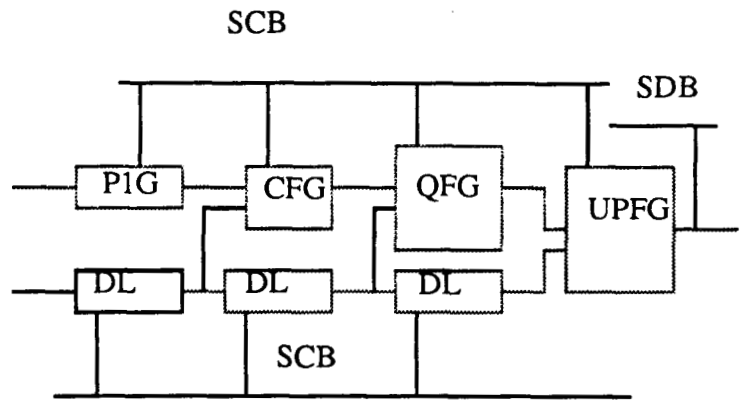

Figure 6. The segmentation stage.

Since the image only consists of two regions and therefore $P^{(r)}(i,j,0) = 1 - P^{(r)}(i,j,1)$, only $P^{(r)}(i,j,1)$ is computed, for every pixel. It is possible to execute to execute these operations in real time because components are now available that execute multiplications and divisions at the rate of one every 40ns. The neighborhood extractor (Figure 3) is based on a series of delay lines (DL), which are circuits that output at every instant the data received in input n clocks earlier, where n is the length of the line. Each delay line is implemented using shift registers. Delay lines are also used in the design to synchronize the pipeline.

The second functional block of the relaxation engine is the segmentation stage network (SSN). This subsystem iterates the probability updating scheme described in section I. It consists of a network of segmentation stages , each capable of executing one iteration of the relaxation algorithm. These stages can be modularly connected in a variety of ways. Two configurations are displayed in Figure 5. The first (Figure 5.a.) consists of four units cascaded, so that the output of a stage is the input of the next one. The second configuration (Figure 5.b.) sacrifices some of the system throughput

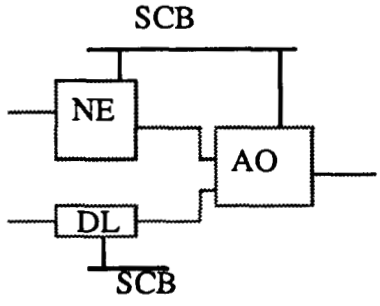

Figure 7. The compatibility function generator.

of the by iterating the algorithm by looping the output of a string of two stages to its input, as many times as is desired.

Each segmentation stage (Figure 6) is responsible for executing one iteration of the relaxation algorithm, that is, for obtaining $P^{(r)}(i,j,k)$ from $P^{(r-1)}(i,j,k)$. The first task of the updating unit is to determine the a priori probabilities p1(0) and p1(1) of the two image regions. The probability function and the a priori probabilities are then output to the compatibility coefficient generator (CFG). This circuitry determines the image compatibility function by running each pixel neighborhood through a network of arithmetic operators, arranged in a sequence, such to reproduce the calculations defined in {2.1.6} and {2.1.7} (Figure 7). Once the compatibility function is computed, it is output, together with the probability function, to the support function generator (QFG). The support function and the probability function, finally are processed by the updated probability function generator (UPFG) to produce the updated probability function. Both the support and the updated function generators are networks of arithmetic operators arranged so to perform {2.1.8} and {2.1.9}.

IV. Conclusions.

In the previous pages we presented the architecture of a blob-detecting system. The system, based on a pipelined processing scheme, allows for real time segmentation of 'blobby' images for scientific and industrial applications. The system is designed to be an inexpensive image analysis peripheral to a commercial personal computer. The design can be expanded, with little effort, to add the capacity to execute other image processing algorithms, characterized by the application of the same procedure on all pixels, and that operate on a neighborhood basis. Algorithms such as template matching, for the recognition of objects, fall in this cathegory. This expansion can be achieved by replacing the compatibility function circuitry with memory, which can be loaded with the desired operator, and replacing the dedicated arithmetic operators network with a programmable digital signal processor.

V. Literature.

[1] Kittler J. and Illingworth J., "Relaxation labelling algorithms - a review," *Image and vision computing,* Vol.3 No.4 (1985), pp. 206-216.