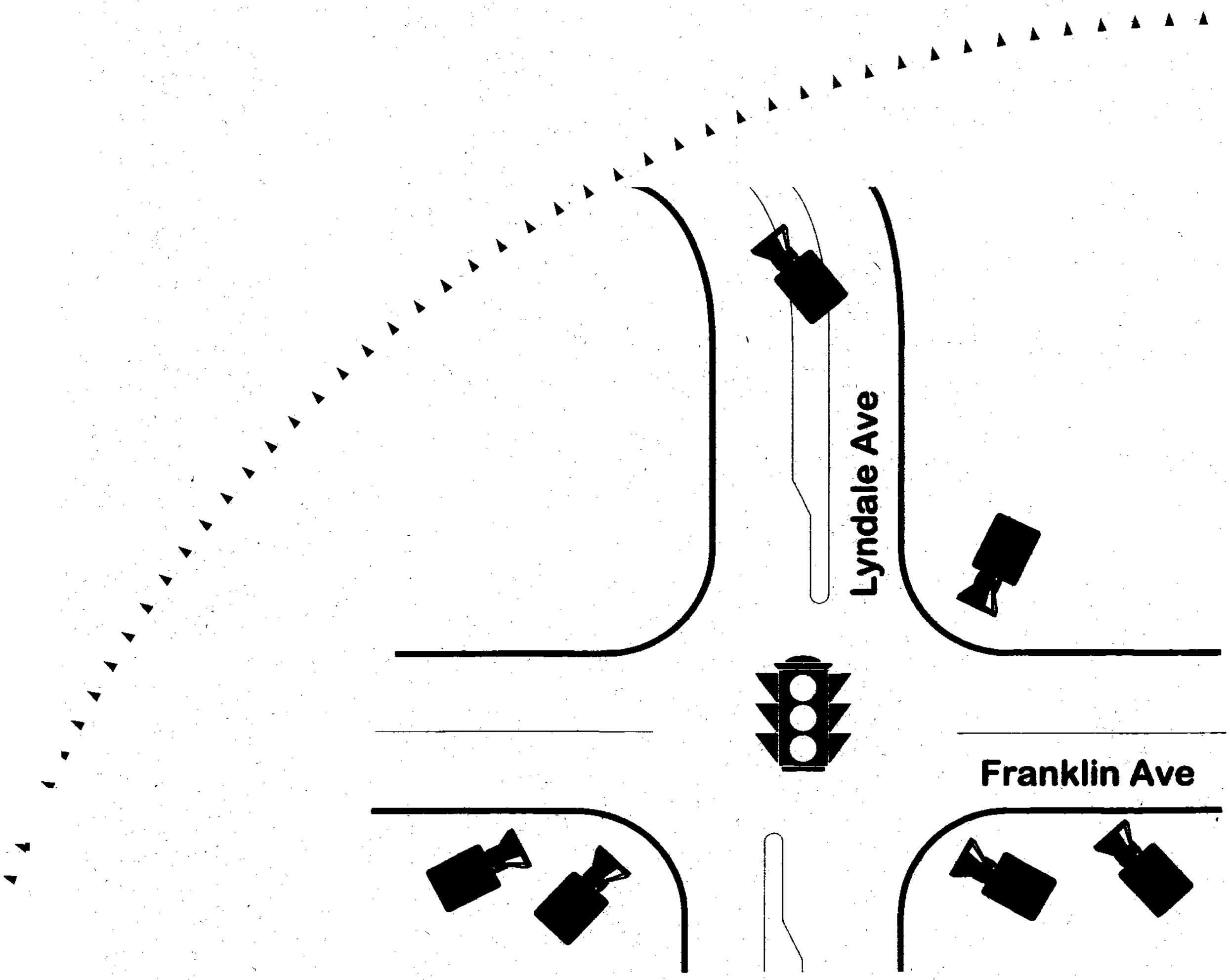
Report Number 96-33

MINNESOTA DEPARTMENT OF TRANSPORTATION

Research



Development and Application of On-Line Strategies for Optimal Intersection Control (Phase III)

Minnesota Guidestar®

UNIVERSITY OF MINNESOTA
CENTER FOR
TRANSPORTATION
STUDIES

Technical Report Documentation Page

1. Report No. MN/RC - 96/33	2.	3. Recipient's Accession No.	
4. Title and Subtitle DEVELOPMENT AND APPLICATION OF ON-LINE STRATEGIES FOR OPTIMAL INTERSECTION CONTROL (PHASE III)		5. Report Date October 1996 6.	
7. Author(s) *Dr. Eil Kwon, Dr. Yorgos J. Stephanedes, Dr. Xiao Liu, Sabhari Chidambaram, Charalambos Antoniades		8. Performing Organization Report No.	
9. Performing Organization Name and Address Department of Civil Engineering University of Minnesota Minneapolis, MN 55455 * ITS Institute, Center for Transportation Studies University of Minnesota Minneapolis, MN 55455		10. Project/Task/Work Unit No. 11. Contract (C) or Grant (G) No. (C) 71788 TOC #137	
12. Sponsoring Organization Name and Address Minnesota Department of Transportation 395 John Ireland Boulevard Mail Stop 330 St. Paul, Minnesota 55155		13. Type of Report and Period Covered Final Report 1994-1996 14. Sponsoring Agency Code	
15. Supplementary Notes			

16. Abstract (Limit: 200 words)

The previous phases of this research reviewed and tested existing intersection control algorithms in a simulated environment. Further, a machine-vision detection system with four cameras was installed at the intersection of Franklin and Lyndale Avenues in Minneapolis, Minnesota, to develop a live intersection laboratory.

Phase III enhanced the live laboratory with two additional cameras covering the intersection proper and the extended approach of southbound Lyndale Ave. A comprehensive operational plan for the laboratory was developed and a new microscopic simulator for the laboratory intersection was also developed. Two types of new intersection control strategies, i.e., one with link-wide congestion measurements and the other based on neural-network approach, were developed and evaluated in the simulated environment. Further, using the data collected from the machine-vision detection system, an automatic procedure to estimate the intersection delay was also developed and applied to compare the performance of fixed-timing control with that of the actuated control strategy.

17. Document Analysis/Descriptors Intersection Control Machine-vision Cellular Automata		18. Availability Statement No restrictions. Document available from: National Technical Information Services, Springfield, Virginia 22161	
19. Security Class (this report) Unclassified	20. Security Class (this page) Unclassified	21. No. of Pages 127	22. Price

DEVELOPMENT AND APPLICATION OF ON-LINE STRATEGIES FOR OPTIMAL INTERSECTION CONTROL (PHASE III)

Final Report

Prepared by

Dr.. Eil Kwon
ITS Institute, Center for Transportation Studies
University of Minnesota
Minneapolis, MN 55455

Dr. Yorgos J. Stephanedes
Dr. Xiao Liu
Sabhari Chidambaram
Charalambos Antoniades
Department of Civil Engineering
University of Minnesota
Minneapolis, MN 55455

October 1996

Published by

Minnesota Department of Transportation
Office of Research Administration
200 Ford Building Mail Stop 330
117 University Avenue
St Paul Minnesota 55155

This report represents the results of research conducted by the authors and does not necessarily represent the views or policy of the Minnesota Department of Transportation. This report does not contain a standard or specified technique.

The authors and the Minnesota Department of Transportation do not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to this report.

TABLE OF CONTENTS

List of Tables

List of Figures

EXECUTIVE SUMMARY

The most common type of intersection control strategy determines the timing plan off-line using arterial or network optimization software. The resulting timing plan is stored in the computer memory for implementation based on on-line criteria. The computer models used for this type of control include TRANSYT7F, NETSIM, SIGOP and AAP. More recently, traffic demand-responsive control strategies with on-line timing generators and adaptive features have been introduced. These include SCOOT, SCATS, PRODYNE and OPAC. Although each software has been individually tested by various agencies [1], no comprehensive effort has been made to quantify and evaluate their performance, especially in terms of the applicability to detection with both loops and video image processing. Further, recent developments in the area of neural networks provide new dimensions in improving the performance of real time intersection control.

In the previous phases of this research, Phases I and II, existing intersection control algorithms were reviewed and tested in a simulated environment. In addition, a video detection system was installed at an intersection where a live laboratory would be developed. The current research, Phase III, extends the previous research efforts by developing a comprehensive operational plan of the live laboratory, where various intersection operational strategies can be tested in a real environment.

A new microscopic simulator was developed for the laboratory and this was a major contribution in Phase III. It uses a cellular automata approach with a simplified car-following model, which was also developed in this research. The new simulator was tested with the data collected from the intersection laboratory and used for evaluating a new control strategy. The new intersection control developed in this research is based on the level of congestion quantified with a newly defined congestion index. The congestion index uses point measurements from either loop or machine vision detection systems. Results from evaluation tests of the new control indicated its potential for improved performance when compared to pretimed and stopline-acutation control.

A neural-network based adaptive control strategy for urban network operations was also developed and tested in a simulated environment with NETSIM. In this scheme, a signal phase is represented by a neuron and all neurons are connected in such a way that no conflict phases occur. The objectives of this connection scheme are to coordinate phases between intersections, and to use queue length to change signal phases adaptively in response to traffic demand in each approach. The neural-network based control was tested using the NETSIM simulator and its performance was compared with that of stopline-actuation based control. The test results showed improved performance by the neural network based control for all three test cases. Because of the scope of this project the new strategy was tested in an isolated intersection.

I. INTRODUCTION

I.1 Background

The most common type of intersection control strategy determines the timing plan off-line using arterial or network optimization software. The resulting timing plan is stored in the computer memory for implementation based on on-line criteria. The computer models used for this type of control include TRANSYT7F, NETSIM, SIGOP and AAP. More recently, traffic demand-responsive control strategies with on-line timing generators and adaptive features have been introduced. These include SCOOT, SCATS, PRODYNE and OPAC.

Although each software has been individually tested by various agencies [1], no comprehensive effort has been made to quantify and evaluate their performance, especially in terms of the applicability to detection with both loops and video image processing. Further, recent developments in the area of neural networks provide new dimensions in improving the performance of real time intersection control. In the previous phases of this research, Phases I and II, existing intersection control algorithms were reviewed and tested in a simulated environment. A machine-vision detection system was also installed at an intersection where a live laboratory would be developed. The current research, Phase III, extends the previous research efforts by developing the operational plan of the live laboratory, where various intersection operational strategies can be tested in a real environment. In addition, a new control strategy that can take advantage of video detection is developed and the feasibility of neural network-based control methods is determined.

I. 2 Research objectives

The major objectives of this research are:

- Development of an operational plan for the live intersection laboratory where control strategies can be tested in a real environment.
- Development of a new intersection control strategy that can be used with both loop and machine-vision detection systems.
- Determination of the feasibility of neural network-based methods for real time intersection control.

I.3 Report organization

This report summarizes the final results of the current project, Phase III. The second chapter briefly describes the existing intersection control strategies. Chapter three develops an operational plan for the live intersection laboratory including the example estimation results of the intersection delay using the machine-vision detection data. The description of the new intersection simulator developed for the intersection laboratory is included in Chapter four. Chapter five develops a new intersection control strategy with machine-vision detection. The development of neural network-based intersection control strategies is included in Chapter six. Finally Chapter seven contains the conclusion and future research needs.

II. OVERVIEW OF INTERSECTION CONTROL STRATEGIES

II. 1. Introduction

Demand responsive traffic-control systems change signal timings in response to traffic flow variations, measured or estimated in terms of traffic volumes and queue lengths. The effectiveness of these control systems depend on the accuracy of models of vehicle state and behavior, prediction models [15], the method of finding the optimum traffic control strategy [16], and the frequency at which a change in control strategy is considered. The development of such systems began in the early 1960's. SCOOT was the first such system that was a success, soon followed by SCATS, OPAC and others..

Several types of traffic model and control logic have been used along the history of demand responsive control-systems. The first of such systems were aimed at isolated intersections. As small networks started being of interest, the control logic had to cope with coordination, computing requirements and safety needs. This resulted in the application of a more traditional approach, based on small changes in offset, cycle and split. As computing power has been increasing , there has been a trend towards more-realistic traffic models.

Various control strategies for isolated intersections are discussed in the next section. Following that is a summary description of the network controllers SCOOT, SCATS, and OPAC, and a detailed description of the network controller CARS. A detailed description of SCOOT, SCATS and OPAC was included in the previous phase report.

II. 2. Control Strategies for Isolated Intersections

II. 2. 1. Introduction

A clear distinction must be made among traffic-control plans which are set in a pre-planned way without information on actual conditions, traffic-control plans which are responsive to overall changes in the pattern in traffic demand, and local actuation - responsiveness at individual intersections to individual vehicle arrivals. This section focuses on the latter: local actuations of signalization at an individual intersection [17]. The ideal vehicle actuated system would select in real time the optimum cycle based on measured parameters of the traffic.

II. 2. 2. Control based on vehicle intervals

This is the most frequently used strategy. In employing this strategy a passage detector is usually placed at each approach of the intersection. A vehicle is detected when approaching a red light and a "call" is sent to the controller for right of way. The controller receives the data from the detector and searches for an appropriate gap in the traffic having the green signal in order to change that signal from green to red. The *"allowable gap" or "vehicle headway"* is defined as the maximum time gap between vehicles, for which the controller would not change the phase.

Basic Control

This is the simplest type of vehicle interval strategy. Under basic control, there are four fundamental timing adjustments in each phase: *minimum green or initial green interval, vehicle interval, maximum green, and all red clearance*. When the green signal is given to waiting traffic, it is granted initially for a fixed short period. This is the minimum green setting. If the waiting traffic does not form a queue that reaches the detector, green will have the minimum green duration. If the waiting traffic extends past the detector or if additional traffic is approaching, the green will change to a predetermined sequence. Actuation by each additional vehicle extends the green but not in a commutative way, i.e., at each actuation the controller erases the unused part of the previous actuation interval. As long as vehicles are spaced apart shorter than the allowable gap, the green will be extended or held. Changes in the value of the allowable gap should occur from the point of view of minimizing delay. Poor adjustment of the allowable gap in dense traffic can cause excessive delays as a result of a prolonged green signal. The allowable gap can also be characterized as the amount of time needed for the last vehicle with right of way to clear the distance from the detector to the stop-line. In order to provide safe stopping time the detector must be positioned at the approach in relation to the approach speed.

Variable Initial Control

This type of control differs from the previous one in that the controllers have variable initial green. These controllers can count vehicles crossing the detector on red and thereby vary the initial or minimum green to clear the actual number of vehicles stored between the stop-line and detector. Using variable initial controllers the single minimum green setting is replaced by three settings: "minimum initial", "maximum initial" and "actuation to maximum initial". Maximum initial is set to clear the number of vehicles that can be stored in the space between the stop-line and the detector. "Actuation to maximum initial" is the setting that ensures that resulting green is sufficient to clear the number of vehicles, actually counted as waiting between the stop-line and the detector. This setting relates to the number of approaching lanes.

Passage Time

"Passage time" refers to the setting of green extension time on controllers that have separate independent attachments for the setting of the vehicle interval and the green extension time. During a gap the passage time enables the last vehicle with right of way to cover the distance between the detector and the stop-line before the change of phase. This feature is very important as it enables the detectors to be placed a certain distance before the stop-line to avoid the dilemma zone. Further, in light traffic conditions, this feature allows drivers to gain green from red before arriving at the intersection.

Time Waiting Gap Reduction

When a call is registered on a street on red the "time waiting gap reduction" feature decreases the vehicle interval exponentially so that shorter gaps are acceptable. As a result the allowable gap at the beginning of a phase is longer than at the end of the phase.

Number Waiting Gap Reduction

This feature is similar to the time waiting gap reduction except that the allowable gap is reduced according to the actual number of cars (queue) waiting at the approach having the red phase. The reduction is performed according to a predetermined relationship between the allowable gap and the queue size.

Gap Reduction by Density

Type 1

The goal is to detect the end of a platoon of vehicles. The allowable gap is set as slightly greater than the spacing of the vehicles over the past ten seconds. If the car spacing increases, indicating the end of a platoon, the gap between vehicles will exceed the allowable gap and a gap change will occur.

Type 2

The allowable gap decreases continuously at a rate depending on the traffic density.

Type 3

For this control strategy there are three important settings: "gap", "headway" and "waste". When the time between successive vehicles actuation exceeds the headway setting, the excess, i.e., the waste increment is accumulated. When the accumulated waste increment equals the waste time setting, the allowable gap is reduced immediately from the gap to the headway setting.

Volume-Density Control

This is the most complex control for isolated intersections. The allowable gap is not determined from a single parameter of the traffic on red or green but from the value of the allowable gap determined from the exponential reduction of the gap that is due to "calls" from the red phase approach, the value of the gap calculated from the number of cars waiting at the red phase and from the value of the gap calculated from the density calculation approach. At any time the allowable gap is the minimum of the three values described above.

Headway Density Control

The allowable gap is reduced in a way similar to that of the waiting procedure. However, the allowable gap is reduced in steps, and each step corresponds to a pre-set number of arrivals on red. The minimum allowable gap setting cannot be exceeded.

Volume Density Variation

This control strategy does not employ an allowable gap feature. It assigns priority to an approach according to the time waiting on red, the number waiting on red and the volume on green.

II. 2. 3. Control based on vehicle speed

The green time is extended in accordance with the vehicle speed measured at the detector position to enable the last vehicle to reach the stop-line at the onset of the next phase. The green extension and vehicle interval have the same value. This strategy was used in British controllers with double pneumatic detectors to measure speed.

II. 2. 4. Lane Occupancy Control

Green is extended by the presence of a vehicle within a long detector loop. An advantage claimed for this strategy is that it is possible to detect the presence of vehicles actually stopped near the stop-line, rather than the less direct method used in variable initial controllers. This method is of limited use.

II. 2. 5. System D

The British Department of Environment Strategies selected this strategy to replace the speed timed vehicle interval equipment. Two or more detectors, usually three, are located on each approach and designated *x, y* and *z* detectors. The controller is similar to the basic, except that provision is made whereby demands may originate from the *x* detector only, while all three detector may extend green.

II. 2. 6. Rate of occupancy control

This strategy employs long loop detectors on each approach similar to lane occupancy control. However, it differs significantly from lane occupancy in that the green may not terminate immediately on the absence of vehicle from the loops.

II. 2. 7. Green Extension System

The primary purpose of this system is to obviate drivers being placed in a dilemma at higher speeds. Two passage detectors are placed at distances which relate to the approach speed. If gap changes occur, most drivers will not be placed in a dilemma. If, however, gap changes do not occur in a given time, the first detector is disconnected and the controller operates on a basic control strategy, i.e., with one passage detector operating per approach.

II. 2. 8. Advance Call

A detector is located well in advance of the stop-line. When there is no demand on an opposing phase, an approaching vehicle passing the advance detector calls green from red, thereby allowing that vehicle to proceed without stopping. When a conflicting demand occurs, passage time is afforded to allow a vehicle passing over the advance detector to reach the detection system at the junction. At that stage the controller operates according to its main strategy, e.g., basic, lane occupancy, etc..

II. 2. 9. Variable Maximums

Some controllers have the facility to vary maximum greens between a high and a low limit. After each vehicle passes over the detector a timing condenser in the maximum green circuit is charged at a low rate for a certain period after which the charging reverts to a high rate. A modification of this strategy involves the extension of the maximum based on the flow at the end of the green instead of the flow throughout the green.

II. 2. 10. Summary

Most of the control strategies for isolated intersections, currently in use, were described. Vehicle interval control strategies are more popular than the others. The ideal strategy would minimize delay, stops and number and severity of accidents over a wide range of traffic conditions. In practice, optimizing on one of these factors may be to the detriment of others. Also, there has been an increasing need for controlling a network of intersections than an isolated one. The following section deals with the existing network controllers.

II. 3. Overview of network control strategies

II. 3. 1. SCOOT (Split Cycle Offset Optimization Technique)

SCOOT [2][3][4][5][6] consists of a traffic model and three optimizers that are used by the modeler to control a network of traffic signals. Signal control parameters are adjusted based on information obtained from loop detectors at upstream intersections. The three basic principles of SCOOT are: Measure of Cyclic Flow Profiles (CFPs) in real time, update of an on-line queue model continuously, and incremental optimization of signal settings. An on-line model stores and updates information on flow profiles, queue data, degree of saturation and congestion every four hours. The SCOOT coordination plan can be adjusted to the latest traffic conditions recorded by the CFPs. This is achieved by optimizing the splits, offsets, and cycle times, i.e., minimizing the performance index.

II. 3. 2. SCATS (Sydney Coordinated Adaptive Traffic System)

The basic premise behind the operation of SCATS [7][8][9][10][11] lies on its ability to explicitly describe data that reflect traffic flow conditions. Although SCATS does not use a mathematical traffic model coupled to a signal timing optimizer, the availability of explicit data describing traffic flow conditions is fundamental to the successful operation of the algorithms. SCATS controls traffic on an area basis instead of individual intersection basis. The principal purpose of the control system is to minimize overall stops and delay and, when traffic demand is at or near the capacity of the system, maximize the throughput and control the formation of queues. This is accomplished by controlling three basic parameters: Cycle lengths, offsets, and phase splits. SCATS is a reactive system, which takes information from the previous time slice and uses it as input for the current time slice.

II. 3. 3. OPAC (Optimization Policies for Adaptive Control)

OPAC [12][13][14] is a computational strategy for real-time demand-responsive traffic signal control. It provides results that are close to the theoretical optimum, requires on-line data that can be readily obtained from upstream link detectors, is suitable for implementation on existing microprocessors (it has not been applied yet on the field), and forms a decentralized control in a network. OPAC follows a decentralized philosophy, that is, it performs the optimization at each intersection through dynamic programming and a rolling horizon. More specifically, it minimizes vehicle delays and percentage of stopped vehicles.

II. 3. 4. CARS (Control Autoadaptivo para Redes Semaforizadas)

II. 3. 4. 1. Introduction

CARS is a third-generation demand responsive traffic control system for networks, arterials and isolated intersections [15]. It is intended to be used in signalized urban areas and follows a centralized, congestion oriented approach.

CARS uses a simulation model called PACKSIM, that handles vehicle packets which move according to an ad hoc model and stop in horizontal queues. Horizontal queues is a major emphasis area and their use, seeks to make the system capable of dealing with congested situations, evaluation of free space within a link and finding a progressive signalization in the arterials. The prediction models used are based on real time measurements. Before CARS makes a decision about a control change, it forecasts the network state. The algorithm tests control changes in variable-size sub-networks. The size of these sub-networks depends on the traffic conditions. Unlike other systems, CARS accepts arbitrary positioning and number of detectors. The number and position of detectors reflect upon the performance of the system, so some restrictions apply in order to preserve the effectiveness of control.

CARS is restricted to urban signalized areas with preferably short links (<400 meters) [15]. The system incorporates a graphical user interface that allows the user to specify the network characteristics easily.

II. 3. 4. 2. Control Timing

CARS can act as pre-timed, fixed control system or as a demand responsive system. CARS interacts with its environment through the detector information and the signal state from the intersections. Like OPAC, and contrary to SCOOT, CARS does not rely on the concept of cycle, split and offset but calculates acyclic settings.

The operation time of the system is discretized in intervals of *delta* seconds [15]. The length of the interval is user defined, usually around 2 seconds. The time that the system completes one cycle of operation (system control cycle, SCC) consists of several delta intervals (usually 5 intervals, i.e., 10 seconds). During the SCC the system performs the following actions: Updates the state of the moving vehicles according to the detector readings collected from the last SCC, sends to the controllers the traffic signal changes corresponding to the next SCC intervals, steps ahead and forecasts the state of the system at the end of the current system cycle, and plans-decides upon the control strategy of the next SCC.

Whereas the stepping ahead and planning processes occur every SCC, every 5 minutes of operation the system substitutes these processes with updating the external arrival predictors, and turning movement ratios, and reconsiders the size and shape of the variable size sub-networks around each demand responsive intersection. At the start of system operation, if no performance or detector data are available, the system acts as in fixed control mode until some time elapses (user defined period) and data are collected.

II. 3. 4. 3. Simulation Model

CARS uses a simulation model called PACKSIM [15]. PACKSIM can be considered a mesoscopic simulator (uses a simple car-following algorithm and does not depend on flow, density and speed). It models the vehicles in the network, predicts the state of the network and tests, prior to applying, control strategy changes. This simulation package is for use in densely signalized networks. The vehicle queues are modeled horizontally, expand backwards and can break up, resulting in links with varying-size groups of stopped vehicles.

The updating of the links in this simulation package is done using a two step procedure. First, PACKSIM predicts the free space of the accepting link, and second, performs the gradual grouping of already stopped and stopping vehicles at the stop-line of the intersections creating new vehicle packets.

II. 3. 4. 4. Vehicle Detection

PACKSIM can operate under an arbitrary number of detectors. Detectors can be positioned across the whole width of the link or just covering single or multiple lanes of the link [15].

II. 3. 4. 5. Feedback Process

The system updates the state of the network according to the detector readings taken from simulation. The process takes as inputs the state of the network at the beginning of the SCC and the readings from the detectors in each delta interval, and produces the state of the network at the end of the SCC.

During each delta interval of the SCC, the simulation model moves the vehicle packets through each detector, compares the real measurements with the model estimates and updates the vehicle packets and the control strategy state (i.e., value of the objective function) [15]. The vehicle packet updating is done by filling the free spaces between two positions or removing vehicles from the link. This could result in creating new vehicle packets or completely removing them in order to match the model with the real measurements.

II. 3. 4. 6. Updating the Predictors

The detector measurements taken in the feedback process are also used in updating the predictors used in the stepping ahead and planning processes (described later). When the system runs under PACKSIM, model measurements are used whereas real detector values are used when the system runs under real-world situations [15]. The vehicle arrival predictors are updated using data from a link-wide detector near the beginning of the link. Free-space predictors are updated after the vehicles are revised by PACKSIM. Free-space is the space available for new vehicles to enter the link.

II. 3. 4. 7. Updating the Turning Ratios

CARS classifies the vehicles measured by the upstream detector according to the phase sequence of the upstream intersection. The vehicle arrivals are accumulated along several phase sequences and are used to update the turning ratios according to an exponential filter [15].

II. 3. 4. 8. Stepping Ahead Process

Before considering the changes in the control strategy, the network state at the end of each control cycle is predicted. This is achieved by advancing the simulation model ahead in time using the vehicle arrival prediction at the network entrances, and the free space prediction at the exiting links of the network. The stepping ahead process begins at the end of the feedback stage when all detector readings are updated. Once all new detector readings are available (usually after the first delta interval) the simulation progresses ahead in time and the state of the system is forecast. A diagram of the stepping ahead process is shown in figure 2.1.

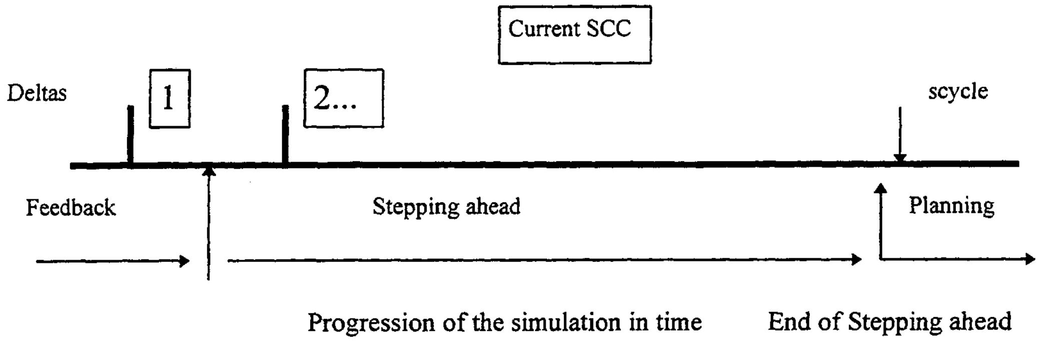


Figure 2.1. Stepping ahead procedure of CARS

II. 3. 4. 9. Prediction of External Arrivals

The forecasting of the network state at the end of each control cycle is done using the prediction of external arrivals. The predictor has to provide vehicle arrivals as number of vehicles and vehicle speed for the remaining time duration of the current control cycle [15]. This is a short-time predictor since each control cycle is of the order of 10 seconds. The prediction method is independent of historical information and adapts to the varying conditions by recalculating the prediction parameters each time the difference between the predicted and real (detected) values is significant.

II. 3. 4. 10. Prediction of External Free Space

The external free-space is predicted the same way as the external arrivals. A necessary condition is the presence of a detector on the entrance link of the network for which the prediction is made [15]. If no detector exists on that link, the link is not modeled. A moving average of five minutes is used [15].

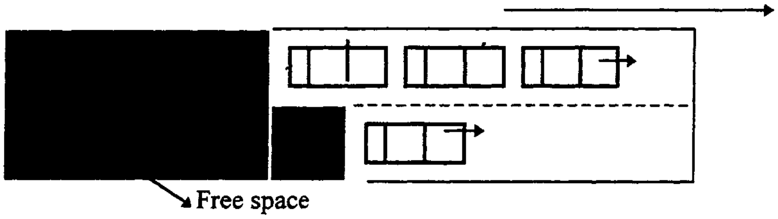


Figure 2.2. Free space concept in CARS

Free space is defined as the space of the link, which is not occupied by vehicles, from the upstream end of the link to the edge of the last vehicle, expressed in equivalent vehicles, as shown in figure 2.2 (shaded part).

II. 3. 4. 11. Planning Process

The planning process considers changing the control strategy of the network to minimize a certain objective function. Starting at the end of the stepping ahead process, for each intersection, the system evaluates the effects of each control strategy during a time horizon, whose duration is dynamically calculated [15], see figure 2.3.

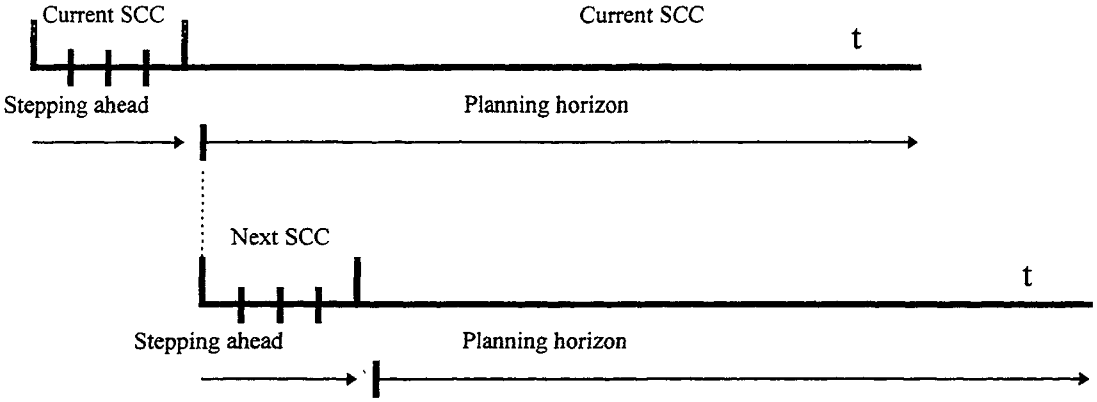


Figure 2.3. The planning process of CARS

Owing to high computational requirements, the planning process is carried out at a sub-network (explained later) around each intersection [15]. The size of the sub-network is variable and dependent on the traffic conditions.

The objective functions that CARS considers are [15]:

1. Number of stopped vehicles.
2. Queue length.
3. Queue volume.
4. Average speed.
5. Exiting volume.

II. 3. 4. 12. Planning (variable) Sub-Networks

As mentioned above the planning task considers changing the control strategy in each intersection of the network [15]. In order to minimize the computational needs of the task the process is carried out at a sub-network around the intersection. The free-space and the vehicle arrivals at the sub-network have to be predicted.

The size of the sub-network depends on the time horizon along which the control strategy change is evaluated, the network state during the time horizon and the control strategies in the network. Further minimization of the computational needs require that the sub-network depend on the time horizon and the average speed in the links of the network. The size of the sub-network is reconsidered every 5 minutes. At the beginning of operations the sub-networks are initialized at their maximum value. In very congested conditions, judging from the average speed, the sub-networks are set on their minimum value. An example of a planning sub-network is shown in figure 2.4 [15].

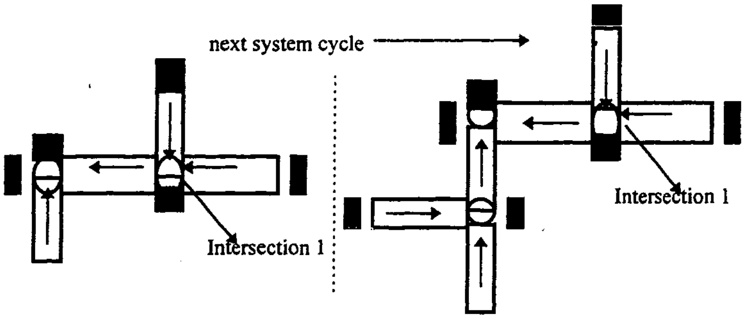


Figure 2.4. An example of a variable planning sub-network around intersection 1

II. 3. 4. 13. Planning Order

During the planning process for a particular intersection the control strategies in the adjacent intersections are influencing the results since they clearly influence the vehicle arrivals and free-space predictions for that intersection. Therefore, the intersection at which a control strategy is decided first, "imposes" its control strategy on the remaining intersections. CARS plans the critical intersections in a network first, then the arterials in a reverse order from the one followed in the next planning process, and finally the rest of the intersections [15].

II. 3. 4. 14. Prediction of Internal Arrivals

Internal arrivals refer to the vehicles entering a link which is internal to the network. The internal arrival prediction is needed for the planning process in order to provide vehicles to the planning sub-networks. These predictions supply the system with the number of vehicles and individual speeds [15].

As described in updating of turning ratios the vehicle arrivals are classified according to the current phase in the upstream intersection. The predictor of arrivals in one phase is updated according to an exponential filter whose parameter is fixed and determined during testing. The vehicle speed is predicted using the same method but is averaged according to the number of vehicles.

II. 3. 4. 15. Prediction of Internal Free-Space

The free-space in a link is measured as the number of vehicles that can enter the link taking into account the speed of the already existing vehicles. This prediction is needed for the planning process. As for the prediction of the internal arrivals, free-space is predicted using the PACKSIM model since detectors do not provide free-space measurements [15]. The free-space is again classified according to the current stage in the upstream intersection and is updated using an exponential filter which uses a fixed value for its parameter as described above.

II. 3. 4. 16. Testing

CARS has been tested by simulation in multiple scenarios, including isolated intersections, arterials and networks. The AIMSUN simulator has been used for these tests. In particular, for performing the simulation, AIMSUN sends detector counts to CARS and receives phase endings from CARS.

During simulation, the internal CARS simulation model, PACKSIM, is first adjusted to the field conditions. For these conditions, the local link parameters, average vehicle speed, average distance between vehicles and average free speed have to be specified. Similarly, the AIMSUN model has to be adjusted.

CARS has been tested against fixed-time control strategies. The tests have been performed at both light and congested traffic conditions. In both cases CARS has showed a 5-30% improvement over fixed time control.

II. 3. 4. 17. Summary

CARS is a third generation demand-responsive traffic control system for networks, arterials and isolated intersections. It uses the concept of rolling horizon to test control strategy changes and the concept of adaptive centralized control based on small variations. The simulation model, PACKSIM, is used to evaluate the effectiveness of each control strategy. CARS is intended to be used in densely signalized urban areas and follows a centralized, congestion-oriented approach. PACKSIM, the simulation model, deals with packets of vehicles moving according to an ad hoc "packet following" model and stopping in horizontal queues. CARS uses prediction models based on real-time measured traffic information. Network state forecasting is performed before considering the changes in a control strategy. The control strategies are tested on sub-networks around the intersection of interest. CARS accepts an arbitrary number and positioning of detectors, but certain rules apply to assure the effectiveness of the demand-responsive control.

II. 4. Summary

Control strategies for isolated intersection and networks were discussed. The ideal strategy would minimize delay, stops and number and severity of accidents over a wide range of traffic conditions. A trade-off is required as optimizing on one of these factors may be to the detriment of others. Owing to the complexity of traffic behavior and changing traffic conditions, no method can determine the optimal type of control for a given intersection or network. Testing these different strategies in the field and evaluating their performance would be desirable and can lead to the development of advanced strategies that share the best of the features of the existing ones. Such testing and evaluation could be accomplished in an intersection laboratory environment.

III. DEVELOPMENT OF INTERSECTION LABORATORY OPERATIONAL PLAN

III. 1. Introduction

Intersection control strategies have long been evaluated by simulation. Such strategies cover a wide range, from simple time-of-day control with pre-determined timing plans to sophisticated demand responsive control. Because of the complexity of traffic behavior and varying prevailing traffic conditions, no method can determine the optimal type of control for a given intersection or network.

Testing new control strategies in a real traffic environment, and thus, refining the control schemes prior to full scale implementation is of critical importance in developing efficient and robust real-time control strategies. Developing an intersection laboratory, where new control strategies can be tested with real traffic is one of the essential elements in developing comprehensive real time traffic management strategies. A laboratory also forms the best platform for field testing incident detection algorithms. Several additional applications of this laboratory would make it priceless to the research community. This chapter addresses the development of an operational plan for a live intersection laboratory with enhancements to the infrastructure that already exists at the intersection.

III. 2. Overview of the intersection laboratory

Prior to initiating the project and following consultations with the traffic engineers from the Minnesota Department of Transportation and the City of Minneapolis, the intersection of Franklin and Lyndale Avenues in downtown Minneapolis had been selected as the site for setting up this intersection laboratory.

The intersection of Franklin and Lyndale in downtown Minneapolis caters to heavy traffic that results in frequent congestion and delays. With a high incident rate, this intersection could provide valuable data for future development of intersection incident detection strategies. Further, the location and distance of the intersection from adjacent intersections make it possible to operate as an isolated intersection, but it can be easily incorporated into coordinated network control. For the above reasons, this site was considered ideal for the laboratory.

In an earlier phase of the project, a machine-vision detection system was purchased and installed at the intersection. In addition, the intersection is equipped with an actuated controller unit. The machine-vision system monitors vehicles on the roadway via processing of video images and provides detector output to the traffic controller. The detector zones in the machine-vision system are user-definable and easy to redefine. Such flexibility makes the machine-vision system all the more suitable for such a laboratory.

III. 3. Existing infrastructure at the intersection laboratory

III. 3. 1. Introduction

In the development of a live intersection laboratory, it is imperative to have detection equipment that permits changes to detector location. The machine-vision system provides this kind of flexibility and furnishes data that cannot be acquired from loop detectors. For this reason, a machine-vision system was installed at the intersection. The intersection is also equipped with an EPAC300 controller unit for actuated control. The features of the machine-vision system and controller unit are discussed in detail in the following sections.

III. 3. 2. Machine vision detection system

The machine-vision vehicle detection system provides a way to automate traffic surveillance and detection using machine vision technology. The system includes hardware and software along with other commercially available components such as the host computer and image sensors.

The machine-vision unit located at the intersection is connected to four cameras. Each camera looks upon the four approaches to the intersection and sends the video signals to the image processor within the unit. These cameras are installed on poles, located at the corners of the intersection, at a height of approximately 35 ft. The orientation and location of the cameras is based on specific guidelines. These guidelines contribute significantly to the efficiency of the system in detecting vehicles.

The detection system and cameras were installed at the Franklin and Lyndale intersection in February 1994. For optimal use, virtual detectors were placed in areas of the image where there were no contrasting features. Detectors are placed at an intersection depending on the use of their output for fulfilling specific objectives. In the case of this intersection, the objective has been to set up a laboratory for testing various control strategies, and for collecting real time data for various studies and research.

The computer that supervises the operations at the machine-vision unit is located at the office of the City of Minneapolis. Communications between the unit at the intersection and the computer at the City office occurs through special cables laid for this purpose. These cables allow direct access to the machine-vision unit.

For control of the intersection, the machine-vision processor is connected to a EPAC300 controller unit at the intersection through the External Interface Module (EIM) port. Testing a new control strategy at the intersection requires the modification of the inputs from the machine-vision unit to the controller.

III. 3. 3. EPAC300 Actuated controller unit

III. 3. 3. 1. Basic Interface Inputs

The basic inputs to the EPAC300 series controller unit are divided into three categories: Per phase, per ring inputs and per unit inputs. These inputs are associated with electrical signals generated external to the unit.

III. 3. 3. 1. 1. Inputs per phase

The EPAC300 series controller unit will provide the following input features on a per phase basis:

Vehicle detector: Used for entering a vehicle demand in the appropriate phase of the controller (service call).

Pedestrian detector: Used for entering pedestrian demand in the appropriate phase of the controller (pedestrian call).

Hold: This phase input is the command that retains the green right-of-way and has different controller unit responses depending upon operation in actuated or non-actuated mode.

Phase omit: Inhibits the selection of a phase, even in the presence of demand. Calls present on an omitted phase will not be serviced and, therefore, are not "serviceable conflicting calls". This omission shall continue until the signal is removed. The phase to be omitted will not present a conflicting call to any other phase, but will accept and store calls. Activation of this input will not affect a phase in the process of timing.

III. 3. 3. 1. 2. Inputs per ring

The EPAC300 series controller unit will provide the following input features on a per ring basis:

Force off: Ring input is used to force the termination of the GREEN timing in the actuated mode. It will only be effective when:

- A call on a conflicting phase can be serviced by forcing termination of the currently active phase (a serviceable conflicting call).
- The minimum green has passed.

Stop time: This ring input, when active, causes cessation of the controller unit ring input for the duration of such activation. Upon removal of the activation, all portions will resume timing from the point it was stopped.

Red rest: This ring input will cause the controller unit to rest on red in the absence of any serviceable conflicting demand. Registration of a serviceable call will result in immediate advance from red rest to green of the demanding phase.

Inhibit Maximum termination: This ring input is used to disable the maximum termination functions of all phases in the ring. The maximum green time-out is one of the conditions that can cause phase termination. When active, this input inhibits the maximum green "time out" from causing the termination of the phase. The input will not inhibit the timing of the maximum green.

Omit Red clearance: Causes the red clearance interval to be omitted and the "all red" delay to be bypassed.

Maximum II selection: This ring input, when active, selects the Maximum II time value or, when inactive, the Maximum I time value. The time value is the maximum green time interval.

Pedestrian recycle: This ring input controls the recycling of the pedestrian intervals.

III. 3. 3. 1. 3. Inputs per unit

The EPAC300 series controller unit will provide the following input features on a per unit basis:

Internal advance: This unit will cause immediate termination of the currently timed interval in the active phase. In the case of concurrent timing of several intervals, the interval that will terminate next without the input will terminate immediately.

Manual control enable: This unit input will place vehicle and pedestrian calls on all phases.

External minimum recall to all vehicle phases: This input will place a recurring vehicle call on all phases. This prevents any phase from unnecessarily dwelling in green, by creating a serviceable conflicting call.

III. 3. 3. 2. Actuated mode without volume density control

The green interval is a variable interval that depends upon vehicle actuations. The green interval time will be limited by the maximum green time function which will commence timing upon registration of a serviceable conflicting call. The minimum green time will not be preempted by a maximum green termination. All green time after minimum is considered the extensible portion of the green or part of the passage time timing period. The passage timer counts the time between successive vehicle actuations. Three time settings are provided for termination of green timing on an actuated phase without volume density control:

- Minimum green: Set upon the storage of vehicles waiting to be served.
- Passage time: Function of vehicle actuations that occur during the green interval
- Maximum green: Maximum length of time a phase may be green depends on the presence or absence of a serviceable conflicting call.

III. 3. 3. 3. Actuated mode with volume density control

Variable initial: The effect of variable initial is to increase the minimum green in a manner that depends upon the number of vehicle actuations accumulated in the phase while it was displaying red or yellow. It cannot go further than the maximum initial and not more than the maximum green time. The computed value is the product of the number of non-green actuations and the seconds/actuations time value.

Gap reduction: The effect of gap reduction on the extensible portion is to reduce the allowable gap between successive vehicle actuations by decreasing the extension time in a manner that depends upon the time waiting on a conflicting red phase. The Gap reduction is accomplished by means of the following functional settings:

- Time before reduction
- Cars before reduction
- Passage
- Minimum gap

- Time to reduce

The “time before reduction period” begins when a phase is green and there is a serviceable conflicting call. The “cars before reduction” count begins in the green extensible portion to calculate the sum of the conflicting cars waiting. Upon completion of the “time before reduction” and the “cars before reduction” a linear reduction of the allowable gap from the passage time level begins. The time period in which this reduction occurs is called, “time to reduce”. The reduction of the allowable gap continues until the gap reaches a minimum value, “minimum gap”.

III. 3. 3. 4. Summary

The basic inputs to the EPAC300 series were classified into three groups and input features of each group were discussed in detail. The front panel of the controller unit provides the interface between the user and the traffic control unit. Data are entered or displayed by selection from menus. Input from the machine-vision presence detectors indicates to the controller the presence or absence of a vehicle.

III. 4. Enhancement of Machine vision detection system

III. 4. 1. Introduction

Major objectives of the project included extensive collection of data and testing of a set of control strategies at the intersection. The scope of collecting data and testing management schemes was limited by the four cameras at the intersection. Presence of additional detectors within the intersection and at upstream of the heavily traveled South-bound approach, would expand the scope of controlling the intersection. For addressing the need for improved control by enlarging the viewing window of the intersection, certain enhancements were completed.

III. 4. 2. Enhancements completed

III. 4. 2. 1. Installation of two new cameras

Incidents are a major cause of delay in arterial networks and intersections. Incident detection and response is, therefore, essential in reducing delay. Identifying incident management as an objective at the intersection introduces the need for additional detectors, and this in turn requires the installation of additional cameras. For instance, none of the four cameras which look at the approaches have a proper view of the mid-intersection area. A camera that looks at the mid-intersection area was, therefore, necessary for incident detection. Further, a camera looking at upstream of the Southbound approach was felt necessary for control as a result of heavy merging traffic from two major arterials and the I-94 freeway exit. For addressing these concerns, two new cameras were installed at the intersection. One camera looks upon the mid-intersection area, the common area used by all vehicles entering the intersection. The other, looks approximately 400 ft upstream of the SB approach, where vehicles from Hennepin Ave and Lyndale Ave merge and join the vehicles coming from the exit of I-94.

Following installation, certain technical problems had still to be overcome in getting the video output to the supervisor computer. The machine-vision software and hardware were updated to handle the six cameras. Following considerable effort, the output from all six cameras is available at the supervisor computer.

With a camera looking into the middle of the intersection, that area has been isolated from the approaches. When installed within the intersection, presence detectors can provide data for testing intersection incident detection algorithms. The other camera can provide data for studying merging traffic. Extensive data can be collected with the six cameras. This paves the way for the development of an Intersection Traffic Database.

III. 4. 2. 2. Design of a new detector configuration

Three are the basic types of machine-vision virtual detector, count detectors that measure volume, presence detectors that indicate the presence of a vehicle, and speed traps that measure speed and volume. Boolean logic can be used to combine the output from different detectors. Further, station detectors are virtual detectors which provide a means for collection of traffic data for later use. The three basic types of detector can be linked to a station detector if their output is to be saved in a file.

Prior to the installation of the two new cameras, presence detectors were placed at the stop-lines of the four approaches, and their output was used by the controller for actuated control of the intersection. Count detectors were placed at the downstream of the approaches on all lanes to measure the downstream volume. For delay estimation the speed of the cars in the approach segments was desired. Therefore, speed traps were installed at the upstream of the approaches on all lanes. Count detectors and speed traps were also placed in the exits of the intersection. Additional count detectors were placed further upstream of the speed traps.

With images from the two new cameras, a new detector layout was prepared for the mid-intersection area and the area upstream of the SB approach. Load index of the machine-vision system is a quantitative measure of the area of the image to be processed by the system. The larger the number of detectors, the higher is the load index. The load index of the machine-vision system was quite high with the existing detectors. It has been noticed that system performance deteriorates with the increase in the index beyond the present value. Therefore, certain detectors had to be removed to accommodate the new detectors for the new camera images.

The detector layouts given in Figs. 3.1. and 3.2. were proposed for the two new cameras and are in use now. Speed traps that can measure speeds as well as volumes have been installed upstream of the South bound approach. For the camera facing the mid-intersection area, many small presence detectors were installed to cover the whole intersection. They are all hooked onto a station detector. A delay time of 20 seconds has been specified. So, when a vehicle stays in the intersection for more than 20 secs, an actuation or call is made indicating an incident at the intersection. With the new version of the machine-vision system, which is yet to be released, it would be possible to get on-line data to the supervisor computer. Real time applications for incident management can be run on the on-line data and can expedite the clearance of the incident.

The detector configuration for the four old cameras was retained with minimal changes. Refer to Figs 3.3. - 3.6. for the detector layout for the four cameras. The combined detector layout for the four cameras is in Fig 3.7.

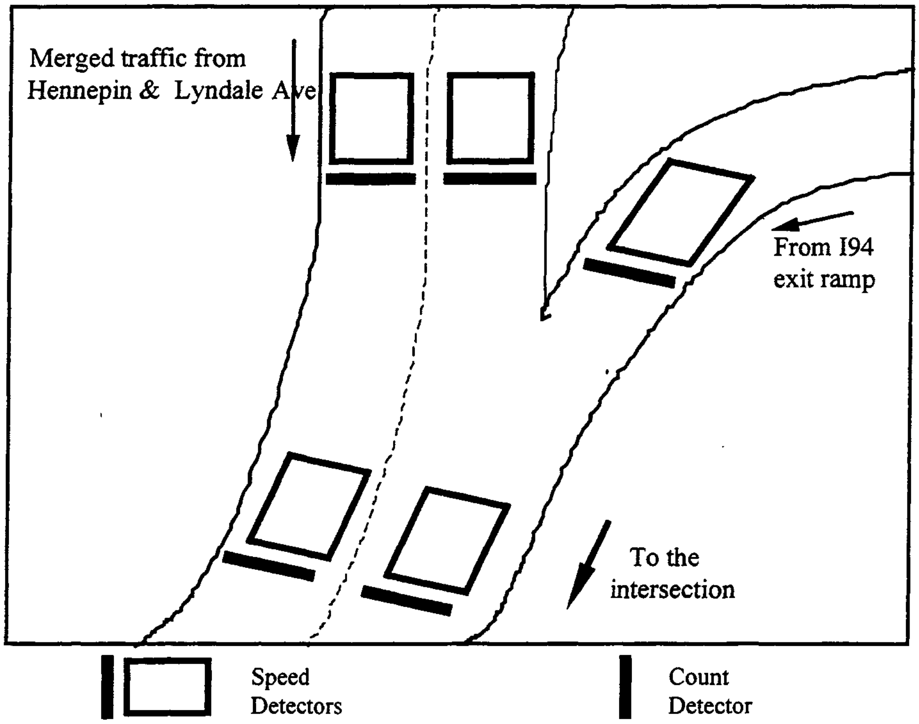


Figure 3.1. Detector layout for the new camera looking upstream of the SB approach

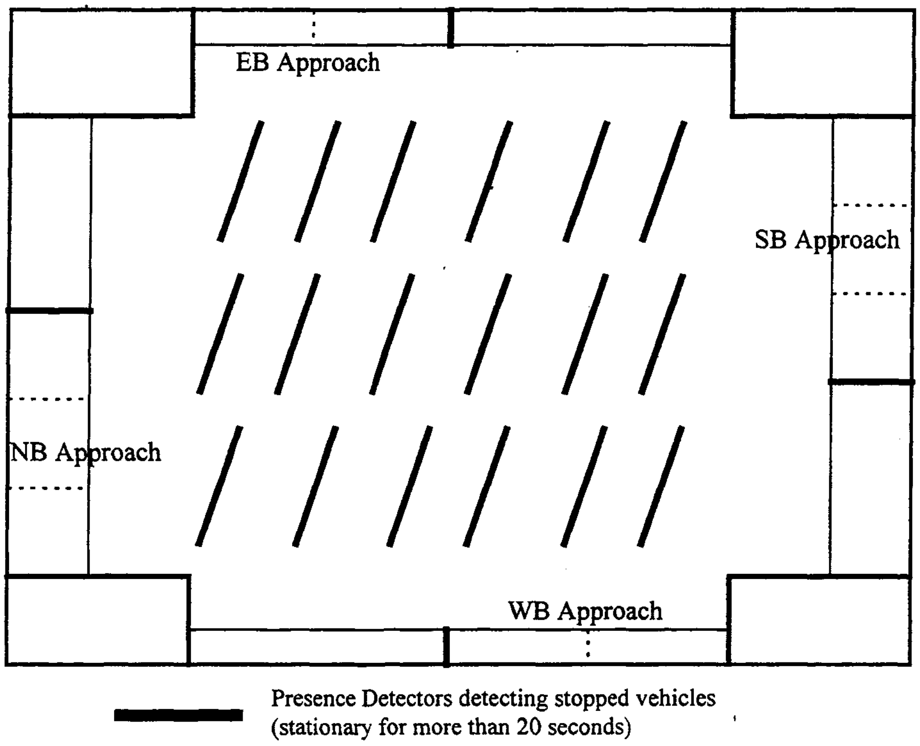

EB Approach
SB Approach
NB Approach
WB Approach
Presence Detectors detecting stopped vehicles
(stationary for more than 20 seconds)

Figure 3.2. Detector layout for the new camera looking into the mid-intersection

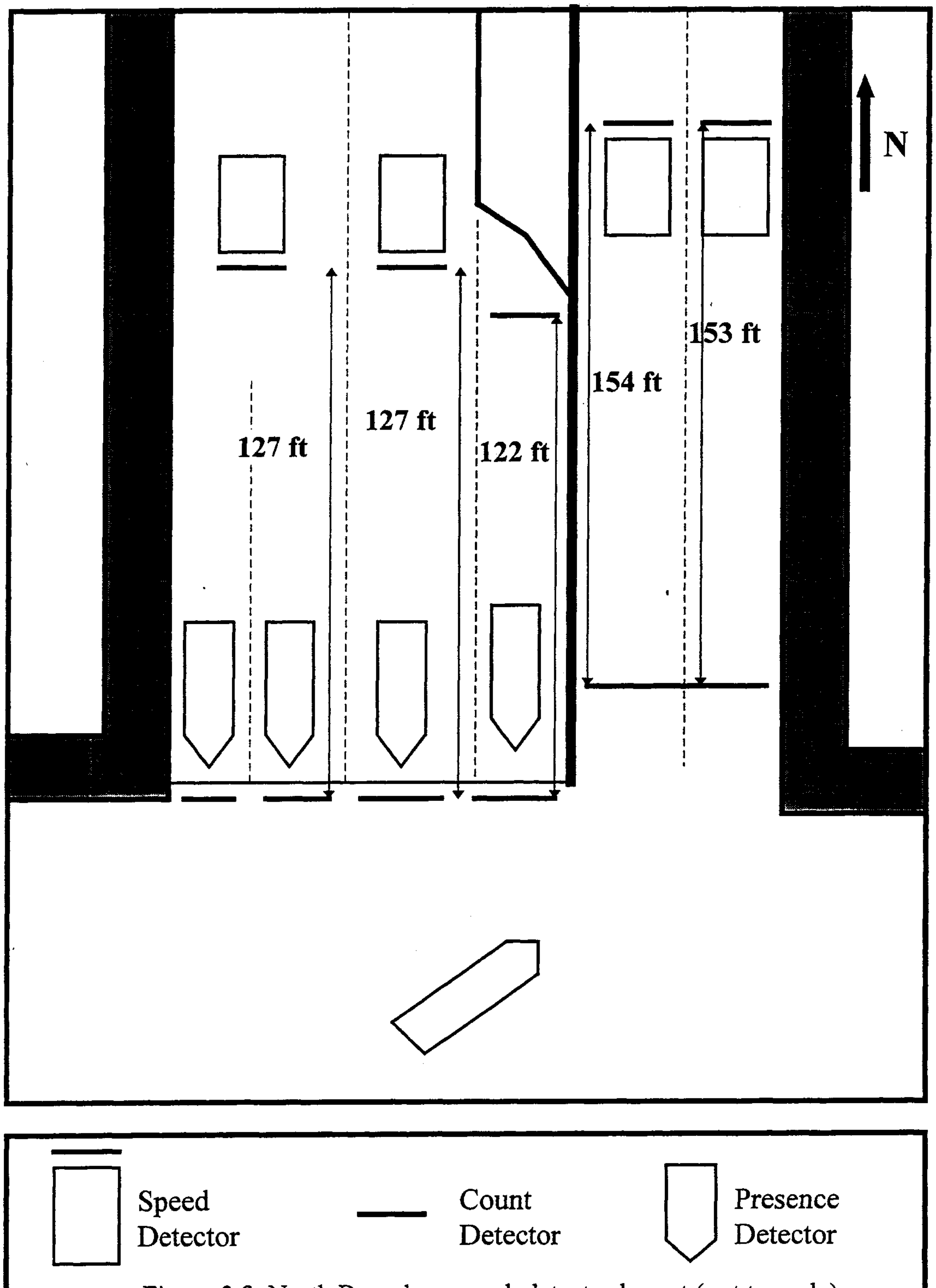


Figure 3.3. North Bound approach detector layout (not to scale)

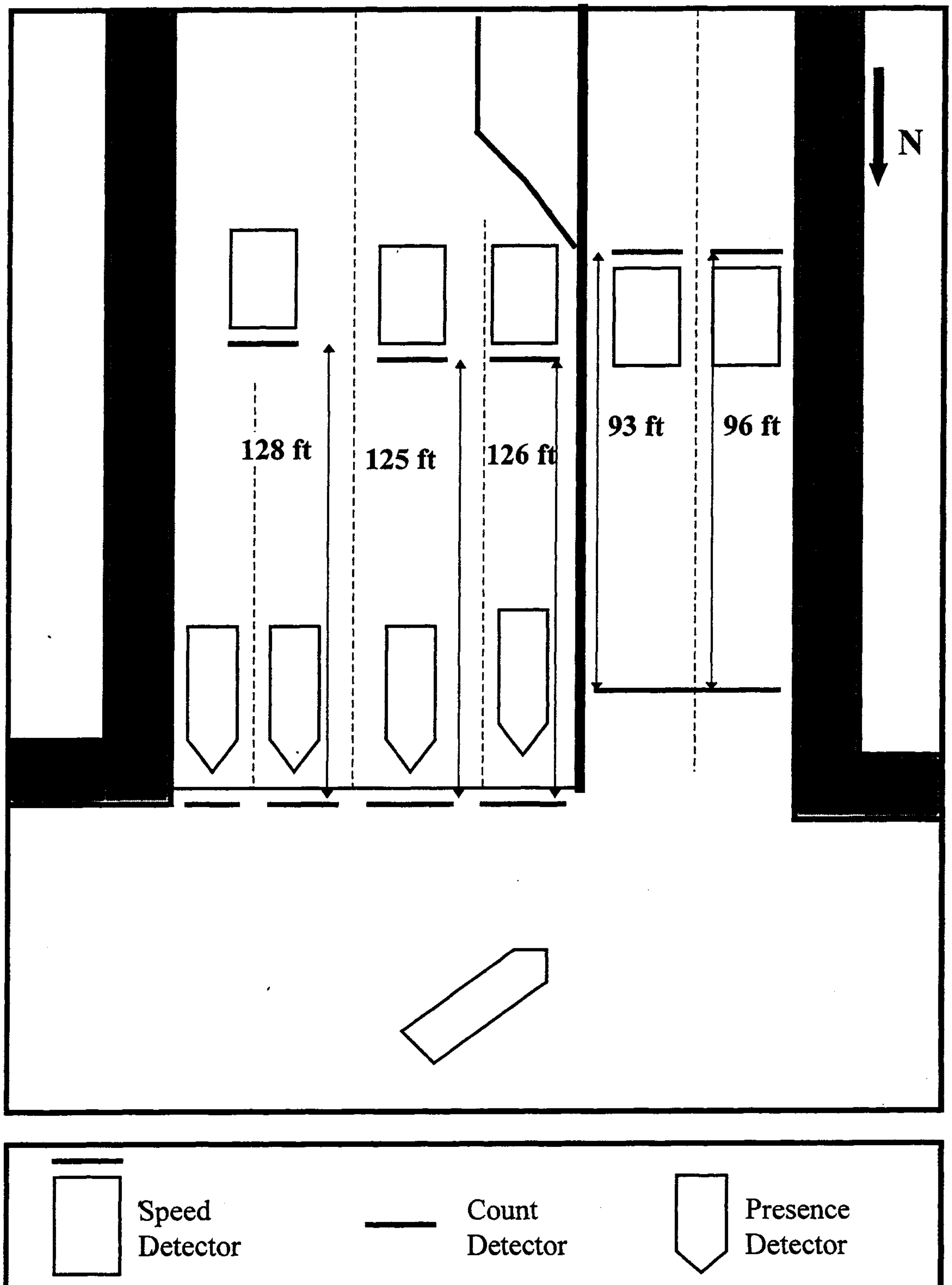


Figure 3.4. South Bound approach detector layout (not to scale)

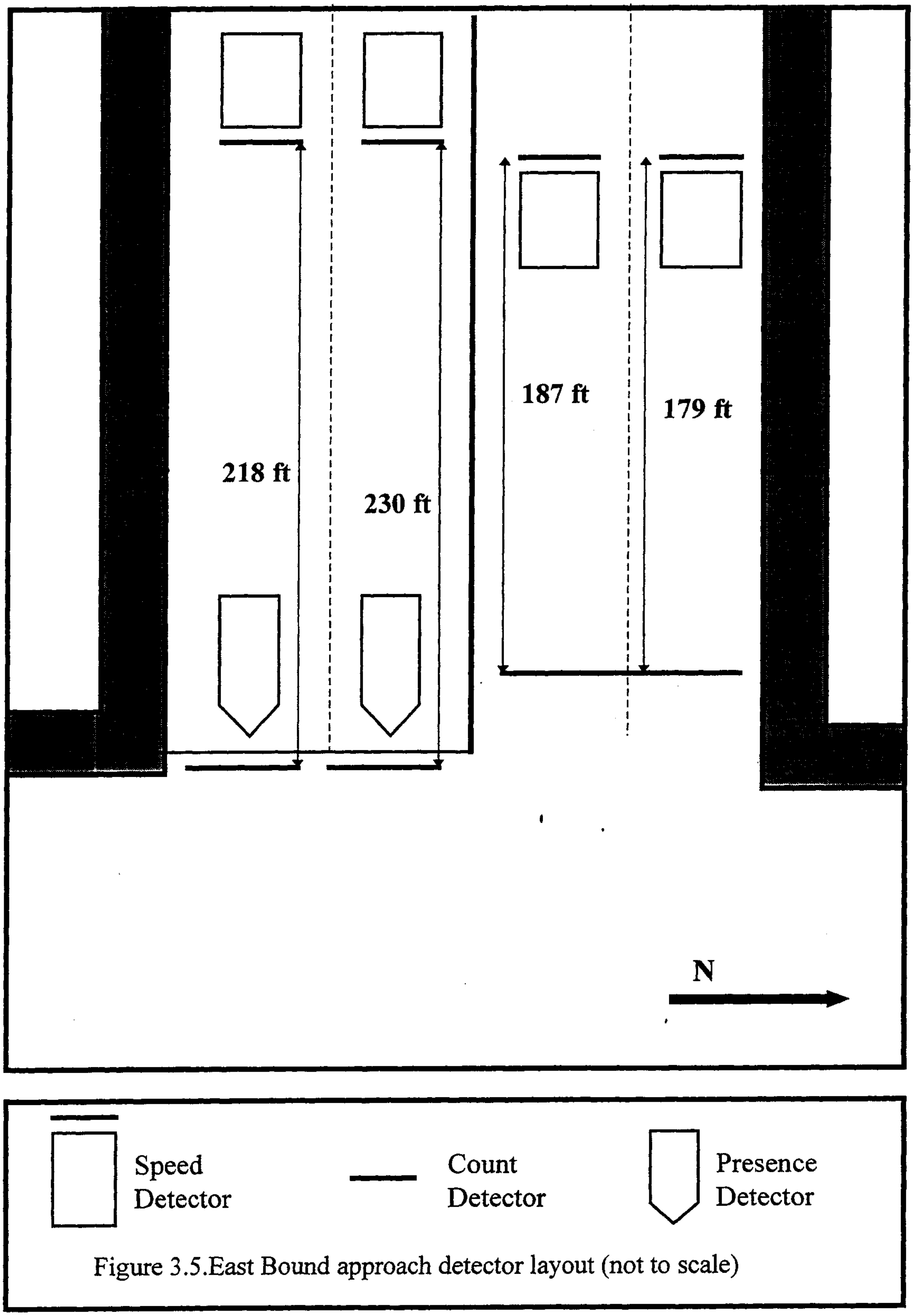


Figure 3.5.East Bound approach detector layout (not to scale)

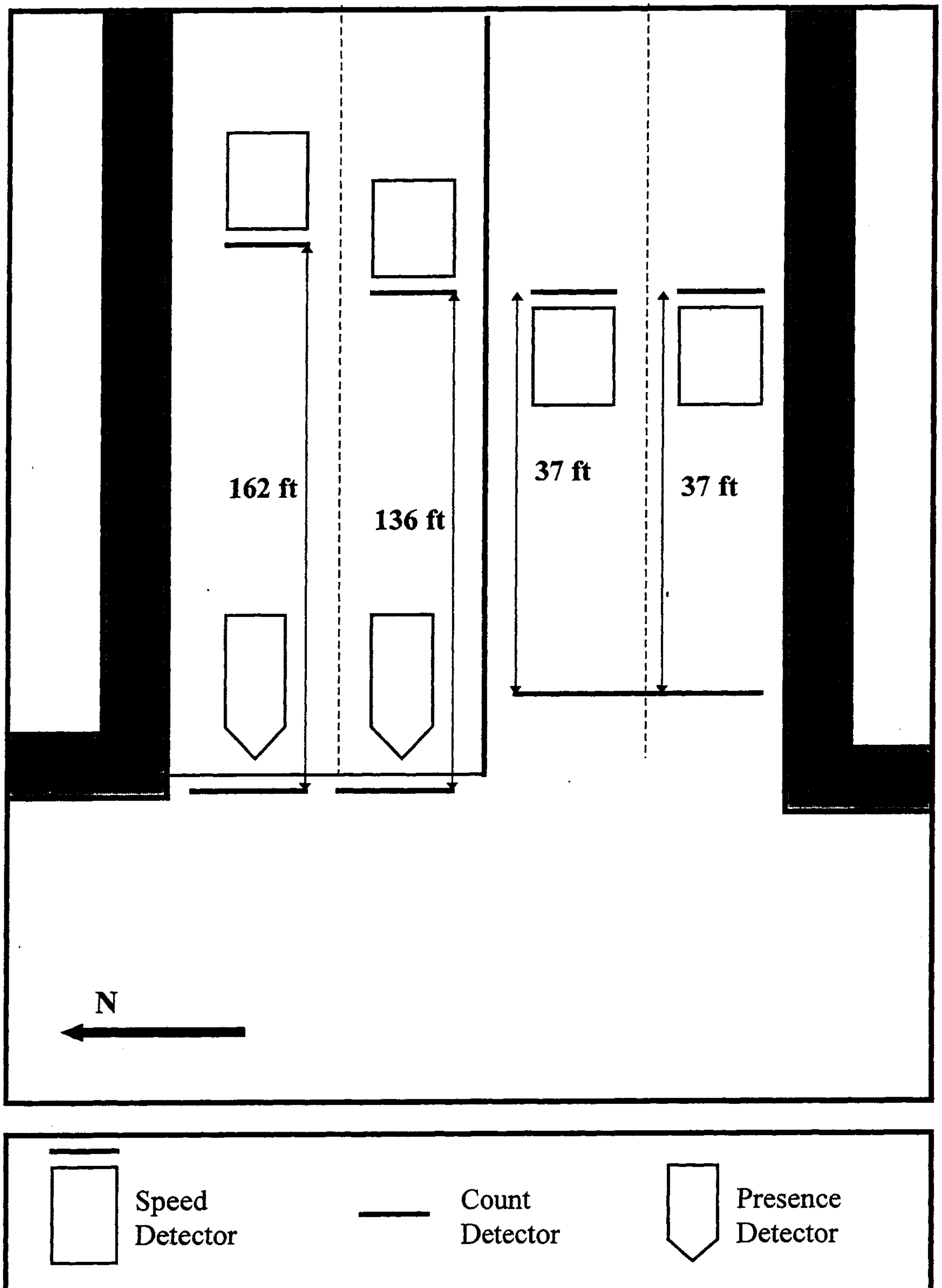


Figure 3.6. West Bound approach detector layout (not to scale)

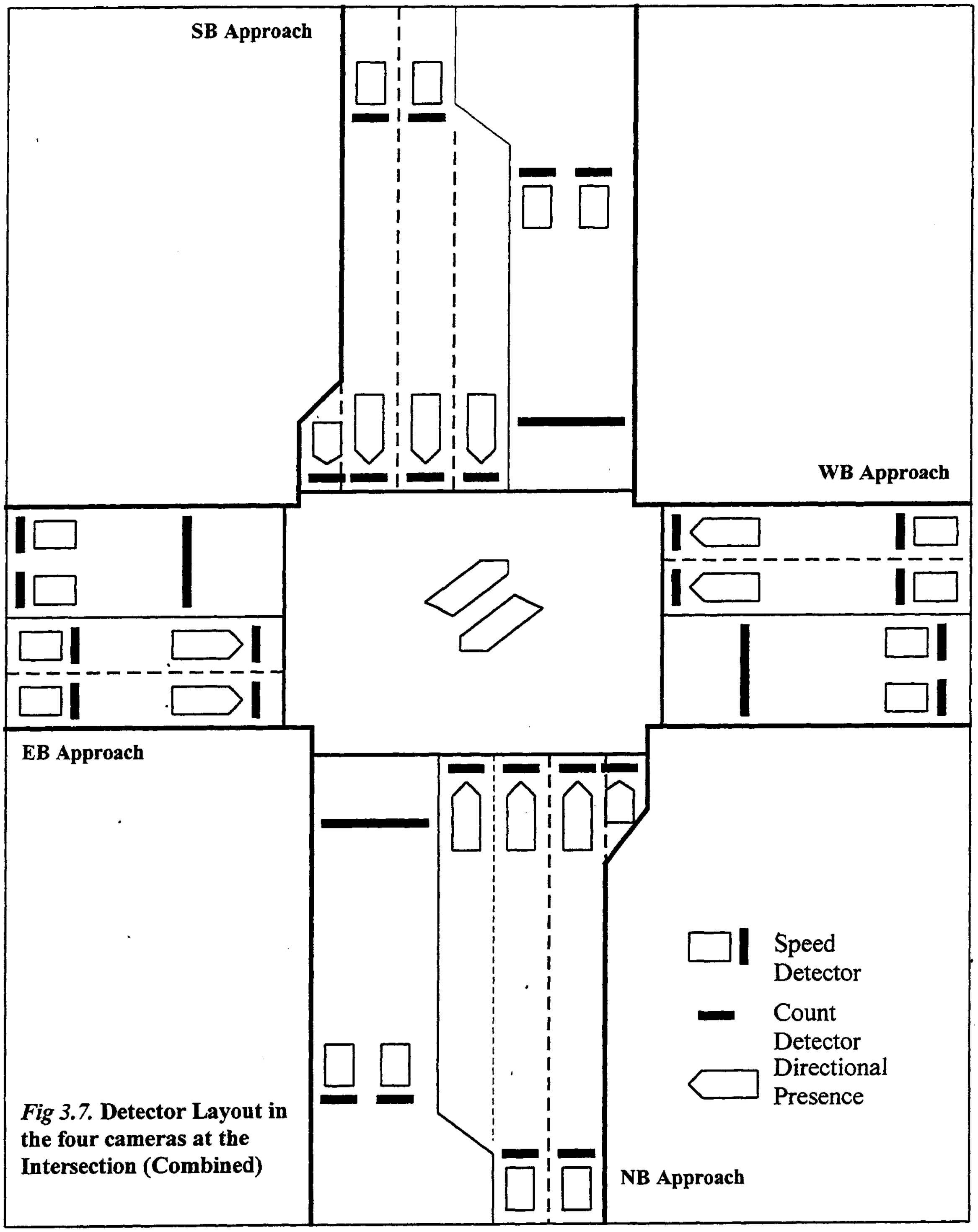


Fig 3.7. **Detector Layout in the four cameras at the Intersection (Combined)**

III. 4. 3. Enhancements needed

III. 4. 3. 1. Development of an intersection data access plan

The supervisor computer is situated in the City of Minneapolis office. Data that are collected are stored in this computer. The objective is to get the data to the ITS lab in the Center for Transportation studies for the development of an intersection traffic database. Bringing data directly to CTS would save a lot of time and effort. There are two ways of accomplishing this objective. The feasibility, advantages and disadvantages of the two methods are discussed in the following sections.

III. 4. 3. 1. 1. Method I - Direct access of the machine-vision unit from the ITS lab

The machine-vision unit has two 9-pin RS-232 communications ports labeled supervisor and modem. The supervisor port is used to connect the processor directly to the supervisor computer using a null-modem cable, while the modem port is used to communicate via modem with the supervisor computer or other coordinated system device. At this intersection, the supervisor port is used for communication with the supervisor computer. The modem port has not been used.

The idea behind this setup involves the use of both the communication ports. Communication could be opened between the ITS lab computer and the machine-vision unit through the modem port. With this connection, the unit becomes accessible from CTS as well as the City, as the supervisor computer is already connected through the supervisor port. A dedicated phone line is also available at the intersection, and can be connected to the modem port. The machine-vision unit can be accessed by dialing into it from the ITS lab. The telephone lines will be used for communication instead of special cables.

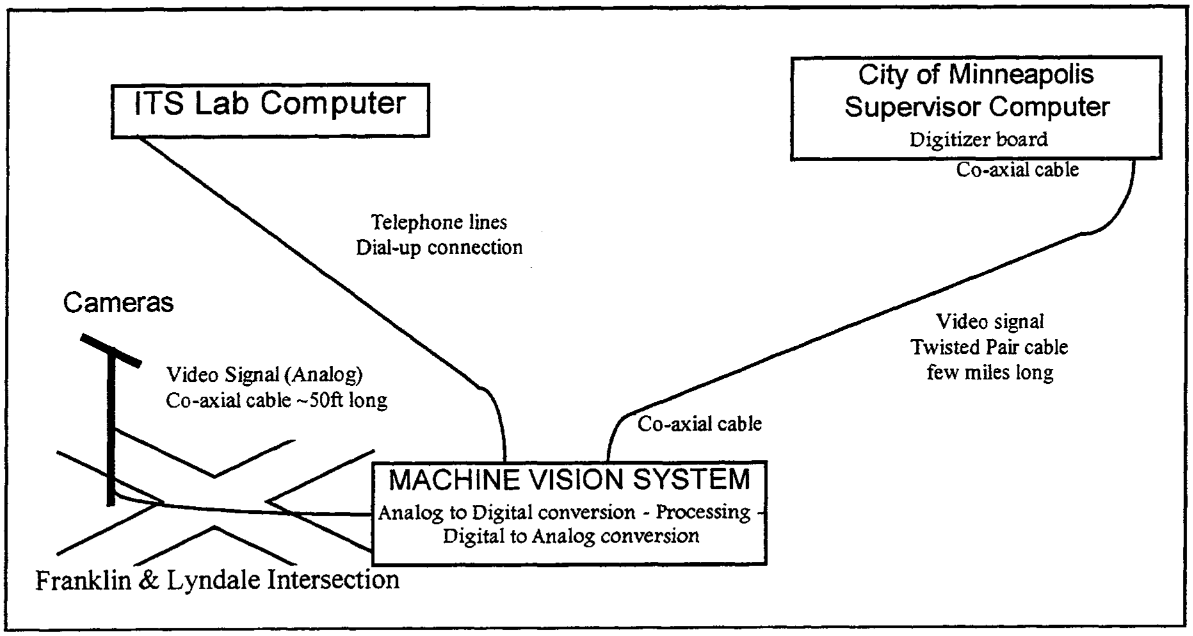


Figure 3.8. Data access plan - Method - I

This method is not feasible now. This is because the current version of the machine-vision system does not allow the use of both the modem and supervisor ports at the same time. The product has to be updated to address this need.

The advantages of this method are that data collected can be brought directly to the ITS lab. No effort is demanded from the City officials. There is ample freedom to select the time of collection, duration of collection, etc. without bothering the City officials.

The disadvantages are as follows. If the machine-vision system is enhanced to allow access through both ports, there has to be some restriction on the use from the ITS lab (modem port). A kind of password protection is required so that the detector files are not altered by 'a user at the ITS lab. For users at the ITS lab, access should only be provided to collect data and recover storage space. There should be no provision for making other changes to the system, as the City of Minneapolis will be held responsible for the consequences of any changes. This security feature is not available in the current version of the machine-vision system. The software has to be further modified to allow users from the City office and the ITS lab to simultaneously log on to the unit, or prevent one user from accessing if the other is already logged on. Thus, major changes have to be made to the machine-vision hardware and software.

III. 4. 3. 1. 2. Method II - Accessing the supervisor computer from the ITS lab

In this method, no communication is assumed between the machine-vision unit and the ITS lab computer. Currently, the City of Minneapolis is not collecting any data for its use. Rather, the City is using the detection system and the cameras for actuated control of the intersection only. Communications between the supervisor computer and the machine-vision unit are opened only when changes are to be made to the detector files. In this method, the data must be collected in the supervisor computer at the City office. These data can be accessed from the ITS lab by dialing into the supervisor computer using a modem.

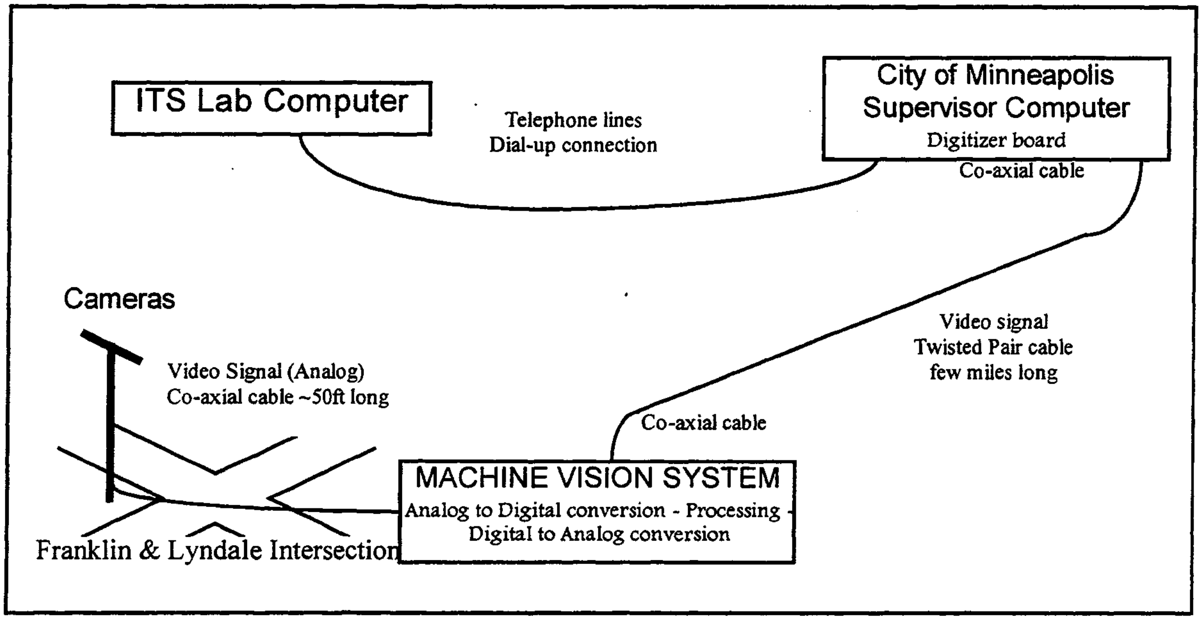


Figure 3.9. Data access plan - Method II

This method could be initiated, but requires certain resources. In particular, a dedicated telephone line has to be installed at the City for this purpose. A fast modem is also required for the transfer of data. To allow remote login into the supervisor computer, appropriate software has to be installed.

The main advantage of this method is that not much effort is required to get the collection process working. The set up will not consume much time. Moreover, no changes are required in the machine-vision software and hardware.

One of the main disadvantages of this method is that the City officials will have to monitor the collection of data. An operator has to open up communications and constantly monitor the operations. The operator has to start the collection, download data from the memory module when full and then recover the storage space for further collection.

III. 4. 4. Summary

Analysis of data obtained from the intersection laboratory is performed at the Center for Transportation Studies. Easy acquisition of data eliminates certain delays in the analysis. Two methods were identified, either of which could facilitate the acquisition of data. In the first method, there is direct access to the machine-vision unit from the ITS lab at the Center for Transportation Studies. In the second, access is provided to the supervisor computer at the City of Minneapolis, where the collected data are stored. Even though it is difficult to accomplish, the first method is preferred because of the relative ease in obtaining the data.

III. 5. Development of a intersection traffic database

III. 5. 1. Introduction

A database is a logically coherent collection of data, designed and built for a specific purpose. The database derives its data from a specific source and has a degree of interaction with the real world, and an audience that is actively interested in the contents of the database.

A database management system (DBMS) is a collection of programs that enables users to create and maintain a database. The DBMS is hence a general-purpose software that facilitates the processes of defining, constructing, and manipulating databases for various applications. The three processes are explained as follows. Defining a database involves specifying the types of data to be stored in the database, along with a detailed description of each type of data. Constructing the database is the process of storing the data itself on a storage medium that is controlled by the DBMS. Manipulating a database includes such functions as querying the database to retrieve specific data, updating the database to reflect changes, and generating reports from the data.

No special purpose DBMS software is used for implementing the computerized traffic database. Here PARADOX, a full-featured relational database management system, is used in implementing the database. A relational database allows the definition of a relationship (called a link) between different tables of data.

III. 5. 2. Advantages of the database approach over the traditional file processing approach

The database approach has a number of advantages over the traditional file processing approach. Some of the basic differences are discussed below. A fundamental characteristic of the database approach is that the database system contains not only the database itself but also a complete definition or description of the database. By contrast, data definition is part of the application program.

In traditional file processing, the structure of the data files is embedded in the access programs, so any changes to the structure of a file may require changing all programs that access this file. In a DBMS, insulation exists between the programs and data. Also a DBMS provides the users with a conceptual representation of the data that does not include many of the details on the ways the data are stored.

Considering the above characteristics of the database approach, it was decided to develop an intersection traffic database containing data obtained from the machine-vision unit located at the Franklin and Lyndale Ave intersection.

Since a database contains related data, a database or table is created for all data collected in a day. In other words, there is a table for each day of available data. Other options of combining days of data would result in a large database which would be difficult to handle with a PC. The queries and processes might consume more time as the access times for a record would increase. More powerful machines like Unix workstations would be required to handle such large databases.

It must also be noted that the database does not contain data collected over a period of 24 hours. Data are usually collected during the Morning Peak, Off Peak and Evening Peak periods, each lasting about 3 - 4 hours. Thus we may have 10 - 12 hours (maximum) of data per day.

III. 5. 3. Designing the database

Designing the database involves identifying the data to be stored in the database and determining appropriate structures to represent and store the data. An efficient design should incorporate all the requirements of the system users and facilitate fast processing of queries. In this case, the users would be University researchers and other traffic engineers. The requirements would be different queries that can be made on the data; for example, a query could be the determination of the exit volume for the intersection for a specific period of a day.

After identifying the data, the following design was chosen. The format is similar to the machine-vision output format, as no additional indices or keys were required. The Average_flow field has been excluded from the database as it can be derived from Volume. The data collected during the three periods, i.e., AM-Peak, Off-Peak and PM-Peak,, on a single day are stored in a single database. Therefore, date information need not be stored in each record. The names of the databases will contain the information about the date on which the data were collected.

Time
Station_ID
Volume
Average_Speed
Class_1
Class_2
Class_3
Time_Headway

Level_of_Service

Space_Mean_Speed

Space_Occupancy

Density

With the value of the fields, Time and Station_ID, each record in the database can be located. There is only one record with a unique set of values for the fields time and station_id.

III. 5. 4. Constructing the database

Constructing the database involves storing the data on a storage medium that is controlled by the DBMS. The database is constructed as a table in PARADOX. Several steps need to be carried out before the data are stored in the database. They are as follows.

Step 1: Conversion of binary file into ASCII

The data obtained from machine-vision are in a binary format. The machine-vision data filter program is used to convert this binary file into an ASCII file.

Step 2: Removal of unnecessary information from the ASCII data file and import of the same into PARADOX

Following conversion, the data file has the following information in a sequential order,

1. Machine-vision unit identification information
2. Station detector numbers and their description
3. Start and stop time (with date) of data collection
4. Headers (format) for the data collected
5. Data

For the data to be imported into PARADOX, the first 3 items should be removed. The removal of headers depends on the procedure used to import the data. This can be accomplished in two ways.

Method 1

1. Delete items 1 to 4 in the data files. Merge the data files for the three periods into one file.
2. Run PARADOX. Choose File | Utilities | Import.
3. Select the data file from the appropriate directory.

   Also in options specify that there are no delimiters for text fields and that all fields are separated by commas.
4. Restructuring the table

   PARADOX automatically determines the data type for each of the columns. To change the selected data type and enter the headers for the columns, restructure the table (Table | Restructure). Time data type should be changed from date to alphanumeric as PARADOX wrongly detects it as a date format.

Method 2

1. Delete items 1 to 3. Modify item 4 in the data file after merging the three files for the three periods.
2. Open the text file in Microsoft Excel. The Import dialog box is opened.
3. Specify that fields are separated by commas and there are text qualifiers in the data.
4. Change column data formats for Time as Text, Level of Service as text. Save the file as a Excel 4.0 worksheet.
5. Import this Excel file into PARADOX. The columns get the headers automatically. If necessary, some of the data types can be changed from numbers to short numbers.

The first step of the two methods has been automated using the CONVERT.CPP program. Depending on the method to be followed, the program deletes the unnecessary information from the data files. The second method is easier and quicker to use than the first one, even if it looks lengthier. All data stored in the database should follow the same method. The other steps cannot be automated as a few processes in other applications are involved.

III. 5. 5. Manipulating the database

Now that the database has been constructed, queries can be made, updates can be done and reports can be generated. For simple queries to be processed, some coding is required in a simple query language. For other large applications where the database is to be used, coding in ObjectPAL is required. ObjectPAL is the integrated programming language for Paradox for Windows, used to develop full-featured Windows database applications.

The program CONVERT.CPP should be executed only after the conversion to the ASCII format is done. In cases where there is data only for two periods of a day, corresponding options have to be checked while executing the program. Two tables can be merged within PARADOX, if they have the same format. Report generation is also very simple in PARADOX.

Some of the queries that would be made on the data are,

1. Determine Total intersection (entry/exit volumes) traffic in a specified period
2. Calculate average approach speed on a specific lane/approach
3. Search for the indication of a incident
4. Determine average headways, occupancy etc.
5. Calculate turning volumes for each approach in specific periods.

III. 5. 6. Summary

A database has been designed to store the data obtained using machine-vision. The database includes a single table with data fields obtained from the machine-vision outputs. Data collected on a specific day are stored in a single database. A part of procedure for constructing the database has been automated using a C++ program. Upon execution, this program merges the data files for a specific day and removes unwanted information. With this complete, the file is ready to be ported into PARADOX or any spreadsheet. Upon importing the data into PARADOX, queries can be run on the data for analysis and research.

III. 6. Estimation of intersection delay with machine vision data

III. 6. 1. Introduction

Intersection delay is an often used indicator of performance of traffic control strategies at intersections. Use of this indicator is imperative in selecting the most effective strategy. It also facilitates proper assessment of the effects of improvements such as signal optimization, road geometry, introduction of traffic control devices and other traffic management schemes. In this work, an effort has been made in developing an intersection delay estimation algorithm to work with machine-vision data.

The machine-vision detectors that are installed at the intersection approaches do not provide continuous monitoring of traffic. The output, especially speed (for speed traps only) and volume, is obtained from these detectors at the end of regular time intervals. Therefore, the delay estimation model must be able to work by accepting input at such intervals.

A delay estimation algorithm has already been developed for such a case. It is a discretized version of the standard input/output traffic model. In each of the approaches, count detectors are installed at the stop-line and speed traps are installed upstream of the approach, for each of the lanes. The segment of the roadway between the two detectors is the section that is analyzed. In this model, delay per segment is defined as the difference between the total time that all vehicles occupy the segment and the time that the same vehicles would occupy the segment if they traversed it at a pre-defined free flow speed. In other words, the delay is the excess time consumed by the vehicles in slowing down or stoppages due to the presence of the intersection.

It is not possible to track each vehicle to measure the delay. So, depending on the number of arrivals, number of departures and the number of vehicles stored earlier in the segment, a fraction of the time slice (half or full or based on speed) is approximated as the delay experienced by the vehicles in that time-slice.

The algorithm is an approximation of the actual delay. It has some shortcomings. It does not take into consideration the status of the signal in the delay calculations. It also cannot account for lane changes and is very sensitive to false detections. Shorter segments with more detectors may provide more accurate results, but the increase in the detectors, past the load index of the machine-vision system, may cause erroneous results. The algorithm needs to be modified to address the above needs.

Along with the changes in the detector layout, the machine-vision system was upgraded. Earlier, data used to be written in ASCII format in the memory of the machine-vision unit; however this is not efficient because it consumes a lot of space. In the current version, data are written as a binary file in the unit and downloaded to the supervisor computer at the office of the City of Minneapolis. After this they are converted to an ASCII format using a filter program for importing them into databases or for further analysis. The format of the file has also changed. Additional data fields, such as level of service, space mean speed and space occupancy, are also available for every time slice.

Owing to the changes made in the detector layout as well as the changes in the machine-vision system, a new program was written for estimating the delay. Delay is one of the important measures of effectiveness that can be obtained from the data. Different control strategies can be applied at the intersection of Franklin and Lyndale Avenues and the data can be analyzed and the delay calculated to arrive at the best control strategy that results in the least delay. The presence of the controller and the machine-vision unit at the intersection increase the flexibility of testing different control strategies.

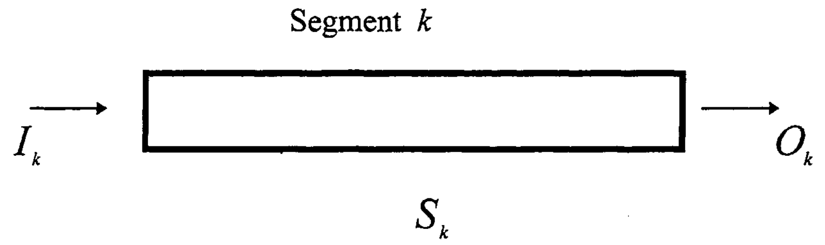
III. 6. 2. Delay estimation algorithm

The algorithm developed to run on machine-vision data to evaluate traffic performance is presented in the next few sections. In this presentation, a segment is defined as a section of the roadway, considered over a lane in this case. As a result, the number of segments per approach is same as the number of lanes. The delay and storage used in the delay equations are also defined. Based on the number of departing vehicles during a time slice and the vehicles stored in the previous time slice, the components of the delay equation vary. All arrivals and departures are assumed to be Poisson distributed.

III. 6. 2. 1. Definition of Delay

In a time period,

Delay per vehicle = (Total time all vehicles spend in traveling through the segment -
Total time required to traverse the segment at free flow speed) /
Total number of vehicles



Length of Segment = L_k

Length of time slice = T_s

Average speed of vehicles traversing the segment = $V_{k,i}$

Time to traverse segment at free flow speed = T_k^F

Input to the segment in time slice i = $I_{k,i}$

Output from the segment in time slice i = $O_{k,i}$

Note: Subscript *k* refers to the segment and *i* refers to the time slice

III. 6. 2. 2. Definition of storage

At the end of time slice *i*, vehicles stored in segment *k*,

$$S_{k,i} = I_{k,i} - O_{k,i} + S_{k,i-1} \qquad ,0 \le S_{k,i} \le S_{k,\max}$$

where, $S_{k,\max}$ is the maximum number of vehicles that can be accommodated in the segment, and average effective length of the vehicle is 22 feet.

III. 6. 2. 3. Delay calculations

$(D_{k,i})$ is the delay in the segment *k* during time slice *i*.

Case 1 : $O_{k,i} \leq S_{k,i-1}$

(Output in time slice *i* ≤ Storage in time slice *i*-1)

$$D_{k,i} = \left[O_{k,i} * T/2_{s}\right] + \left[\left(S_{k,i-1} - O_{k,i}\right) * T_{s}\right] + \left[I_{k,i} * T_{s}/2\right] - \left[O_{k,i} * T_{k}^{F}\right]$$

Delay equation components

First term : Time spent by output vehicles which departed during the present time slice.

Second term : Time spent in the segment by the vehicles that were stored during the previous time slice, but could not depart even by the end of the present time slice.

Third term : Time spent by input vehicles occupying the segment in the present time slice.

Fourth term : Time spent by the output vehicles in the segment if they traversed at a pre-defined free flow speed.

Case 2 : $O_{k,i} > S_{k,i-1}$

(Output in time slice i > Storage in time slice i-1)

$$D_{k,i} = \left[S_{k,i-1} * T_{s}/2\right] + \left[\left(O_{k,i} - S_{k,i-1}\right) * \left(L_{k} / V_{k,i}\right)\right]$$
$$+ \left[\left(I'_{k,i} + S_{k,i-1} - O_{k,i}\right) * T_{s}/2\right] - \left[O_{k,i} * T_{k}^{F}\right]$$

Delay equation components

First term : Time spent in the segment by the vehicles that were stored during the previous time slice, but departed during the present time slice.

Second term : Time spent in the segment by the vehicles that entered and departed during the present time slice.

Third term : Time spent in the segment by the vehicles that entered, but could not depart by the end of the present time slice.

Fourth term : Time spent by the output vehicles in the segment if they traversed at a pre-defined free flow speed.

Delay per vehicle in segment *k*, $$D_k = \frac{\sum_i D_{k,i}}{\sum_i O_{k,i}}$$

(summation over all time slices)

III. 6. 3. Traffic data collection and analysis

III. 6. 3. 1. Introduction

The machine-vision detector layout at the intersection illustrated in Section 4 was used to collect traffic data. These data were used to obtain the traffic demand and delay on the approaches, and measure the turning volumes.

A total of 60 sets of traffic data were collected. A set refers to data collected over a period of 2 - 4 hours during AM-Peak, Off-Peak or PM-Peak periods. Of the 60 sets, 29 sets were obtained when the intersection was under actuated control and the rest, when it was under pretimed control. In actuated control data sets, 14 were obtained in the AM-Peak period, 7 in the Off-Peak period, and 8 in the PM-Peak period. In pretimed control data sets, 11 were obtained in the AM-Peak Period and 10 each in the Off-Peak and PM-Peak periods. The data were collected between the February 15th 1996 and April 15th 1996. Data could not be collected during all three periods of a day or on consecutive days owing to operational problems. Specific sets of data were selected for analysis to obtain reliable results. The erroneous sets were discarded.

Although data were collected over 2 - 4 hour periods, the periods used in the analysis are 06:30 - 08:30 (AM-Peak), 11:00 - 13:00 (Off-Peak), and 16:00 - 18:00 (PM-Peak), i.e., the periods of interest to the City of Minneapolis.

III. 6. 3. 2. Traffic flow and turning movements

From the collected data, the average traffic demand on the approaches and turning volumes were determined for the three periods. The average traffic demands are shown in

Figure 3.10. From the data it can be seen that the northbound and southbound approaches carry high traffic volumes. The demand on the westbound and southbound approaches is the lowest during the AM-Peak period and follows an increasing trend during the other two periods. Eastbound approach is the least traveled of the four approaches. Figure 3.11 shows a different grouping of the same data. The demand on the approaches can be compared for a specific period.

Figures 3.12 - 3.14 show the turning percentages for each period for all four approaches. In the three periods, it can be seen that the southbound and eastbound approaches have a high left-turning percentage. The westbound approach has a high right-

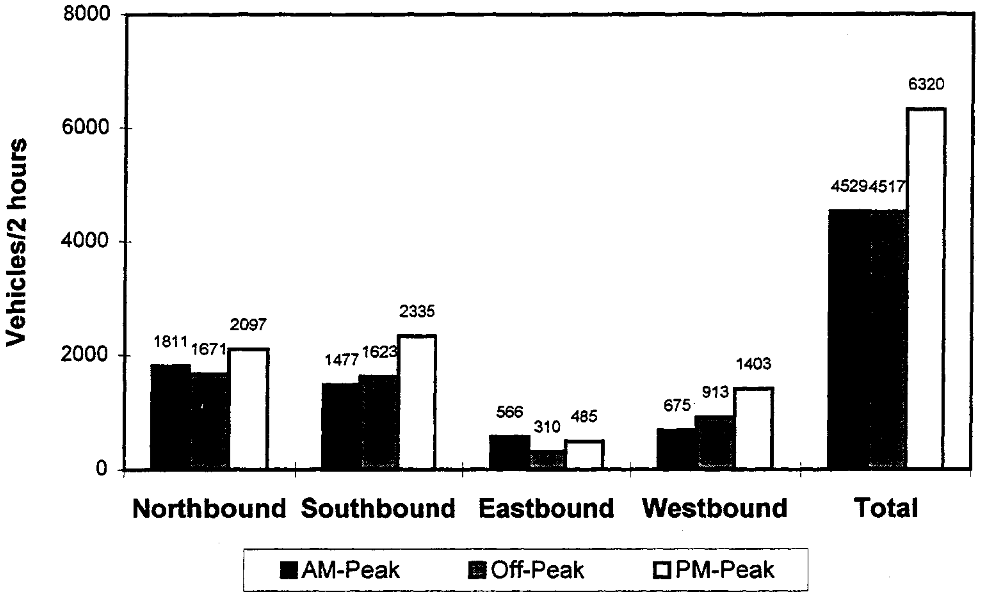


Figure 3.10. Average demand by approach

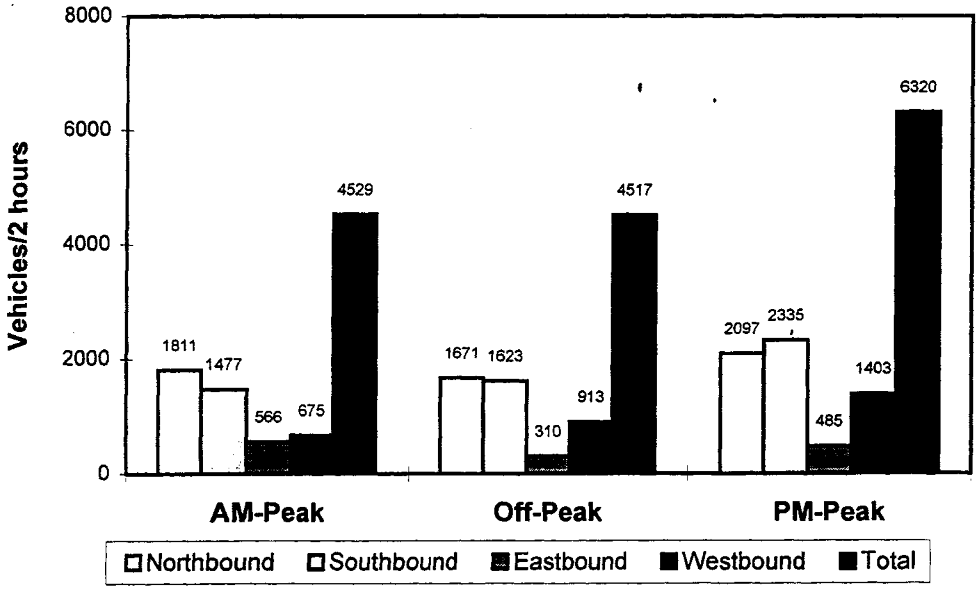


Figure 3.11. Average demand by time

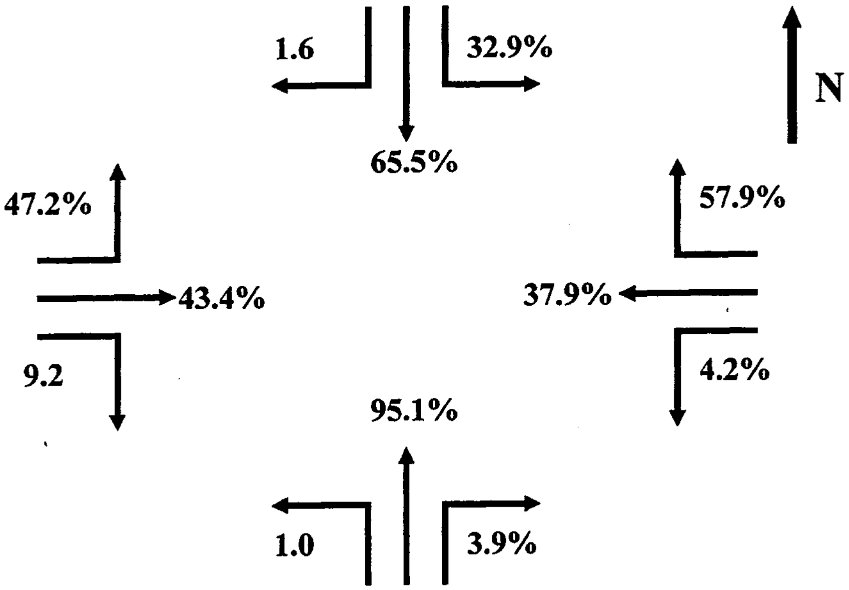


Figure 3.12. Turning movements during AM-Peak period

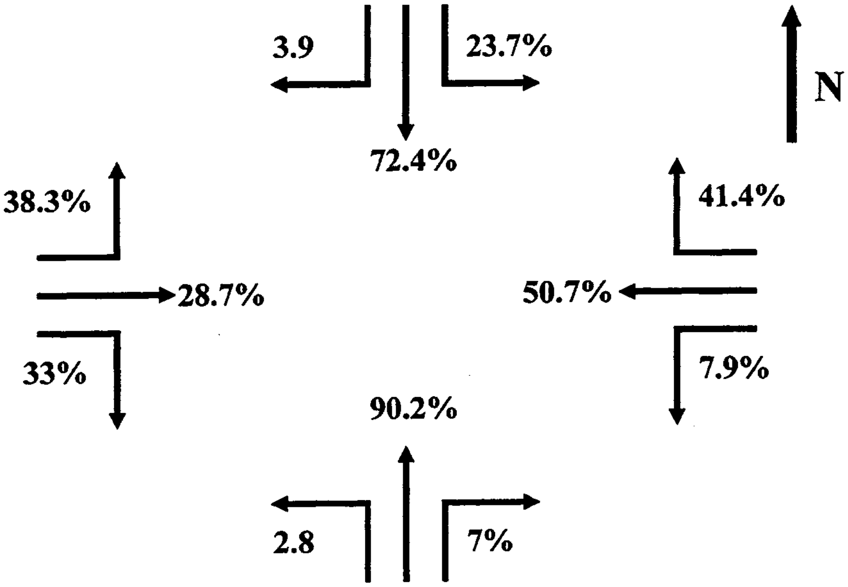


Figure 3.13. Turning movements during Off-Peak period

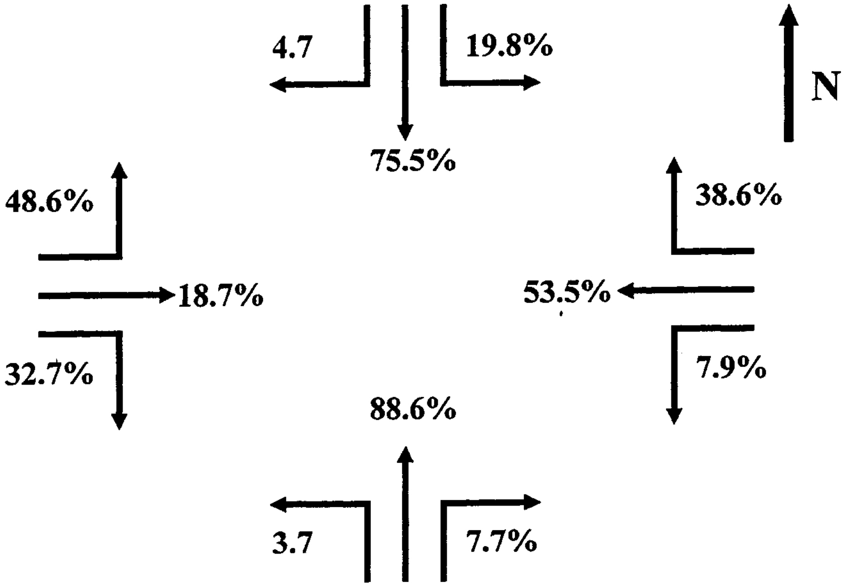


Figure 3.14. Turning movements during PM-Peak period

turning percentage. Even though the northbound approach possesses a left-turn lane, only about 3 in 100 vehicles make a left turn.

III. 6. 3. 3. Delay estimation

The algorithm described in the earlier sections was used to estimate the delay. As mentioned in Section 6.3.1. data were obtained with the intersection operating under actuated and pretimed control. The results of the estimation are analyzed and evaluated in the following sections.

III. 6. 3. 3. 1. Pretimed control

Table 3.1 shows the total delay in vehicle hours obtained over the two hour periods. For each period and approach, figures 3.15 and 3.16 show the average delay per vehicle in minutes. The total delay values across time periods cannot be directly compared against each other as the total delay depends on the volume, which could be different for each period. Therefore, the average delay per vehicle is used in this evaluation.

From the figures, in two of the three periods, southbound approach has a higher average delay per vehicle than northbound, owing to high-left turning volumes. Owing to a higher volume in the AM-Peak period, the total delay is higher for the northbound than the southbound approach. Owing to a high left-turning volume and a shorter green time, the eastbound approach also has a high average delay per vehicle. Another reason might be that the accumulated errors are large because of light traffic demand on the eastbound approach. The short green interval explains the high value of the average delay per vehicle for the westbound approach.

Table 3.1. Total delay in veh-hrs (two-hr periods) - Pretimed control

	AM-Peak Period	Off-Peak Period	PM-Peak Period
Northbound	14.51	8.67	11.56
Southbound	12.08	11.02	10.66
Eastbound	10.7	9.08	13.12
Westbound	9.42	11.6	10.58
Total	46.71	40.36	45.92

The estimated delays for the northbound, southbound and westbound approaches are reasonable in the three periods of study. However, certain discrepancies exist in the estimated delays for the eastbound approach. The algorithm might accumulate large errors in light traffic, since such traffic may contribute to increased calculation errors.

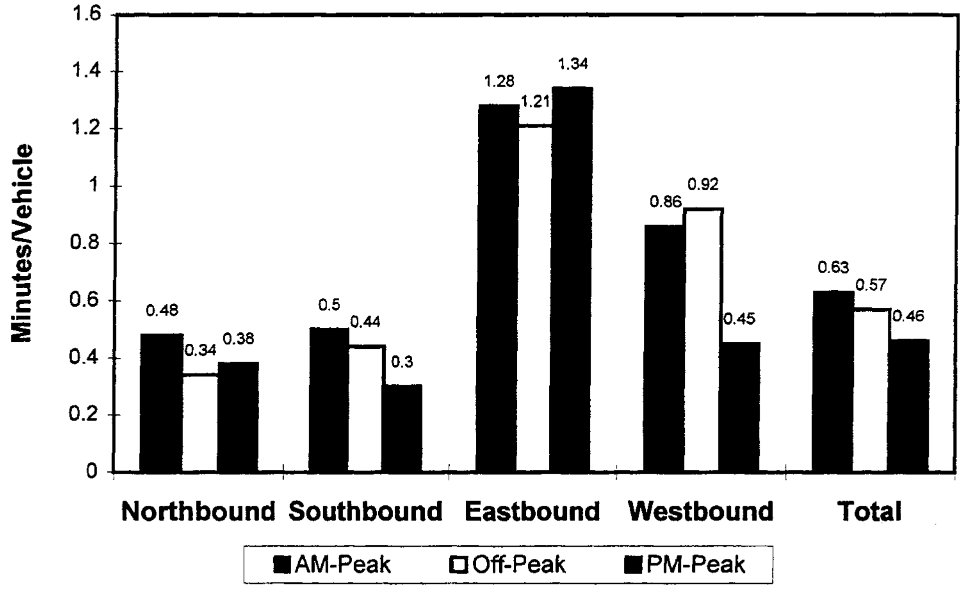


Figure 3.15. Average delay by approach - Pretimed control

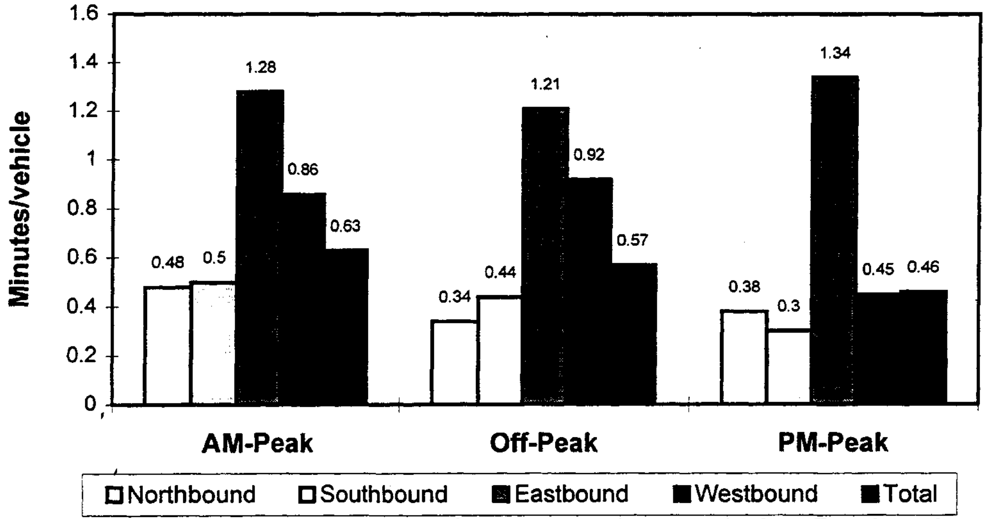


Figure 3.16. Average delay across time periods - Pretimed control

III. 6. 3. 3. 2. Actuated control

Table 3.2 summarizes the total delay in vehicle hours obtained over the two-hour periods. Figures 3.17 and 3.18 illustrate the average delay per vehicle in minutes during each of the periods and for each of the approaches. The values of the average delay per vehicle obtained with actuated control data sets follow the same trends as values obtained from the pretimed control data sets. High left-turning volumes cause a high average delay per vehicle. High volumes at an approach cause high total delay for that approach. The average delay per vehicle on the eastbound approach is high owing to the same reasons mentioned in the previous section. The algorithm may cause errors in the case of light traffic, resulting in a high average delay per vehicle.

Table 3.2. Total delay in veh-hrs (two-hr periods) - Actuated control

	AM-Peak Period	Off-Peak Period	PM-Peak Period
Northbound	15.62	8.03	11.19
Southbound	11.02	10.46	11.76
Eastbound	12.13	9.25	12.11
Westbound	8.99	10.61	9.62
Total	47.76	38.35	44.68

III. 6. 3. 3. 3. Pretimed versus actuated control

Figures 3.19 - 3.21 compare the average delay per vehicle obtained from the pretimed and actuated control data sets during three time periods, i.e., AM-peak, Off-peak and PM-peak. During the AM-peak period, except for the northbound approach, the average delay decreased when the intersection was under actuated control. However, the average delay per vehicle did not change when the whole intersection is considered. The reason for this is that the northbound traffic forms a significant portion of the total-traffic during the AM-peak period.

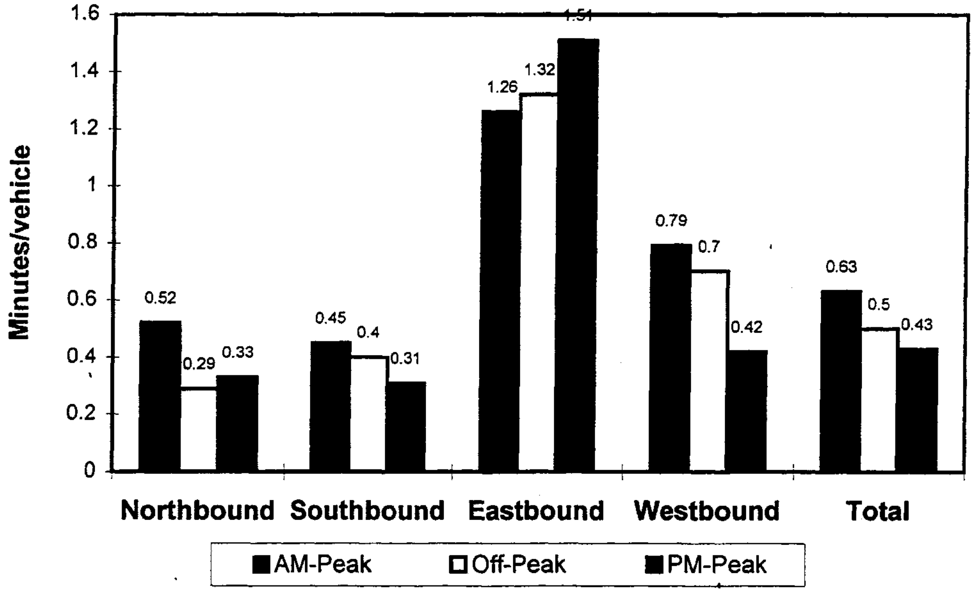


Figure 3.17. Average delay by approach - Actuated control

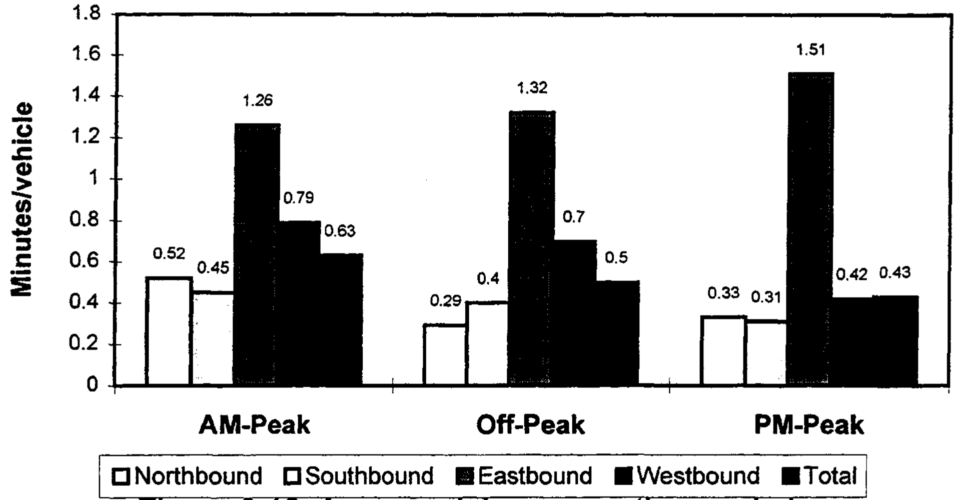


Figure 3.18. Average delay across time periods - Actuated control

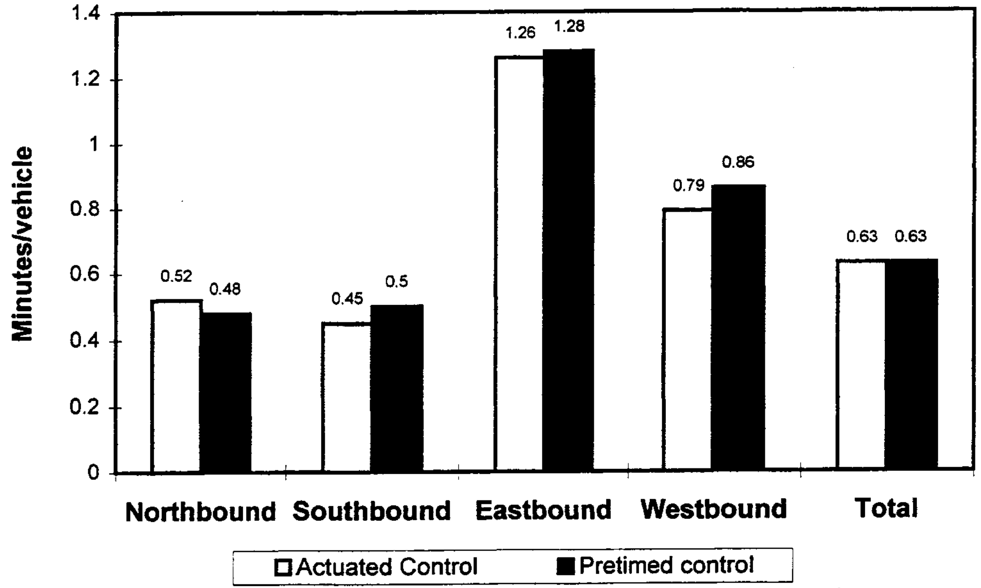


Figure 3.19. AM-Peak - Comparison of average delay

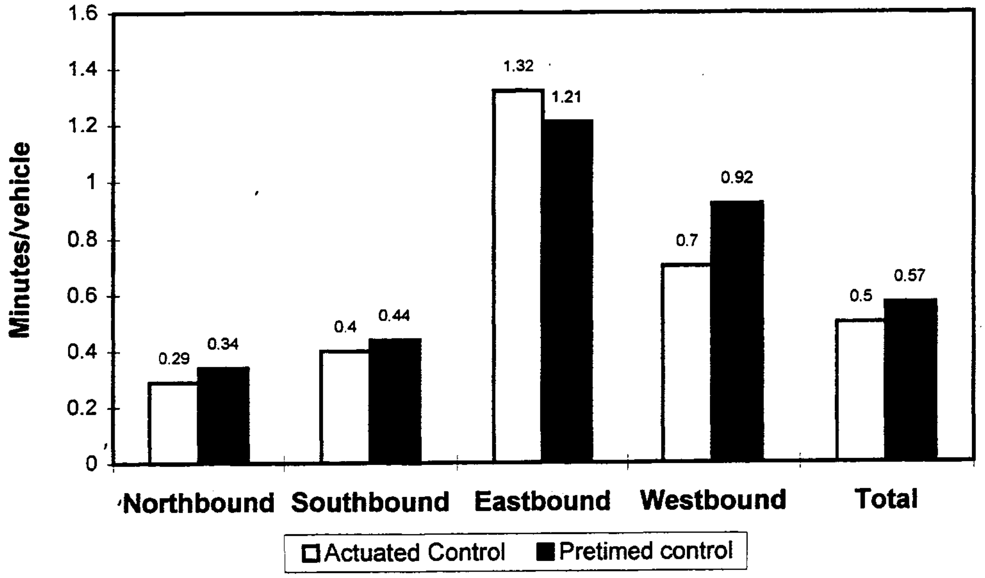


Figure 3.20. Off-Peak - Comparison of average delay

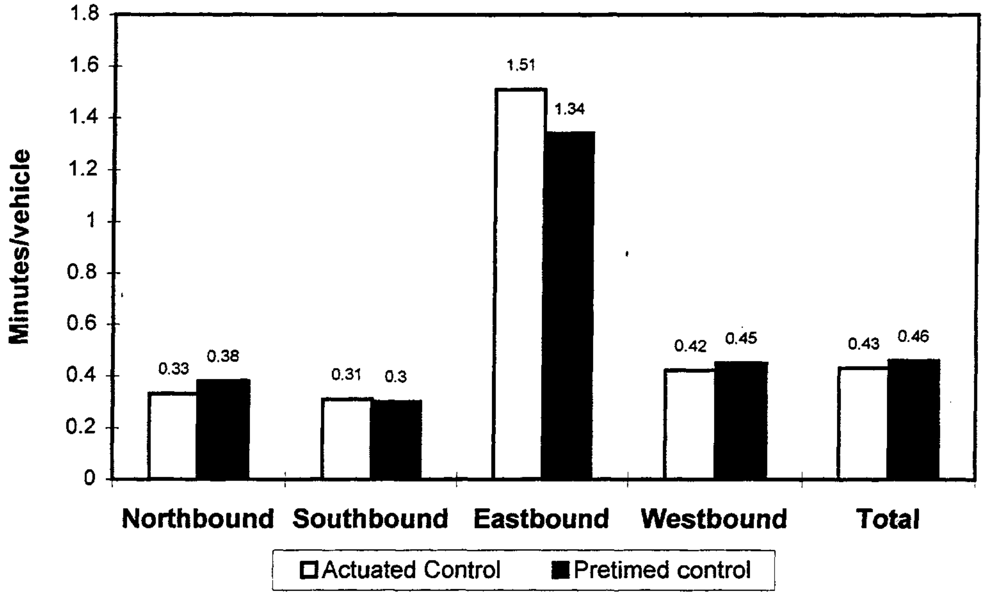


Figure 3.21. PM-Peak - Comparison of average delay

During the Off-Peak period, except for the eastbound approach, the average delay decreased when the intersection was under actuated control. When the intersection was considered as a whole, the overall delay per vehicle decreased from 0.57 min/veh (pretimed) to 0.50 min/veh (actuated). During the PM-peak period, with actuated control, the average delay per vehicle decreased in the southbound and eastbound approaches. The average delay for the intersection also decreased, but not substantially. Not many conclusions can be drawn from such small differences in delay values as there may also exist errors in the estimation of the delay. Further, the traffic data obtained from machine-vision are not 100% accurate. For example, errors in the order of 10% may exist in the calculation of total volumes at the upstream and downstream of an approach.

III. 6. 3. 3. 4. Summary

The maximum reduction in the average delay, with actuated control, has been during the Off-peak period. Actuated control is demand responsive, and is best suited for fluctuating volumes. This implies that the benefits of the actuated control strategy are significant when the volumes are not high. During the AM-peak and PM-Peak periods, the intersection is close to saturation and the benefits of actuated control are not realized.

III. 6. 3. 4. Summary

The traffic demand and turning movements for each approach was determined from the data. Using the delay estimation algorithm developed earlier, results under the pretimed and actuated control strategies were presented and compared. The algorithm was found to be inefficient in low traffic conditions causing high delays. The algorithm is sensitive to lane-changes and false detections, and the resulting errors that accumulate over time.

III. 6. 4. Hurdles in obtaining data

Other than some problems with the machine-vision hardware which prevented data collection early on, the following problems also contributed to the delay in operations.

1. No fixed load limit for the machine-vision unit

   There is no proper (quantified) limit to the number of detectors that can be installed. When this unknown limit is exceeded there is phase lag in the display of the vehicle actuations on the screen, which is the only indication. This was mistaken for a delay in transmission, as the display on the screen would be among the jobs with the least priority, when compared to detection. The error was noticed when the data obtained were analyzed. An approximate value has been found to be the limit of number of detectors that can be installed, and detectors were removed to reduce the load, and allow the system to operate within this limit, before accurate data could be collected.

2. A bug in the data interval filter program

   The filter program was used to convert the binary data files obtained from the machine-vision system to ASCII files. All data collected after December 31st 1995 could not be converted to ASCII formats owing to a error in the program. The error was detected and corrected by Image Sensing Systems.

III. 6. 5. Summary

A delay estimation algorithm that works on machine-vision data, developed during the last phase of the project, was used. Changes were made to the C executable modules to account for the changes in the detector layout. Data were collected at the test intersection over a period of two months with the new detector layout. Turning volumes and traffic demands were determined, delays were estimated for pretimed and actuated control operations, and the results were analyzed.

III. 7. Operational plan for the intersection laboratory

III. 7. 1. Introduction

The machine-vision system is flexible enough to allow re-positioning of detector locations and data collection at various intervals. With the actuated control unit receiving inputs from the machine-vision system, various control strategies can be tested. With the necessary equipment installed, an operational plan is required to fully utilize the resources. The applications within the present environment include, but are not limited to, data collection, testing control strategies and incident detection.

III. 7. 2. Possible applications under the present environment

III. 7. 2. 1. Data collection

Developing a comprehensive intersection traffic database has been a primary objective of this project. With additional cameras at the intersection, detectors can be placed within the intersection as well as upstream of the SB approach, other than the approaches to the intersection. With more coverage from the six cameras, data can be collected extensively from the different parts of the intersection in building up this database.

It has be noted that the performance of the machine-vision system deteriorates with the increase in the number of detectors beyond a limit. This limit cannot be exactly quantified as a number because different types of detector consume the resources in different proportions. An indication of crossing the limit could be a phase lag in the display of detector actuations on the screen.

III. 7. 2. 2. Testing control strategies

The main utility of this intersection laboratory is in providing a test bed for experimenting with different control strategies. Pretimed control and different kinds of vehicle actuated control can be implemented at the intersection. A limitation in such an implementation could be due to the lack of user-definable features in the controller unit. If the control strategy involves any logical decision making to be done before the outputs are sent to the controller, it may not be possible to implement it without special hardware.

III. 7. 2. 3. Incident detection

With incidents being the major cause of delays in surface street networks, incident detection is essential. Further, in real time ITS applications, incident detection has to be performed. With the present version of the machine-vision system, data are not obtained on-line. However, the presence of detectors within the intersection helps in recording the incidents, the traffic flow during an incident, etc. Such data would be useful for testing different intersection incident detection algorithms. Without incident data from surface streets, such testing would otherwise, be accomplished by simulation.

III. 7. 3. Future applications

With the latest version of the machine-vision system, data collection can go a rung further up in the ladder with on-line data collection. Currently, the data collection has to be monitored by downloading data and retrieving storage space on the machine-vision system memory.

The new version will include an Interface Developer's Kit, which allows the user to develop custom applications that can collect, process and disseminate real-time measured traffic data collected from multiple machine-vision processors via the scope server. The user can develop applications to collect and process traffic data to analyze traffic trends in real-time or to archive data for off-line analysis. The unit can communicate with variable message signs and display current traffic data in report or graphical form. Additional useful applications could be developed.

One of the most important uses of the new software would be in incident detection. The incident detection algorithm would be based on real-time data and trigger off incident alarms.

III. 7. 4. Summary

Possible applications of the laboratory equipment were discussed. Data collection and testing of traffic management schemes can be carried out without many changes to the system. For incident detection, major changes would be required in the detector files, to keep the load index within limits. With enhancements in the machine-vision system, further applications can be developed to fully utilize the laboratory.

III. 8. Summary

This chapter deals with the development of an operational plan for the live intersection laboratory. The intersection selected as the laboratory is equipped with a system for machine vision detection, and EPAC300 Actuated Controller Unit for actuated control. Thus, various control strategies can be tested at the laboratory. Intersection delay time, the well accepted indicator of performance of traffic management strategies, is used to evaluate the different control strategies and arrive at the most efficient strategy. The delay algorithm developed in the previous phase of the project was used to estimate the delay. An intersection traffic database has been designed and constructed with the available data obtained from the machine-vision system. This database was created using the PARADOX relational database management system. Applications of this intersection laboratory were also outlined. With some enhancements to the software and the setup, the laboratory will prove to be very valuable to researchers and traffic engineers.

IV. DEVELOPMENT OF MICROSCOPIC SIMULATOR FOR INTERSECTION LABORATORY

IV.1 Introduction

Developing a simulator that can emulate the intersection laboratory environment is of critical importance in operating the laboratory and developing new control strategies. In this research, a new microscopic simulator was developed for the following purposes;

- Efficient and realistic representation of the traffic behavior at the intersection.
- Capability to emulate current and future features of the machine vision detection system at the lab.
- Flexible modular design that allows
  - Interaction with other control modules outside the simulator, e.g., real-time data acquisition and incident detection,
  - Efficient calibration of control parameters,
  - Testing new traffic models.

Figure 4.1 shows the major modules of the laboratory simulation environment. The rest of this chapter summarizes the structure and modeling approaches of the intersection simulator.

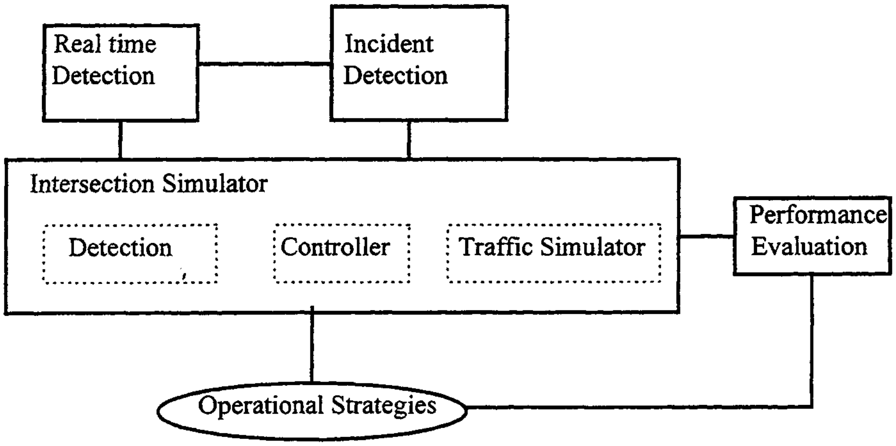


Figure 4.1. Framework of intersection laboratory simulation environment

IV.2 Structure of Intersection Laboratory Simulator: InterLab

IV.2.1 Data structure

The intersection laboratory simulator (InterLab), developed in this research adopts a cellular automata (CA) approach with a new microscopic car-following model. CA models use discrete representation of space (cell) and time, and each cell is updated following a given set of rules. In the context of road traffic, a road is represented by boxes which can contain vehicles. A set of rules for vehicle movements is then developed to move all the vehicles from one box to another. This approach combines microscopic resolution and computational efficiency.

Figure 4.2 shows the geometry of the intersection laboratory in a discretized format. In the current version of InterLab, each approach is modeled as a two-dimensional array with 20 foot-cells; such modeling is based on the assumption that the sum of average vehicle length and minimum headway is approximately 20 feet. Each cell contains the following set of data, updated every 1 second;

- Vehicle existence (Yes/No)
- Current vehicle destination.
- Current vehicle speed.
- Distance to front vehicle.
- Front vehicle speed.

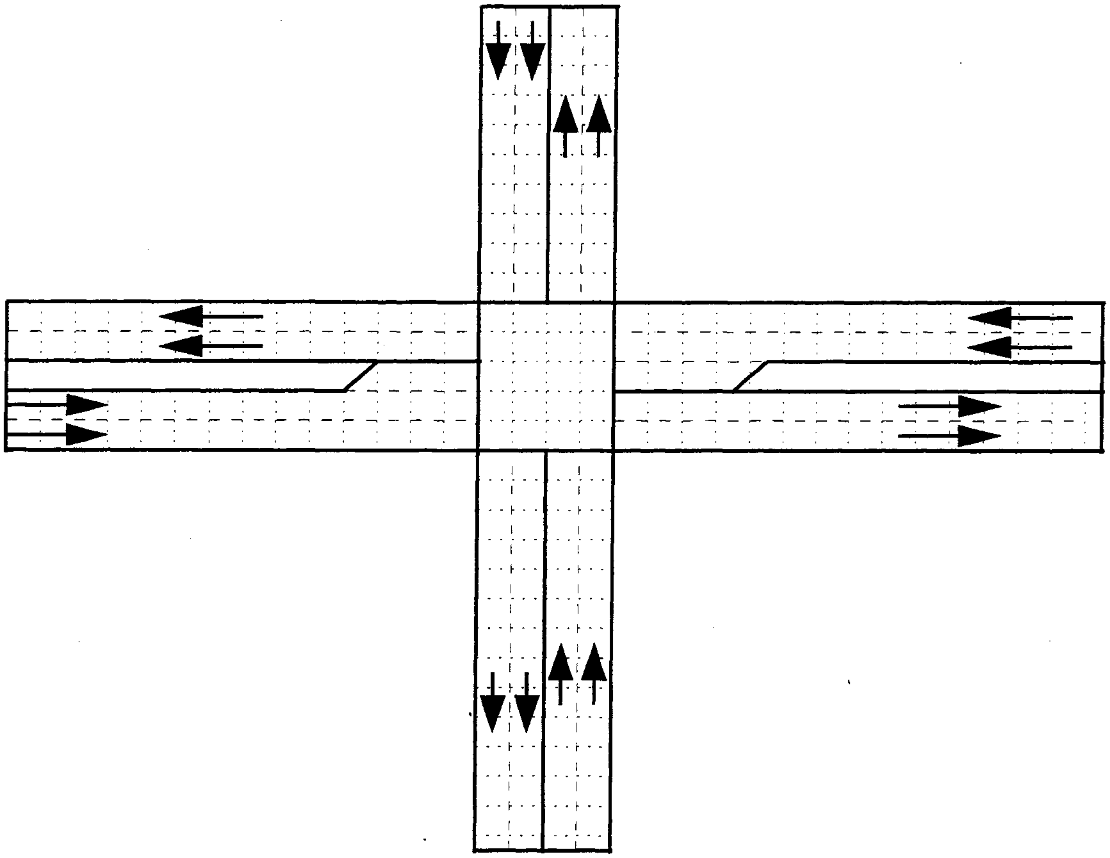


Figure 4.2 Discretized intersection laboratory for InterLab

IV.2.2 Software structure

InterLab was developed using a graphical programming language, LabView [18]. It creates programs in block diagram form with built-in input/output user interface. LabView programs are called virtual instruments (VIs) and adopt a hierarchical and modular structure. A VI can be a subVI within another VI and each VI can be represented as a graphical icon. The data dependency among different VIs can be explicitly visualized in a block diagram, which is a source code. The principle that governs LabView program execution is called data flow; this principle contrasts with the control flow method of executing a conventional, instruction-driven program. Figure 4.3 illustrates the block diagram, i.e., source code, of the main VI of the InterLab software, which consists of the following major VIs;

1) New vehicle generation VI

2) Update the location of all vehicles VI

3) Detection and measurement VI

4) Signal control VI

The data flow among the above modules is clearly indicated in the block diagram; this flow is executed every 1 second during the user-specified simulation period.

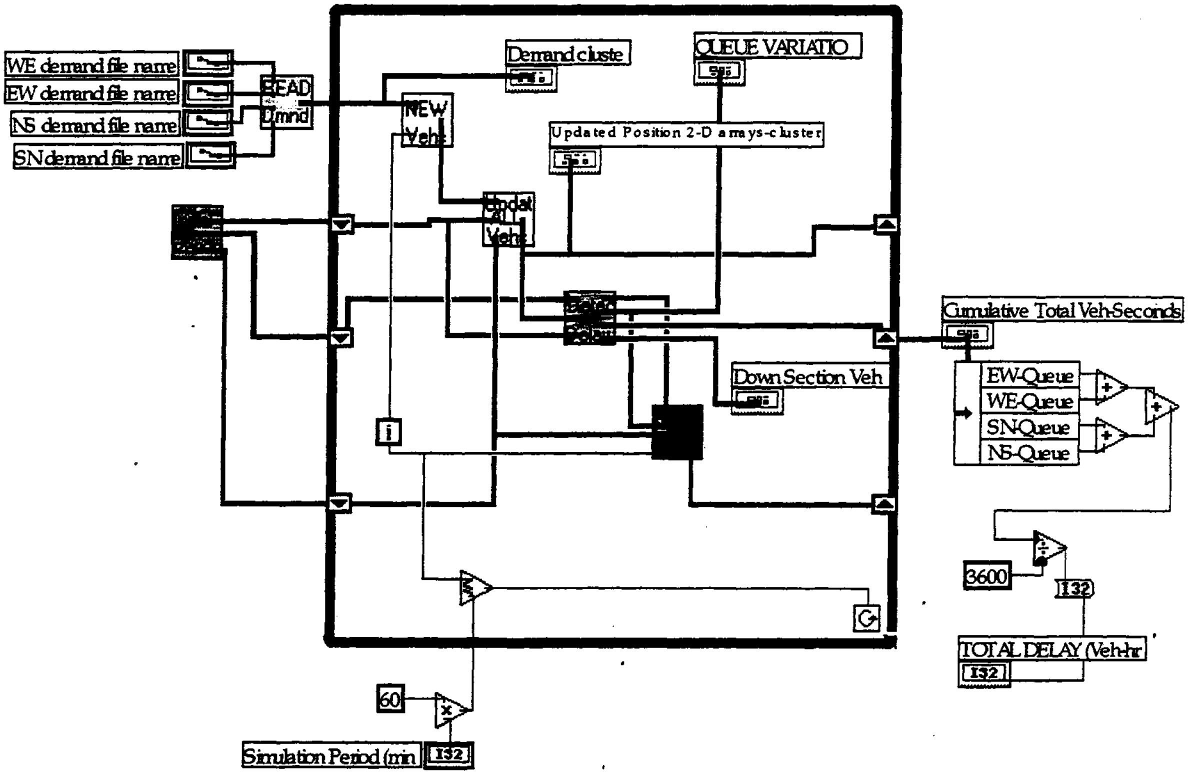
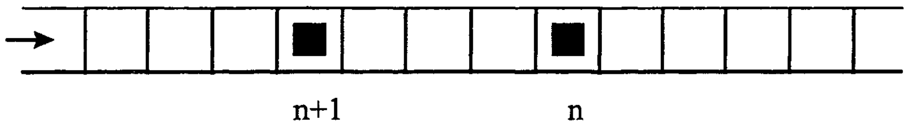


Figure 4.3 Block diagram of the main VI of InterLab

IV.3 Microscopic modeling of vehicle movements

IV.3.1 Development of a car-following model

The model developed in this research updates the location and speed of each vehicle every one second as follows:



For calculating the speed $V_{n+1}(t+1)$ of vehicle, n+1, at the next time step:

$V_{n+1}(t+1) = V_{n+1}(t) + \Delta v$

Find the headway (space), D, from vehicle n+1 to the front vehicle, n;

If $D \geq D_{max}$, then $\Delta v = \alpha$,

where, D_{max}: headway for maximum acceleration,

α: maximum acceleration per second.

Else, if $V_n \geq V_{n+1}$, then

$\Delta v = \text{Min}\ [\ D^* \ \alpha / D_{max},\ (V_n - V_{n+1})]$

where, $V_{n+1}(t) + \Delta v <=$ Max. Speed

if $V_n < V_{n+1}$, then

$\Delta v = (V_n - V_{n+1}) * [1 - \beta (D - D_{min}) / D]$

where, D_{min} = minimum headway,

β = deceleration parameter.

Max. Speed = Speed limit

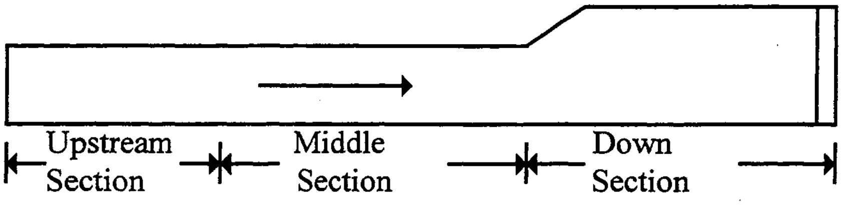
IV.3.2 Basic procedure for updating vehicle location

```
For each time step (1 second),
    For each roadway,
        For each vehicle,
            Find front vehicle information, i.e., distance and speed,
            Determine new speed,
            Determine moving range,
            Find any front vehicle in the moving range,
            Determine new location
            Move to new location
```

Figure 4.4 shows the vehicle update block-diagram for the downstream roadways.

IV.3.3 Treatment of lane changing

To reflect the drivers' lane changing behavior at the roadways approaching an intersection stopline, each approach is divided into three sections and each section is assumed to have different lane-changing characteristics:



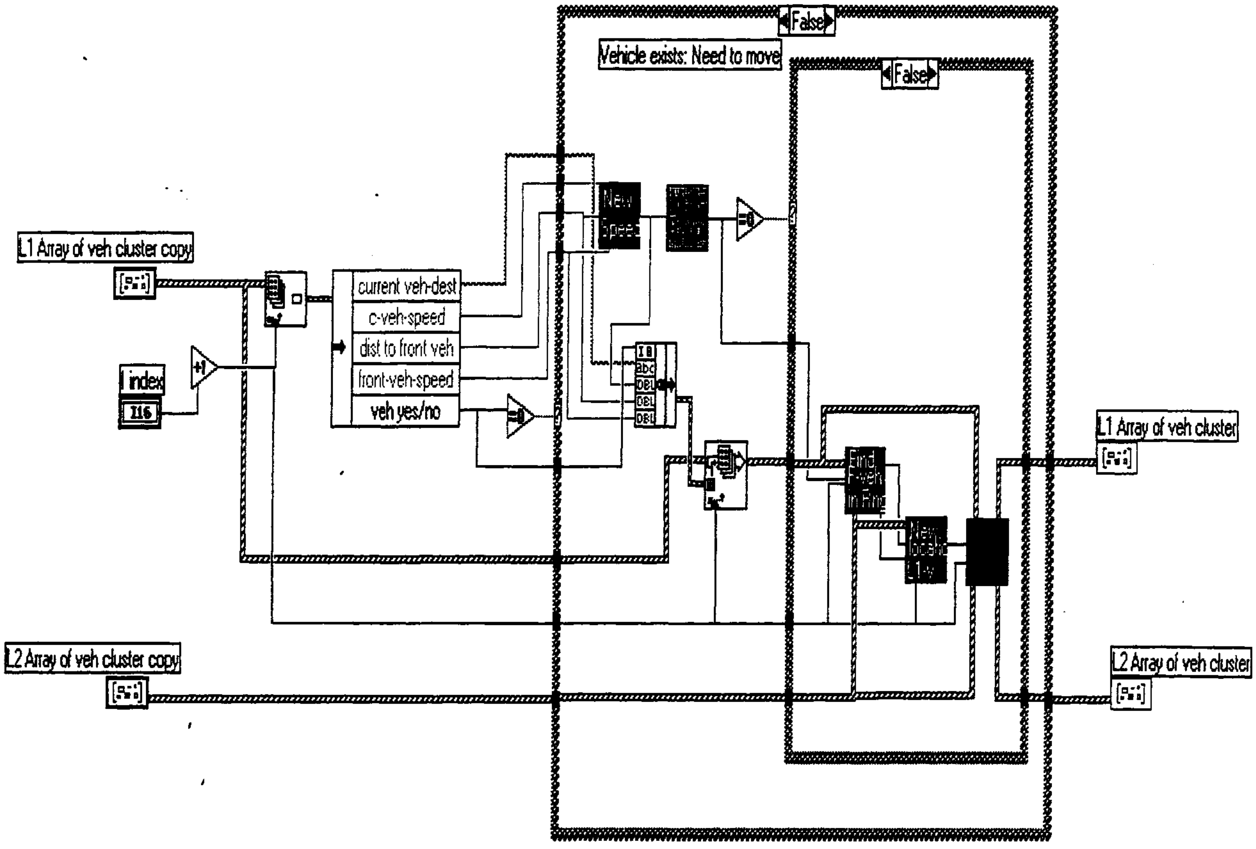


Figure 4.4 Block diagram for updating vehicle location at downstream approaches

Upstream section: Free lane-changing zone. Vehicles change lanes if possible.

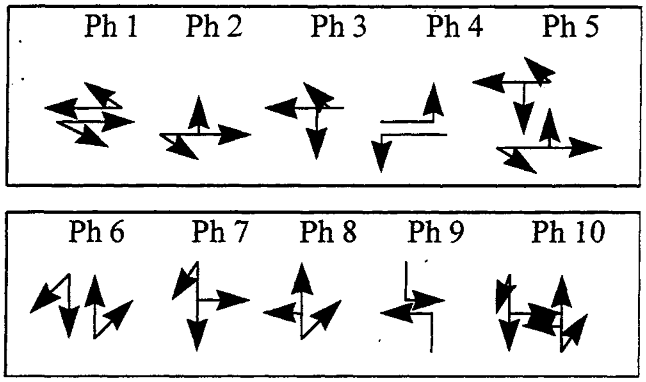
Middle section: Restricted lane-changing zone. Vehicles change lanes depending on their destinations, e.g., left-turning vehicles change to the left lane, and through vehicles change to either right or left lane depending on the availability.

Down section: No lane-changing zone.

For each section, a different lane-changing algorithm was developed and incorporated into the "new location determination" module. The length of each section depends on the geometry of an intersection.

IV.3.4 Modeling vehicle movements at intersection

InterLab models explicitly the movements of vehicles at the intersection at different signal phases. The current version of InterLab handles the following set of signal phases;



In the above phase definitions, Phases 1 - 5 are for East/West approaches with left-turn pockets and Phases 6- 10 for North/South approaches at the intersection laboratory.

For each phase, the vehicles at the intersection are first moved to appropriate downstream roadways depending on their destinations. The vehicles at the upstream approaches are then moved into the intersection following destination-specific paths. For example, Figure 4. 5 illustrates the vehicle movement paths for Phase 1.

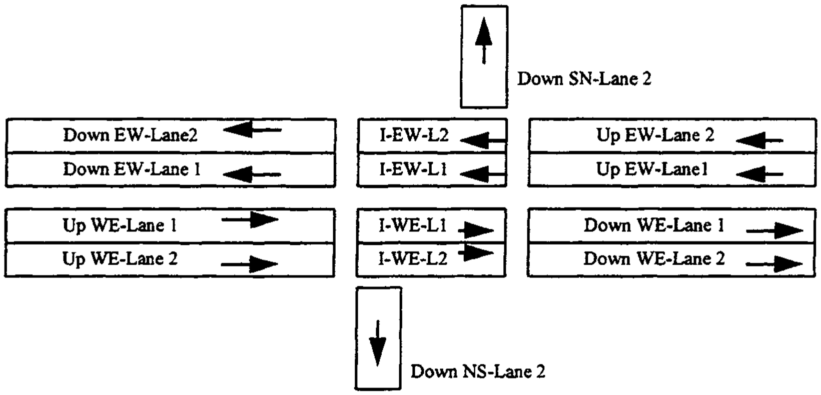


Figure 4.5. Vehicle movement paths for Phase 1

IV.3.5 Modeling vehicle arrival at upstream section boundaries

The vehicle arrival pattern at the upstream boundaries is modeled using a modified uniform distribution approach. This approach ensures the generation of the exact number of vehicles as specified by user for each time interval. At each time step during the simulation, i.e., every one second, the algorithm determines if a new vehicle needs to be generated as follows;

```
step 1: Headway = Round [User defined headway]
     2: Headway Error = User defined headway - Headway
     3: Headway  = Round [headway + Headway Error]
        Go back to step 2;
```

The source code for the above algorithm is shown in Figure 4.6

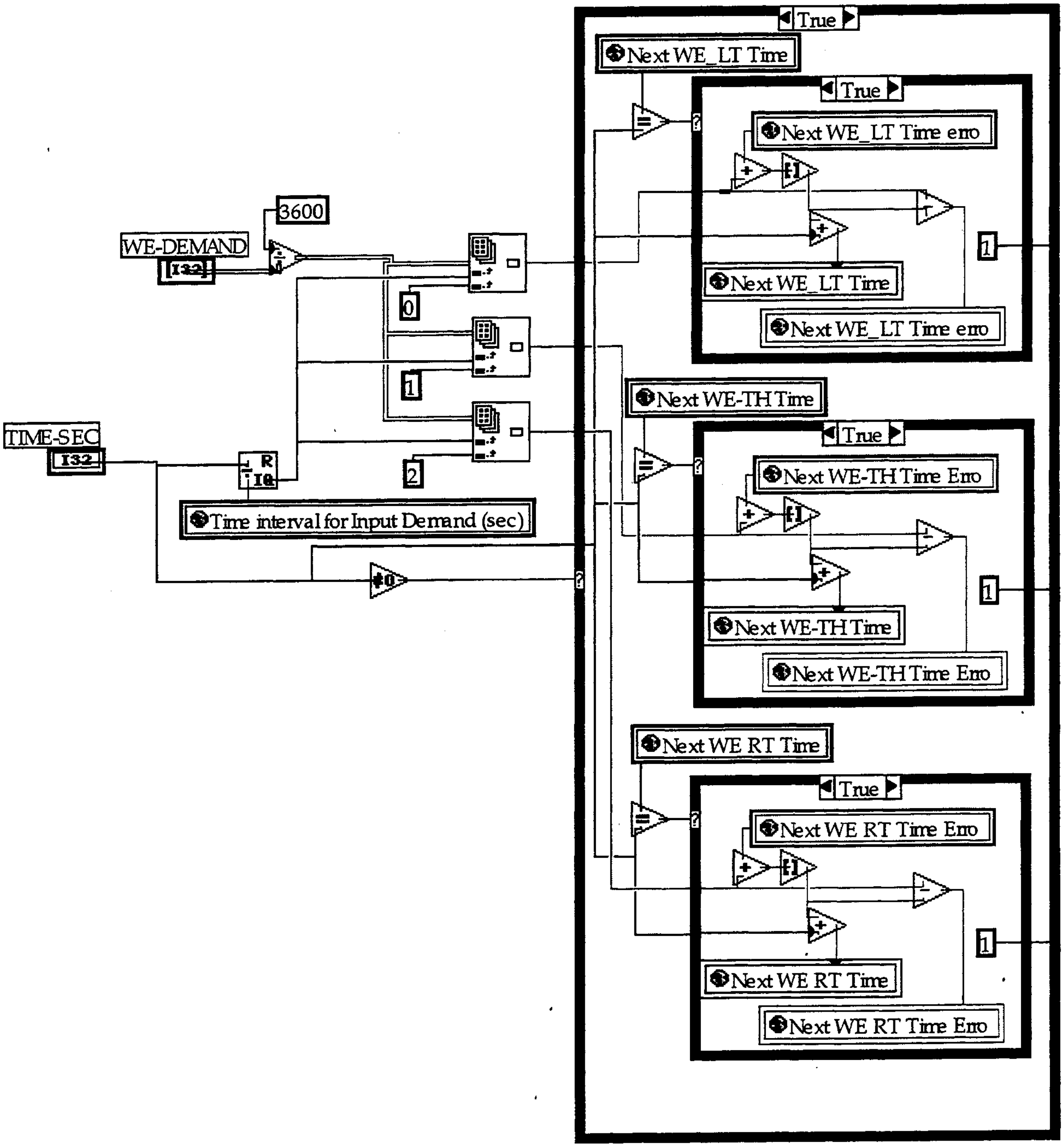


Figure 4.6 Source code for the new vehicle generation module

IV.4 Testing the simulator

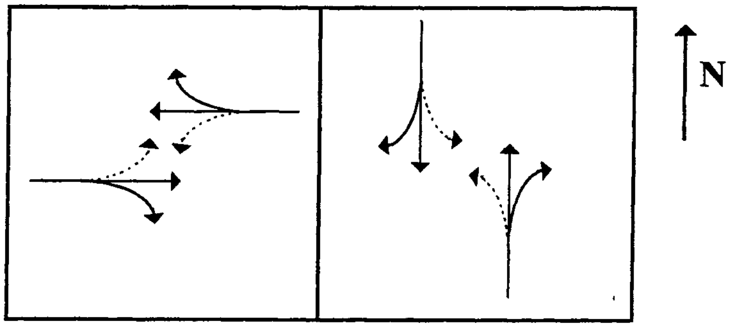
A model that emulates real-time behavior can be tested qualitatively and quantitatively. The former method involves application of basic tests for which the results are known. For example, an increase in approach demand results in delay increase for that approach. These simple tests can also be considered as adequacy checks on the model. Measures of effectiveness that are available for qualitatively testing intersection control strategies include the total travel time, i.e., total vehicle hours of travel for all vehicles at the intersection. Quantitative testing involves calibration, which estabilishes the values of model parameters that best replicate real traffic behavior.

IV.4.1 Qualitative testing

For qualitative testing of the new intersection simulator, the real data from the field were not used. Simple phase sequences and constant volumes were, instead, used to check the adequacy of the model. These tests were later followed by complex ones. In a typical simple test, a base case scenario is run initially and the resulting travel times are noted. Changes are made to this base case and the results are studied. If the changes made in the travel times is a logical outcome of the changes to the input, the model is said to behave satisfactorily. Many such tests were conducted to test the different aspects of the model. In the following sections, three simple tests, conducted to illustrate the adequacy of the model, are shown.

Base case

The geometry of the test intersection is similar to the Franklin and Lyndale intersection. The east-west approaches have two through lanes and a left turn pocket, and the north-south approaches have two lanes. There are two phases (each with all movements); green times are 40 seconds for the EW approaches, and 30 seconds for the NS approaches, which is shown in fig 4.7. The simulation period is 30 minutes. Turn and through volumes for the approaches are shown in table 4.1 below.



Phase 1
40 seconds

Phase 2
30 seconds

Figure 4.7. Phase sequence for testing

Table 4.1. Demand data for testing

Approach	Left turn volume	Through volume	Right turn volume
Northbound, Southbound	50 veh/hr	500 veh/hr	100 veh/hr
Eastbound	200	950	200
Westbound	250	1100	100

The total vehicle hours obtained for the base case are shown in table 4.2.

Table 4.2. Total vehicle hours for base case

	Total vehicle time for the simulation period
Northbound	36293 veh sec
Southbound	36293 veh sec
Eastbound	43149 veh sec
Westbound	44321 veh sec
Intersection	44.46 veh hr

Test case 1

In this test, the green time for the EW phases (all movements) was increased by 5 seconds and the new value was 45 seconds. The expected result is total travel time reduction in the eastbound and westbound approaches, and an increase in the other two. The travel times obtained in the base case and test 1 are compared in table 4.3. The obtained results follow expected trends, thereby upholding the adequacy of the model.

Table 4.3. Comparison of base and test case 1

	Base case: **Total vehicle time**	**Test case 1:** **Total vehicle time**
Northbound	36293 veh sec	39073 veh sec
Southbound	36293 veh sec	39073 veh sec
Eastbound	43149 veh sec	42577 veh sec
Westbound	44321 veh sec	42299 veh sec
Intersection	44.46 veh hr	45.28 veh hr

Test case 2

In this test, the through volume demand for the Eastbound approach was increased from 1100 vph to 1200 vph. The travel times obtained are compared with the base case travel times in table 4.4.

Table 4.4. Comparison of base and test case 2

	Base case: **Total vehicle time**	**Test case 2:** **Total vehicle time**
Northbound	36293 veh sec	36293 veh sec
Southbound	36293 veh sec	36293 veh sec
Eastbound	43149 veh sec	44937 veh sec
Westbound	44321 veh sec	44610 veh sec
Intersection	44.46 veh hr	45.04 veh hr

The total travel times in the north and southbound approaches have not been affected. The increase in the demand for the eastbound approach has increased the total travel time in that approach. As left turns from the eastbound approach are permitted left turns, the number of the eastbound gaps that can be accepted by westbound left-turning vehicles has decreased with the increase in the eastbound through volume. Therefore, the total travel time for the westbound approach has also increased. The values of the total travel times obtained seem reasonable and, therefore, support the adequacy of the model.

Test case 3

In this case, the phase for the eastbound approach was changed to a leading phase by 10 seconds. Therefore, the total green time for the eastbound approach became 50 seconds, and the westbound green time remained at 40 second. The total travel times are shown in table 4.5.

Table 4.5. Comparison of base and test case 3

	Base case: **Total vehicle time**	**Test case 3:** **Total vehicle time**
Northbound	36293 veh sec	40382 veh sec
Southbound	36293 veh sec	40382 veh sec
Eastbound	43149 veh sec	40444 veh sec
Westbound	44321 veh sec	46527 veh sec
Intersection	44.46 veh hr	46.59 veh hr

The eastbound total travel time has decreased, while the total travel times in the other three approaches have increased owing to an increase in their red times. The number of total vehicle hours for the intersection has also increased. The results look rational and support the objectives of the model.

IV.4.2 Model calibration and quantitative testing

Calibration of model parameters using real data is essential for obtaining credible results from the simulator. It is not expected that the results from a simulator will match real data, but the closer the performance the more reliable the results would be in further simulations.

The data for this testing were collected from the Franklin and Lyndale intersection laboratory using the machine vision system. In particular, volume, speed, occupancy, etc. data were obtained at intervals of 20 sec. Volume data were aggregated and mean vehicle speed averaged over a three minute period. For determining the period length, it was noted that if this duration is very short, traffic will exhibit a large number of sudden fluctuations; conversely, if it is too long, traffic will be void of changes. The data used was obtained over an hour from 4 pm to 5 pm (PM peak period) on 03-11-96. The demand patterns are shown in tables 4.6 and 4.7 for the four approaches. The phase sequence used in the calibration is shown in Figure 4.8. The speed data were obtained from virtual speed traps placed in the approaches between 120 - 220 feet from the intersection stoplines. The volume data were used as the input to the simulator for generating arrival patterns. The simulator only generates uniform arrival patterns; such patterns are deterministic and this allows immediate control over the input vehicle values. The duration of the simulation period was one hour.

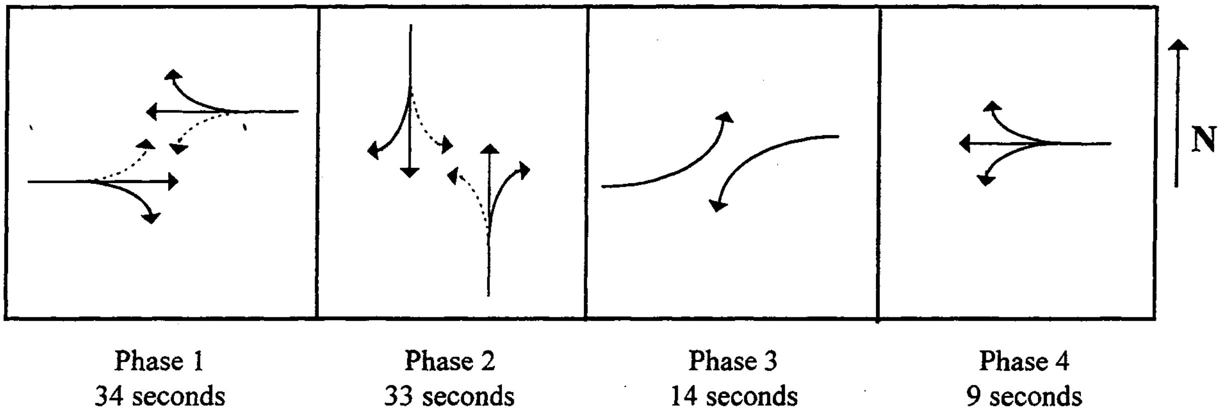


Figure 4.8. Phase sequence used in calibration

Table 4.6. 3-minute demand pattern; Northbound and Southbound approaches

Time	Northbound volume (vph)			Southbound volume (vph)		
period	Left	Through	Right	Left	Through	Right
16:00 - 16:03	20	898	62	163	657	20
16:03 - 16:06	40	1119	121	251	965	104
16:06 - 16:09	0	720	80	225	990	45
16:09 - 16:12	20	1500	40	289	829	82
16:12 - 16:15	41	699	20	279	580	21
16:15 - 16:18	0	978	162	221	955	44
16:18 - 16:21	40	1080	80	191	1148	21
16:21 - 16:24	82	1016	62	287	1127	66
16:24 - 16:27	40	500	160	428	970	22
16:27 - 16:30	0	1713	167	280	1037	43
16:30 - 16:33	102	678	20	398	1087	155
16:33 - 16:36	85	747	128	111	605	44
16:36 - 16:39	20	1752	208	274	1544	42
16:39 - 16:42	103	975	82	146	930	24
16:42 - 16:45	40	1459	81	435	1143	62
16:45 - 16:48	61	718	81	376	923	41
16:48 - 16:51	42	1474	84	186	1328	46
16:51 - 16:54	60	720	220	290	919	111
16:54 - 16:57	62	1076	62	69	739	92
16:57 - 17:00	21	1058	21	0	1382	78

Table 4.7. 3-minute demand pattern; Eastbound and Westbound approaches

Time period	Eastbound volume (vph)			Westbound volume (vph)		
	Left	Through	Right	Left	Through	Right
16:00 - 16:03	110	60	30	40	500	160
16:03 - 16:06	60	20	20	20	120	280
16:06 - 16:09	100	1	119	80	440	520
16:09 - 16:12	40	40	60	60	300	180
16:12 - 16:15	120	0	140	80	500	420
16:15 - 16:18	100	100	100	20	200	400
16:18 - 16:21	120	20	100	100	280	240
16:21 - 16:24	140	0	20	60	180	100
16:24 - 16:27	140	80	100	20	600	80
16:27 - 16:30	90	100	70	20	480	200
16:30 - 16:33	90	60	30	100	420	480
16:33 - 16:36	100	140	80	100	560	320
16:36 - 16:39	60	40	80	20	180	340
16:39 - 16:42	140	100	80	80	340	560
16:42 - 16:45	20	80	20	20	80	300
16:45 - 16:48	140	20	160	120	560	480
16:48 - 16:51	100	20	60	100	380	280
16:51 - 16:54	140	80	0	120	480	240
16:54 - 16:57	240	100	120	40	680	180
16:57 - 17:00	140	0	140	100	380	160

The car following model developed in this research uses a speed equation with four parameters: α, maximum acceleration per second; β, the deceleration coefficient; D_{min}, the minimum headway; D_{max} , the maximum headway for full acceleration. Because of the number of parameters, testing all possible combinations of parameter values was not practical. Therefore, the minimum and maximum headways were fixed at 1 and 4 cells, i.e., 20 and 80 feet, respectively. α and β values can be changed to obtain speeds that match speeds obtained in the field. The maximum speed limits in the approaches and while turning within the intersection can also be altered. A number of combinations were tried and a sample of the average speed values on two major approaches over the one-hour period is presented in table 4.8. The detectors, to obtain the speeds, were placed at the same location as in the field intersection laboratory. The pretimed control strategy used by the City of Minneapolis at the intersection laboratory was used with the same phase sequence and duration. As the table indicates, for a particular set of the speed equation parameters and speed limits, the average speed values obtained from simulation are quite close to real data. It should be noted that speed values are averaged over an hour and that nothing can be speculated about the speed distribution in general.

Table 4.8. Speed data comparison

Case	α	β	Max. speed	Max turn speed	Eastbound	Westbound
					simulated	simulated
1	12.0	0.9	35	15	30.0 mph	24.6 mph
2	10.0	0.9	35	15	23.9	24.1
3	7.0	0.9	35	15	20.1	22.4
4	10.0	0.9	30	12	19.7	19.3
5	10.0	0.5	30	12	17.2	18.4
6	8.0	0.5	25	12	17.1	16.2
Real Data					19.6	19.0

IV.5 Summary

In this chapter, a new microscopic simulator for the intersection laboratory was developed using a cellular automata modeling approach with a simplified car-following model. This approach allows computational efficiency with microscopic resolution. Further, detailed algorithms were developed to model the movements of the vehicles at the intersection following their destinations. The new simplified car following model involves a smaller number of parameters than that of the conventional models, and this simplifies the calibration effort. The software was written with LabView, a graphical programming language, whose data flow programming concept allows clear visualization of the interrelationships among different modules. With the built-in user interface and block-diagram type source code, the development and maintenance of the software can be more efficient than with the traditional, instruction-driven programming languages. The test results with the real traffic data collected from the intersection laboratory shows promising performance of the microscopic models, developed in this research, which have less parameters than conventional microscopic models.

V. DEVELOPMENT OF INTERSECTION CONTROL STRATEGY WITH MACHINE-VISION DETECTION

V.1 Introduction

Developing practical control strategies that are effective, but adequately simple to be implemented with existing detection systems is of critical importance in managing urban traffic networks. The control strategy should be able to respond to link-wide level of congestion, which needs to be quantified with the measurements by the detection systems available in the field. In this study, an intersection control strategy is developed using a new link congestion index, which quantifies the level of congestion at each link using the data that can be measured by either loop or machine-vision detection systems. With the new control, by adjusting the parameters in the congestion index formulation, different weights can be applied on each approach depending on the operational objectives. The new control strategy was evaluated in the simulated environment of the intersection laboratory and its performance was compared with those of pretimed and stopline-actuation based control strategies. The rest of this chapter summarizes the formulation of the congestion index, new control procedure and evaluation results.

V.2 Formulation of congestion index and new control procedure

For the link shown in Figure 5.1 the level of congestion is estimated as follows;

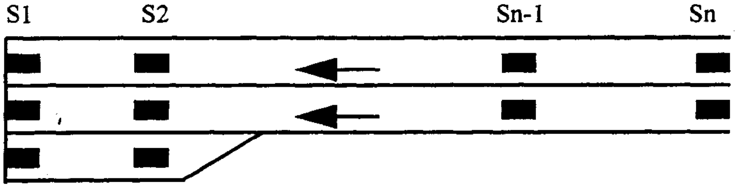


Figure 5.1. Example link (S: Detector location)

$C_{i,t} = \sum \beta_{i,j} D_j$

where, Ci,t = Congestion index for approach i at end of time interval t,

$\beta_{i,j}$ = Parameter for detector location j at approach i,

$D_j = (P_j + V_j)/(1 + V_j)$;

where, $P_j = 1.0$; if detector j is occupied at end of time interval t,

$= 0.0$; otherwise

V_j = number of vehicles crossing detector j during time interval t.

In the above formula;

$D_j = 0$; if $P_j = V_j = 0$; No traffic during the last time interval.

$0 < D_j < 1$; if $P_j = 0$ and $V_j \neq 0$; Congestion level depends on number of vehicles passed.

$D_j = 1$; if $P_j = 1$ and $V_j = 0$; Vehicle did not move during the last time interval.

$V_j \neq 0$; Vehicles passed but detector is still occupied.

As indicated above, the congestion index tries to quantify the level of congestion by combining occupancy information with count data on a 0 to 1 scale for each time interval. Further, the variables in the formula can be estimated with the data collected from the conventional loop detectors as well as the machine-vision systems.

The new control strategy developed in this study uses the cumulative values of the congestion index estimated every one second for the current and next phases. With a given phase sequence, minimum/maximum phase duration and extension intervals, the algorithm compares the cumulative congestion level of the current phase and that of the next phase and determines if an extension is needed. Although the current version of control strategy has a simple fixed-type structure in terms of phase and extension intervals, the use of the congestion index allows the possibility of introducing variable minimum/maximum phase and extension intervals depending on the level of congestion at each approach. The algorithms of the new control strategy and the stopline-actuation based strategy have been coded into the intersection laboratory simulator for

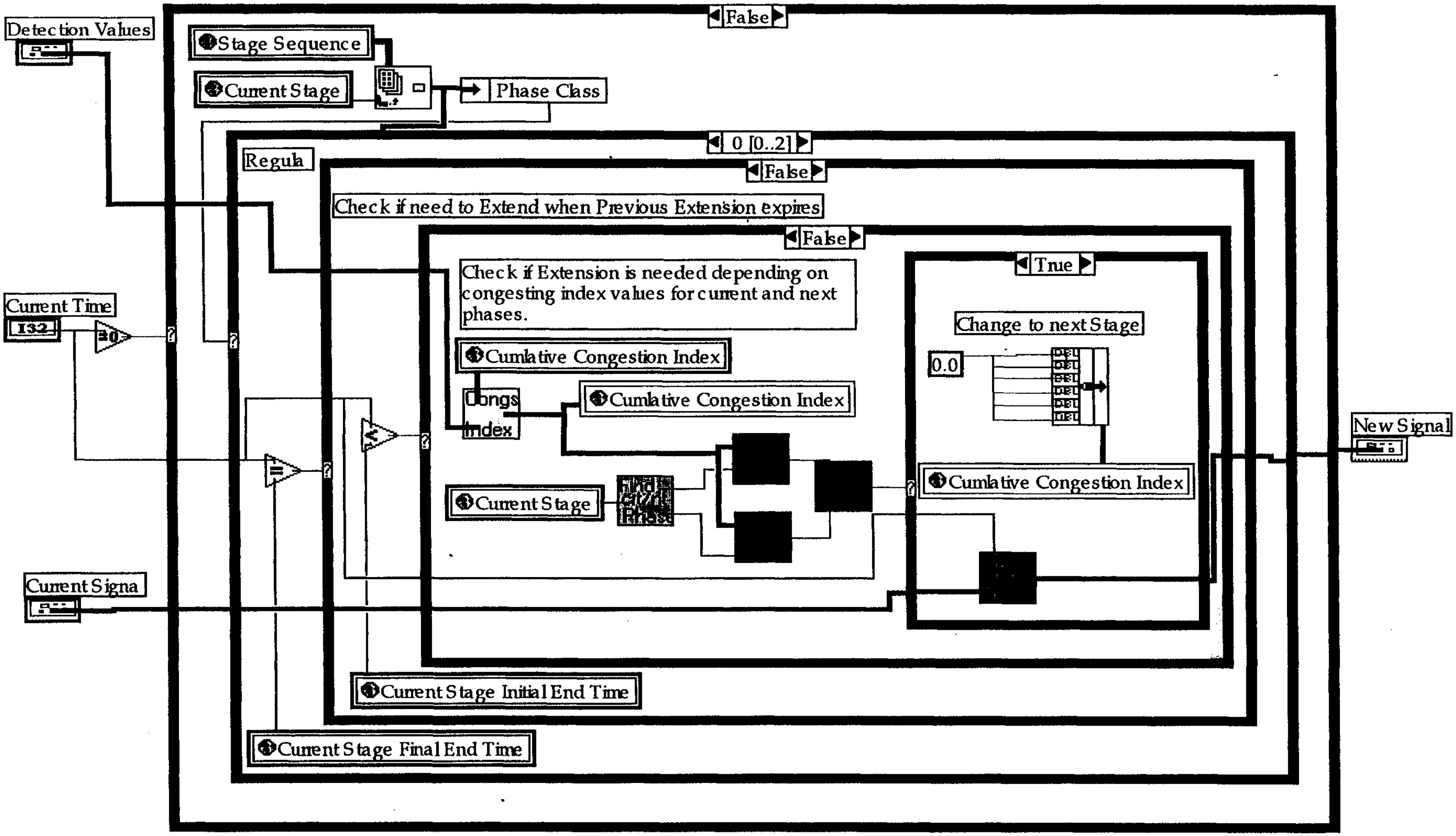

False
Detection Values
Stage Sequence
Current Stage
Phase Class
0 [0..2]
Regula
False
Check if need to Extend when Previous Extension expires
False
Check if Extension is needed depending on congesting index values for current and next phases.
True
Change to next Stage
Current Time
I32
0.0
Cumlative Congestion Index
Congs Index
Cumlative Congestion Index
Cumlative Congestion Index
Current Stage
New Signal
Current Signa
Current Stage Initial End Time
Current Stage Final End Time

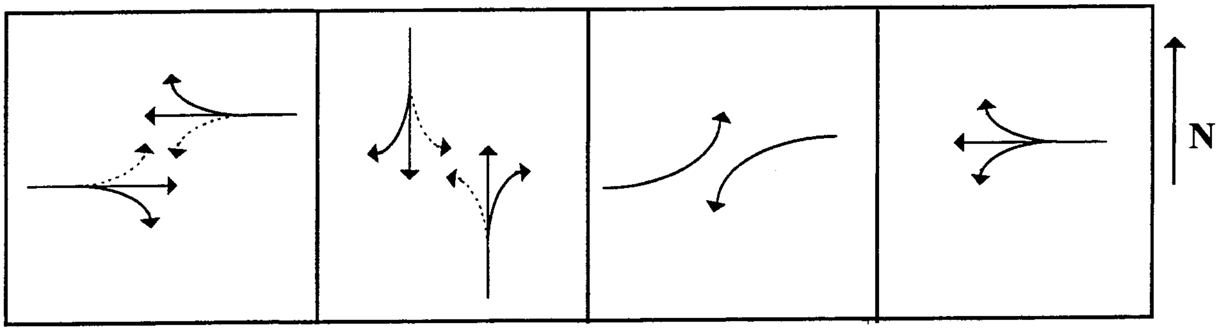
Figure 5.2. Source code of new control strategy

the evaluation. Figure 5.2 shows a portion of the source code of the new control strategy. The current version of the new control uses the detection from three locations per approach including stopline. The test results of the new control in the simulated environment are summarized in the following section.

V.3 Evaluation of the new control strategy

In this section, the effectiveness of the new control strategy over pretimed and stop-line actuated control was analyzed in the simulated laboratory environment elucidated earlier. The measure of effectiveness is the total vehicle hours at each approach or whole intersection. The volume data collected Data were collected over an hour from 4 pm to 5 pm (PM peak period) on 03-11-96 was used as the input demand for evaluation.

To compare the performance of each control strategy, the key control parameters, i.e., the phase sequence, the maximum duration and extension intervals of each phase, remained unchanged across the strategies. Pretimed control used the maximum interval of each phase. The new control strategy calculates the congestion index of a phase based on the presnece and actuation information from the detectors placed at three fixed locations in each approach. This index is used to make decisions regarding the extension of a phase. By changing the weights associated with three locations, the index can be calculated in different ways. The three weights must always add up to 1. Tables 5.3 - 5.5 summarize the simulation results of each control strategy with same demand pattern and maximum phase duration, but with different minimum and extension intervals. The phase duration and the phases used in simulating pretimed control is shown in fig. 5.3. The definition of a phase is different from the one used by the City of Minneapolis. The demand patterns for the four approaches are given in tables 5.1 and 5.2.



Phase 1
34 seconds

Phase 2
33 seconds

Phase 3
14 seconds

Phase 4
9 seconds

Figure 5.3. Phase sequence for pretimed control

Table 5.1. 5 minute demand pattern - Northbound and Southbound approaches

Time period	Northbound volume (vph)			Southbound volume (vph)		
	Left	Through	Right	Left	Through	Right
16:00 - 16:05	36	874	86	177	710	25
16:05 - 16:10	0	1188	96	268	1023	89
16:10 - 16:15	36	900	12	280	677	51
16:15 - 16:20	12	887	133	249	972	39
16:20 - 16:25	61	1030	97	235	1154	39
16:25 - 16:30	24	1255	149	357	1020	39
16:30 - 16:35	109	803	60	294	990	120
16:35 - 16:40	25	1105	154	216	1124	40
16:40 - 16:45	73	1461	98	313	1066	37
16:45 - 16:50	61	910	49	331	1140	53
16:50 - 16:55	75	1153	200	192	772	68
16:55 - 17:00	12	959	37	43	1247	102

Table 5.2. 5 minute demand pattern - Eastbound and Westbound approaches

Time period	Eastbound volume (vph)			Westbound volume (vph)		
	Left	Through	Right	Left	Through	Right
16:00 - 16:05	108	24	36	36	372	264
16:05 - 16:10	60	0	72	48	276	312
16:10 - 16:15	96	24	120	84	468	360
16:15 - 16:20	108	72	108	72	216	360
16:20 - 16:25	168	0	72	48	360	108
16:25 - 16:30	102	12	78	12	468	144
16:30 - 16:35	108	12	48	72	492	312
16:35 - 16:40	120	96	108	72	348	528
16:40 - 16:45	36	72	36	48	108	360
16:45 - 16:50	120	0	108	108	468	456
16:50 - 16:55	180	24	60	107	565	168
16:55 - 17:00	168	60	132	72	456	180

Table 5.3. Performance of Pretimed control

	Total vehicle hour (for one hour simulation)
Northbound	8551 veh sec
Southbound	92457 veh sec
Eastbound	94448 veh sec
Westbound	91979 veh sec
Intersection	**79.84 veh hr**

Table 5.4. Performance of Actuated Control

Case #	Northbound (veh sec)	Southbound (veh sec)	Eastbound (veh sec)	Westbound (veh sec)	Total (veh hr)
Case 1 (5,5,5,4)	91736	97617	95046	8268	81.30
Case 2 (5,5,5,3)	90872	97843	101485	7860	82.79
Case 3 (10,10,6,4)	89424	99846	97444	7938	81.85
Case 4 (20,20,8,4)	90683	96444	89800	8039	**79.16**
Case 5 (21,21,8,4)	90646	95012	89586	8212	**78.74**

The five cases in table 5.4 were obtained by changing the extension interval and/or the minimum duration of each phase. The maximum duration of each phase is same as used in the pretimed control. The minimum duration and extension interval set for each case are given in parantheses. The first three numbers are the minimum duration used for the first three phases. The minimum duration for phase 4 or the lead was fixed at 9 seconds. The last number within the parantheses is the extension interval. In the last two cases, the total travel time is lower than what was obtained with pretimed control and this indicates the potential of actuated control to outperform pretimed control under certain conditions as expected.

Table 5.5. Performance of New control strategy with congestion index

	α	β	γ	**North-bound (vh sec)**	**South-bound (vh sec)**	**East-bound (vh sec)**	**West-bound (vh sec)**	**Total (vh hr)**
1	0.1	0.2	0.7	88612	94139	76079	7981	**74.11**
2	0.2	0.2	0.6	89235	93714	68969	7686	**72.11**
3	0.2	0.3	0.5	90161	93360	69861	7652	**72.51**
4	0.2	0.5	0.3	91571	96118	62892	7637	**71.73**
5	0.2	0.6	0.2	89454	94556	64523	7323	**71.07**

The five cases in table 5.5 have the same minimum phase duration and the extension intervals as in the Case 4 of the stopline-actuated control in table 5.4. As indicated in table 5.5, all cases with the new control strategy show lower total vehicle hours than those of pretimed and actuated control. Parameter α refers to the stop-line detector, β to the intermediate upstream detector and γ to the upstream boundary detector. Stop-line actuated control can be emulated by setting β and γ as zero. The lowest total travel time among the five cases tried in this testing was obtained with case 5, which has the the highest β value. Optimization of performance of the new control strategy has not been an objective in this part of the research, and future work in this direction can lead to improved results.

V.4 Summary

A new intersection control strategy was developed. The new strategy is based on the congestion index, which quantifies the level of link-wide congestion using point measurements, i.e., presence and vehicle counts. The performance of the new control was evaluated with the new microscopic intersection simulator developed in this research. Using the same demand pattern, phase sequence and maximum phase duration, three control strategies, i.e., pretimed, stopline-based actuated and new congestion index based control strategies, were simulated. In pretimed control, the maximum interval of each phase was used as the phase duration. Comparative performance analysis indicates that the new control performed consistently better than pretimed and stopline actuated control. Further research should address the introduction of variable minimum and maximum phase intervals based on the concept of congestion index. Optimization of the parameters in the congestion index and extension to network control also need to be studied.

VI. INTEGRATION OF OPTIMIZATION ALGORITHM AND DEVELOPMENT OF NEURAL NETWORK-BASED CONTROL STRATEGY

VI.1 Introduction

In this chapter, we develop and evaluate an urban-network traffic-adaptive control strategy, which is based on neural networks. First, we briefly review neural network models and describe the backpropagation neural network and simulated annealing optimization algorithms. Second, we briefly describe the Hopfield optimization neural network model on which the strategy is based, the specific neural network structure for intersection traffic control, and the strategy and its implementation in C++. Third, we describe the work related to NETSIM modification in order to simulate the new strategy, and present the simulation results that compare the new strategy to current actuated control.

VI.2 Optimization algorithms for intersection control: Backpropagation neural network and simulated annealing algorithm

Artificial neural networks(ANNs), whose structures are based on our present understanding of biological nervous systems, have been studied for many years in the hope of achieving human-like performance in various practical computational applications. These models are composed of many nonlinear computational elements operating in parallel and arranged in patterns reminiscent of biological neural network. Computational elements or nodes are connected via weights that are typically adapted during use to improve performance. Research has shown that ANNs have greatest potential because of their much higher computation rate and fault tolerance.

Neural network classifiers are non-parametric and make weaker assumptions concerning the shapes of underlying distributions than traditional statistical classifiers. They may thus prove to be more robust when distributions are generated by nonlinear processes and are strongly non-Gaussian. They can perform three different tasks. First, they can identify which class best represents an input pattern, where it is assumed that input has been corrupted by noise or some other process and classes are given. Second, the classifiers can be used as an associative memory, where the class exemplar is desired and the input pattern is used to determine which exemplar to produce. A third task they can perform is to vector quantize or cluster the N inputs into M clusters. Several neural network models can be used as classifiers very well, such as the Hopfield net, the Carpenter/Grossberg classifier, Kohonen's self organizing model, and Multi-layer perceptron.

The Hopfield net can be used as an associative memory or to solve optimization problems. The Carpenter/Grossberg classifier has been designed to form clustered and is trained without supervision. Kohonen's self organizing model can produce self-organizing feature maps similar to those that occur in the brain. Multi-layer perceptrons are feed-forward nets with one or more layers (hidden layers) of nodes between the input and output nodes. These additional layers contain hidden units or nodes that make the input and output not directly connected. Since the backpropagation training algorithm was first proposed, Multi-layer perceptron has become the most popular model, especially in the pattern recognition field.

Backpropagation neural network

Backpropagation neural network is a feed-forward multi-layer perceptron with one or more, hidden layers of nodes between the input and output nodes, which can be trained using the backpropagation algorithm. Although it cannot be proven that this algorithm converges, its applications have been shown to be successful in several computational problems. A three-layer network with one hidden layer is shown in Figure 6.1, where:

$$Y_m = f\Big(\sum_{k=0}^{k=K-1} W_{km} Z_k - \theta_m \Big) \text{ , m=0, 1,...,M-1} \qquad (6.1)$$

$$Z_k = f\Big(\sum_{i=0}^{i=N-1} W_{ik} X_i - \theta_k \Big) \text{ , k=0,1,...,K-1} \qquad (6.2)$$

W_{km} is the connection between nodes k and m, usually called weight.

The capabilities of this network stem from the nonlinear function f() used within nodes. Each node in the hidden and output layers simply sums weighted inputs from the previous layer and passes the result through a nonlinearity. Several such alternative nonlinearities are shown in Figure 6.2.

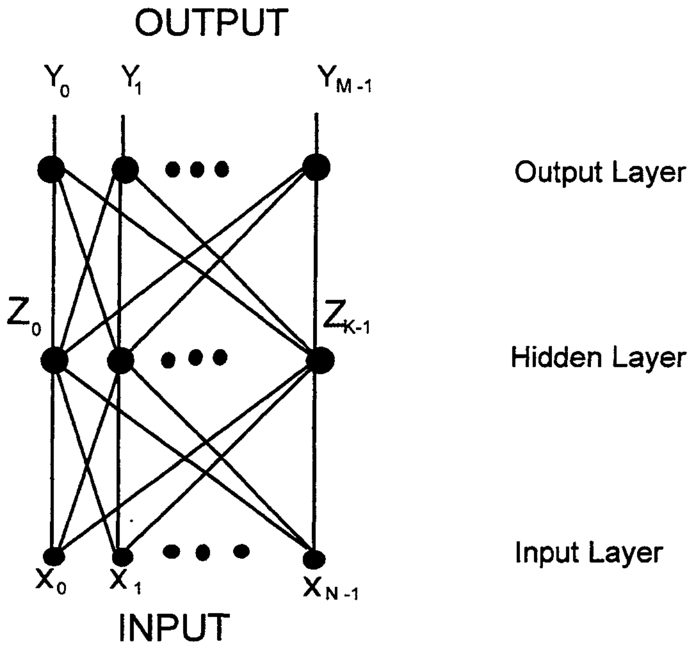


Figure 6.1 Multi-layer neural network

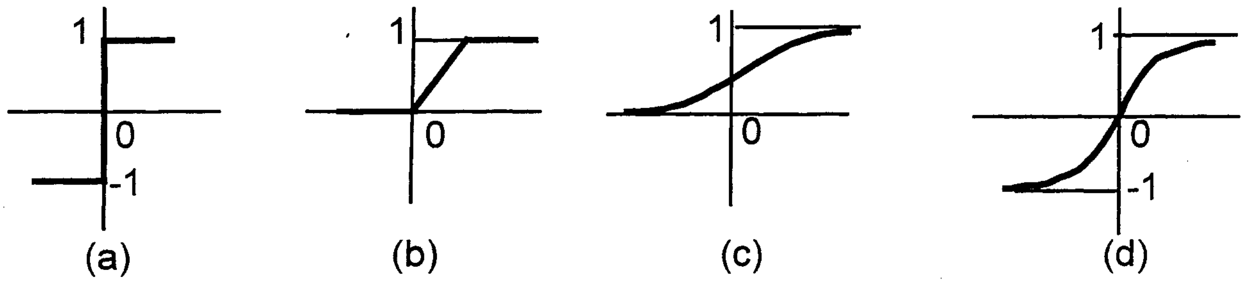


Figure 6.2 Sigmoid functions

The backpropagation training algorithm is an iterative gradient algorithm designed to minimize the mean square error between the actual and desired output of a multi-layer feed-forward perceptron. Mathematically it requires continuous differentiable nonlinearities. The following sigmoid logistic nonlinearity (shown in Figure 6.2 (c)) is most commonly used. $f(\alpha)=1/(1+e^{-(\alpha-\theta)})$ (6.3)

where θ is an internal threshold or offset.

The training algorithm is briefly described in 5 steps as below.

Step 1. Initialize Weights

Set all weights to small random values from -1 to 1.

Step 2. Present Inputs and Desired Outputs

Present an input vector (x0, x1, ..., xN-1) and specify the desired output vector (d0, d1, dM-1).

Step 3. Calculate Actual Output

Use the sigmoid nonlinearity from (6.3), and formulas (6.1) and (6.2) to calculate output vector (y0, y1, ..., yM-1) .

Step 4. Update Weights

Use a recursive algorithm starting at the output nodes and working back to the hidden layer. Update weights by

$$w_{ij}(t+1)=w_{ij}(t)+\eta\delta_j x_i' \qquad (6.4)$$

In this equation $w_{ij}(t)$ is the weight from hidden layer i or from an input to node j at time t, x_i' is either the output of node i or is an input, η is a user-defined constant, called gain, and δ_j is an error term for node j. If node j is an output node, then

$$\delta_j = \text{yj(1-yj)(dj-yj)},$$

where dj is the desired output of node j and yj is the actual output.

if node j is an internal hidden node, then

$$\delta_j = x_j'(1-x_j')\sum_k w_{jk}\,\delta_k$$

where k is over all nodes in the layers above node j.

Convergence to the desired output is sometimes faster if a momentum term $\beta(w_{ij}(t) - w_{ij}(t-1))$ is added and weight changes are smoothed by

$$w_{ij}(t) = w_{ij}(t) + \eta\,\delta_j x_{i'} + \beta(w_{ij}(t) - w_{ij}(t-1))$$

where $0 < \beta < 1$.

Step 5. Repeat by Going to Step 2.

The generally good performance found for the backpropagation algorithm is somewhat surprising considering that it is a gradient search technique that may find a local minimum in the LMS (Least Mean Square) cost function instead of the desired global minimum. To improve performance and reduce the possibility that a local minimum occurs several suggestions can be made, such as allowing extra hidden nodes, lowering the gain term η used to update weights, and making many training runs starting with different sets of random weights or with determined initial weights based on some criteria.

Simulated annealing optimization algorithm

Simulated annealing, proposed by Kirkpatrick, is a general purpose combinatorial optimization technique. This algorithm is an extension of a Monte Carlo method developed by Metropolis et al. to provide an efficient simulation of a collection of atoms in equilibrium at a give temperature. In each step of the algorithm, an atom is given a small random displacement and the resulting change, E , in the energy of the system is computed. If E > 0, the displacement is accepted, and the configuration with the displaced atom is used as the starting point of the next step. The case E>0 is treated probabilistically; the probability that the configuration is accepted is P(E) = $\exp(-\Delta E / k_B T)$. Random numbers uniformly distributed in the interval (0,1) are a convenient means of implementing the random part of the algorithm. One such number is selected and compared with P(E) . If it is less then P(E), the new configuration is retained; if not, the original configuration is used to start the next step.

Using the cost of function in place of the energy and defining the configurations by a set of parameters $\{X_i\}$, the Metropolis procedure can be used to generate a population of configurations of a given optimization problem at some effective temperature, i.e., a control parameter. The simulated annealing process consists of first "melting" the system being optimized at a high effective temperature, then slowly lowering the temperature in stages until the system "freezes" and no further changes occur. At each temperature, the simulation must proceed long enough for the system to reach a steady state. The sequence of temperatures and the number of attempted rearrangements of the $\{X_i\}$ to reach equilibrium at each temperature can be considered an annealing schedule.

A simulated annealing algorithm differs from a conventional iterative improvement algorithm in that, it not only accepts a solution with a better objective function value but also accepts a solution with worse objective function values probabilistically. When the temperature T is high, the probability of accepting a worse solution is relatively large, making it relatively easier for the simulated annealing algorithm to escape from a basin with local minimum value to another basin. It is expected that the algorithm will arrive at the basin with the global minimum value eventually in the sense of probability.

Let f(X) be the objective function and X, the parameter set, and simulated annealing algorithm can be restated:

```
Set initial temperature T (positive)  and initial feasible solution Xold ;
          Calculate  fold=f(Xold) ;
          Set Xbest=Xold, fbest=fold ;
          Repeat
              for ( i=1 to number of iteration ) {
                     Xnew=perturbation(Xold) ;   ( Heuristic )
                     fnew=f(Xnew) ;
                     r=rand() ;   o < r < 1 .
                     if ( ( fnew < fold  )  or  ( r< exp((fold -fnew)/T) ) {
```

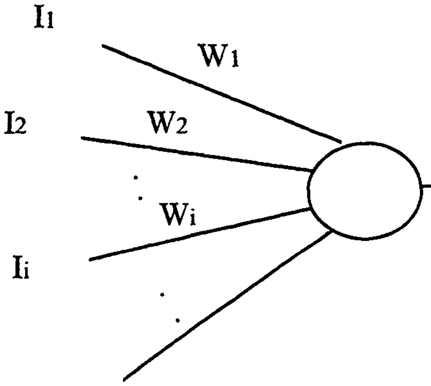
```
                    Xold = Xnew;  fold =fnew ;
                    if ( fnew < fbest) {
                        Xbest=Xnew ; fbest=fnew
                    }
                }
            }
            T = ßT ;   ß < 1.
        until (stopping criterion is met or certain number of steps)
Output Xbest and fbest as the best known solution and objective function value.
```

VI.3 Integration of optimization algorithm and development of neural network-based control strategy

The goal of an Advanced Traffic Management System (ATMS) is to manage existing transportation resources through the use of intelligent traffic control systems in order to maximize the efficiency and effectiveness of all transportation modes in an integrated framework. From a systems point of view, realizing this goal requires finding an optimal solution for a very complex control problem. Responding to this need, the features of neural network systems are appropriate for addressing the intelligent and adaptive control aspects of the ATMS. In particular, Hopfield neural network model is a proven method for solving complex but well-defined optimization problems.

VI.3.1 Hopfield optimization neural network

The neuron (Figure 6.3) is the basic element in a neural network. Neurons receive inputs (Ii)



$$O = [1 + \tanh(\sum W_i I_i / x_0)] / 2$$

Figure 6.3. Neuron

from interconnections with other neurons in the network. These interconnections are weighted (Wi) based upon their contributions to the neuron. Weighted interconnections are summed internally to the neuron, and neuron output is either a signmoidal signal (can be from 0 to 1 or from -1 to 1) or just two states (0 and 1 or -1 and 1). Highly-interconnected neural networks are shown to be extremely effective in computing. The networks can rapidly provide a solution to a problem formulated in terms of optimizing function, often subject to constraints. The Hopfield model uses a fully interconnected network of neurons (shown in Figure 6.4) to arrive at a function to be optimized, and that function is called, "energy function."

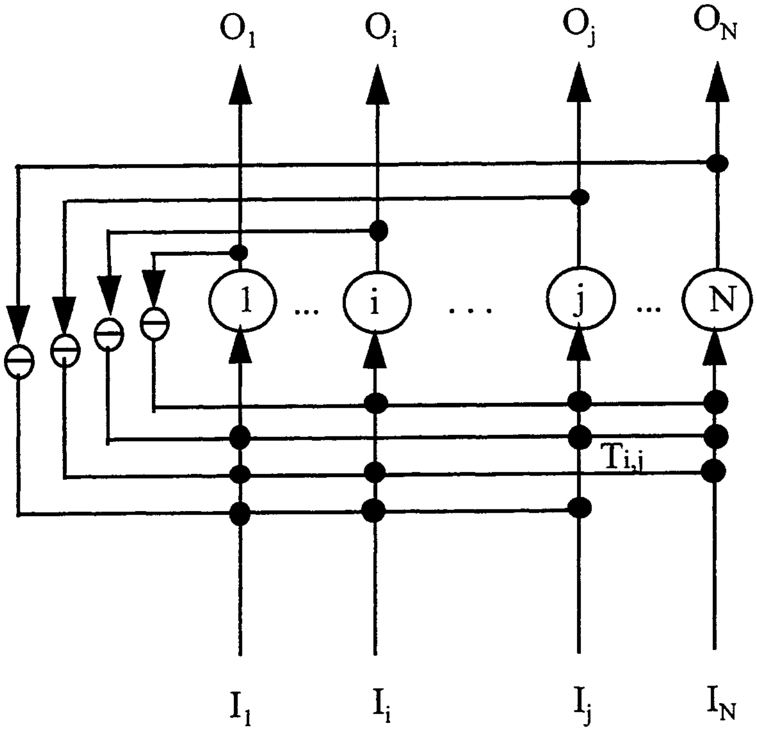


Figure 6.4 Hopfield fully connected neural network model

The dynamics of the discrete-time Hopfield network are given by

$$U_i = \sum_{j=1}^{N} T_{i,j} O_j + I_i$$

$$O_i = g(U_i)$$

where

I_i are the input biases,

$T_{i,j}$ are the interconnection weights,

O_j are the neuron outputs,

U_i are the internal states,

$$g(x) = [1 + \tanh(x / x_0)] / 2$$

as shown in Figure 6.5.

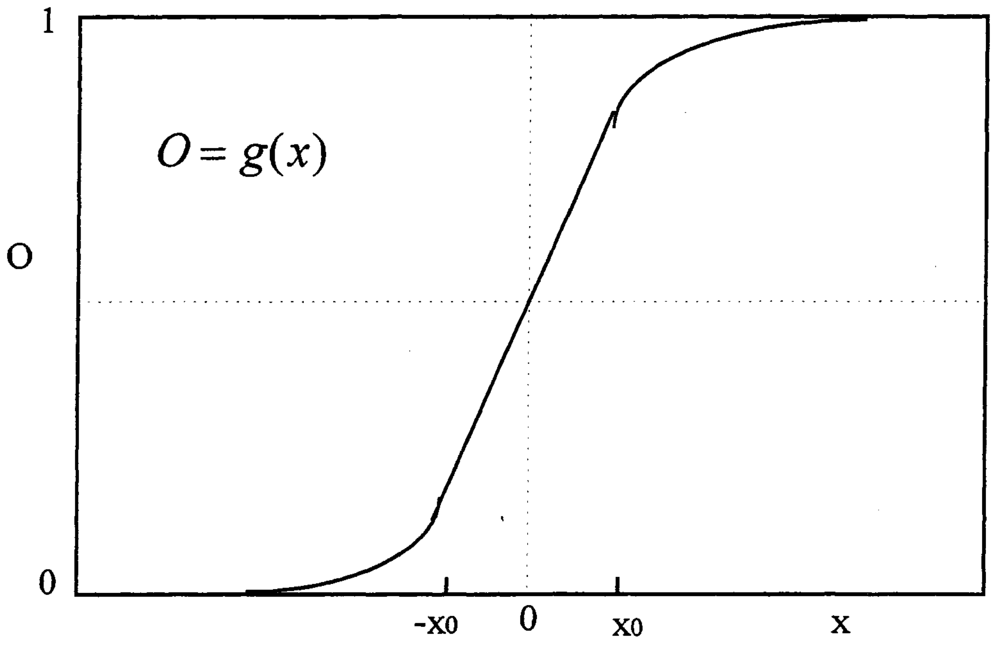


Figure 6.5. The internal-output relation for the neuron

and, i, j = 1, 2, ... N.

Hopfield and Tank show that the dynamics of this model with symmetric connections ($T_{i,j} = T_{j,i}$) always converge to a stable state, in which all neurons remain constant and energy function

$$E = -1/2\sum_{i=1}^{N}\sum_{j=1}^{N} T_{i,j}O_iO_j - \sum_{i=1}^{N} O_iI_i \qquad (6.5)$$

is minimized.

VI.3.2 Traffic control strategy

The difficulty to any application using Hopfield network is the determination of a suitable energy function that corresponds to an index to be optimized. In particular, for traffic control in an urban network, the goal is to maximize the traffic flow and minimize delays over a given time period. To achieve this goal, first, we coordinate all intersections in the network by maximizing bandwidths; second, we change signal phases of individual intersections by adapting them to traffic demand.

Assume a double-ring 8 phase controller for an intersection, as shown in Figure 6.6. Each phase has two states, Red and Green (Yellow clearance period can be considered as Green), and can be directly modeled as a neuron in a Hopfield network. As shown in Figure 6.7, neuron N_I corresponds to phase I, where I = 1, 2, ..., 8. When the neuron output is 1, the phase is Green. When the neuron output is 0, the phase is Red. Considering barrier and phase conflict, we can determine interconnections as shown in Figure 6.7. For instance, when phase 1 is Green, i.e., output O_1 is 1, phases 2, 3, 4, 7, and 8 must be Red. This constraint can be guaranteed by inputting inverted O_1 to neurons 2, 3, 4, 7, and 8 with large weights.

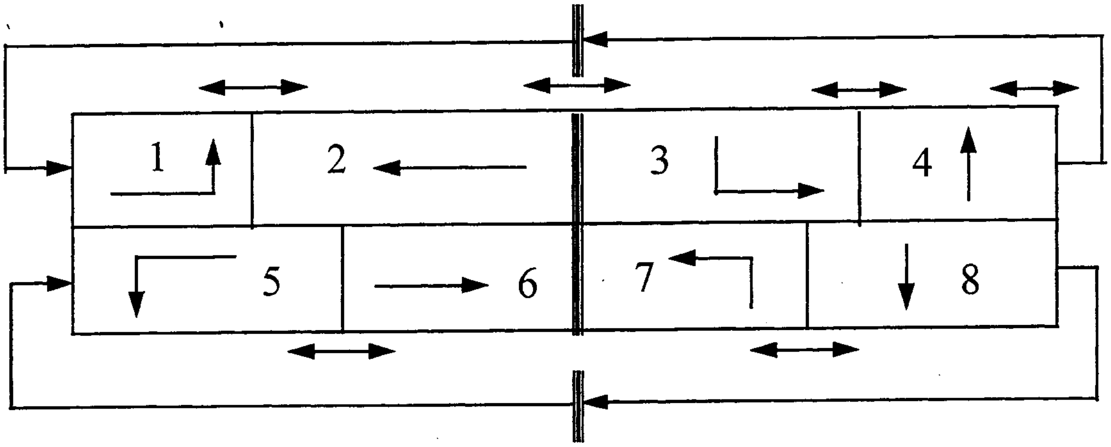


Figure 6.6. Double-ring 8 phase controller

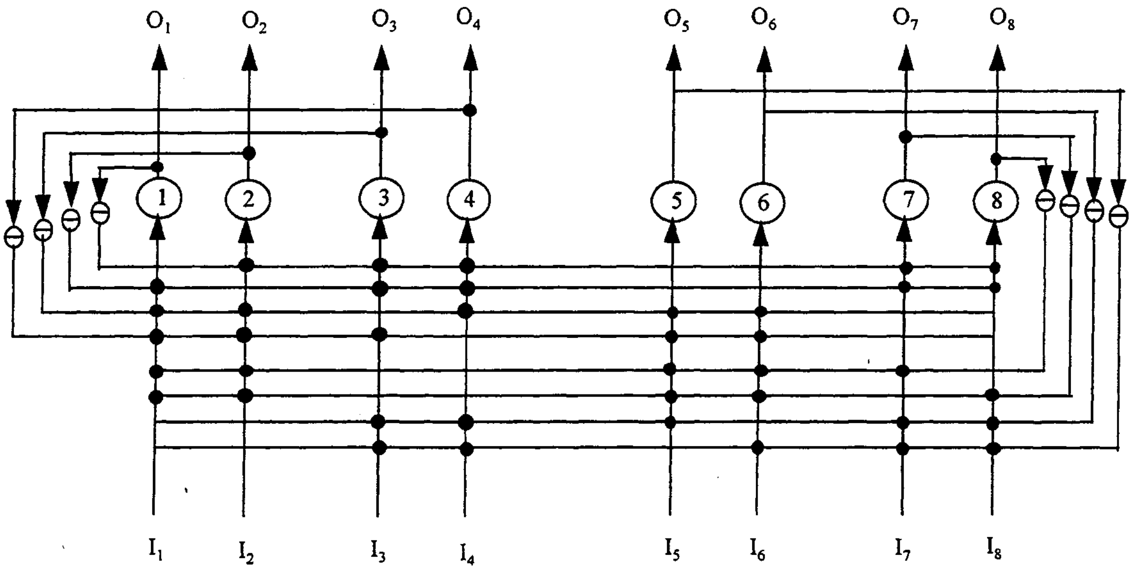


Figure 6.7. Neural network structure for double-ring 8 phase controller

Following construction of the sub-network for this individual intersection, we connect all sub-networks and determine bias Ii. From Figure 6.6, two phases are available for dispersing traffic at each approach. For example, phases 1 and 6 are for east-bound traffic (including left-turn). To maximize the bandwidth between these two phases and phases 3 and 6 at the upstream intersection of this approach, we connect the output of phases 3 and 6 at the upstream intersection to the input of phases 1 and 6 by extending the off-set, i.e., the time period vehicles travel from the upstream intersection to the down stream one at the desired speed). Similarly, we connect the output of phases 1 and 6 to the input of phases 3 and 6 at the upstream intersection. Two are the reasons for constructing this kind of connection. One is that connections must be symmetric, and the other is that connections should allow phases to affect each other. In order to respond to increased demand at an approach, we use queue length as input bias to each respective phase. In particular, we accomplish this by multiplying a factor such that we strongly favor extension of the current phase.

The energy function described above can be expressed as:

$$E = -1/2\sum_{i=1}^{N}\sum_{j=1}^{N} T_{i,j}O_iO_j - \sum_{i=1}^{N} O_i[(\alpha + \delta(i - P_c)\alpha_c)Q_i + \beta P_{i,up}] \tag{6.6}$$

where, α, α_c, and β are constant, Q_i is the queue for phase i, P_c is current phase, and $P_{i,up}$ is upstream phase status (1 or 0).

Minimizing this index implies potential maximization of bandwidths or throughput (or minimization of delay).

For a given intersection, we should deal with certain constraints before using the neural network to make a control decision. These constraints are:

- A phase cannot change to the next one within the minimum green, yellow and red clearance periods.
- When the phase reaches maximum green, it should change to the next phase automatically.
- Certain phases in two rings (e.g., 2 and 6) have to change to their next phases simultaneously to cross the barrier.
- If demand in left-turn bay is low, the exclusive left-turn phase should be skipped.

Further, special consideration should be given to exclusive left-turn phase. Before changing to this phase, intuitively the queue length input for the phase should be the queue length of the approach, rather than queue length in left-turn bay. Our simulation also shows that exclusive left-turn phase should end based on occupancy of the upstream surveillance detector in the bay. For instance, when occupancy is less than a threshold (e.g., 0.6), the phase should end.

When varying a phase between minimum and maximum green is feasible, the system calls the neural network method every second to determine whether this phase should end To be sure, following the end of a phase, the system moves to its fixed yellow period and does not immediately change to the next phase. The input (or bias) to each neuron is the weighted summation of its corresponding queue length, the delayed output of the two neurons of the corresponding two phases at the upstream intersection, and the delayed output of the two neurons of the corresponding two phases at the downstream intersection.

VI.3.3 Implementation in C++

We have implemented a general Hopfield neural network using C++. The header and source files are given in Appendix A. A **neuron** class is declared for a neuron, and a **network** class for the network. The network is declared a friend in the neuron class. The program follows the procedure described for setting inputs, interconnection weights, and updating.

VI.4 Testing neural network-based optimized control

The Hopfield neural-network strategy developed in the previous section is designed for urban street networks. However, in this project we have focused on a live laboratory at an isolated intersection. For the new control strategy, isolated intersection control is a special case within urban network control.

VI.4.1 NETSIM modification and interface development

NETSIM is used to simulate the traffic with both the new control and existing actuated control strategies. As described above, we implemented the new strategy in C++. However, since NETSIM was written in FORTRAN, an interface has been developed to exchange necessary data between NETSIM and the new control module. For simulating the new strategy the NETSIM input file can be the same as the one used for NETSIM to simulate any other strategies. Therefore, users do not need to prepare any data files specifically for the new control strategy.

For NETSIM to simulate the isolated Franklin-Lyndale intersection, we introduce a "dummy intersection" at each approach. Generally the signals for a dummy intersection are always green for both entering and exiting traffic. However, we have modified these signal plans so that the dummy intersections generate platoon demand patterns. Two are the reasons for this modification; first, platoon patterns are closer to the real traffic patterns, and second, the change allows the user to take advantage of the coordination feature of the new strategy.

VI.4.2 Simulation of new strategy with NETSIM

For the geometry and traffic demand data collected at the Franklin-Lyndale intersection with the machine vision system under current actuated control, the new strategy has been simulated with NETSIM.

The geometry is given in Figure 6.8, including surveillance detector layout. All detectors are presence type. For the control phases shown in Figure 6.9, the corresponding neural network structure at the intersection is shown in Figure 6.10 .

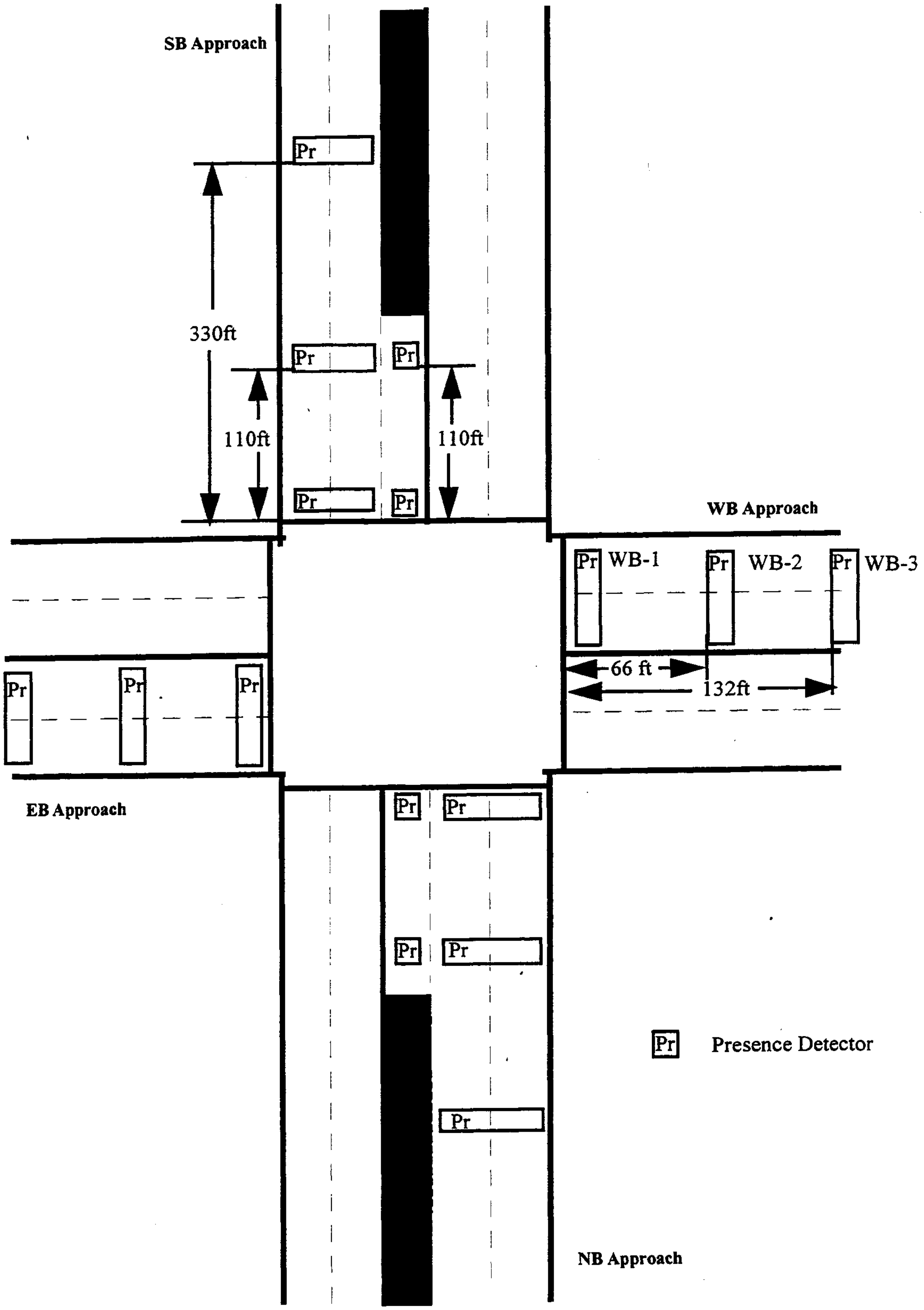


Figure 6.8. Geometry and surveillance detector layout (not on scale)

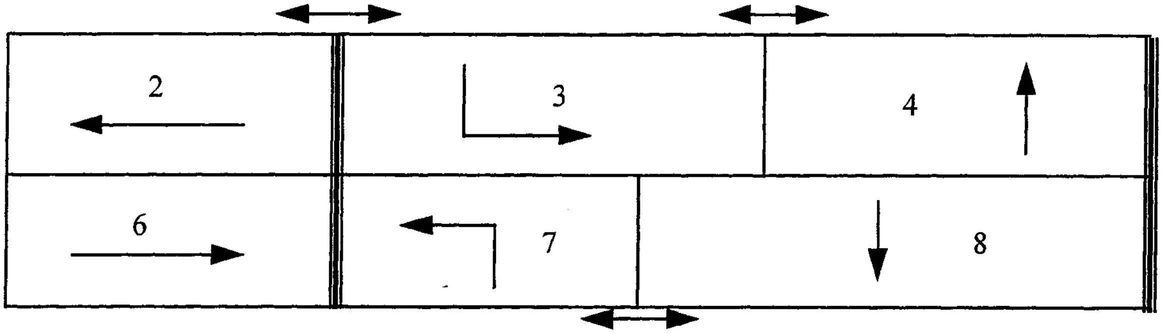


Figure 6.9. Current control phases for Franklin-Lyndale intersection

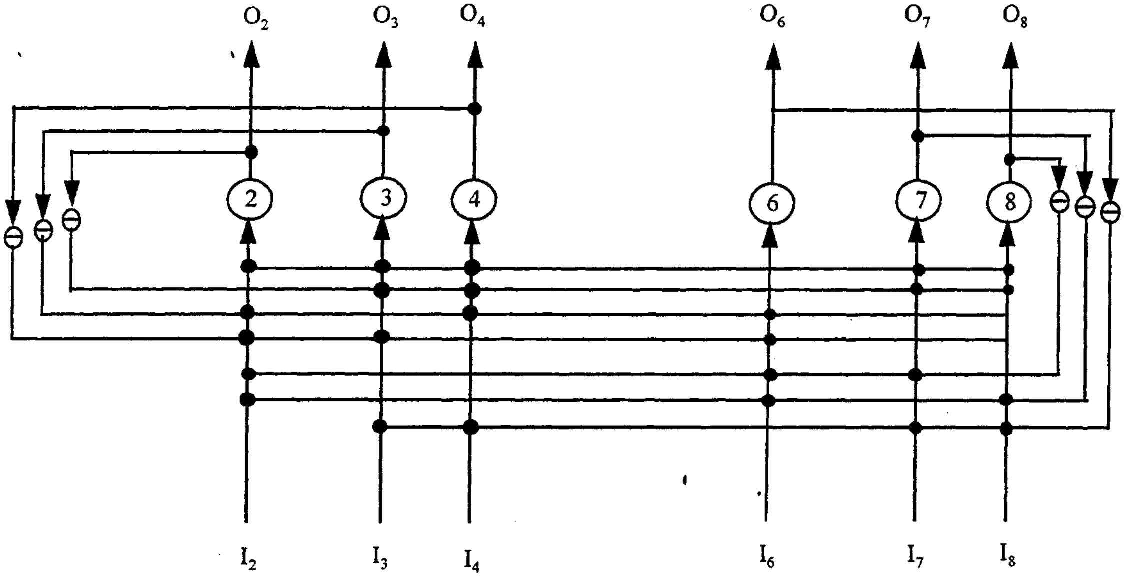


Figure 6.10. Neural network structure for Franklin-Lyndale intersection

Three cases, AM-Peak, PM-Peak, and Off-Peak, have been simulated. The simulation results, and comparison with simulation results from the current actuated control plan are presented in the next section.

VI.4.3 Comparison with simulation results from current actuated control

The current control plan has been developed by traffic engineers at the City of Minneapolis. It is a fully stop-line vehicle actuated control policy applied to all traffic conditions, i.e., AM-Peak, PM-Peak, and Off-Peak. The demands for all 3 cases and turning percentages for each of 3 cases are shown in Figures 3.10, 3.12, 3.13, and 3.14 in chapter 3. Main parameters of this control plan are given below. All units, except Vehicle Recall are in seconds. Yellow and red clearance times are slightly different from real data because the new control strategy cannot handle fractions of a second.

Table 6.1. Main parameters for current control plan *

Phase	Minimum Green	Maximum Green	Yellow Clearance	Red Clearance	Passage Time	Vehicle Recall
2	10	31	3	2	3	
3	6	25	3	2	3	
4	10	44	3	2	3	Minimum
6	10	31	3	2	3	
7	6	8	3	2	3	
8	10	61	3	2	3	Minimum

* All units, except Vehicle Recall are in seconds.

The neural network strategy uses the original parameters, i.e., those of table 6.1. Surveillance detector layout has been given in Figure 6.8. Because no method can reliably calculate or detect the exact value of queue length, here we use discrete queue length. For example, referring to Figure 6.8, if detector WB-3, which is 132 ft from the stop line, is occupied in the previous second, we consider the queue length in this approach is 6 vehicles. Otherwise, if detector WB-2, which is 66 ft from stop line, is occupied, the queue length is 3 vehicles (here we assume vehicle length is 22 ft).

Values of simulated MOEs from the application of the new neural network control strategy at the Franklin-Lyndale intersection are summarized below:

Table 6.2. AM-Peak MOEs

	Ave. Delay (Sec./veh.)	Stops (%)	Ave. Speed (MPH.)
Current	29.6	64.0	11.0
Neural Network	22.8	55.7	13.1
Improvement (%)	23.0	13.1	19.0

Table 6.3. PM-Peak MOEs

	Ave. Delay (Sec./veh.)	Stops (%)	Ave. Speed (MPH.)
Current	31.9	67.1	10.5
Neural Network	26.4	63.5	11.9
Improvement (%)	17.1	5.4	13.5

Table 6.4. Off-Peak MOEs

	Ave. Delay (Sec./veh.)	Stops (%)	Ave. Speed (MPH.)
Current	24.0	62.3	12.6
Neural Network	21.6	52.3	13.5
Improvement (%)	10.0	16.1	7.4

VI.4.4 Analysis of results

Analysis of the detailed output data from NETSIM indicates that the current control plan is not well balanced for all approaches. Specifically, all three MOEs in the Southbound approach are substantially better than those in other approaches. In contrast, the neural network strategy results in well balanced MOEs for all approaches. More-detailed analysis should be based on additional NETSIM output, such as cycle length change, phase skipping, and queue length in each lane. To accomplish this, the NETSIM source code needs to be further modified.

The neural network control strategy produces MOE values that are improved over those of the current actuated control plan, for all cases. From tables 6.2-4, average delay per vehicle improves 10.0% - 23.0%; stop percentage, 5.4% - 16.1%; and average speed improves 7.4% - 19.0% over current values. Comparing these three cases, it can be noticed that for both average delay and average speed AM-peak period is most improved, and Off-peak period is least improved. However, for stop percentage Off-peak is most improved and PM-peak is least. Refering to Figure 3.10, total demands in AM-peak and Off-peak are almost the same. For these two periods, average delay and average speed produced by the current control plan are significantly different, 23% and 15%, respectively. But the differences for average delay and average speed produced by the new strategy are just 5.5% and 3%, respectively.

Based on these preliminary testing and analysis, it seems to be that the new strategy is more adaptive to traffic than the current control plan. We believe that the current control plan can be further refined or a more sophisticated actuated control plan may be developed. Similarly, the new strategy is also preliminary and further improvement is expected.

VI.5 Summary

We have proposed an urban-network traffic-adaptive control strategy, based on the Hopfield optimization neural network. In this scheme, each neuron represents a control phase. All neurons are connected in a way that guarantees no conflict phases exist. The input bias of each neuron is a weighted summation of its corresponding queue length and the delayed output of the two neurons of the corresponding two phases at the upstream intersection. The objectives of this connection scheme are to coordinate phases between intersection, and use queue length to control phase change adaptively to traffic demand in each approach.

Compared with the current stop-line actuated control, the new strategy results in improved MOEs for all three test cases. The strategy uses existing NETSIM input files so that users do not need to prepare any new file specifically for this strategy.

Because of the scope of this project we only tested the new strategy in an isolated intersection. Simulation results were only compared with those from the current stop-line actuated control plan. The new strategy is still preliminary and needs to be refined and further tested in an urban network

VII. CONCLUSIONS AND FUTURE RESEARCH NEEDS

This report summarized the final results of the project, "Development and application of on-line strategies for optimal intersection control, Phase III". First, state-of-the art network control strategies were briefly reviewed. This review focused on CARS, a new traffic-adaptive network control approach, that uses a mesoscopic traffic model, PAKSIM, for on-line simulation. The PAKSIM model combines realistic representation of traffic behavior with computational efficiency and, with frequent real time update, can be a useful tool for on-line optimal control.

Second, the intersection laboratory was enhanced with installation of two additional cameras to cover the intersection proper and the East bound Lyndale approach. Further, a comprehensive operational plan for the intersection laboratory was developed including a procedure to estimate intersection performance using the real data collected from the machine-vision detection system. An example application of the procedure was also conducted to compare the performance of pretimed and actuated control strategies at the laboratory. An additional important product of this research is a new microscopic simulator developed for the laboratory environment. The new simulator uses a cellular automata approach with a simplified car-following model also developed in this research. The new simulator was tested with the data collected from the intersection laboratory and used to evaluate the new control strategy developed in this research. The new strategy is based on the newly defined congestion index, which quantifies the level of link-wide congestion at each approach using point measurements. The new control strategy was compared with those of pretimed and stopline-actuation control and test results indicated the potential of the new strategy for improved performance over the existing. Finally, a neural network approach integrating the Hopfield optimization scheme was applied to develop an adaptive control strategy for managing urban networks. The preliminary test results with the NETSIM simulator showed improved performance compared with stopline-actuation control.

Future research needs include the establishment of communications between the field laboratory and the ITS laboratory, University of Minnesota, for automatic data collection. The new microscopic models need to be refined with more comprehensive real data from the laboratory intersection. Finally, the hardware/software environment of the intersection laboratory needs to be expanded, so that new control strategies can be tested and evaluated with real traffic conditions.

REFERENCES

[1] TRIS Literature Review

[2] Bretherton, R.D. and Bowen, G.T. "Recent Enchancements to SCOOT-SCOOT Version 2.4", IEE: Third Interantional Conference on Road Traffic Control, 1-3 May 1990, pp 95-98.

[3] Houncel, N.B., McLeod, F.N. and Burton, P. "SCOOT: A Traffic Database", IEE: Third International Conference on Road Traffic Control, 1-3 May 1990, pp. 99-103.

[4] Robertson, D.I. and Bretherton, R.D. "Optimizing Networks of Traffic Signals in Real Time- The SCOOT Method", IEEE Transactions on Vehicular Technology, Vol. 40, No 1, February 1991, pp. 11-15.

[5] Hunt, P.B., Robertson, D.I., Bretherton, R.D. and Wintoh, R.I. "SCOOT- A Traffic Responsive Method of Coordinating Signals", TRLL Laboratory Report 1014, 1981.

[6] Bretherton, R.D., Rai, G.I, "The use of SCOOT in Low Flow Conditions", Traffic Engineering and Control, Vol. 23, No. 12, December 1982, pp. 574-576.

[7] Traffic Engineering and Control, December 1989, p. 601.

[8] Sims, A.G. "SCAT- The Sydney Co-ordinated Adaptive Traffic System, Symposium on Computer Control of Transport, Sydney, Australia, Feb. 1981, pp.22-26.

[9] J.E Longfoot, "SCATS-Development of Management and Operation Systems", International Conference on Road Traffic Signalling, London, March 30- April 1, 1982.

[10] P.R. Lowrie, "SCATS- Sydney Coordinated Adaptive Traffic System: A Traffic Responsive system for Controlloling Urban Traffic", Road and Traffic Authority of N.S.W Sydney, Australia, 1990.

[11] Sims A.G., Finlay, A.B., "SCATS. Splits and Offsets Simplified (S.O.S)", Proceedings from the 12th ARRB Conference, August 1984.

[12] Bretherton R.D., Robertson, D.I. "Optimum Control of an Intersection for any Known Sequence of Vehicle Arrivals", Proceedings to the 2nd IFAC/IFIP/IFORS Symposium on Traffic Control and Transportation Systems, Monte Carlo, 1974.

[13] Barierre J.L., Henry J.J. "Decentralization vs Hierarchy in Optimal Traffic Control", 5th IFAC/IFIP/IFORS Conference on Control in Transportation Systems, Vienna, 1986.

[14] Gartner N.H., "OPAC: Strategy for Demand -Responsive Decentralized Traffic Signal Control", IFAC Symposium on Control, Computers, Communications in Transportation, Paris, France, Sept. 1989 pp.241-244.

[15] Barcelo' J., Grau Rafel "CARS: An Experience in Demand-Responsive Traffic Control", 1992.

[16] Barcelo' J., Grau Rafel "A review of Objective Functions in the CARS Demand-Responsive Control System", 1993.

[17] Staunton. M. M., "Vehicle Actuated Signal Controls for Isolated Locations",1976.

[18] National Instruments, LabView User Manual for Version 4.0, January 1996.

APPENDIX A

C++ Source Code for
The Hopfield Neural Network Implementation

APPENDIX A.
C++ Source Code for The Hopfield Neural Network Implementation

```
//inter_nn.h

#include <iostream.h>
#include <stdlib.h>
#include <math.h>
#include <stdio.h>

#define MXSIZ 9


class neuron
    {
    protected:
    int phase ;
    float output;
    float activation;
    friend class network;

    public:
    neuron() { };
    void getnrn(int);
    };

class network
    {
    public:
    float pra;

    neuron (phase)[MXSIZ];

    float Qu[MXSIZ];
    float outs[MXSIZ];
    float acts[MXSIZ];
    float mtrx[MXSIZ][MXSIZ];

    network() { };
    void getnwk(float);
    void asgninpt(int,int,int,int,int,int,int,int,int,int);
    void iterate(int,float,float,float);
    void getacts(float,float);
    void getouts(float);

//print functions

    void prmtrx();
    void practs();
    void prouts();
    };
```

```
//inter_nn.cpp

#include "inter_nn.h"
#include <stdlib.h>
#include <time.h>


//generate random noise


int randomnum(int maxval)
{
// random number generator
// will return an integer up to maxval

return rand() % maxval;
}

void neuron::getnrn(int i)
   {
     phase = i ; //cit = i;

     output = 0.0;
     activation = 0.0;
   };


//set up network

void network::getnwk(float a)
   {
     int i,j;

//        pra = a;

     for(i=1;i<9;++i)
     {
       phase[i].getnrn(i) ;
     }

     //find weight matrix

     for(i=1; i<9;++i)
     {
       for(j=1; j<9; j++)
       {
         mtrx[i][j] = 0. ;
       }
     }

     mtrx[2][3] = -a ;
     mtrx[2][4] = -a ;
     mtrx[2][7] = -a ;
     mtrx[2][8] = -a ;
```

```
        mtrx[3][2] = -a ;
        mtrx[3][4] = -a ;
        mtrx[3][6] = -a ;

        mtrx[4][2] = -a ;
        mtrx[4][3] = -a ;
        mtrx[4][6] = -a ;

        mtrx[6][3] = -a ;
        mtrx[6][4] = -a ;
        mtrx[6][7] = -a ;
        mtrx[6][8] = -a ;

        mtrx[7][2] = -a ;
        mtrx[7][6] = -a ;
        mtrx[7][8] = -a ;

        mtrx[8][2] = -a ;
        mtrx[8][6] = -a ;
        mtrx[8][7] = -a ;


        prmtrx();
    }



//print weight matrix

void network::prmtrx()
    {

        cout<<"\n" ;
        cout<<"\t\t2\t3\t4\t6\t7\t8\n\n" ;
        cout<<"2\t"<<"\t"<<mtrx[2][2]<<"\t"<<mtrx[2][3]<<"\t"<<mtrx[2][4]
                <<"\t"<<mtrx[2][6]<<"\t"<<mtrx[2][7]<<"\t"<<mtrx[2][8] ;
        cout<<"\n";
        cout<<"3\t"<<"\t"<<mtrx[3][2]<<"\t"<<mtrx[3][3]<<"\t"<<mtrx[3][4]
                <<"\t"<<mtrx[3][6]<<"\t"<<mtrx[3][7]<<"\t"<<mtrx[3][8] ;
        cout<<"\n";
        cout<<"4\t"<<"\t"<<mtrx[4][2]<<"\t"<<mtrx[4][3]<<"\t"<<mtrx[4][4]
                <<"\t"<<mtrx[4][6]<<"\t"<<mtrx[4][7]<<"\t"<<mtrx[4][8] ;
        cout<<"\n";
        cout<<"6\t"<<"\t"<<mtrx[6][2]<<"\t"<<mtrx[6][3]<<"\t"<<mtrx[6][4]
                <<"\t"<<mtrx[6][6]<<"\t"<<mtrx[6][7]<<"\t"<<mtrx[6][8] ;
        cout<<"\n";
        cout<<"7\t"<<"\t"<<mtrx[7][2]<<"\t"<<mtrx[7][3]<<"\t"<<mtrx[7][4]
                <<"\t"<<mtrx[7][6]<<"\t"<<mtrx[7][7]<<"\t"<<mtrx[7][8] ;
        cout<<"\n";
        cout<<"8\t"<<"\t"<<mtrx[8][2]<<"\t"<<mtrx[8][3]<<"\t"<<mtrx[8][4]
                <<"\t"<<mtrx[8][6]<<"\t"<<mtrx[8][7]<<"\t"<<mtrx[8][8] ;
        cout<<"\n";
```

```
   }


//present input to network, should get from detectors

void network::asgninpt(int CurPhase1, int Change1, int NextPhase1, int CurQ1, int NextQ1,
                int CurPhase2, int Change2, int NextPhase2, int CurQ2, int NextQ2)
   {
      int i;

      for(i=1;i<9;i++)
      {
         acts[i] = 0. ; // put all neurons in sleep
         Qu[i] = -100. ;
      }

      //find initial activations

      if(Change1 == 1)
      {
         Qu[CurPhase1] = 1.1*CurQ1 ;
         Qu[NextPhase1] = NextQ1 ;
         acts[CurPhase1] = 1.1*CurQ1 ;
         acts[NextPhase1] = NextQ1 ;
      }

      if(Change2 == 1)
      {
         Qu[CurPhase2] = 1.1*CurQ2 ;
         Qu[NextPhase2] = NextQ2 ;
         acts[CurPhase2] = 1.1*CurQ2 ;
         acts[NextPhase2] = NextQ2 ;
      }


      //print activations

      practs();
   }


//find activations

void network::getacts(float dlt,float tau)
   {
      int i, j;
      float r;

      for(i=1;i<9; i++)
      {
         r = -acts[i]/tau ;
         for(j=1;j<9;j++)
         {
            r += mtrx[i][j]*outs[j] ;
         }
         r += Qu[i] ;
         r *= dlt ;
         acts[i] += r ;
       }

    }


 //find outputs

 void network::getouts(float la)
    {
       float b1,b2,b3,b4;
       int i;

       for(i=1;i<9;++i)
       {
         b1 = la*acts[i];
         b4 = b1/10.0;
         b2 = exp(b4);
         b3 = exp(-b4);
         outs[i] = (1.0+(b2-b3)/(b2+b3))/2.0;
       }
    }


// print outputs

void network::prouts()
    {
      int i;
      cout<<"\nthe outputs\n";
      for(i=1;i<9;++i)
      {
          cout<<outs[i]<<"  ";
          cout<<"\n";
      }
    }

//print activations

void network::practs()
    {
       int i;
       cout<<"\n the activations:\n";
       for(i=1;i<9;++i)
       {
          cout<<acts[i]<<"  ";
          cout<<"\n";
       }
    }

//iterate specified number of times
```

```
void network::iterate(int nit,float dlt,float tau,float la)
    {
    int k;

    for(k=1;k<=nit;++k)
        {
        getacts(dlt,tau);
        getouts(la);
        }
    cout<<"\n";
    practs();
    cout<<"\n";
    prouts();
    cout<<"\n";
    }



void main()
    {

//numit = #iterations;
//dt = delta t;
// ------------------------------------
// parameters to be tuned are here:

    float a=100. ;
    float dt=0.01;
    float tau=1.0;
    float lambda=3.0;
//--------------------------------------
    int numit=300;

//create network and operate

    network *tntwk = new network;
    if (tntwk==0)
        {
        cout << "not enough memory\n";
        exit(1);
        }
    tntwk->getnwk(a);
    tntwk->asgninpt(2,1,3,5,10,6,1,7,10,5);
    tntwk->getouts(lambda);
    tntwk->prouts();
    tntwk->iterate(numit,dt,tau,lambda);
    cout<<"\n";
}
```