Real-Time Imaging **6**, 241–249 (2000)
doi:10.1006/rtim.1999.0190, available online at http://www.idealibrary.com on IDEAL®

A Parallel Pipeline Based Multiprocessor System For Real-Time Measurement of Road Traffic Parameters

Real-time measurement and analysis of road traffic flow parameters such as volume, speed and queue are increasingly required for traffic control and management. Image processing is considered as an attractive and flexible technique for automatic analysis of road traffic scenes for the measurement and data collection of road traffic parameters. In this paper, the authors describe a novel image processing based approach for analysis of road traffic scenes. Combined background differencing and edge detection techniques are used to detect vehicles and measure various traffic parameters such as vehicle count and the queue length. A *RISC based multiprocessor system* was designed to enable real-time execution of the authors algorithm. The multiprocessor system has nine processing modules connected in a parallel pipeline fashion. Results shows that the authors multiprocessor system is able to provide measurement of traffic parameters in real-time. Results are presented for *real tests* of our system by analysing traffic scenes on the highways of Singapore.

M.Y. Siyal[1], M. Fathi[1] and M. Atiquzzaman[2,3]

[1]*School of EEE, Nanyang Technological University, Nanyang Avenue, Singapore 639798*
Tel: +65 790-4464, Fax: +65 792-0415
[2]*Department of Elect. & Comp. Eng., University of Dayton, Dayton, Ohio 45469-0226, USA*
Tel: +1937 229 3183, Fax: +1937 229 4529, E-mail: atiq@engr.udayton.edu

Introduction

Processing of road traffic images has been the subject of vigorous research for many years. It is well known that a fully automatic image processing system will have a major impact on both the collection and analysis of road traffic data [1–4]. Many techniques have been proposed to speed up the analysis of road traffic images, and some intelligent approaches have been developed to compensate for the effects of variable ambient lighting, shadows, occlusions, etc. in the road traffic images. Many of these algorithms are either not accurate or cannot be applied to real-time, real-world scenes.

The authors have developed a number of algorithms, which can be used to analyse road traffic scenes for obtaining traffic flow parameters [1,5]. Here, the algorithms required for pre-processing of the images are filtering and segmentation. Pre-processing is followed by *background differencing* to isolate the vehicles from the images [5]. However, this technique of background differencing has problems of accurately updating the background frame and automatically selecting a suitable threshold value used in the isolation of the vehicles from the background. *Edge detection based segmentation* of traffic scene has the advantage of being

[3]Author for Correspondence.

less sensitive to variation of ambient lighting and shadows [6]. However, the combination of background differencing and edge detection technique has the advantage of being able to eliminate stationary vehicles, shadows and the road markings, and is less sensitive to variation of ambient lighting [2]. This combined vehicles detection technique was used to detect moving vehicles, and measure traffic parameters such as vehicle count and queue length.

Since real-time analysis of traffic scenes involves very fast processing and analysis of sequences of traffic images, a large volume of data to be processed in a very short period of time; thus that it is very expensive in terms of computing power requirements. Therefore, for real-time image processing tasks, various forms of *parallelism* in the processing algorithms must be exploited to reduce the computing requirements. So far, a wide range of general and special purpose computers have been developed. To achieve, real-time processing it is necessary to design special purpose multi-processor systems consisting of a number of cooperating special purpose processors, each executing a part of the task. Besides multi-processors, powerful RISC type processors have also been an attractive research topic [7–9]. Computer systems built with RISC type processors are more cost effective than multi-processor systems using special purpose processors. This system can execute, in real-time, the image processing algorithms for analysis of road traffic scenes.

The proposed multiprocessor system described here has three identical pipelines, each having three processing modules. A RISC type array processor was designed, which is used in the processing modules of the pipelines. The array-processing module was designed with high degree of parallelism to speed up the image processing operations. By combining data parallelism found in array processors into the architecture, a high degree of concurrency has been achieved. The array-processing module has nine processing elements (PE) connected in a mesh fashion. Because 3×3 masks are used in image processing algorithms for traffic analysis, image area operations with a 3×3 mask in the design of the authors processor was implemented. However, other area processing with larger masks can be easily realized through several 3×3 basic mask operations. The authors show that their system is able to provide information on traffic parameters in real-time. Results of real tests for analysing traffic scenes on the highways of Singapore are presented here.

The rest of this paper is organized as follows. First the experimental setup is described and this is followed by the authors algorithm to analyse images of traffic flow in roads. The design of the authors proposed pipelined multiprocessor system using array processors is discussed next. Accuracy of the results of measurements carried out in a real highway are then presented and are followed by the concluding remarks.

The Processing Scheme

A CCD camera installed over a road, which provided the raw traffic flow images, was used and the images were analysed in order to extract the necessary traffic flow information. The processing scheme was directed into three steps called, *vehicle detection, vehicle count and queue parameters*. One window per lane was placed across the road (see Fig. 2). Three windows were used, with each pipeline in the proposed multiprocessor system being dedicated to a window. This is because in Singapore (and in many countries), roads/express-ways have three lanes. Each pipeline has three processing modules responsible for the vehicle detection, vehicle count and queue parameter extraction operations. The multiprocessor system (to be described later) is interfaced to a host computer, which in this case is a Pentium based PC. The picture output from the camera system is fed to the system. The data acquisition and control circuit controls and manages the data. Control signals from the camera system are used to synchronize and load the image from the camera system into the multiprocessing system.

The Vehicle Detection Process

A common and simple vehicle detection technique is the *background differencing technique*. This technique is based on a pixel-by-pixel comparison of a background image of a traffic scene (without any moving vehicles) and the current frame of the scene. The technique has been successfully used by a number of researchers [3,4]. The background image needs to be periodically updated in order to account for changes in ambient lighting. In practice, the effectiveness of this method depends on the accuracy of the background updating technique and the selection of a suitable threshold value. Various researchers [1,2,4] have proposed different algorithms for updating the background. However, they don't work well under all lighting and weather conditions. Therefore, they are not suitable for real-world applications, such as traffic analysis as carried out in this study.

An alternative technique to the background differencing is based on edge detection. Edge-detection based techniques are generally more effective than the background differencing technique. In fact, the edge information of the objects remains significant despite the variation of the ambient lighting. Analysis of traffic images has proved that various surfaces and different parts and colors of a vehicle create significant edges. Even cars having the same color as the surface of the road, reflect more light and can be more easily detected.

There are two types of edge-based detection techniques: conventional gradient based edge detectors and morphological edge detectors. The conventional gradient-based edge detection operations have found wide acceptance in image processing applications. However, morphological edge detectors have shown better performance than conventional edge detectors while having a lower computational cost [10]. The authors have developed a novel morphological edge detector, consisting of separable median filtering and morphological operators, called SMED (Separable Morphological Edge Detector). The SMED has a lower computational requirement and better performance, compared to the other morphological edge-detection operators. This operator can be defined as:

$$\mathrm{SMDED} = [D(s(f)) - E(S)f))] \tag{1}$$

Where $S(f)$ is the result of applying a separable median filtering to the original image f.D( ) and E( ) are dilation and erosion operators, respectively.

The algorithm used here for vehicle detection is based on applying low pass filtering, combined background differencing and SMED edge detector on windows located across the road. Following the application of the vehicle detection operation, the pixels having values that are greater than the threshold are used to recognize vehicles. The above procedure of vehicle detection is shown in detail in Figure 1.

The size of the window has a strong effect on the performance of the whole system. The window size should be located vertical to the road direction for vehicle counting. As the width of a window is increased up to the length of vehicles, the accuracy increases but the computation time increases, which slows down the whole system On the other hand, the width of the window is dependent on the road type, as in highways (expressways, motorways, etc.) the vehicles move more rapidly than other roads. On highways the width of the windows should be more, as there is a risk of missing vehicles on some frames.

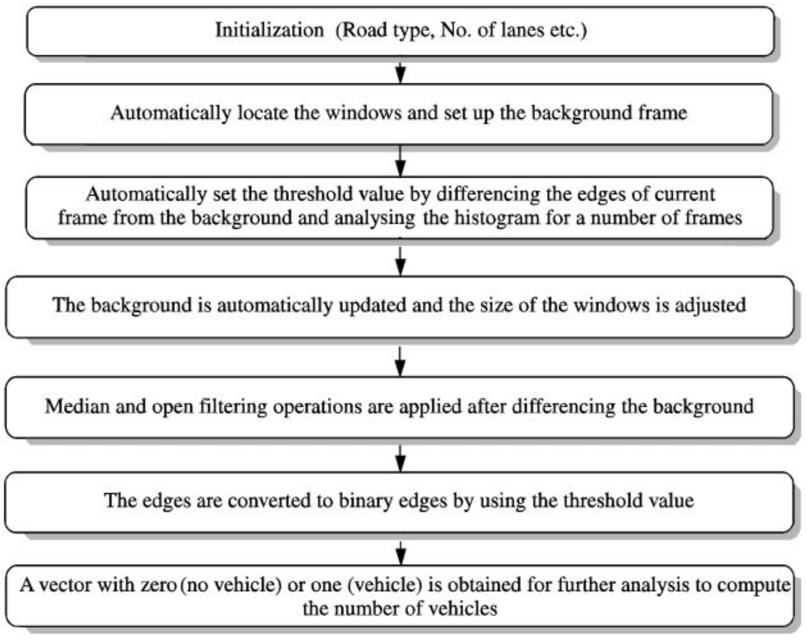
Initialization (Road type, No. of lanes etc.)
Automatically locate the windows and set up the background frame
Automatically set the threshold value by differencing the edges of current frame from the background and analysing the histogram for a number of frames
The background is automatically updated and the size of the windows is adjusted
Median and open filtering operations are applied after differencing the background
The edges are converted to binary edges by using the threshold value
A vector with zero (no vehicle) or one (vehicle) is obtained for further analysis to compute the number of vehicles

Figure 1. The flow chart of the vehicle detection algorithm.

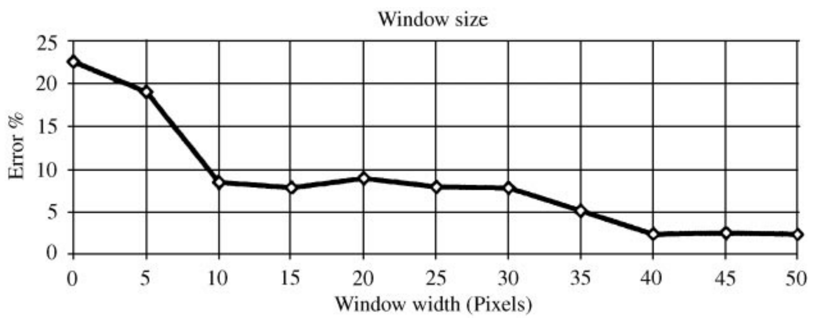

Figure 2. Graph of window width vs. percentage of errors for detecting vehicles.

The authors have conducted extensive experiments in order to determine the best window size for various situations. The percentage of errors of vehicle detection algorithm vs. window width for a road is shown in Figure 2. The graph of the computation times against window width for the same road is shown in Figure 3. As it can be seen, a compromise can be made between the accuracy needed and the processing speed required by selecting the width of the window.

In the system used here, the location of the windows, the number of lanes on the road, and the road type are manually determined by an operator as a part of the initialization procedure. The size of windows is computed and located automatically across the road. The width of the windows is automatically adjusted every 30 min to account for slight changes in traffic conditions. The width of a window is computed as follows:

$$E = L^* V \tag{2}$$

Where L is the length of each window (equal to the width of the lane) and V is a speed factor that is determined initially by the operator, and later on is continuously automatically adjusted by comparing the relative speed of vehicles. If the vehicles are moving at a higher speed, the width of the window is increased.

Following the computing of windows and their sizes, the vehicle detection program uses the co-ordinates of each window to detect vehicles. Figure 4 shows the placing of windows on the road for a typical vehicle detection operation.

Measurement of Traffic Parameters

As shown in Figure 4, in order to detect vehicles, windows of appropriate sizes are placed across each lane of the road. Here, the SMED edge detector is applied to each window, and the number of edge points is compared with a threshold value to decide, whether the window contains a vehicle. The threshold value is automatically calculated by analysing the histogram for each window.

Following the application of edge detection operation to the windows, a status vector is created. To count the number of vehicles, the status vector is analysed. In this manner, for each frame, if a vehicle is detected in a window, a '1' is stored at the status vector of the

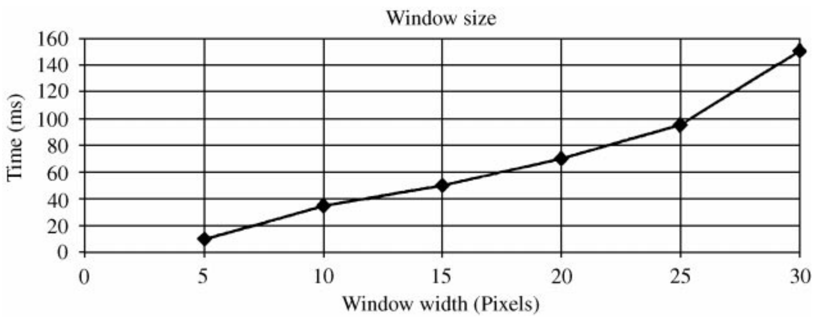

Figure 3. Graph of window width vs. processing speed for detecting vehicles.

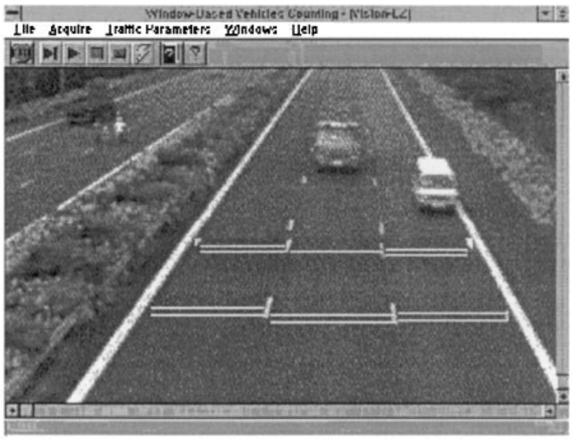
Window-Based Vehicles Counting
File Acquire Traffic Parameters Windows Help

Figure 4. The placing of windows on the road.

window; otherwise a '0' is stored at the status vector of the window. The group of 1s (ones) corresponds to a vehicle and a group of 0s (zeros) corresponds to the distance between two vehicles. An isolated 1 alone could be considered as a motor cycle or a fast moving vehicle.

The vehicle detection operation along with a motion detection operation is used for queue length measurement [10]. In this case, the window is located along the road direction. The algorithm used to detect and measure queue parameters consists of two operations, one involving motion detection and the other vehicle detection. These operations are applied to a window to detect the size of the queue. Since microcomputer systems operate sequentially, a motion detection operation is first applied. If the algorithm detects no motion, the vehicle detection operation is used to decide whether there is a queue or not. The reason for applying the motion detection operation first is that the traffic scenes that we analysed for queue detection are expected to contain vehicles. In this case, vehicle detection gives mostly the positive result, while in reality there may not be any queue at all.

The method for motion detection is based on differencing two consecutive frames and applying median filters for removing noise. The vehicle detection algorithm is only applied when the motion detection algorithm detects "no motion". To reduce the amount of data and to eliminate the effects of minor motion of the camera, the key region has to be at least a 3 pixels wide profile of the image along the road. In this method, a median filtering operation is firstly applied to the key region (profile) of each frame and then, the difference of the two profiles is compared to detect motion. When there is motion, the difference of the profiles is larger than the case when there is not motion. Therefore, the motion can be easily detected.

Design Considerations

The desire to run applications faster and the availability of cheaper and faster microprocessors and other components has resulted in much research activity in the field of computer architecture. Many computer experts have come up with various design methodologies. In order to decide the structure of the computer for the multiprocessor system, pipeline, array processor and multiprocessor structures were explored, as these techniques are widely used in today's modern computers.

Array processors no doubt are best suited to image processing based problems as the processing element's organization matches exactly with the 2-dimensional nature of images. However, as the size of the array processor increases, the cost and complexity of the system increases [11]. Since most sensors (e.g. camera) deliver data sequentially, an efficient conversion to parallel format is necessary for array processor systems. In most array processor systems, the picture is first stored in an intermediate place and then loaded into the machine; therefore, useful execution time is wasted in converting sequential data into a suitable parallel data format [12].

On the other hand pipelining takes full advantage of the sequential nature of image data. As the pictures are generated, they can be directly input to the pipeline. Once the pipeline is filled, the throughput is equal to the input data rate. Pipeline systems also cost much less than array processor systems. The advantage of the pipeline structure is that the basic speed is constant. The delay from the input to output is constant. Furthermore, common input and output devices such as video camera and monitor can be easily interfaced to a pipeline, since the pipeline contains only one common data stream [7].

Obviously, there is limit to the parallelism achieved in the pipelining. Although attractive, pipelining alone will not achieve the goal of executing image processing algorithms in real-time. This will only come with parallel processing and array processing.

Thus the authors used parallel pipeline architecture for the multiprocessor system, and designed a RISC type array processor with nine processing elements for the processing module of the pipeline. To exploit parallelism, the whole image was partitioned into sub-images (windows), which are processed by different pipelines in parallel. The rationale behind a distributed approach is that the image data can be accessed in a parallel manner by all pipelines within the system.

The Multiprocessor System

Since the architecture for this project had to reflect image processing algorithms developed for traffic analysis, which were of serial form, was adopted the pipeline architecture as the parallel processing structure for the authors system. The main operations for the multiprocessor system are:

- detect vehicles;
- count vehicles;
- calculate the queue parameters.

In order to reduce the load on the system and improve its response time, each pipeline is dedicated to process one window. Two frame buffers M1 and M2 serve as the communication interface between the camera system and the multiprocessor system, and between the multi-processor system and the host computer respectively, as shown in Figure 5. The camera system transmits the lines of a picture, serially one after another, and is

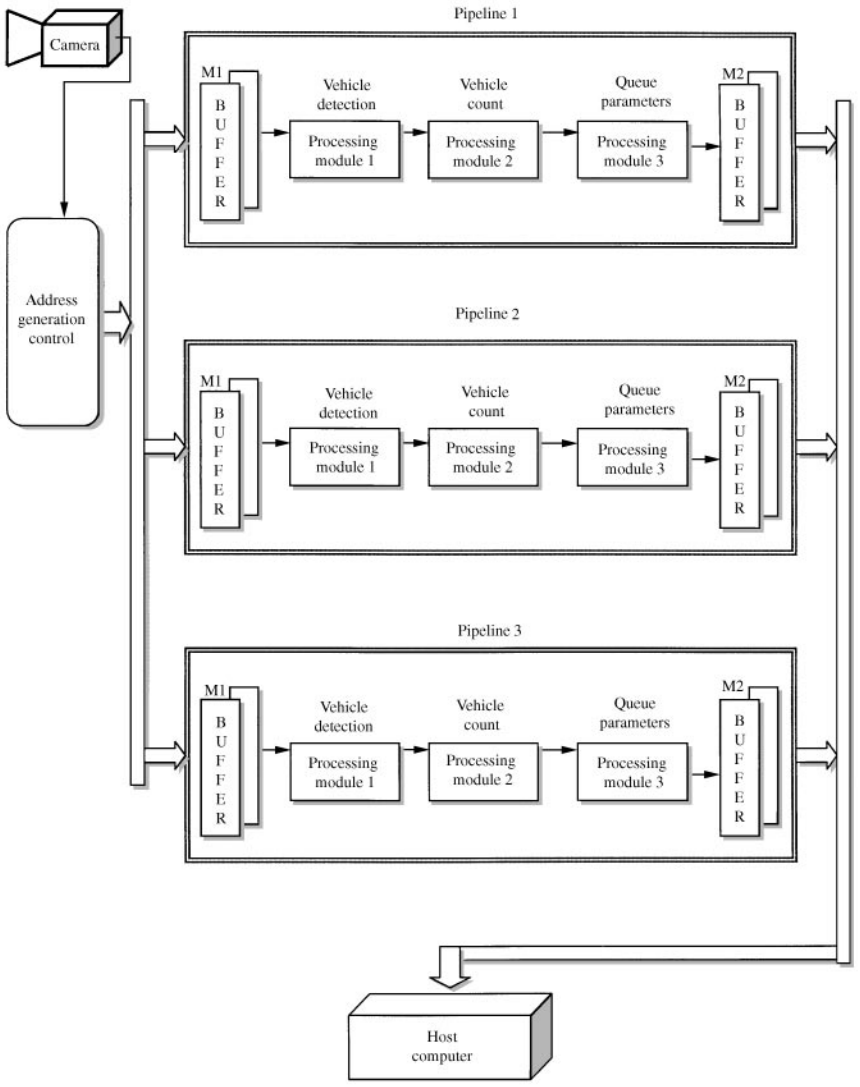

Figure 5. Block diagram of the multiprocessor system.

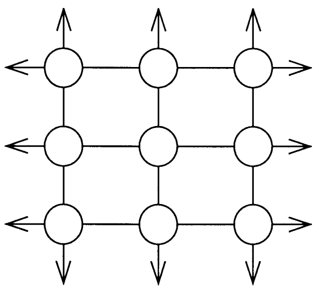
Figure 6. The array processor.

connected to all pipelines via their local memories. The pixels, which are required by a pipeline, are clocked into the pipeline by a data acquisition and control circuit. The address generation circuit drives its input from the camera system and generates addresses of the pixels, which are being output from the camera system.

As soon as the pipelines receive that data, they start executing the algorithms. After completing their tasks, each pipeline sends its data to memory (M2) for storage. These data are accessed by the host computer to present the traffic information in graphical form.

The Array Processor Module

Figure 6 shows the block diagram of the array-processing module. The mesh of 3×3 processing elements forms this computation module. The nine PEs are fully inter-connected. They could therefore, work synchronously, the same way as mesh-connected array processors work. Each PE in the array would access its corresponding input, along with inputs from its neighboring PEs.

As shown in Figure 7, every PE consists of a PALU (Processing Arithmetic and Logical Unit) providing

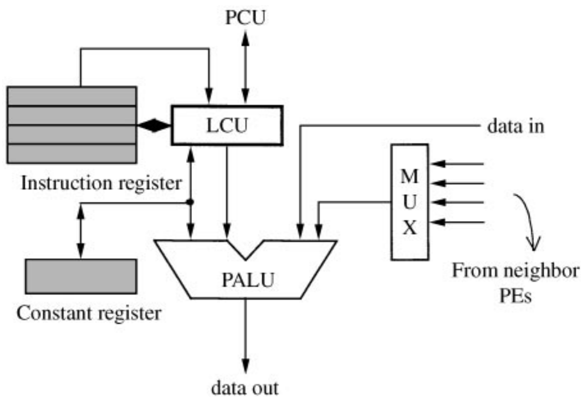

Figure 7. Structure of the PE.

Boolean as well as arithmetic functions, instruction and constant registers, and a LCU (Logic Control Unit). Each processing element has a local memory, which is used to store instructions and the data. Initially, both instructions and constant registers in each PE are loaded with the corresponding instructions and the data. After the initialization phase, the PEs execute their instructions. During the process, every PE in the array executes the operations defined in its own instruction register simultaneously. Note that the whole process in the array is well connected and is controlled by a host system.

The processing module has 64 instructions with 8-bit format. Apart from instructions for normal functions, there are few distinctive instructions to take advantage of the special architecture of the processor.

Results

The authors have conducted extensive experiments for continuous periods of 6–8 hours. The result of counting vehicles under various weather conditions is shown in Figure 8. As can be seen, the error is around 15%. The

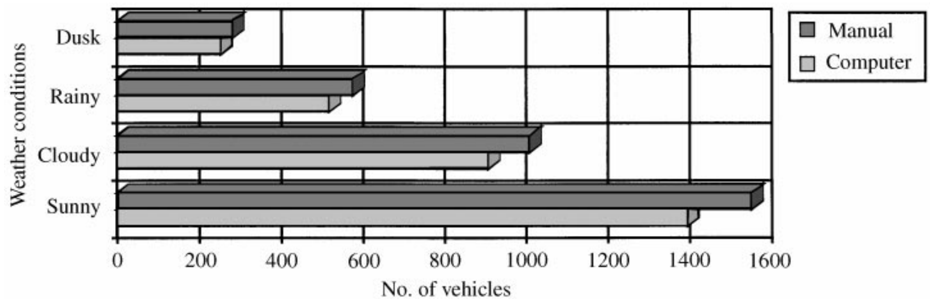

Figure 8. Results of counting vehicles.

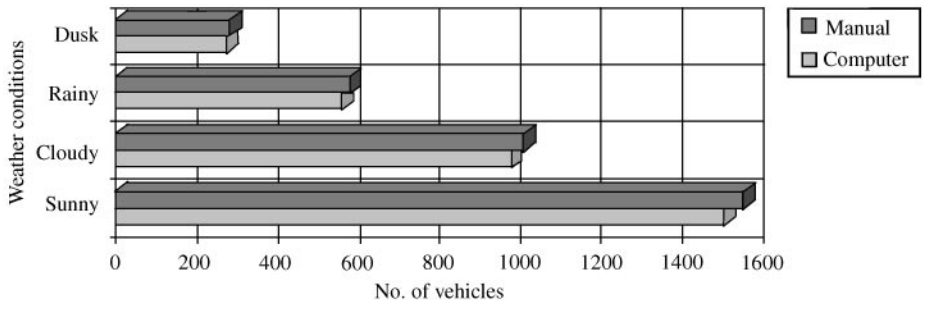

Figure 9. Results of counting vehicles by placing windows between the lanes.

authors believe that these errors are due to drivers supposedly driving one lane, but instead straddle two lanes. To further improve the accuracy of the vehicle counting operation, the windows are placed in between two lanes. With this approach the accuracy achieved was more than 97% (Figure 9).

Comparison of the results of queue parameter measurement using the authors system and manual measurement of the queue is shown in Figure 10. As can be seen, the system is able to detect the queue with a good accuracy.

The multiprocessor system together with the host computer is able to provide traffic information in real-time. During the experiment, the results were reported online in graphical form. The authors system is able to execute image processing algorithms at a rate of 15 frames per second, which is more than sufficient for real-time operation, as only key regions of the image (windows) are processed and not a full image.

Conclusions

In this paper, a multiprocessor system based on a RISC type programmable image processor for executing image-processing algorithms for road traffic applications was described. The processor is designed to take advantage of parallelism in mesh-connected array processor for image area processing. Due to this small mesh-connected PE array, a high degree of concurrency has been achieved.

A novel vehicle detection technique based on edge detection and key region processing was also introduced in the paper. To implement the algorithm in real-time, a low computational cost technique for edge detection, which eliminates noise and detects thin edges has been used. The information extracted by the vehicle detection algorithm was used for measuring other traffic parameters, such as the volume of vehicles and queue length. The algorithm was implemented on the proposed multiprocessor system, and the results obtained demonstrate that the system can provide traffic parameters from road traffic images in real-time.

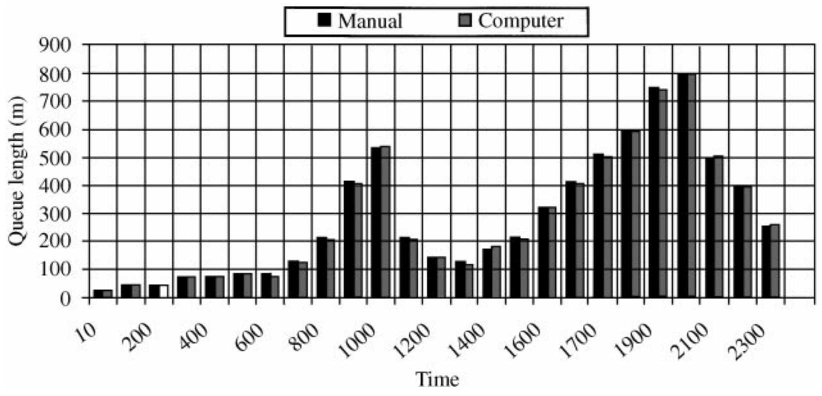

Figure 10. The result of queue parameters.

References

1. Dickinson, K.W. & Waterfall, R.C. (1984) Image processing applied to traffic: Practical experiences. *Traffic Engineering and Control*, Feb. 1984, pp. 60–67.
2. Fathy, M and Siyal, M.Y "An image detection technique based on morphological edge detection and background differencing for real-time traffic analysis", Pattern Recognition Letters, Vol. 16 (1995), pp. 1321–1330.
3. Hoose, N. (1991) *Computer image processing in traffic engineering*, Taunton, Research Studies Press.
4. Ali, A.T and Dagles, E.L., Recent progress in Real-Time Image Analysis for real-world traffic analysis, ICARVC'92, second international conference on Automation, Robotics and Computer Vision proceedings, vol. 1, pp. 10–15, 1992.
5. Siyal, M.Y and Fathy, M "A window-based image processing technique for quantitative and qualitative analysis of road traffic parameters", to appear in IEEE Transactions on Vehicular Technology, September 1998.
6. Fathym, M and Siyal, M.Y "A window-based edge detection technique for measuring road traffic parameters in real-time", Real-Time Imaging I, pp. 297–305, 1995.
7. Navaux, P.A and Cesar, A.F "Performance evaluation in image processing with GAPP array processor", Microprocessors and Microprogramming', pp. 71–82, March 1995.
8. Abnous, A and Bagherzadeh, N "Architectural Design and Analysis of VLIW Processor", Computer and Electrical Engineering', Vol. 21, pp. 119–142, 1995.
9. Aleksic, "CISC vs RISC processors for Graphics: A Simulation Study", Microprocessors and Microprogramming, Vol. 37, pp. 45–48, 1993.
10. Fathy, M and Siyal, M.Y, "A Real-Time Image Processing Approach To measure Traffic Queue Parameters", IEE Proceedings on Image, Vision and Signal processing, Vol. 142, No. 5, pp. 297–303. October, 1995.
11. Konstantinides and Bhaskaran, Z "Monolithic Architectures for Image Processing and Compression", IEEE Computer Graphics and Applications, pp. 75–86, 1992.
12. Tokhi, M.O and Hossain, M.A "CISC, RISC and DSP processors in real-time signal processing and control", Microprocessors and Microsystems, June 1995.
13. Turner, C.J and Bhavsar, V.C "Parallel implementations of convolution and movements algorithms on a multi-transputer system", Microprocessing and Microprogramming', pp. 283–290, June 1995.