

# VEHICLE DETECTING AND TRACKING IN VIDEO FOR INCIDENT DETECTION

Nitin Agarwal  
Nicholas Andrisevic  
Kiran Vuppla  
Marian S. Stachowicz  
[mstachow@d.umn.edu](mailto:mstachow@d.umn.edu)

Laboratory for Intelligent Systems  
Department of Electrical and Computer Engineering  
271 Marshall W. Alworth Hall  
University of Minnesota Duluth  
Duluth, MN 55812 USA  
Phone: (218) 726-6147  
Fax: (218) 726-7267

Andrey Burlev  
Konstantin Seleznev

GadgetSoft, Novosibirsk, Russia  
[www.gadgetsoft.com](http://www.gadgetsoft.com)

Published by  
Minnesota Department of Transportation  
Office of Research Services  
Mail Stop 330  
395 John Ireland Boulevard  
St. Paul, Minnesota 55155-1899

Disclaimer: This report represents the results of research conducted by the authors and does not necessarily represent the view or policy of the Minnesota Department of Transportation and/or the Center for Transportation Studies. This report does not contain a standard or specified technique.

## **Acknowledgments**

1. Northland Advanced Transportation Systems Research Laboratory, (NATSRL), for providing funding and assistance on this project.
2. The Employees of the Minnesota Department of Transportation, for their assistance in the data collection for this project.
3. Special thanks to Nic Bacigalupo, MIS Manager for the Duluth Mn/DOT office, who helped by providing access to the cameras that Mn/DOT has installed around the Duluth highways.

## Table of Contents

<b>Chapter</b>	<b>Page No.</b>
1. Introduction	1
2. Vehicle Detection Algorithm	3
2.1. System Overview	3
2.2. Detection of Moving Areas	4
2.3. Determination of the Threshold for Finding Contrast Regions	4
2.4. Background Refresh	5
2.5. Separation of the Areas Found	5
2.6. Tracking the Moving Areas	5
2.7. Attribution of the Moving Areas to Vehicles	7
3. Data Collection for Experimentation	9
3.1. Data Acquisition	9
3.2. Accounting for Variance in Conditions	9
3.3. Glossary	11
4. Possible Algorithm Applications	13
4.1. Video Traffic Surveillance	13
4.2. Automatic Incident Detection	14
4.3. Electronic Rumble Strips	15
4.4. Automatic Number Plate Recognition	15
4.5. Car Park Systems for Occupancy	16
4.6. Car Park Systems for Security	16
4.7. Fire Detection	17
4.8. Electronic License Plate	17
4.9. Bus Lane Enforcement	18
5. Conclusion	19
6. References	21
<b>Appendix A</b>	<b>A-1</b>
<b>Appendix B</b>	<b>B-1</b>
<b>Appendix C</b>	<b>C-1</b>

## Tables and Figures

<b>Figure No.</b>	<b>Page No.</b>
Figure 1: Distribution histogram of the difference between the background and the foreground	4
Figure 2: Mask and the contours for the frame	5
Figure 3: Correspondence between areas from frame to frame	6
Figure 4: The case when the way of an area from frame to frame cannot be followed	6
Figure 5: Correspondence between areas 1 to 1 for p+1 frames	7
Figure 6: Experiment on data collected using Grab.exe	9
Figure 7: Experiment on low quality video with Grab.exe	10
Figure 8: Experiment of video clips that were collected using dirty lenses camera on Grab.exe software	10
Figure 9: Experiment on video clip of an intersection using Grab.exe	11
Figure 10: Queue Length Measurement and Intersection Control [11]	14
Figure 11: An Electronic Rumble Strip [8]	15
Figure 12: A system for Automatic Number Plate Recognition [16]	16
Figure 13: Fire Detection using Video Imaging [13]	17
Figure 14: An Electronic License Plate [15]	17
Figure 15: View from a Camera Installed on a Bus [14]	18

## **Executive Summary**

Video cameras are widely used in Traffic Management Centers (TMCs), and the numbers are steadily increasing. For example, within Minneapolis and St. Paul, (Twin Cities), the number of cameras used for traffic management is reaching near 250. Other Minnesota cities are following this trend, e.g. Traffic Operational Centers (TOCs) in Duluth and St. Cloud also make use of a large number of cameras for traffic management.

The purpose of this project is to research and experiment on the application of vehicle detection software in traffic control systems, specifically to detect abnormal traffic situations, such as sudden lane change, vehicle going off the road and possibly vehicle accidents.

The automation of tasks such as surveillance of the roadways, would help save on the maintenance of the road and help respond more effectively to abnormal situations, such as accidents. This would make traveling much easier for the travelers as it would save them time and money and would make traveling safer, since dangerous road conditions would be more effectively addressed. Furthermore, the members of the Department of Transportation would be better able to perform their jobs, without being impeded by tedious and repetitive tasks. All in all, the automation of certain aspects in a traffic control system proves to not only increase the effectiveness and robustness of the system, but also increases the quality of service and performance of the traffic control system.

This research project consists of three parts:

### **Vehicle Detection Algorithm:**

This phase describes the algorithm employed in the development of the software. Main steps in the algorithm are:

1. Detection of moving areas.
2. Tracking the moving areas.
3. Matching the moving areas to the vehicles.

### **Data Collection for Analysis:**

In this phase various experiments were conducted to determine exactly what features of the software required further testing and analysis. Before this is done, it must be determined what type of video clips are considered valid or invalid data, based on features such as content, quality, etc. Once this is accomplished, the methods by which the data is collected are chosen based on how effectively they produce the desired data. Next the resources for collecting the data are described as well as when and where the videos were captured, and the types of equipment to be used.

The results of the experiments are analyzed and used to determine if further experimentation is required, and also to form conclusions and comments on the performance of the software.

### **Possible Algorithm Applications:**

This phase involved exploration and evaluation of the applicability of the algorithm in various types of traffic control situations. Application of the algorithm is discussed for each of these situations. An evaluation of the usability of the algorithm is reported for each of these situations.



# Chapter 1

## Introduction

The purpose of this project is to research and experiment on the application of a vehicle detection algorithm in traffic control systems, specifically to detect abnormal traffic situations, such as sudden lane change, vehicle going off the road and possibly vehicle accidents.

Video cameras have been widely used in Traffic Management Centers (TMCs), and the numbers are steadily increasing. Even within Twin Cities alone, the number of cameras used for traffic management is reaching near 250, and other cities are following this trend, e.g. Traffic Operational Centers (TOCs) in Duluth and St. Cloud use a large number of cameras as well.

The initial task of this research is to focus on detecting vehicles from video images and using it for computation of traffic density. An algorithm is developed for this purpose and is discussed in the Chapter 2 of the report.

The third chapter discusses the data collection techniques and experimentation with the data. While gathering data for experimentation, the main goal was to collect data with various light and weather conditions.

Chapter 4 discusses the possible areas in which the algorithm can be applied and finally, the last chapter ends with the conclusion of the report.

The algorithm is implemented with the program Grab.exe. Grab.exe has been developed by Konstantin Seleznev and Andrey Burlev of GadgetSoft Inc. in cooperation with members of the Laboratory for Intelligent Systems, (LIS), at the University of Minnesota, Duluth. The main 3 sections of the report are listed below and are discussed in the following chapters:

- Vehicle Detection Algorithm
- Data Collection for Experimentation
- Possible Algorithm Applications



## Chapter 2

### Vehicle Detection Algorithm

Traffic control systems are based on a wide range of detectors to estimate the parameters of moving objects. Magnetic loop detectors are often used today to count vehicles passing through them. Systems based on video cameras have certain advantages. In addition to vehicle count they can determine some other traffic parameters such as changes of trajectory, dynamic parameters of the motion, non-standard motion. A camera can be set up much easier than a loop detector. The use of video-based detection and control systems will improve planning and building the roads by providing a way to accurately measure traffic density. The presented system uses one camera fixed on a high site, e.g. a bridge or a post, and looking downwards at the traffic scene. The system requires setting the geometrical parameters of the camera's position.

This section describes the algorithm for vehicle detection and tracking by video image of traffic scene taken by stationary camera. The algorithm consists of three stages: detection of moving areas, tracking of the moving areas, and attribution of the areas to vehicles on the traffic scene. The system can handle moving objects that partially or completely overlap, and obtain such parameters as Number of vehicles per hour, Average vehicle velocity, Density of vehicles per unit distance, Average spacing between vehicles. The algorithm has been tested on real traffic scenes and proved to be an effective tool to obtain traffic parameters.

#### 2.1 System Overview

The presented algorithm consists of three parts.

1. Detection of moving areas.

This part of the algorithm detects areas that contrast to the road. Simultaneously, the image of the road is gradually corrected according to weather and light changes and camera position.

2. Tracking the moving areas.

After the detection of contrasting areas, for each frame we have 2D areas showing the positions of the vehicles (or parts of vehicles) on the road. The aim of this part of the algorithm is to set up a correspondence between the areas in the previous frame

$A_{k-1} = \{a_{k-1l}, l = 1..n_{k-1}\}$  and the areas in the current frame  $A_{k1} = \{a_{k1}, l = 1..n_k\}$ , i.e.

for each  $a_{k-1p} \in A_{k-1}$  to find a set of such  $a_{kq} \in A_k, q \in N_{k-1}$ , building up the

correspondence under the assumption of small travel of the areas from frame to

frame, and for each  $A_k = \{a_{kl}, l = 1..n_k\}$  we find a set  $a_{k-1q} \in A_{k-1}, q \in P_k$ .

3. Matching the moving areas to the vehicles.

A vehicle can consist of a number of areas and an area in the frame can belong to a number of vehicles. That can happen in a sunny weather when the shadow of the vehicle moves together with the vehicle or when one vehicle overlaps another on the

image. This part of the algorithm sets up the correspondence between moving areas and vehicles. For each frame  $k$  and for each vehicle  $v_{k,p} \in V_k$  we find a set  $a_{kq} \in A_k, q \in C_k$ . Then, for the detected vehicles the trajectories and dynamic parameters are found.

## 2.2 Detection of Moving Areas

Detection of the areas can also be split into three steps.

1. Determination of the threshold for finding contrasting regions
2. Background refresh
3. Separation of the areas found

## 2.3 Determination of the Threshold for Finding Contrast Regions

Let  $F_{i,j} = (Fr_{i,j}, Fg_{i,j}, Fb_{i,j}), i = 1..W, j = 1..H$  be the value at  $(i, j)$  in the current frame, where  $Fr, Fg, Fb$  correspond to the color component on the image,  $W, H$  - are width and height of the frame. Let  $B_{i,j} = (Br_{i,j}, Bg_{i,j}, Bb_{i,j})$  correspond to background image. Now we calculate  $T_{ij} = \max\{|Fr_{ij} - Br_{ij}|, |Fg_{ij} - Bg_{ij}|, |Fb_{ij} - Bb_{ij}|\}$ , and then build a distribution

histogram for  $T_{ij}$ ,  $\Gamma(x) = \sum_{i,j} \begin{cases} 1, & \text{if } T_{ij} = x \\ 0, & \text{otherwise} \end{cases}$ , where  $x = 0..255$ . The distribution graph

will have a maximum near zero because the area of contrasting regions is usually a small part of the area of the whole frame. Thus, the number of pixels where the difference in color between the background and the frame is small will be considerable. Let

$T_m = \max_x \Gamma(x)$ , then we find such  $T$  that  $\Gamma(T) = \beta T_m, T > \arg \max_x \Gamma(x)$ . In our case we took  $\beta = 0.05$ .

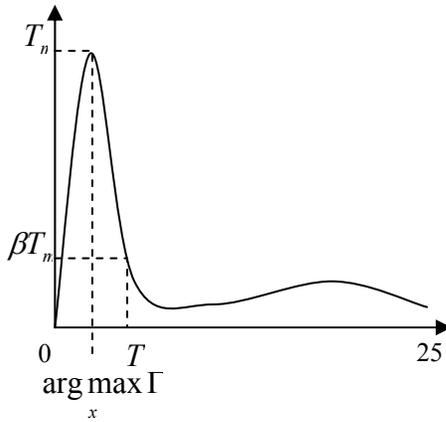


Figure 1. Distribution histogram of the difference between the background and the frame.

## 2.4 Background Refresh

To calculate the background we use the following scheme. For the first frame we put  $B_{ij} = F_{ij}$ , then we calculate mask  $M_{ij} = \begin{cases} 1, & \text{if } T_{ij} > T \\ 0, & \text{otherwise} \end{cases}$ , and now for all  $B_{ij}$  such that  $M_{ij} = 0$  we calculate the refreshed value of the background  $B_{ij}^k = \alpha F_{ij}^k + (1 - \alpha)B_{ij}^{k-1}$ , where  $k$  corresponds to the number of the frame. This means that only those pixels are refreshed that lie beyond contrasting regions. Experiments have shown that the optimal value is  $\alpha = 0.1$ . If light or camera position parameters change then areas will appear for which  $M_{ij} = 1$  throughout many frames, e.g. due to a motion of a shadow from a stationary object or when the camera is displaced, and the whole field of the image will have mask 1. Now we calculate the change mask  $MC_{ij}^k = \sim (M_{ij}^{k-N} \& M_{ij}^{k-N+1} \& \dots \& M_{ij}^k)$ , where  $N$  is the threshold number of frames for the change mask. Now for all  $B_{ij}$  such that  $MC_{ij} = 0$  we refresh background using the change mask  $B_{ij}^k = \hat{\alpha} F_{ij}^k + (1 - \hat{\alpha})B_{ij}^{k-1}$ . We used  $\hat{\alpha} = \alpha$ .

## 2.5 Separation of the Areas Found

For each frame  $F_{ij}$  we have a mask  $M_{ij}$ , and find contours of each connected area where  $M_{ij} = 1$

For frame  $F_{ij}^k$  we get a set of areas  $A_{kl} = \{a_{kl}, l = 1..n_k\}$  characterized by the rectangles containing them  $B_{kl} = \{b_{kl}, l = 1..n_k\}$ . Note that we take into account contours that encircle the area from outside, i.e. if a contrasting area has holes then contours encircling the holes are not used. Also we do not include into the set of areas contours that have small area  $S(b_{kl}) < S$ , we took  $S = 20$  which removed regions corresponding to noise from the vibration of the camera.

## 2.6 Tracking the Moving Areas

Suppose we have two sets of rectangles  $B_{kl} = \{b_{kl}, l = 1..n_k\}$ ,  $B_{k-1} = \{b_{k-1l}, l = 1..n_{k-1}\}$  for the current and the previous frames, respectively. We set up a correspondence between  $a_{k-1p} \in A_{k-1}$  and  $a_{kq} \in A_k$  if  $b_{k-1p} \in B_{k-1}$  intersects with  $b_{kq} \in B_k$  and



Fig. 2. Mask and contours for the frame.

$$b_{kq} = \arg \max_{b_{kl}, l=1..n_k} \frac{S(b_{k-1p} \cap b_{kl})}{S(b_{k-1p})}, \text{ similarly we set up a correspondence between } a_{kq} \in A_k \text{ and}$$

$$a_{k-1p} \in A_{k-1} \text{ if } b_{kq} \in B_k \text{ intersects with } b_{k-1p} \in B_{k-1} \text{ and } b_{k-1p} = \arg \max_{b_{k-1l}, l=1..n_{k-1}} \frac{S(b_{kq} \cap b_{k-1l})}{S(b_{kq})}.$$

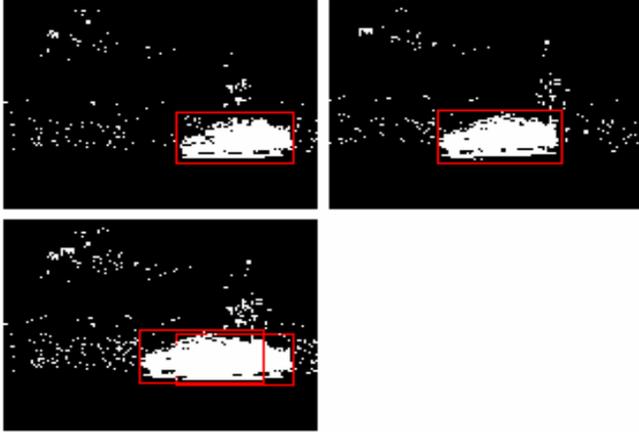


Figure 3. Correspondence between areas from frame to frame

Thus for each two frames we have a correspondence graph between the previous and the current frames. Note that one frame can correspond to many frames and many frames can correspond to one frame. On the other hand, there can be situations when one cannot follow the way of a region from frame to frame, see figure 4 (the way of the area can be tracked by the ribs from the previous frame to the current one and from the current frame to the previous one).

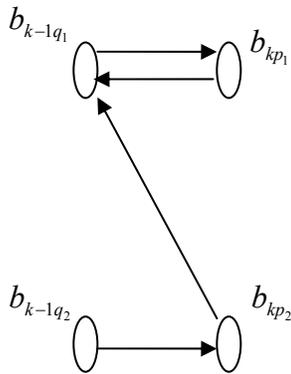


Figure 4. The case when the way of an area from frame to frame cannot be followed.

Thus we will take the following algorithm of adding ribs. First we add ribs going from the areas in the previous frame to the areas in the current frame, and then we add ribs going from the areas in the current frame to those in the previous frame. When adding a frame we control that each connected component of the graph would not have more than one vertex with grade over 1, meaning a graph which includes only vertexes

corresponding to the current and the previous frames but not to the entire sequence of preceding frames. As a result, when we unite the entire sequence of correspondences from frame to frame we can follow the travel of the region throughout a sequence of frames.

## 2.7 Attribution of the Moving Areas to Vehicles

Here we assume that a vehicle can consist of a number of areas and an area can correspond to more than one vehicle. We also assume that the dimension of the vehicle form a rectangle covering the rectangles that form the vehicle. Let us denote the weight of an area inside the vehicle as  $w_{klp} = \frac{S(b_{kp} \cap v_{kl})}{S(v_{kl})}$ , where  $k$  is the number of the frame,  $p$  is the number of the area in the frame,  $l$  is the number of the vehicle in the frame. The area can be found in three major situations.

For an area in the current frame  $a_{kp}$  there is a corresponding area in the previous frame  $a_{k-lq}$ , then we set up a correspondence between  $a_{kp}$  and vehicles corresponding to  $a_{k-lq}$  in case if  $w_{klp} > W$ , we used  $W = 0.1$ . If for  $a_{k-lq}$  there is no correspondence to any vehicle then we check for how many frames back the area has had a correspondence from frame to frame as 1 to 1 (see fig. 5). If such a correspondence is found throughout  $P_1$  frames then a vehicle is created, and  $a_{kp}$  is attributed to the vehicle.

An area can disappear, i.e. for a region in the previous frame  $a_{k-lq}$  there is no area in the current frame. In that case a vehicle may have no regions left, and we mark the vehicle as gone. If a vehicle that has gone does not reappear for  $P_2$  frames (see c) then it is removed from the set of vehicles. We used  $P_2 = 3$ .

An area can reappear, i.e. for  $a_{kp}$  there is no area in the previous frame. We check the intersection of this region with vehicles that have gone and if  $w_{klp} > W$ , then we admit correspondence of that region to the gone vehicle  $v_{kl}$  and the vehicle is restored as normal.



Figure 5. Correspondence between areas 1 to 1 for  $p+1$  frames.



## Chapter 3

### Data Collection for Experimentation

The data collection phase of the project consists of several stages. The first stage is to decide when and where the traffic videos should be recorded, contacting the necessary parties to schedule use of equipment and facilities if needed, and obtaining the required equipment to perform the data collection. Also during this stage, careful planning is done to ensure that undesirable outside interference with the recording process, as well as other obstacles, are kept to a minimum.

#### 3.1 Data Acquisition

The next stage involves the actual data collection. The primary traffic video source for this project was The Minnesota Department of Transportation, (Mn/DOT), facilities in Duluth, Minnesota. Nic Bacigalupo, the MIS Manager for the Duluth Mn/DOT office helped in providing access to the cameras that Mn/DOT has installed around Duluth. Using a laptop computer and Axis Camera Explorer, traffic videos were recorded directly from the camera in real-time, and then converted into movie file format to be useable by Grab.exe software. Traffic footage was also recorded near the University of Minnesota Duluth campus to provide an additional source of data.



Figure 6: Experiment on data collected using Grab.exe

#### 3.2 Accounting for Variance in Conditions

Data was collected in the form of .avi video files. These videos were recorded at an approximate frame rate of 30 frames per second. The data was collected in a manner that accounted for the wide range of weather, equipment, and traffic conditions. In a real world situation, diverse weather conditions would exist and may affect the performance of the software. These conditions include a bright sunny day, on a foggy day, on a rainy day and even during different times of day such as morning time, afternoon time, and evening time and during dusk.



Figure 7: Experiment on low quality video with Grab.exe

Data was also collected at a frame rate of less than 30 frames per second. This was done to allow observation of the Grab.exe software's behavior when the bandwidth is less than optimal, causing a decrease in recording performance, and when the video quality is less than optimal, i.e. low frame rate, blurry picture, and low resolution.



Figure 8: Experiment of video clips that were collected using dirty lenses camera on Grab.exe software

Since cameras would be mounted on various platforms and would likely be overlooking the road from various angles and positions, care was taken to gather video data that represented these cases. While shooting videos, the camera was placed at various locations such as under a bridge, on a pole, on the road median, by the side of the road and various pan, tilt and zoom settings were experimented with.

Traffic monitoring software would be required to work with cameras at locations with diverse traffic conditions. For example, on a straight highway, on a curved road, on a

crossroad, on a cloverleaf, in a tunnel, and so on. Therefore, videos were also shot to account for these considerations.



Figure 9: Experiment on video clip of an intersection using Grab.exe

### 3.3 Glossary

Road Color Difference: The contrast of the road from the background to foreground.

Camera Height: The height at which the camera is located on the pole.

Camera Direction: The change in view of the camera.

Field of View: The distance to which the camera can view the vehicles

Car Color Difference: The contrast of the vehicles from the background to foreground.



## Chapter 4

### Possible Algorithm Applications

This section discusses the automating of various tasks required by the Department of Transportation. These tasks can be done manually; however doing so requires a significant amount of funding and manpower. Discussed below are some of the tasks which when automated will reduce a lot of the overhead on the human operators, make traveling safer, increase the cost effectiveness of fuel use, can decrease travel time, and make the traffic control system more robust.

Algorithm application is discussed for each of these tasks. An evaluation of the algorithm's usability is reported for each of these tasks.

#### 4.1 Video Traffic Surveillance

Video images can be used to keep a watch on traffic in the various roads of a town or on a highway. Several video cameras are installed on the roads on which the traffic needs to be checked. These images could be transferred to the operator working in the Traffic Management Center through a broadband connection. A computer algorithm to look for certain behavior in the traffic can analyze the images from these cameras. This could include

- **Intersection Control:** The video image from a busy traffic intersection could be analyzed to find if the traffic at the intersection is causing congestion [11]. If this is the case, then the traffic at the previous intersections could be alerted about the congestion at the approaching intersection so that the vehicle owners could take precautionary measure, such as an alternate route or just be mentally prepared about the traffic ahead. If this information could be made available to them beforehand, i.e., before they left their homes or offices then they could possibly delay their trip so as to avoid the congestion ahead.
- **Queue Length Measurement:** Similar to the above discussion, if there is significant amount of traffic congestion on a certain road, then the traffic approaching the congested area could be warned beforehand about the situation so that the vehicle owners could take alternate measures in advance to handle the situation. If they are warned well in advance of the congestion, then they could avoid the congested road. And, if that is not possible, then they could compensate for the delay, such as by rescheduling appointments and meetings.



Figure 10: Queue Length Measurement and Intersection Control [11]

- **Travel Time:** It would help the commuters, if existing software could approximate their travel time depending on the route taken and the amount of traffic on that route prior to the commencement of their journey [11]. Using software like this, the commuter could travel a route with the least traffic and subsequently, least travel time.

**Algorithm Usability:** With some modifications, the algorithm could be adapted to the above-mentioned applications. For example, instead of identifying just individual vehicles, it needs to identify a group of vehicles and find how dense they are. If the density is high, it suggests that the road is crowded with chances of a big queue of vehicles and travel time can be higher on this road.

## 4.2 Automatic Incident Detection

Video Imaging could also be used to look for incident conditions on roads. These conditions include an accident, sudden change of lane, a vehicle going off the road and so on. This would help the Department of Transportation to regulate the traffic flow on a road. For instance, if an accident occurred and was detected by the software, which would then alert the necessary authorities, who will notify and dispatch the required emergency response teams to the location of the accident.

Furthermore, appropriate action can be taken to avoid congestion or roadblock on the road due to the mishap. In addition, if an incident occurs, its video could be preserved for the analysis, which could be useful for insurance companies and the court of law to find the cause of the accident. Similarly, if a vehicle is being driven recklessly which changes lanes quickly and is over the speed limit, then appropriate action could be taken against the driver of the vehicle to avoid any accident.

**Algorithm Usability:** The algorithm could be applied for the above task after several enhancements. The software is already capable of identifying and tracking vehicles. If an additional module is added to it, which could track the trajectory of the vehicle and identify any sudden change in the trajectory, the software could be implemented for the above purpose.

### 4.3 Virtual Rumble Strips

Currently, rumble strips are made on roads by the Department of Transportation to check for vehicles going off the road and to prevent them from going into the fields and ditches. Installation of rumble strips is expensive and time consuming. It even deteriorates the life of the tires of the vehicles when they run over them. A simple alternative could be a vehicle with a camera installed on it and equipped with software that could continuously monitor the vehicle's lateral position on the roadway and signal the driver's predicted run-off in time to prevent a mishap [8]. This system would be a simple, inexpensive and maintenance free alternative to the present rumble strips.

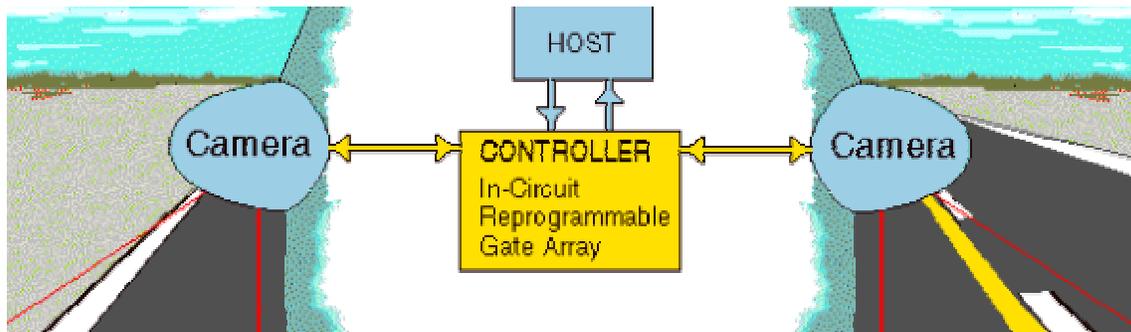


Figure 11: An Electronic Rumble Strip [8]

In addition, the above system could be used to alert the driver if he is going in the wrong direction on a road-way. This could be accomplished by the fact that on interstates, the line drawn on one side of the road is yellow in color while on the other side it is white in color.

**Algorithm Usability:** This algorithm may not effectively perform this application, as the basic idea is quite different in this case than what the algorithm is originally designed for. This algorithm requires the camera to be on a fixed platform and tracks the moving vehicles; however, this would require the camera to be fixed on a moving vehicle and would have to identify the steady lines on the either sides of the road.

### 4.4 Automatic Number Plate Recognition

Automatic number plate recognition could be used as an identification technology for traffic surveillance, toll collection, traffic management and many other projects where accuracy, speed and automation are essential objectives. If a high security building requires all the vehicles only with a valid permit to enter the building, then a surveillance system could be installed at the gates of the building. This system could be equipped with software that would decode the number plate of the vehicle from its image and would compare this number from the ones stored in the database with the permit to the building [16]. If there were a match, then the gate would open for the car and if not the gate would remain locked. This would help avoid the need of a human operator at the entrance gates of the buildings.



*Figure 12: A system for Automatic Number Plate Recognition [16]*

**Algorithm Usability:** In order to accomplish this, the algorithm would have to undergo considerable modifications. Once the software identifies a vehicle and its' License Plate, a software would be required which could identify the number on the plate and compare it with the ones in the database.

#### **4.5 Car Park Systems for Occupancy**

Multiple video Cameras could be installed in the multi-story car parking. The cameras could be installed so that they can cover the entire parking lot. If there is an empty slot for a car in the parking structure, the software would identify the vacant spot and indicate it at the entrance of the gate so that a car entering the parking structure could go straight to the empty slot without wasting time and fuel in finding a slot, saving money and reducing pollution.

**Algorithm Usability:** The algorithm can be used to identify vacant slots in a car park and could give directions to a new car entering into the parking area showing the directions to an empty slot.

#### **4.6 Car Park Systems for Security**

Many thefts of cars take place at the car parks where there is nobody to attend the cars. A mechanism could be installed which could take the picture of a car while it is exiting the parking space and compare it with a picture which was taken while the car was entering. After the image comparison the vehicle could be let to leave the parking if a close match was found or may be stopped for further inspection if there was a mismatch. This system could help reduce the car theft from parking lots just by using this simple comparison.

**Algorithm Usability:** The algorithm could be effectively used for the above task. The software could be used to take the pictures of the cars while entering and exiting the parking area. By comparing the two pictures, a suspicious car could be identified.

## 4.7 Fire Detection

Certain sections of roads, such as tunnels, are susceptible to particular hazards, such as fires, which can create a critical situation in short amount of time. As such, detection and response time are crucial in this case. A Fire Detection mechanism that is installed in these tunnels would alert the incoming traffic to any such incident so that they could wait outside until the emergency response teams have dealt with the situation.

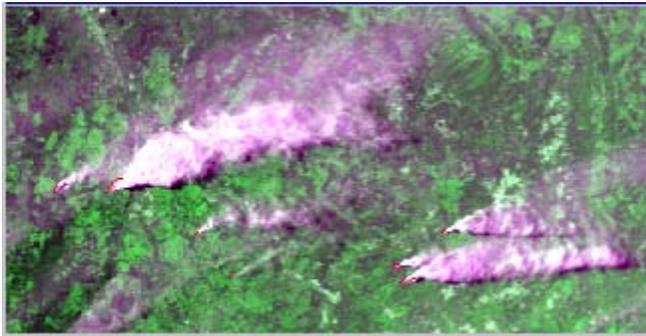


Figure 13: Fire Detection using Video Imaging [13]

**Algorithm Usability:** It would require a completely different algorithm to detect fire. This algorithm is not designed for this purpose.

## 4.8 Electronic License Plate

For the law enforcement agencies, this system could be quite useful. An electronic license plate could be installed on the vehicles, which is different from the conventional number plate as it would be a flat panel monitor instead of a metal sheet. Any registration number could be displayed on the monitor depending on the requirement.



Figure 14: An Electronic License Plate [15]

**Algorithm Usability:** This application again does not come under the scope of this algorithm. This is significantly different from all the other applications discussed in this paper.

## 4.9 Bus Lane Enforcement

Certain roads in the cities have bus lanes that are meant specifically for use by the local city buses. It must be made sure that these lanes are used only by the permitted vehicles and by no other vehicles. To achieve this, cameras would be installed in the front of the buses so that they could see the vehicles in front of them [14]. The image from this camera could be processed to extract the registration number of the vehicle and compared with the roster of numbers in the database with the permit to drive on the bus lane. If this number does not match with any of the entries in the database, then a ticket could be issued to the owner of the vehicle.



*Figure 15: View from a Camera Installed on a Bus [14]*

**Algorithm Usability:** This application is quite similar to the one discussed for Automatic Number Plate Recognition. Hence, the same could be implemented here with little modifications.

## **Chapter 5**

### **Conclusion**

The algorithms were tested with digitized videos of traffic scenes. The values of Number of vehicles per hour, Average vehicle velocity, Density of vehicles per unit distance, Average spacing between vehicles have been obtained. All the values were measured according to pixels on the image, i.e. the geometrical parameters were found in pixels and velocity was given in pixels per second. Further we are going to develop an algorithm for automated determination of the position of the camera referring the road by the averaged character of the vehicles' travel, and also to improve the adoptability of the algorithms to the changes in light and small displacements of the camera.

Some or all of the above mentioned possible algorithm applications could be applied to real life traffic control and management systems depending on the requirements. The automation of tasks such as surveillance of the roadways, would save on the maintenance of the road and help respond more effectively to abnormal situations, such as accidents. This would make traveling much easier for the travelers as it would save them time and money and would make traveling safer, since dangerous road conditions would be more effectively addressed.

Furthermore, the members of the Department of Transportation would be better able to perform their jobs, without being impeded by tedious and repetitive tasks. All in all, the automation of certain aspects in a traffic control system proves to not only increase the effectiveness and robustness of the system, but also increase the quality of service and performance of the traffic control system.



## Chapter 6

### References

- [1] D. Beymer, P. McLauchlan, B. Cofman and J. Malik. A Real-Time Computer Vision System for Measuring Traffic Parameters, in *Proc. IEEE Computer Vision and Pattern Recognition*, San Juan, Puerto Rico, June, 1997, pp.495–501.
- [2] C. Isaac and M. Gerard. Detecting and Tracking Moving Objects for Video Surveillance, in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2000.
- [3] M.F. Philip and M.W. Murray. A Unifying Framework for Structure and Motion Recovery from Image Sequences, in *Proc. of Fifth International Conference on Computer Vision*, June 23–25, 1995, pp. 314–320.
- [4] N. Garber and L. Hoel. *Traffic and Highway Engineering*. Pacific Grove, CA: Brooks/Cole Publishing Company, 1999.
- [5] A. Burlev, K. Seleznev, M. S. Stachowicz. Vehicle Detecting And Tracking Using Color in Video, *Unpublished Paper*, 2003.
- [6] AlphaWorks, (2002). *Real-Time Video Traffic Surveillance Software*. Retrieved June 20, 2003 from <http://www.alphaworks.ibm.com/tech/videosurveillance>
- [7] Peek Traffic, Inc., (2000). *VideoTrak - The Next Generation Detection Standard...* Retrieved June 19, 2003 from <http://www.stansellelectric.com/html/peek-pvs.html>
- [8] Williamson, G. A. (2002). *Automotive Lane Tracker*. Retrieved June 21, 2003 from <http://www.williamson-labs.com/lane.htm>
- [9] Betke, M., Haritaoglu, E., Davis, L. S. (1996). *Multiple Vehicle Detection and Tracking in Hard Real Time*. Retrieved June 14, 2003 from <http://citeseer.nj.nec.com/betke96multiple.html>
- [10] Intertraffic.com, (2002). *Carmen Free Flow ANPR Software*. Retrieved June 19, 2003 from [http://www.intertraffic.com/marketplace/mypage/products\\_detail.asp?mypageid=385&productid=602](http://www.intertraffic.com/marketplace/mypage/products_detail.asp?mypageid=385&productid=602)
- [11] Citilog, (2003). *Leader for Video Detection solutions in the Traffic Industry*. Retrieved June 12, 2003 from <http://citilog.itnetwork.fr/>
- [12] Golden River, (2000). *Traffic Data for the Decision Maker*, Retrieved July 17, 2003 from <http://www.goldenriver.com/>

[13] ScanEx, (2003). *Detection of forest fires*. Retrieved July 19, 2003 from <http://www.scanex.ru/stations/fire.htm>

[14] Steven J Warren, (2003). *UK Speedtrap Guide. The only UK Site with Comments from the Police and the most up to date information on UK Police SpeedTraps*. Retrieved July 19, 2003 from <http://www.ukspeedtraps.co.uk/gatso16.htm>

[15] Electro-Optical Technologies Inc, (2003). *State of WI License Plate*. Retrieved July 19, 2003 from <http://users.rcn.com/lnelson/wi.html>

[16] Hi-Tech Solutions, (2003). *Welcome To Hi-Tech Solutions*. Retrieved July 19, 2003 from <http://www.htsol.com/>

# **Appendix A**

## **Parallelization of Fuzzy Relational Equations**

# Appendix A

## Parallelization of Fuzzy Relational Equations

Deepa Krishnamoorthy

Marian Stachowicz

Masha Sosonkina

University of Minnesota

1049 University Drive, Duluth, MN 55812, USA

### Abstract

Many scientific and engineering problems involve the use of fuzzy relational equations of the form  $A \circ R = B$ , where  $A$  is the input vector,  $R$  is the relational system and  $B$  is the output vector. A lot of research has been carried out to find an appropriate solution to fuzzy relational equation. For some applications the above system can be very large involving several hundreds of fuzzy equations for which a solution is to be found. A uniprocessor system could take a long time to solve large problems since it has limited memory resources and computational power. Parallelization of large problems into smaller ones and distributing them to various processors where each processor computes its own solution is found to be very efficient. Through this paper we show that such a method for finding solution of fuzzy relational equation yields good results in terms of savings in time required for finding the solution.

**Keywords:** Fuzzy relations, computation of fuzzy relational equation

### 1. Introduction

Fuzzy relational equations [2, 4, 5] are useful in many engineering and scientific applications like fuzzy controllers and system analysis. The notion of fuzzy relational equations is associated with the concept of composition (max-min) of binary relation. In the max-min composition the membership grades (degree of association of fuzzy relations) are combined by finding the maximum of the minimums of the corresponding membership grades of the rows of the first fuzzy relation and the columns of the second fuzzy relation.

The equation

$$A \circ R = B \quad (1)$$

where  $\circ$  denotes the max-min composition [2] is called a fuzzy relational equation. By fuzzy composition we mean that [1]

$$\max_{j \in J} \min (a_i, r_{ij}) = b_{ij} \quad \forall i \in I, \forall j \in J$$

If the fuzzy relation  $\mathbf{R}$  is viewed as a pure fuzzy system, then by the max-min composition the systems output  $\mathbf{B}$  can be easily computed without any restrictions given input  $\mathbf{A}$  and the system  $\mathbf{R}$  itself.

There are two types of problems that can arise [7].

- The first problem arises when input  $\mathbf{A}$  and output  $\mathbf{B}$  are given and relation  $\mathbf{R}$  is to be determined. This problem is called the “fuzzy identification” problem.
- The second problem is called the “fuzzy deconvolution” problem, where relation  $\mathbf{R}$  and the output  $\mathbf{B}$  are known and input  $\mathbf{A}$  is to be found.

The fuzzy deconvolution problem is the problem of our interest for which a solution is to be found using a parallel method that we propose and discuss in the following sections.

In this paper, we propose parallelization of fuzzy relational equations by using a row wise distribution of the matrix, which is used to represent the system of equations. Parallelization of fuzzy equations is appropriate in cases where the relational equation set involved is quite large. On a uniprocessor system, large problems could result in inappropriate consumption of memory and could also slow down the solution process. By using a parallel method the problem can be broken down into smaller independent problems, which can be solved simultaneously by each processor involved in the computation. This saves memory as against the uniprocessor system, by distributing the work to other processors. Also the time involved to solve the problem can reduce considerably. In the following sections, we will discuss details about the implementation approach and the algorithm used.

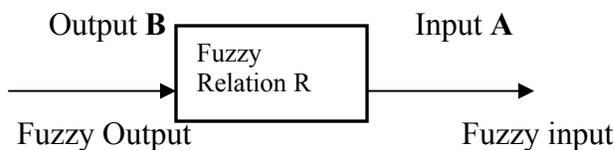
## 2. Fuzzy Relational Equations and Parallelization

### 2.1 Fuzzy Relational Equations

We are interested in finding a solution to the system of equation given in (1)

We can view  $\mathbf{A}$  as the fuzzy input,  $\mathbf{B}$  as the fuzzy output and  $\mathbf{R}$  as the fuzzy relation.

The following is a diagrammatic view of the problem of “fuzzy deconvolution” for which a solution is to be found.



**Fig 1:** The fuzzy deconvolution problem

We propose to find the input vector **A** given the fuzzy relation **R** and the output vector **B** by using a parallel method, details of which is given in the next few sections.

## 2.2 Existence of solution

The **height** of a vector or relation is defined as the highest membership grade in the set. The membership grade is a value in the interval [0,1] defining the degree of membership to the set in consideration [2].

The criterion for existence of solution for the fuzzy relation equation system is that the height of the relation **R** be less than the height of the input vector. For the problem at hand the output vector **B** acts as the input to the system [6]. So for our case

$$\text{height [B]} < \text{height [R]}$$

Once we can establish that this is true we know that the solution exists.

## 2.3 Notion of parallelization

Fuzzy relational equations can be easily parallelized since they are represented as a set by a matrix and each equation in this set is independent (does not depend on the solution of other equations) of each other.

The following example shows how sets of fuzzy equations look in a matrix form.

$$\mathbf{R} = \begin{vmatrix} 0.5 & 0.7 & 0.8 \\ 0.6 & 0.6 & 0.4 \\ 1.0 & 0.5 & 0.3 \end{vmatrix} \quad \mathbf{B} = \begin{vmatrix} 0.3 \\ 0.6 \\ 1.0 \end{vmatrix}$$

Once we get the system in this format we can find the input vector **A** that satisfies this equation given output vector **B** and relation **R**. Some applications could have systems to the order of thousands of equations.

By our approach, for larger problem, the matrix **R** can be broken down into several independent problems by using a row wise distribution. The number of equations distributed to each processor depends on the number of equations in the problem and also the number of processors available for the computation. Each processor also gets the known output vector **B**, which is used in finding the local solution. Once the local solution is found, the global solution is obtained by aggregating the local solutions computed from each processor. This can be achieved when each processor broadcasts its local solution to all other processors.

## 3. Message Passing Interface

**MPI** (Message Passing Interface) [3] is a widely used standard for writing message-passing programs to communicate between processors. In the message-passing model of

parallel computation, the processes executing in parallel have separate address spaces. Communication occurs when a portion of one process's address space is copied into another process's address space. A fixed set of processes is created at program initialization, one process per processor.

Some basic functions of **MPI** are listed below.

- MPI\_Init** : Initiate an MPI computation
- MPI\_Finalize** : Terminate a computation
- MPI\_Comm\_size** : Determine the number of processes
- MPI\_Comm\_rank** : Determine the process identifier
- MPI\_Send** : Send a message
- MPI\_Recv** : Receive a message

MPI has a rich set of easy to use library functions for parallel tasks. We chose to use **MPI** for parallelizing fuzzy equations since it is easy to use and its implementation is readily available.

## 4. Implementation

The relational matrix **R** and known output vector **B** are stored in files. Once MPI is initialized, the matrix and known vectors files are accessed and the values are read in row wise.

The total number of rows going to each processor is determined by using the following formula:

$$\text{Local number of rows} = \text{total number of rows} / \text{number of processors}$$

In case the number of rows cannot be broken down equally, the last processor will be given the extra rows as a simple solution to this problem. Since each processor reads from the same file, the start and end points of reading are determined initially to avoid some processors reading data meant for other processors.

### 4.1 Finding the Upper Level Solution

Given fuzzy relation **R** and output vector **B** of equation (1), the aim is find the input vector **A**. The solution that satisfies this relational equation is given as follows [2]

$$\begin{aligned} \mathbf{R} \alpha \mathbf{B} &= \mathbf{1} \quad ; \text{if } r \leq b \\ &= b \quad ; \text{if } r > b \end{aligned} \quad (2)$$

where  $\alpha$  is an operator that determines the output value depending on the values of a row of **R** and vector **B** as given in the formula (2).

Once the local solution is found in each processor using the above formula, it is broadcast to all other processors. Each processor in turn stores the solution coming from other processors according to the rank of the processors so that each processor will have the entire solution in the same sequence.

## 4.2 Example for Relational Matrix

If the relational matrix **R** is given as

$$\mathbf{R} = \begin{pmatrix} 0.8 & 0.5 & 0.4 & 0.3 & 0.9 & 0.6 & 0.9 & 0.5 & 0.9 & 0.5 & 1.0 \\ 0.7 & 0.9 & 0.4 & 0.6 & 0.9 & 0.9 & 0.8 & 0.2 & 0.2 & 0.5 & 0.6 \\ 0.2 & 0.4 & 0.4 & 0.2 & 0.3 & 0.0 & 0.5 & 0.7 & 1.0 & 0.4 & 0.3 \\ 0.3 & 0.2 & 0.4 & 0.5 & 0.4 & 0.5 & 0.3 & 1.0 & 0.8 & 0.9 & 0.7 \\ 0.4 & 0.1 & 0.9 & 0.5 & 0.9 & 0.7 & 0.3 & 0.7 & 0.5 & 0.9 & 0.9 \\ 0.2 & 0.3 & 1.0 & 0.7 & 0.9 & 0.3 & 0.8 & 0.7 & 0.0 & 0.3 & 0.9 \\ 0.1 & 0.2 & 0.9 & 0.7 & 0.6 & 0.9 & 0.1 & 0.9 & 0.6 & 0.4 & 0.0 \\ 0.0 & 0.2 & 0.0 & 1.0 & 0.3 & 0.2 & 0.1 & 0.3 & 0.2 & 0.9 & 0.2 \\ 0.7 & 0.4 & 0.6 & 0.3 & 0.6 & 1.0 & 0.8 & 0.2 & 0.5 & 0.4 & 0.0 \\ 0.0 & 1.0 & 0.8 & 0.1 & 0.3 & 0.8 & 0.6 & 1.0 & 0.6 & 0.5 & 0.2 \\ 0.4 & 0.7 & 0.5 & 0.6 & 0.9 & 0.8 & 0.9 & 0.6 & 0.5 & 0.6 & 0.9 \end{pmatrix}$$

This is an 11 x 11 relational matrix representing a set of 11 Fuzzy equations. The matrix is square since it has equal number of rows and columns.

## 4.3 Example for Input Matrix

The fuzzy input is a 1 x 11 matrix as given below in the transposed form

$$\mathbf{B}^T = \begin{pmatrix} 0.9 & 0.4 & 0.3 & 0.5 & 0.6 & 0.1 & 1.0 & 0.8 & 0.9 & 0.3 & 0.2 \end{pmatrix}$$

Each value represents the value on each row of the output matrix **B**. This is read simultaneously into each processor

## 5. Solution

For the example, suppose there are only two processors. Since there are 11 rows, there cannot be an equal distribution to each processor. In this case as mentioned before, we send 5 rows to each processor, which accounts for 10 rows and the eleventh row, which is the extra row for this case, is sent to the last processor (second processor). Observe that since the height of the relational matrix is not less than the height of the output matrix, a solution exists. The data in each processor looks as follows

**Processor 1:**

$$\begin{pmatrix} 0.8 & 0.5 & 0.4 & 0.3 & 0.9 & 0.6 & 0.9 & 0.5 & 0.9 & 0.5 & 1.0 \\ 0.7 & 0.9 & 0.4 & 0.6 & 0.9 & 0.9 & 0.8 & 0.2 & 0.2 & 0.5 & 0.6 \\ 0.2 & 0.4 & 0.4 & 0.2 & 0.3 & 0.0 & 0.5 & 0.7 & 1.0 & 0.4 & 0.3 \\ 0.3 & 0.2 & 0.4 & 0.5 & 0.4 & 0.5 & 0.3 & 1.0 & 0.8 & 0.9 & 0.7 \\ 0.4 & 0.1 & 0.9 & 0.5 & 0.9 & 0.7 & 0.3 & 0.7 & 0.5 & 0.9 & 0.9 \end{pmatrix}$$

**Processor 2:**

$$\begin{pmatrix} 0.2 & 0.3 & 1.0 & 0.7 & 0.9 & 0.3 & 0.8 & 0.7 & 0.0 & 0.3 & 0.9 \\ 0.1 & 0.2 & 0.9 & 0.7 & 0.6 & 0.9 & 0.1 & 0.9 & 0.6 & 0.4 & 0.0 \\ 0.0 & 0.2 & 0.0 & 1.0 & 0.3 & 0.2 & 0.1 & 0.3 & 0.2 & 0.9 & 0.2 \\ 0.7 & 0.4 & 0.6 & 0.3 & 0.6 & 1.0 & 0.8 & 0.2 & 0.5 & 0.4 & 0.0 \\ 0.0 & 1.0 & 0.8 & 0.1 & 0.3 & 0.8 & 0.6 & 1.0 & 0.6 & 0.5 & 0.2 \\ 0.4 & 0.7 & 0.5 & 0.6 & 0.9 & 0.8 & 0.9 & 0.6 & 0.5 & 0.6 & 0.9 \end{pmatrix}$$

According to the formula (2) we have local solution (in transposed form)

*Both Processors*

**Processor 1:**

$$\begin{pmatrix} 0.1 & 0.1 & 0.2 & 0.1 & 0.1 \end{pmatrix}$$

**Processor 2:**

$$\begin{pmatrix} 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \end{pmatrix}$$

The local solution found in this manner is broadcast to each processor. The final solution in each processor looks as follows

**Final solution:**

$$\begin{pmatrix} 0.1 & 0.1 & 0.2 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \end{pmatrix}$$

This is a simple example of the solution procedure of fuzzy relational equation. Most applications will have larger set of fuzzy relational equations. We will in the following section, provide the experiments and results of computer simulations and compare the time required for computing solution of large fuzzy equation sets on several processors as against a uniprocessor system.

## 6. Experiments and Results

Experiments were carried out for various sizes of the relational matrix. Given below are the results obtained for 11 x 11, 25 x 25, 100 x 100 and 400 x 400 dimensional matrices. The values were generated randomly and experiments were run on SunBlade100 Unix systems with UltraSparc Iie processors, having a clock speed of 500MHZ and each machine having about 384MB of memory. The table's show that the experiments were carried out for a total of 11 processors and the time in seconds has been calculated. The time on each processor is displayed according to the processor rank or identification number in the pool of processors. We show that as the numbers of processors are increased, the time required for finding solution decreases. Also as the problems get bigger the savings in time is more significant as seen from the tabulated results.

Number Of Processors													Max
1	0.033												0.033
2	0.061	0.091											0.091
3	0.172	0.113	0.175										0.175
4	0.157	0.144	0.190	0.262									0.262
5	0.312	0.183	0.222	0.277	0.261								0.312
6	0.231	0.178	0.255	0.292	0.253	0.295							0.295
7	0.215	0.202	0.156	0.212	0.190	0.165	0.168						0.215
8	0.224	0.202	0.138	0.209	0.207	0.178	0.175	0.188					0.224
9	0.115	0.111	0.113	0.110	0.100	0.102	0.084	0.081	0.092				0.115
10	0.115	0.116	0.118	0.104	0.098	0.100	0.084	0.074	0.091	0.096			0.118
11	0.040	0.036	0.038	0.039	0.033	0.034	0.032	0.029	0.030	0.027	0.022		0.040
Time on Processor(sec)	0	1	2	3	4	5	6	7	8	9	10		

**Table 1: Experiment with 11 Fuzzy Equations**

Number Of Processors													Max
1	0.016												0.016
2	0.042	0.112											0.042
3	0.041	0.011	0.145										0.145
4	0.043	0.015	0.017	0.013									0.043
5	0.040	0.014	0.017	0.014	0.010								0.040
6	0.046	0.023	0.025	0.022	0.019	0.016							0.046
7	0.057	0.035	0.038	0.034	0.029	0.031	0.027						0.057
8	0.051	0.032	0.033	0.031	0.026	0.029	0.023	0.021					0.051
9	0.053	0.036	0.038	0.036	0.032	0.039	0.030	0.029	0.027				0.053
10	0.070	0.054	0.060	0.057	0.052	0.053	0.052	0.040	0.040	0.044			0.070
11	0.059	0.050	0.054	0.052	0.044	0.047	0.049	0.042	0.038	0.040	0.035		0.059
Time on Processor(sec)	0	1	2	3	4	5	6	7	8	9	10		

**Table 2: Experiment with 25 Fuzzy Equations**

Number Of Processors												Max
1	3.251											3.251
2	0.864	0.834										0.864
3	0.431	0.401	0.405									0.431
4	0.265	0.238	0.240	0.236								0.265
5	0.228	0.203	0.205	0.202	0.198							0.228
6	0.199	0.175	0.178	0.173	0.169	0.168						0.199
7	0.188	0.166	0.170	0.170	0.164	0.161	0.159					0.188
8	0.185	0.167	0.167	0.166	0.164	0.161	0.154	0.158				0.185
9	0.155	0.135	0.138	0.135	0.130	1.33	0.129	0.126	0.124			0.155
10	0.126	0.116	0.114	0.112	0.107	0.111	0.102	0.104	0.096	0.102		0.126
11	0.117	0.105	0.104	0.101	0.096	0.094	0.095	0.097	0.092	0.089	0.088	0.117
Time on Processor(sec)	0	1	2	3	4	5	6	7	8	9	10	

**Table 3: Experiment with 100 Fuzzy Equations**

Number Of Processors												Max
1	245.3											245.3
2	196.2	196.5										196.5
3	152.2	152.9	152.6									152.9
4	83.11	83.42	83.11	83.70								83.70
5	51.10	51.22	51.08	51.19	51.07							51.22
6	37.74	37.72	37.72	37.73	37.71	37.78						37.78
7	30.15	30.24	30.25	30.19	30.28	30.33	30.24					30.25
8	25.87	25.88	25.86	25.83	25.86	25.88	25.85	25.8				25.87
9	19.67	19.65	19.65	19.65	19.65	19.55	19.65	19.65	19.64			19.67
10	15.55	15.54	15.54	15.54	15.54	15.54	15.54	15.38	15.31	15.5		15.55
11	12.29	12.28	12.27	12.26	12.26	12.26	12.26	12.25	12.25	12.2	12.2	12.29
Time on Processors (sec)	0	1	2	3	4	5	6	7	8	9	10	

**Table 4: Experiment with 400 Fuzzy Equations**

## 7. Discussion

We observe from the tables that parallelization works well for problems involving a large number of equations. The tables also show a maximum value for each row in the last column. This value is the minimum amount of time that a pool of processors has to wait until a solution can be obtained. Table 1 is the experiment conducted for an 11 x 11 matrix. This problem is very small involving only 11 rows. We observe that the time for computation does not show any significant decrease as we increase the number of processors.

We can think of the following simple equation

## **Total time for computation = Parallelization time + solving time**

For smaller problems the parallelization time could be significantly higher due to the communication overheads encountered while exchanging data between processors. Same is the case for a 25 x 25 matrix involving only 25. Table shows the results obtained for this case. Observe that there is not much decrease in time since the problem is still relatively small. Table 3 is the experiment conducted for a 100 x 100 matrix involving 100 rows. The results obtained for this experiment however shows a significant decrease in time as the number of processors is increased. Here we start to see the effectiveness of parallelization. Table 4 is the experiment conducted for larger matrix of size 400 x 400. We can observe that there is quite a bit of saving in time as the number of processors is increased and as the problem gets bigger. In fact a super linear speedup is achieved for larger problems (Tables 3 and 4). This result is due to the size of the problem that cannot be solved efficiently on a single processor (or a small number of processors).

## **8. Conclusion**

Through this paper we proposed to show that fuzzy relational equations of the form  $A \circ R = B$ , could be parallelized to find a solution. Our experiments show that for large problems, parallelization works efficiently as the number of processors is increased. As we can see from the results of 11 x 11 and 25 x 25 relational matrix, since the problems are relatively small, parallelization does not work very well. Parallelization has communication overheads for the processors involved and for smaller problems this overhead overcomes any advantage of distributing the problem to several processors. As the problems get larger as in the case of 100 x 100 and 400 x 400 relational matrices as shown in the experiments, these overheads become less significant and parallelization works by reducing the time required as against the uniprocessor system. The entire matrix is not stored on one particular processor and only significant portion is read into each processor saving memory and reducing computation time. This makes parallelization a good choice for solving problems involving large number of fuzzy relational equations.

## **9. Acknowledgements**

This work is supported by Northland Advanced Transportation systems Research Laboratories (NATSRL), University of Minnesota Duluth.

## **10. References**

- [1]Y. Wu, S. Guu and J. Liu, “*An Accelerated Approach for Solving Fuzzy Relational Equations With a Linear Objective Function,*” *Fuzzy Sets Syst.*, vol 10 No.4, Aug 2002.
- [2]G.J. Klir and B. Yuan, “*Fuzzy Sets and Fuzzy Logic*”, NJ: Prentice-Hall, 1995
- [3]W.Gropp, E. Lusk and A. Skjellum, “*Using MPI: Portable Parallel Programming with the*

*Message-Passing Interface*", second edition, MIT press, 1999

[4]T. Terano, K.Asai and M. Sugeno, "*Fuzzy Systems Theory and Its Applications*", Academic Press, 1992

[5]K. Tanaka,"*An Introduction to Fuzzy Logic for Practical Applications*", Rassel Inc, 1996

[6]M.Stachowicz, L.Beall,"Fuzzy Logic Package for Mathematica", Wolfram Research Inc, 2000

[7]L. Wang, "*A Course in Fuzzy Systems and Control*", NJ: Prentice Hall, 1996



## **Appendix B**

### **Sigmoidal Function Based Optimal Weighted Vector Directional Filters**

## Appendix B

### Sigmoidal Function Based Optimal Weighted Vector Directional Filters

Rastislav Lukac<sup>1</sup> and Marian S. Stachowicz<sup>2</sup>

<sup>1</sup> Slovak Image Processing Center

Jarkova 343, 049 25 Dobsina, Slovak Republic, lukacr@ieee.org

<sup>2</sup> Laboratory for Intelligent Systems, Department of Electrical and Computer Engineering, University of Minnesota Duluth, 274 MWAH,

10 University Drive Duluth, MN 55812, USA, mstachow@d.umn.edu

#### Abstract

In this paper, we provide a new directional optimization of a recently developed class of weighted vector directional filters (WVDFs) that represent a powerful tool for removing of impulsive noise, bit errors and color artifacts in color images. Depending on the weight coefficients, the WVDFs can be designed to perform a wide range of smoothing operations. The proposed optimization process based on sigmoidal approximation of the sign function and utilizing local information allows to adapt the WVDF behavior to statistical properties of both noise and useful signal, saves the memory space and is easy to implement. The proposed sigmoidal function based optimal WVDFs find the application in denoising of standard color images and digitized old color photos, they are able to remove impulsive noise and outliers, and provide excellent signal-detail and color chromaticity preservation.

**Note:** The complete paper in this appendix is available on the CD as a pdf file.

## **Appendix C**

**Presentation at Mn/DOT Duluth on the Research Day  
Vehicle Detection and Tracking in Traffic Control Systems**

## Appendix C

### Presentation at Mn/DOT Duluth on the Research Day Vehicle Detection and Tracking in Traffic Control Systems

Marian S. Stachowicz

A. Burlev, K. Seleznev,

N. Andrisevic, N. Agarwal, K. Vuppla

2 Laboratory for Intelligent Systems, Department of Electrical and Computer Engineering, University of  
Minnesota Duluth, 274 MWAH,  
10 University Drive Duluth, MN 55812, USA, mstachow@d.umn.edu

#### Abstract

This presentation was made by Marian S. Stachowicz at the Mn/DOT Duluth on the Research Day, i.e. on 11/13/03. This presentation has been organized as follows

- Research Objective
- Summary of Research Methodology
- Summary of Previous Work
- Literature Search
- Implementation
- Comparison
- Expected Benefits and Users of this Research
- Snapshots of the software
- Possible Applications

**Note:** The complete presentation in this appendix is available on the CD as a ppt file.