2009 IEEE International Advance Computing Conference (IACC 2009)
Patiala, India, 6–7 March 2009

A Memory Efficient Algorithm for Real Time Object Counting

Subramanian S.[1, 2] and Bhadrinarayana L. V.[1]

[1]*Electrical and Electronics Engineering Department, Birla Institute of Technology and Science, Pilani, India*
[2] *Mechanical Engineering Department, Birla Institute of Technology and Science, Pilani, India*

Abstract

Object Counting is a challenging problem with different solutions based on the available computing power and the nature of data to be processed. Memory efficiency, simplicity and speed are very important requisites for algorithms to be used in modern day systems which include distributed systems and wireless sensor networks where it is extremely advantageous to do basic preprocessing of data from the sensors in the nodes itself. Reduced computing power available in the nodes poses a challenge and this can be overcome by the use of algorithms with simple steps. We present an algorithm to monitor continuity, count objects and measure parameters such as area by isolating patterns and objects from data based on simple computations. Memory efficiency, speed and flexibility of the proposed algorithm have been discussed. It has many broad applications that include counting objects on a conveyer, monitoring people in a traffic signal, pattern isolation from multicolored images and monitoring continuity in real time image data from sensors. We describe the experimental setup used to implement this algorithm.

Keywords: *Counting Algorithm, Object Counting, Image Separation, Pattern Isolation*

1. Introduction

Object counting is a very common task which many systems perform. There is a need to count objects from still images or from streaming data where information arrives in packets. It becomes a challenging problem when different objects are not easily distinguishable. When processing real time data, the information about a complete object may not be available and waiting for the complete data to arrive will lead to a waste of time and memory especially when they are implemented in systems with limited resources such as nodes of a mobile ad-hoc network. Algorithms for images are relatively simpler with lesser constraints. Specific applications of counting algorithms demand different approaches and different results and hence this is a problem that has been analyzed and studied in detail. For tree counting, the color spectrum in images has been used for segmentation and either discrete object or biomass approach is used for counting [1]. Sizes of different objects are assumed to be similar for using the biomass approach since the discrete object approach will lead to wrong results when two trees are touching. Skeleton of the objects in images has also been used for separation before counting [2]. In [3], the background image has been used as a reference for counting the number of moving objects. Each of these methods is computationally demanding and is more applicable for the specific applications they have been designed for. Counting identical objects has been done using High Level Auto Correlation (HLAC) where the similarity between objects is used for fast real time processing and counting [4]. Similarity between objects to be counted has also been exploited to count multicolored objects using Focused Color Intersection [5]. For applications involving robots, genetic algorithms and learning algorithms have also been used in shape recognition from the outputs of a 1-D sensor array [6]. In this present work we discuss our algorithm in detail and how it can be used even when computing resources of the system are comparable to that of a RISC microcontroller or microprocessor. The steps involved make this algorithm very close to hardware implemented algorithm and has very good pipelining efficiency. Algorithm is independent of the nature and relationship amongst different objects. The proposed algorithm can also handle special configurations such as objects distinctly occurring inside other objects. The algorithm will be discussed with respect to binary data or images where pixels have only two possible states. This work is further subdivided into three more sections. In section 2 we describe the algorithm and all the steps involved. In section 3, we discuss some salient features of the algorithm and some sample cases. The experimental setup is explained in section 4.

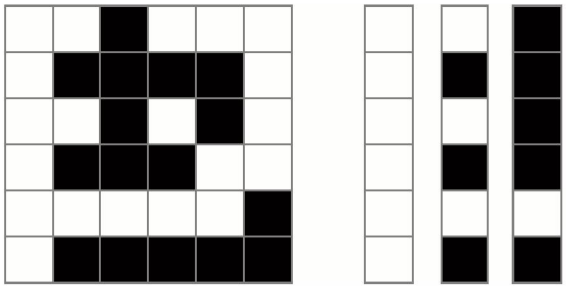


Figure 1. Small binary image and its first, second and third columns

2. Algorithm

2.1. Principle

The whole image is processed column by column. This method of processing makes it directly applicable to streaming data inputs as well for real time

978-1-4244-2928-8/09/$25.00 ©

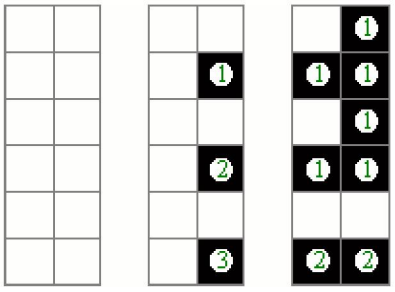


Figure 2. The instantaneous object numbers during consecutive iterations of the algorithm

processing. This algorithm works by comparing the real time input with the one set of previous values stored in the memory. The currently available elements are tagged and numbered by monitoring the continuity between these two sets of values and by checking for completed objects, the object count is increased.

A sample input is shown in figure 1 where the input is a binary image and the square blocks represent pixels of the image. An instantaneous count of currently active objects is maintained by algorithm by analyzing the current inputs with the previous inputs i.e., initially the first column arrives where there are no objects. When the second column arrives, the first column alone is available in the memory. From the two columns, the instantaneous number of object is found to be 3. During the start of the next iteration of the algorithm, the modified second column with the object number tag is stored in place of the first column. When the third column arrives, the instantaneous object count is reduced to 2 after analyzing the previous object tags and the current data. The exact algorithm is discussed later. The first three iterations of the algorithm with the sample object in fig. 1 as input are depicted in fig. 2. All properties and parameters of specific objects are linked according to the instantaneous tags.

2.2 Memory Requirements

For a system having S sensors in a 1-D array, the number of pixels in one dimension of the image is S. Therefore, the maximum number of objects possible at a given instant, M, in one set of values of the 1-D array is given by

$$M = \left\{ \frac{S+1}{2} \right\} \tag{1}$$

where {} denotes the greatest integer function. The minimum number of bits required to represent these M unique objects is the size of the tag (all bits zero in tag represents absence of an object and object tag numbers start from 1) and is given by

$$N = \left\{ \log_2 \left(\left\{ \frac{S+1}{2} \right\} + 1 \right) \right\} \tag{2}$$

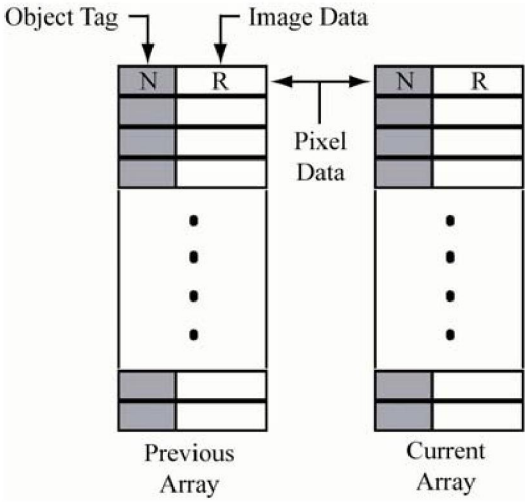


Figure 3. Memory Map depicting genotype of each memory location – Object tag and Image Data

Each pixel has a tag of N bits to uniquely represent the object number. Let the number of data bits in each pixel be R. The total amount of memory required for each column, B (bits), is obtained as

$$B = S\left(\left\{ \log_2 \left(\left\{ \frac{S+1}{2} \right\} + 1 \right) \right\} + R \right) \tag{3}$$

The total memory required, in bits, for processing any arbitrarily long image or streaming media with N rows is $2B$. The memory map of the memory resources required is shown in fig. 3 where the object tags on the processed or previous array (Array 1) store the object numbers for that particular location at the beginning of the iteration and the object tags in the current array (Array 2) store the finally updated tags. Though the tags in the same row of the two arrays may hold different numbers, they represent the same object. In the case of a binary image, the data part of each pixel is only 1 bit.

2.3 Procedure

The complete procedure is described for a binary image with 1bit data (R = 1). The algorithm can be used for multi-bit data by using threshold values. The algorithm treats one row of the two arrays together at a time. Variable 'PNO' stores the modified object tag of the previously processed row i.e. the object tag in the previous row of Array 2. Variable 'NOB' stores the number to be assigned to a new object.

Step 1. *Transfer the results of previous iteration to Array 1 from Array 2. Clear PNO. Set NOB=1*

Step 2. *The data bits of the first row of the two arrays are 'OR 'ed*

Step 3. *An Initial tag is assigned using the following conditions:*
***If** 'OR' value is 0 & PNO=0,*

Then *tag (Array 2) = 0*
Else If *'OR' value is 0 & PNO ≠ 0,*
Then *tag (Array 2) = 0, Increment NOB*
Else If *'OR' value is 1 & PNO = 0,*
Then *tag (Array 2) = NOB*
Else If *'OR' value is 1 & PNO ≠ 0,*
Then *tag (Array 2) = PNO*

Step 4. *PNO = tag(Array 2) and return to step 3 till all the rows are completed.*

Step 5. *Tag(Array 2) of any row is modified to a lesser value that appears in the tag of any other row of Array 2 if they share the same tag in Array1 or appear adjacently.*

Step 6. *Completed objects are detected by checking for objects that have appeared only in Array 1.*

Step 7. *Properties that are being monitored, such as area, length are updated by matching the tags*

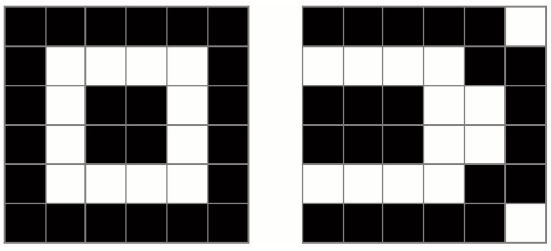
After the completion of step 7, the whole procedure is repeated after the next column of data from the image is retrieved or after the new sensor outputs have arrived in the case of real time processing.

3. Salient Features

For streaming data, a frequency domain or spectrum analysis demands more memory and is time consuming where as this algorithm treats single images as well as streaming data equally. Objects placed in any configuration, given that all different objects do not touch each other, can be identified. Special sample cases are shown in figure 4. In fig. 4(a), one object appears inside another closed object and in fig. 4(b), two long objects merge and partially enwrap a smaller object in between the two arms. The ability of the algorithm to identify the two objects in such cases makes it an indispensable tool for simple processing. One important feature is the availability of intermediate results at the end of each iteration and the need for only one traversal of the image for completion of the algorithm.

4. Experimental Setup

The algorithm is implemented on an 8-bit RISC microcontroller, PIC16F877, using assembly language



(a) (b)

Figure 4. Some special configurations of objects

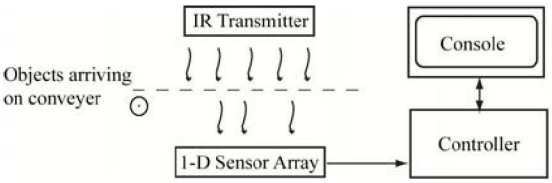


Figure 5. Schematic of the experimental setup

programming to ensure higher efficiency and test the feasibility of implementing the algorithm using basic computational processes and to verify its ability to perform in real time. A 4 MHz crystal is used as the clock. A schematic of the implemented setup is shown in fig. 5. IR LEDs are used as transmitters and the object are placed on a moving conveyer. The conveyor is driven using a DC motor. The direction of motion of the conveyer is perpendicular to the plane of paper and moving outward. This is shown as a dot enclosed by a circle in fig. 5. The conveyer is made of a material transparent to IR rays or a series of belts is used ensuring a gap between the transmitter and receiver. When an object blocks the path of IR rays traveling from IR LED to the IR sensor, the output is different from its steady state value. Thus digital outputs are obtained from a simple conditioning circuitry which is connected to a 1-D array of 16 IR diodes acting as receivers. These 16 digital outputs are connected to the input ports of the microcontroller and are used in the computation as per the set sampling rate. To demonstrate the measurement of parameters, the areas of the finished objects are displayed using 4 seven segment display as soon as the particular object is completed. The total number of objects that have passed through the conveyor is displayed using 2 seven segment displays. The area is calculated from the linear velocity of the conveyor, v (m/s), sampling rate f (samples/s), distance between two sensors, d (m), and the number of pixels in the particular object , C, using the following relation:

$$Area = \left[\frac{vd}{f}\right]C \qquad (4)$$

The term within brackets is the equivalent area corresponding to one pixel or one sensor.

The maximum number of objects that can occur simultaneously (assuming that objects are at least as big as one pixel) is 8 since only 16 sensors were used. Therefore using equation (2), the number of tag bits required is found as 4 (tag numbers: 0001-1000) and only one bit of information is available from the sensor. Therefore, one 8 bit register is sufficient to address memory requirements of one pixel and hence 32 registers are used for the algorithm and 16 registers are used for storing the area of 8 objects. For the implemented system, N=4 and R=1. The sensors are placed with a uniform spacing , d = 1cm between them. The mechanical design features of the conveyor are important but have been ignored in the present work.

5. Conclusion

In the current work, an algorithm to count clearly separated objects, positioned in any complex orientation, has been described. It works on a principle which is heuristic and can be used for images and real time data. The memory requirements have been computed and reduced need for computing resources has been highlighted. This method exhibits a flawless performance and is a very appropriate preprocessing tool for images and other data especially in nodes of sensor networks and ad-hoc systems where there is a shortage of most resources such as memory, power, and computing capacity.

This algorithm can be used to monitor continuity in objects and with further improvements has widespread applications like checking size of objects on a conveyor, simplification of mazes, character isolation and recognition etc. Future development of this algorithm for 3D arrays can lead to new techniques in areas such as Image cryptography.

References

[1] A. Sokkarie and J. Osborne, "Object Counting and Sizing", *proc. of IEEE southeastcon '94*, April 1994, pp. 380-382

[2] H. Sossa, G. Guzman, O. Pogrebnyak and F. Cuevas, "Object Counting without Conglomerate Separation", p*roc. of 4th Mexican Int. Conf. on Computer Science (ENC' 03)* , Sept 2003, pp. 216-220

[3] C. Pornpanomchai, F. Stheitstheienchai and S. Rattanachuen, "Object Detection and Counting System", p*roc. of 2008 Congress on Image and Signal Processing(CISP '08),* May 2008, vol. 2, pp. 61-65

[4] T. Kobayashi, T. Hosaka, S. Mimura, T. Hayashi and N. Otsu, "HLAC Approach to Automatic Object Counting", *proc. of ECSIS Symposium on Bio-inspired Learning and Intelligent Systems for Security (BLISS '08),* Aug 2008, pp. 40-45

[5] V. V. Vinod and H. Murase, "Counting Multi-Colored Objects using Active Search", p*roc. of International Conference on Information, Communications and Signal Processing (ICICS '97),* Singapore, Sept 1997, vol. 1, pp. 185-189

[6] K. Ohtani and M. Baba, "A New Method for Shape Recognition of a Pillar-like Object Using 1-D Ultrasonic Sensor Array and GA", p*roc. of 41st SICE annual conference (SICE '02)*, Osaka, Aug 2002, vol. 2, pp. 1275-1279