# Extraction of Road Traffic Data from Image Sequences

M. Atiquzzaman* and M.G. Hartley**

*Department of Computer Science and Computer Engineering, La Trobe University, Melbourne 3083, Australia. Email: `atiq@LATCS1.lat.oz.au`

**Department of Electrical Enginering and Electronics, University of Manchester Institute of Science and Technology, Manchester M60 1QD, England.

**Abstract**
Image processing involves processing a vast amount of data, and hence requires a large amount of computing power. Multiprocessor systems have been suggested as sources of large amount of computing power. For efficient execution of an algorithm in a multiprocessor system, the structure of the algorithm should mirror the architecture of the multiprocessor system. A linear array multiprocessor architecture for low level image processing is described in this paper. The application under consideration relates to a road traffic situation where vehicles must be counted and velocities determined in order to permit online urban traffic control strategies and freeway incident detection. A parallel algorithm to perform the above task and its multiprocessor implementation are described in this paper.

## 1 Introduction

The processing of real-world images of road traffic scenes for real-time extraction of traffic flow parameters presents a number of problems. The most significant problem is the requirement of a large amount of computing power, arising due to the real-time constraint. Information regarding traffic flow consists of parameters like speed of vehicles, headway between vehicles, and classification of vehicles (e.g. cars, busses, etc.). The objective of the research work described in this paper is to propose a cost-effective and scalable multiprocessor architecture, supported by an efficient parallel algorithm to extract the traffic flow information in real-time.

This paper describes the architecture of a proposed linear-array multiprocessor system and the parallel algorithm to extract the traffic flow parameters. A prototype of the system can be built using off-the-shelf components to reduce the development cost and implementation efforts. Because of regularity in the structure of a linear array, it is scalable and suitable for implementation in VLSI. Parallel algorithms mirroring the linear array architecture are presented.

## 2 System Architecture

The architecture of the proposed linear array multiprocessor is shown in figure 1. It consists of a linear-array of $P$ processor modules (PM), each of which (except the ones at the edges) is connected to its two neighbours by dedicated interprocessor links. Each module is also connected to two busses – the *video bus* (VB) and the *common bus* (CB). Images from a camera are loaded into the PMs through the video bus. The PMs pass the results of processing to the host computer through the CB which is also used for non-neighbour interprocessor communications, if such a need arises. The PMs are small computers in themselves with their own RAM, EPROM, and CPU. The PMs work asynchronously, but can also be made to work synchronously if SIMD operation is desired.

An $M \times M$ image is loaded into the system, each PM receiving a subimage of size $(\frac{M}{P} + 2) \times M$. To reduce the interprocessor communication, two lines of pixels at the two boundaries of a subimage are replicated in the neighbouring PMs. Concurrent loading and processing of sub-images [1] is advocated in the design. This means that a PM starts processing as soon as its subimage from an image has been loaded, while other sub-images of the image are still being loaded into other PMs. Processed results from all PMs are passed through the CB to the host computer, which is responsible for accumulating the results and determining the required parameters. A chip-level design of the system can be found in [2].

## 3 Parallel Algorithm

The traffic parameters mentioned previously are determined from the area and centroid of the vehicle as shown below. A background image of the road scene is defined to be one with only stationary components, i.e., an image with no moving cars. In order to determine the area and the centroid, the moving vehicle in an image, $C = \{c_{i,j}\}$, is isolated from the image by subtracting the image from the background image, $B = \{b_{i,j}\}$, to produce a difference image, $D = \{d_{i,j}\} = \{c_{i,j} - b_{i,j}\}$. This is followed by construction of a binary edge image $\hat{D}$, by operating on each row of $D$ using a first-order one-dimensional gradient edge-detector, and then applying a threshold $\tau$ to the output of the edge-detector. The vehicle is assumed to be of convex shape. The assumption has been found to be true for real-world images.

$$\hat{D}(i,j) = \begin{cases} 1 & \text{if } |d_{i,j-1} - d_{i,j+1}| > \tau \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where, $\tau$ is a threshold, whose value is selected by the operator depending on the type of image. Each PM contains a subimage from each of $B$ and $C$ as described in section 2. All PMs operate on their subimages to obtain subimages of $\hat{D}$. Because of the convex nature of the object, each PM will have two edges in a line of $\hat{D}$ which is overlapped by the vehicle. Each PM will, therefore, have between 0 to $2(M/P)$ edges. All PMs send coordinates of the edge pixels in its subimage to the host computer, which computes the area, centroid, and perimeter of the object using Simpson's rule. The area $A$ is given by [3]

$$A = \frac{d}{3}(f_1 + f_n + 2\sum f_o + 4\sum f_e) \tag{2}$$

where, $d$ is the width of a line, and $f_1$ and $f_n$ are the lengths of the first and last strips of the object respectively as shown in figure 2. $\sum f_o$ and $\sum f_e$ are the sum of the lengths of the odd and even strips respectively. The host computer computes the length of a strip by subtracting the two pairs of edge coordinates corresponding to the strip.

$$f_i = |x_{1,i} - x_{2,i}| \tag{3}$$

where, $x_{1,i}$ and $x_{2,i}$ are the x-coordinates of the left and right edges respectively of the $i-$th strip of the image. The coordinates $(\bar{x}, \bar{y})$ of the centroid of the vehicle are determined by

$$\bar{x} = \frac{\sum_{i=1}^{n} x_{c,i} A_i}{\sum_{i=1}^{n} A_i} \tag{4}$$

$$\bar{y} = \frac{\sum_{i=1}^{n} y_{c,i} A_i}{\sum_{i=1}^{n} A_i} \tag{5}$$

where, $A_i$ is the area of the vehicle in the $i-$th strip of the object, $(x_{c,i}, y_{c,i})$ are the coordinates of the center of the $i-$th strip, and $n$ is the number of strips forming the object. $y_{c,i}$ is obtained from strip number $i$, and $x_{c,i}$ is obtained from $x_{1,i}$ and $x_{2,i}$ as follows.

$$x_{c,i} = x_{1,i} + \frac{x_{2,i} - x_{1,i}}{2} \tag{6}$$

If $\Delta P_i$ is the distance between two edge points, $(x_{1,i}, y_{1,i})$ and $(x_{1,i+1}, y_{1,i+1})$, of two consecutive strips, then the perimeter $P$ is given by

$$P = \sum \Delta P_i \tag{7}$$

where,
$\Delta P_i = \sqrt{(x_{1,i} - x_{1,i+1})^2 + (y_{1,i} - y_{1,i+1})^2}$. The area and perimeter of the vehicle is used to classify the vehicle. The speed $S$ of the vehicle is given by

$$S = \frac{\sqrt{(\bar{x}(t) - \bar{x}(t+\Delta t))^2 + (\bar{y}(t) - \bar{y}(t+\Delta t))^2}}{\Delta t} \tag{8}$$

where, $(\bar{x}(t), \bar{y}(t))$ and $(\bar{x}(t+\Delta t), \bar{y}(t+\Delta t))$ are the coordinates of the centroid of a vehicle at time $t$ and $(t + \Delta t)$ respectively.

Centroids of all the vehicles in a lane of the road are determined as described above. The algorithm is repetitively applied to all the lanes of the road. Distance between the centroids of two vehicles in a lane provides a measure of the distance between the vehicles.

Table 1: Performance figures for typical low level image processing algorithms on $100 \times 100$ images using the Transputer.

| Operation | Maximum speed of operation (images per second) |
|---|---|
| $3 \times 3$ averaging | 1220 |
| $3 \times 3$ edge detection | 1987 |
| 4-neighbor connectivity fill, shrink, & expand | 600 |
| Centroid determination | 747 |

## 4 Results

The above algorithm for collection of traffic flow parameters from real-world images were implemented in an 8086-based testbed system. The values of the parameters were found to be within 5% of the actual values. Results indicate that real-time operation (1 frame/sec) on $100 \times 100$ images is possible with a linear array multiprocessor system with ten 8086 processors. Each processor is then responsible for ten consecutive lines of the image. Simulation results show that the generation of a binary edge image on a linear array system using 10 processors can be carried out at a rate of 8 frames/sec. The algorithm described in section 3 was also implemented in a Transputer-based system. Detailed results regarding the speed of operation of some typical image processing algorithms, using Transputers in the proposed architecture are given in Table 1 [2]. The figures represent the maximum rate, in images per second, at which the operations can be carried out using Transputers in the PMs. The above architecture and algorithm would be very useful for implementation as a versatile and cheap traffic data collection equipment for use by traffic engineers.

## 5 Conclusion

A linear array multiprocessor system and a parallel algorithm to extract, in real-time, information regarding the traffic flow from images of road scenes have been described in this paper. The algorithm does not require any communication among processors, which is usually considered to be a bottleneck in multiprocessor systems.

## References

[1] M. Atiquzzaman, "Performance modeling of multiprocessor systems for different data loading schemes," Tech. Rep., Computer Science Dept., La Trobe University, Melbourne, Australia., 1993.

[2] M. Atiquzzaman, *Algorithms and architectures for automatic traffic analysis.* PhD thesis, University of Manchester Institute of Science and Technology, England, March 1987.

[3] P.S. Sarma, "A microcomputer-based system for area measurement," *IEEE Transactions on Instrumentation and Measurement*, vol. IM-33, no. 3, pp. 168–171, September 1984.

Figure 1: A linear array mulitprocessor system.

Figure 2: Partitioning an image among the processor modules.