Fifth IFSA World Congress (1993), 907 - 910

Traffic Fuzzy Control: Software and Hardware Implementations

M. Jamshidi[1], R. Kelsey[2], and K. Bisset[3]

[1] *CAD Laboratory for Intelligent & Robotic Systems, Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM, USA*
[2] *Department of Computer Science, New Mexico State University, Las Cruces, NM, USA*
[3] *Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, USA*

ABSTRACT

This paper describes the use of fuzzy control and decision making to simulate the control of traffic flow at an intersection. to show the value of fuzzy logic as an alternative method for control of traffic environments. A traffic environment includes the lanes to and from an intersection, the intersection, vehicle traffic, and signal lights in the intersection. To test the fuzzy logic controller, a computer simulation was constructed to model a traffic environment. A typical cross intersection was chosen for the traffic environment, and the performance of the fuzzy logic controller was compared with the performance of two different types of conventional control. In the hardware verifications, fuzzy logic was used to control acceleration of a model train on a circular path. For the software experiment, the fuzzy logic controller proved better than conventional control methods, especially in the case of highly uneven traffic flow between different directions. On the hardware side of the research, the fuzzy acceleration control system showed a marked improvement in smoothness of ride over crisp control.

1. INTRODUCTION

There are two types of conventional controllers for traffic flow systems: one uses a preset cycle time to change lights; the other combines preset cycle times with proximity sensors which can activate a change in the cycle time or the lights. In the case of a less-traveled street which may not need a regular cycle of green lights, a proximity sensor will activate a change in the light when cars are present. This type of control depends on having some prior knowledge of traffic flow patterns in the intersection so that signal cycle times and placement of proximity sensors may be customized for the intersection.

Fuzzy logic control, an alternative to conventional control, can control a wider array of traffic patterns at an intersection. A fuzzy-controlled signal light uses sensors that count cars, instead of proximity sensors which only indicate the presence of cars. This provides the controller with traffic densities in the lanes and allows better assessment of changing traffic patterns. As traffic distributions fluctuate, a fuzzy controller can change the signal light accordingly.

2. SOFTWARE IMPLEMENTATIONS

2.1 Simulation

The computer simulation was written in the C++ computer programming language using object-oriented programming techniques. Object-oriented programming allows the simulation to be built in a modular fashion, making it both expandable and easily maintainable. Thus new features are easily added to the simulation or current features easily modified with little impact on the rest of the program.

The graphics portion of the simulation was written using the X Window System as described by Nye [1,2], specifically the Xt Intrinsics library and the Athena (Xaw) and Hewlett-Packard (Xw) widget libraries, collectively called the X Toolkit. The use of the X Toolkit provides a set of widgets (graphical objects such as buttons or sliders) which can be combined to create the simulation. Their use allows the low-level graphical details to be ignored.

The X Toolkit (like X Windows in general) is event-driven. The simulation responds to events, such as a button press, generated by the user. Any one of a number of actions can be performed at any time to greatly enhance the interactive nature of the simulation.

The crux of the simulation problem is the modeling of traffic flow to mimic reality, specifically the motion of the vehicles relative to one another. Physical equations were used to describe the motion of a car based on the car immediately in front of it (the leading car). Equation (1) describes the acceleration of a car based on the velocities and positions of the car itself and the leading car. This is a time-delay differential equation which comes from traffic flow theory presented by Haight [3], Montrol [4], Morris, et al. [5], and Malek-Zavarei and Jamshidi [6]. Equations (2) and (3) are standard classical physics equations for the velocity and position of an object based on the object's acceleration. v(t) and x(t) are the velocity and position, respectively, of the car at time τ. v'(t) and x'(t) are the velocity and position of the leading car. τ is the driver reaction time and v is the free velocity of the car (the velocity at which the car would drive without interference from other cars).

$$a(t + t\ ) = 4\ v(t)[v(t) - v'(t)] / [(x(t) - x'(t))^2] + 0.2(v\ free - v(t)) \quad (1)$$

$$v(t) = v(t-1) + a(t)dt \quad (2)$$

$$x(t) = x(t - 1) + v(t)dt + (1/2)\ a(t)dt^2 \quad (3)$$

The free velocity of each car is randomly set from a normal distribution with a mean of the "posted" speed limit. As observed in reality, this provides the simulation with cars moving at the speed limit and cars moving both faster and slower than the speed limit. The driver reaction time, τ, is the time expended when a driver sees a disturbance, the brain processes the information, and the driver takes appropriate action. This time is equal to 1.5 seconds.

The simulation has a variety of modes and other configuration inputs which may be selected from an input file when the program is executed. These inputs include the type of controller (timed, proximity, or fuzzy) which may be used to control the traffic signal; whether to use graphics or batch mode; flow rates; percentage of turning cars; and how long (in simulation time) the simulation should run.

Flow rate in cars per hour may be specified separately for each of the four lanes. This parameter may be either a single value (cars per hour) or a series of flow-time pairs (two numbers: flow and time). If a single value is given, that flow rate is then used for the entire simulation run. The cars are actually generated and entered into the simulation randomly to obtain the specified flow rate, and to make the flow pattern or distribution of cars less uniform. The flow-time pairs give the flow rate at certain times during the simulation run. The simulation interpolates between these pairs to provide a smooth transition between flows. Flow-time pair inputs are useful for determining how the controller will perform in a given intersection over the course of an entire day or other time period.

The simulation has two operating modes: a graphics mode displays a scaled traffic environment, and a batch mode executes without graphics. The graphics mode animates the movement of cars in lanes directed to and from the intersection. and displays statistics about cars which have moved through the traffic environment. These statistics include flow rate (cars per hour), average waiting time, and average driving time. The average waiting time (in seconds) is an average of the time all cars spend at zero velocity in the simulation. The average driving time is the average of the time all cars completing the simulation spend driving the length of the simulation. The fuzzy controller version of the graphics simulation also has "pop up" windows available that display: the degree of membership for each of the inputs, which fuzzy rules are being fired, and a list of the fuzzy rules. This information gives insight into the workings of the fuzzy controller.

The simulation's graphics mode allows interactive change of certain inputs originally set in the input file, including the flow rate in each direction and the cycle time length of the lights. Due to the computational overhead of generating graphics, the batch mode can execute as much as ten times faster than the graphics mode. This makes the batch mode ideal for running exhaustive tests using

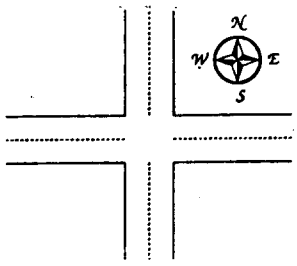

Figure 1. Traffic Cross Intersection.

different traffic patterns for the three different controllers.

The statistics generated by the simulation are used to calculate a cost function (equation 4) as a means of evaluating fuzzy controller performance against that of the conventional controllers. $Wait_{mean}$ is the average waiting time, $Drive_{mean}$ is the average drive time, and $k(Cars_{out}/Cars_{in})$ is the number of cars exiting and entering the traffic environment. Cost is a constant which is equal to 100.

$$Cost = Wait_{mean} / k(Cars_{out} / Cars_{in}) Drive_{mean} \quad (4)$$

2.2 Fuzzy Controller

A fuzzy logic controller was designed for a typical cross intersection traffic environment with a lane directed to and from the intersection at each compass point, as shown in Figure 1.

The fuzzy logic controller was implemented using the Togai InfraLogic Fuzzy C Compiler [7], which accepts a fuzzy source code file as input and compiles this file into a module of C source code. The fuzzy source code, much like natural language, describes each input and output to the fuzzy controller and the fuzzy membership functions associated with those inputs and outputs. Also described with natural language is the fuzzy rule base which maps the combinations of inputs to the outputs. The C source code generated by the Fuzzy C Compiler can then be compiled (by a C compiler) within a C program or as a separate routine to be called by other programs.

There are three inputs into the fuzzy controller, the average density of traffic behind the green lights, the average density of traffic behind the red lights, and the length of the current cycle time. The four membership functions that describe the densities of traffic at the green and red lights are labeled Zero, Low, Medium, and High; a sample of these functions appears in Figure 2. The membership functions for the green and red densities were chosen to be different because the densities of the cars in those two cases will be different: cars stopped at a red light are much closer together than cars moving through a green light, therefore the density of the cars at the red light will be higher than that of the cars moving through the green light. (This detail was actually discovered while observing the graphic display of the simulation. The first version of the fuzzy logic controller used the same membership functions for green and red densities.) The third input, current cycle time, has three membership functions describing time in seconds: Short, Medium, and Long; their membership functions are similar to those in Figure 2.

The densities are taken from two sensors placed on the road, one at the intersection and the other 150 feet before the light. The rear sensor increments a counter every time a car passes over it while the forward sensor decrements the same counter. In this way a count is obtained of the number of cars in the 150 feet before the light. This contrasts with conventional control, which places a proximity sensor at the light and can sense only the presence of cars waiting at a light, not how many cars are waiting.

The output of the fuzzy controller decides whether to change the state of the light (for example, green north-south to green east-west) or keep it the same. The membership functions for change are No, Probably No, Maybe, Probably Yes, and Yes. These functions represent a degree (fuzzy value) of a binary value, 1 being Yes and 0 being No, and were chosen similar to Figure 2.

Other attempts at fuzzy control of traffic signals, particularly by Gallegos and Nguyen [8], have used the fuzzy controller to make incremental changes in the times of the cycle in each direction. We rejected that approach because a controller which could make immediate changes to the light based on the current conditions was preferable.

The fuzzy rule base maps the combination of the inputs to the output to decide whether to change the light. In the fuzzy source code

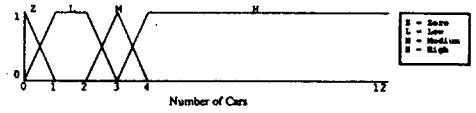

Figure 2. Membership Functions: Traffic Densities at Green Lights.

file the fuzzy rule base is a list of *if/then* statements. The number of rules is equal to the number of input combinations. For example, if there are two inputs with three membership functions each, then the number of rules will be nine.

It is possible to scale down the number of fuzzy rules when certain input combinations are unnecessary. The fuzzy traffic controller design presented would yield a rule base of 48 rules, but for certain combinations the cycle time input has no meaning. For example, when the green traffic density is zero and the red traffic density is zero then the input for time does not matter. The desired output (change) is the same for any amount of time. Thus one rule takes the place of three.

2.3 Testing and Results

Testing of the fuzzy controller involved the simulation of 625 different parameter sets. Figure 3 shows the traffic flow for a 12-hour period of the simulation. These parameter sets vary the flow of traffic in each of four lanes from 0 to 1200 cars per hour, with an increment of 300 (hence, 625 combinations). Each of these parameter sets is simulated for one hour of simulation time; the generated statistics include the average flow rate (cars/hour), average driving time, and average waiting time. Results of the fuzzy controller simulations, in the form of the statistics, are compared with the same 625 parameter sets using a conventional controller (cycle time controller). Overall, the fuzzy controller shows a modest increase in average flow rate. In particular, when traffic densities are highly uneven, the fuzzy controller shows a substantial increase in average flow rate. More significant, perhaps, than the average flow rate is the comparison of the average waiting time. For the fuzzy controller, the average waiting time decreased by 49 percent when compared with the conventional controller.

Another set of parameter files was constructed to simulate a "day in the life" of a traffic signal. The objective was to simulate and observe how well the fuzzy controller handled traffic during the span of a 12-hour day, including morning rush, morning lull, lunchtime rush, afternoon lull, "school's out" rush , and evening rush. Again, the fuzzy-controller simulation was compared with a conventional-controller simulation. The fuzzy controller handled 10 percent more cars than the conventional controller, and also showed an 8 percent increase in average flow rate, a 64 percent decrease in average waiting time, and a 29 percent decrease in maximum waiting time.

The 625 different combination simulations were useful in identifying strengths and weaknesses in the fuzzy controller. Using this information, we were able to modify aspects of the fuzzy controller and improve its performance. The "day in the life" simulation and the corresponding results are more interesting because they provide a "real world" example, which could easily be applied and modified to simulate an actual intersection in a city.

The test suite of 625 parameter sets was eventually collapsed to 69 sets due to the nature of the cross intersection and the duplicated traffic patterns in each lane. Since the inputs are the same for each of the four lanes and the traffic environment is symmetrical, it is unnecessary to repeat the traffic patterns for each lane. The results of the 625 parameter sets showed little to no difference in repeated traffic patterns from one lane to another. This is to be expected. What little difference was observed could be explained by the random nature of the input traffic flows.

The 69 parameter sets vary traffic flow from 0 to 1400 cars per hour with an increment of 350. Each of these parameter sets is also simulated for one hour of simulation time. This scaled-down test suite allows evaluation results with less computational time and no loss in coverage of the problem space.

Later testing included an improved conventional controller which combined proximity sensors with set cycle times. This controller performed much better than the set cycle time controller, but was still unable to outperform the fuzzy controller.

The traffic environment underwent some changes in a later version of the simulation, to better reflect reality. The changes all amounted to re-scaling of certain measurements, including the size of

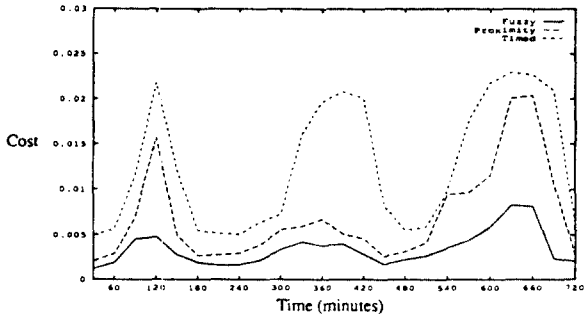

Figure 3. A Comparison of Fuzzy and Non-fuzzy Traffic Controllers.

cars, length and width of lanes, and sensor placement in the lanes. Additional testing, using the new version of the simulation, showed no relative change in controller performance among the three different controllers.

In Figure 3, comparison of the costs (waiting times) of the fuzzy controller and two standard methods clearly shows that the fuzzy controller drastically improves on the waiting times and thus the costs.

3. HARDWARE IMPLEMENTATION

3.1 Introduction

This section describes the use of fuzzy logic in controlling the acceleration of a model train. Velocity is a property which humans do not notice–it is acceleration that can cause passenger discomfort. By selecting a comfortable acceleration and maintaining it as a constant, a smooth increase in velocity and thus a smooth and comfortable ride can be achieved.

The acceleration problem, maintaining a constant acceleration and/or deceleration for a moving object (for example, a train), has a fuzzy nature. Fuzzy logic was chosen as a means of control for this problem, the idea being that a constant acceleration must be achieved as quickly as possible and then maintained. This is also true of deceleration when bringing the train to a stop. We asked: can fuzzy logic control do better than a human operator in this area?

The first part of the problem involves the selection of a comfortable acceleration constant. This constant is obviously different for a model train than a real train. We identified a comfortable acceleration by visually inspecting a water cell being pulled by the model train: at the correct constant acceleration, the water level in the cell maintains an even surface during increase in the train's velocity. From this visual state the acceleration is calculated with known time and distance. Once the acceleration is selected, the more interesting problem of control can be explored. The following sections describe the hardware involved in prototyping the problem and solution.

3.2 Model Train Layout

The model train travels on a circular track; a circle is used because any angular momentum experienced by the train becomes uniform in a circle. (An oval, for example, would contribute angular momentum only at the curved ends of the oval.) The circle is three feet in diameter from center of track to center of track. While circling, the train pushes an arm which turns an optical encoder at the center of the track circle. The optical encoder generates a number of pulses per revolution. These pulses are fed to a computer which calculates the velocity and acceleration of the train. The arm and optical encoder eliminate the need for attaching wires directly to the moving locomotive. Thus there is no need for an electrical umbilical cord which could become tangled during running of the train. The optical encoder sits at the center of the circle; an arm connects the locomotive and optical encoder. The model locomotive pulls a gondola car which has a water cell mounted inside; mounted to the bottom of the gondola car, between the wheel bases, is a lead weight which helps balance the weight of the water and water cell.

The computer uses the difference of the selected and current accelerations as an input to the fuzzy controller. The output of the fuzzy controller is an amount of change in acceleration, which might be an increase, a decrease, or nothing. This actually results in the computer increasing, decreasing, or not changing the power to the train. Deceleration follows the same scenario, except that the selected value is negative.

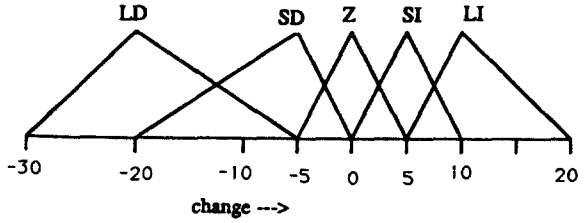

Figure 4. Membership Functions for the Train's Input Power Change.

3.3 Computer Input

The movement of the train pushing the arm turns the optical encoder. The rotation of the optical encoder generates digital pulses at the rate of 128 pulses per revolution, or one pulse every 2.8 degrees. The pulses accumulate in a six-bit counter which eliminates the need for the computer to be constantly reading pulses. Also, since the pulses are digital, there is no need for an analog-to-digital conversion. The six-bit counter is constructed of two cascading four-bit counters. Through experimentation, it was determined that one four-bit counter was insufficient for counting pulses when the train was moving at full velocity. The train's top velocity generates more than 16 pulses per read (strobe of the game and serial ports) by the computer. The six-bit counter allows accumulation of a maximum of 64 pulses.

The six-bit counter is connected to the computer through the game port (four lines used) and the serial port (two lines used). This gives six lines for reading the six-bit value from the counter. Use of the game and serial ports simplifies the interface between the computer and the input from the train. No communications protocols are necessary.

3.4 Fuzzy Controller

The fuzzy controller consists of one input and one output. The input to the controller is the difference in the selected acceleration and the train's current acceleration. The membership functions for diff (the input) are Very Negative, Negative, Zero, Positive, and Very Positive (VN, N, Z, P, and VP, respectively) and are similar to those in Figure 2. The output from the controller is an amount of change in power. The membership functions for change (the output) are Large Decrease, Small Decrease, Zero, Small Increase, and Large Increase (LD, SD, Z, SI, and LI, respectively) and are shown in Figure 4. The fuzzy rule base mapping the input to the output is shown in Figure 5. The number of rules is determined by the number of membership functions in the input, which is a total of five. These membership functions and rules will work for deceleration as well because the selected deceleration value will be the negative of the selected acceleration value.

```
IF diff is Z THEN change is Z
IF diff is P THEN change is SI
IF diff is VP THEN change is LI
IF diff is N THEN change is SD
IF diff is VN THEN change is LD
```

Figure 5. Fuzzy Rules for the Train Acceleration Control System.

3.5 Computer Output

The output from the fuzzy controller (change in power) is added to the current power being output to the train. The computer represents this range of power (in volts) as a numerical range between zero and 255, and outputs a value in this range via the parallel port to a digital-to-analog (D/A) chip. Use of the parallel port for output from the computer again simplifies the interface and avoids the need for communications protocols. The D/A chip converts the eight input lines to one analog output line. The output from the chip is a voltage in the range of zero to 5 volts and is connected to a power regulator circuit. A 15-volt transformer serves as the power supply for the train. This supply is varied by the power regulator circuit as directed by the D/A chip output. The power regulator circuit has an output (to the train track) in the range of 4 to 10 volts along with an increased current to drive the train. Experimentation showed that 4 volts is just below the amount needed to keep the train moving.

3.6 Software Interface

The software interface consists of three parts: the input module, the fuzzy controller module, and the overall driver program. The input module and driver program are written in C compiled with the Turbo C compiler. The fuzzy controller module is written in Togai InfraLogic fuzzy source code and compiled into C with the Togai Fuzzy C Compiler [7]. These programs are all executed on a personal computer. The input module reads the data lines from the counter off the game and serial ports and builds the six-bit value representing the number of pulses. A driver program was written in C to go with the Togai Fuzzy C

language code. The driver program initializes everything, computes the acceleration, calls the fuzzy controller, and writes the output to the parallel port.

3.7 Results

The results are heavily based upon visual inspection of the water cell and observation of the train itself. The hope was that the surface of the water in the water cell would angle "uphill" as the train increased acceleration. When a constant acceleration is achieved the water should level itself, and when deceleration begins the water level should angle "downhill." If this state actually occurred, however, the angle was so small it could not be observed; perhaps the maximum velocity was not large enough. Because of this it was also difficult to select an acceleration for the model train which represented a "comfortable" (for humans) acceleration in real trains. The model train accelerates to maximum so quickly that there is not much to observe.

Two methods of acceleration increase were observed, fuzzy controlled and crisp proportional controlled. The crisp control method is analogous to setting the "throttle" wide open. The train made a "jackrabbit" start and the water in the water cell showed erratic movement–similar to driving a car while holding a cup of coffee: the coffee sloshes around or even spills when the car abruptly stops or starts. With the fuzzy controller, the train's acceleration was much less abrupt. The small amount of movement in the water cell was much less than that observed during crisp control.

3.8 Problems and Limitations

The main problem with this experiment is that of granularity. The computer measures its internal time in clicks. There are 18.2 clicks per second, which calculates to approximately 0.05 seconds per click. In 0.05 seconds at full speed the train travels approximately two pulses. More accurate accelerations could be calculated with more accurate timing of the pulses. This would require a computer whose internal clock generated more clicks per second. Although the lack of communications protocols simplifies the design of the system, it also causes a potential problem: that of reading the values of the counter through the game and serial ports. It is possible for the counter to receive a pulse during the time the computer is reading from those ports, causing an erroneous value to be input to the computer. Hence, a check was added to identify and discard any calculated velocities which appeared out of range (faster than the observed maximum velocity of the train). Another limitation involved the power regulator circuit. Although the power supply offered a maximum voltage of 15 volts, the power regulator circuit was able to vary a range of only 4 to 10 volts. An improved power regulator circuit could obtain smoother control of the train and allow the train to be accelerated over a longer period of time. This would allow the fuzzy control a greater effect.

4. DISCUSSIONS AND CONCLUSIONS

Improvements were made to the fuzzy controller after study of the test data. The number of inputs was increased to five: two green traffic densities, two red traffic densities, and the time input. Previously, the green input density was an average of the traffic densities in the two green directions. The same was true for the red input density. By making an input for each green direction and each red direction, we can obtain more accurate traffic densities. This change in the number of inputs increases the number of possible rules, but through the use of "and" and "or" operators, fuzzy input values can be combined in one rule. This technique kept the number of rules from increasing. The controller was also improved by making minor changes to the shapes of some membership functions in the controller inputs. These changes were made after careful study of the test results and observation of the graphic display of the simulation. Each change to a membership function was followed by additional testing to verify performance.

By way of preliminary conclusions, the initial results show that larger quantities of traffic are "handled" by fuzzy control methods than by conventional control methods. (A "handled" car is one that has exited the simulation after having passed through the intersection.) Fuzzy control shows greatest improvement over conventional methods when the traffic flow is highly uneven. The fuzzy controller can change the lights as necessary to achieve the maximum throughput, rather than be limited to a set cycle time for changing signal lights. Another, perhaps more important, result is that the average time spent waiting in traffic decreases with the use of fuzzy control versus conventional control. This means individual cars spend less time waiting and more time moving.

Fuzzy control has other benefits as well. The fuzzy controller handles a wide range of continually changing traffic patterns. This characteristic makes the fuzzy controller re-usable, a quality not shared by conventional controllers. This concept, even with modest or no improvement in controller performance, makes a fuzzy controller more desirable. For example, all cross intersections may have the same fuzzy controller without requiring individual controller configurations based on observations at an intersection. Also, the cost of a traffic-control system may actually decrease because of the system's re-usability.

For the future, the fuzzy controller is combined with a neural network to construct an adaptive fuzzy/neural controller. An adaptive controller such as this is similar to a standard fuzzy controller, with the exception that the membership functions are not fixed but can change over time. The controller observes the results of its actions and attempts to improve its performance based on these observations. An adaptive controller thus shows promise by being able to learn the peculiarities of a particular intersection while retaining the ability to handle unexpected situations.

Hardware implementation of the fuzzy control of the model train does offer an improved method of controlling the acceleration. The desired amount of improvement was not achieved, but improvement over an "open throttle" was observed. Fuzzy control of the train displayed a smoother movement from "rest" to "maximum velocity" and back to "rest." The small scale of this experiment affected what could be observed and the problems discussed in the previous section dampened the potential results. Further refinements are under way for this fuzzy acceleration control of a model train.

REFERENCES

[1] Nye, A., *Xlib Programming Manual*. Sebastopol, California: O'Reilly and Associates, Inc., 1988.

[2] Nye, A. and T. O'Reilly, *X Toolkit Intrinsics Programming Manual*. Sebastopol, California: O'Reilly and Associates, Inc., 1990.

[3] Haight, Frank A., *Mathematical Theories of Traffic Flow*, New York: Academic Press Inc. 1963.

[4] Montrol, E. W., "Acceleration Noise and Clustering Tendency of Vehicular Traffic", *Theory of Traffic Flow*, ed. R. Herman, New York: Elsevier Publishing Company, 1961.

[5] Morris, R. W. J. and P. G. Pak-Poy, "Intersection Control by Vehicle Actuated Signals", *Vehicular Traffic Science*, ed. R. Herman and R. Rothery, New York: American Elsevier Publishing Company, Inc. 1967.

[6] Malek-Zavarei and M. Jamshidi, *Time-Delay Systems - Analysis, Optimization, and Applications*, North Holland Publishing Co., Amsterdam, 1986.

[7] Togai InfraLogic, Inc. (1990), *Fuzzy C-Development Systems User's Guide*, Irvine, CA.

[8] Gallegos, R. and T. Nguyen, "Fuzzy Logic Traffic Application", CAD Laboratory Internal Report, University of New Mexico, Albuquerque, NM. 1991.