

Clemson University

TigerPrints

All Theses

Theses

December 2019

Agent-Based Resilient Transportation Infrastructure with Surrogate Adaptive Networks

Andre Aaron Apostol

Clemson University, andre.apostol95@gmail.com

Follow this and additional works at: https://tigerprints.clemson.edu/all_theses

Recommended Citation

Apostol, Andre Aaron, "Agent-Based Resilient Transportation Infrastructure with Surrogate Adaptive Networks" (2019). *All Theses*. 3230.

https://tigerprints.clemson.edu/all_theses/3230

This Thesis is brought to you for free and open access by the Theses at TigerPrints. It has been accepted for inclusion in All Theses by an authorized administrator of TigerPrints. For more information, please contact kokeefe@clemson.edu.

AGENT-BASED RESILIENT TRANSPORTATION INFRASTRUCTURE WITH
SURROGATE ADAPTIVE NETWORKS

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Mechanical Engineering

by
Andre A. Apostol
December 2019

Accepted by:
Dr. Cameron J Turner, Committee Chair
Dr. Ardalan Vahidi
Dr. Mashrur Chowdhury

Abstract

Connected autonomous intelligent agents (AIA) with enhanced decision making through machine learning can improve intersection performance and resilience for the transportation infrastructure. An agent is an autonomous decision maker whose decision making is determined internally but may be altered by interactions with the environment or other agents. Implementing agent-based modeling techniques to advance communication for more appropriate decision making will provide great benefits to autonomous vehicle technology.

A new algorithm is proposed to improve the decision-making process of autonomous vehicles and intelligent traffic signals, specifically at city like intersections. This is completed by understanding vehicle to vehicle (V2V), vehicle to infrastructure (V2I), and infrastructure to infrastructure (I2I) communication and using gathered data to ensure these agents make more appropriate decisions given the circumstances. These vehicles and signals are modeled to adapt to the common traffic flow of the intersection and ultimately find an optimum flow that will decrease average vehicle time to ultimately reduce inefficiency through each intersection. Considering each light and vehicle as an agent and utilizing communication between these agents will enable opportunity for data transmission. **Improving agent-based I2I communication and decision making will provide performance benefits to traffic flow capacities.**

Evaluations were completed comparing intersections with fixed, coordinated, and adaptive timing signals. A fixed timing signal is an intersection using a fixed maximum green light time with no opportunity for adjustment. The coordinated signals adapt and change light status based on the current light status of adjacent intersections. Adaptive signals add in a recognition of vehicle load in one direction and adjust their own status either based on the load at the individual intersection or a neighboring light status change with the intent to improve traffic flow.

To compare these scenarios given a specific example of 160 total vehicles present on the road in a 2x2 intersection grid setup, inefficiency was reduced from 50% to 45% given the relationship between ideal average time compared to actual average time for vehicles proceeding through an intersection. Overall tests were run to compare the different light signal options based on the number of vehicles on the road and maximum green light time in one direction. The results were consistent and overall inefficiency was reduced using an adaptive traffic signal to recognize upcoming vehicles combined with the ability to adjust based on adjacent intersection light status changes.

Table of Contents

1.	Introduction.....	1
1.1	Introduction.....	1
1.2	Research Problem and Motivation	2
1.3	Thesis Statement.....	4
1.4	Research Outline	5
1.4.1	Chapter 2: Literature Review	5
1.4.2	Chapter 3: Simulation Method	5
1.4.3	Chapter 4: Simulation Results.....	6
1.4.4	Chapter 5: Conclusions and Future Work.....	6
2.	Literature Review	7
2.1	Current Infrastructure Technology	7
2.2	Recent Autonomous Vehicle Technology Advancements	8
2.3	Partially Observable Markov Decision Process (POMDP)	11
2.4	Connected Vehicle Technology.....	12
2.5	Potential for Communication Improvement.....	19
3.	Simulation Method	21
3.1	Research Campaign	21
3.2	Introduction of Autonomous Vehicle and Intelligent Traffic Signal Agents	21
3.2.1	Defining an Agent Based Autonomous Vehicle	21
3.2.2	Defining an Agent Based Intelligent Traffic Signal.....	23
3.2.3	Coupled System of Autonomous Vehicles and Intelligent Traffic Signals ..	24
3.3	The MATLAB Model.....	25
3.3.1	Modeling a Single Fixed Timing Signal Intersection	25
3.3.2	Intersection Grid with Fixed Timing Signals.....	52
3.3.3	Implementing Coordinated Traffic Signals	57
3.3.4	Implementing Adaptive Traffic Signals	60
3.4	Simulation Goals of the MATLAB Model.....	63
4.	Simulation Results.....	64
4.1	Fixed Timing Evaluation Results.....	64
4.1.1	1x1 Intersection Setup	65

4.1.2	2x2 Intersection Setup	68
4.1.3	Overall Fixed Timing Results	70
4.2	Coordinated Signal Evaluation Results	73
4.2.1	Graphic Displays of Coordinated Light Sequence	74
4.2.2	Overall Coordinated Light Sequence Results	77
4.3	Adaptive Signal Results	80
4.3.1	Graphic Displays of Adaptive Light Sequence	81
4.3.2	Overall Adaptive Light Sequence Results	83
4.4	Light Sequence Comparison	87
5.	Conclusions and Future Work	89
5.1	Research Conclusion	89
5.2	Future Work	89
5.3	Thesis Reflection	89
6.	References	93
7.	Appendix	97

List of Figures

FIGURE 1: AGENT BASED MODELING APPROACH.....	3
FIGURE 2: VEHICLE RECOGNITION TECHNOLOGY.....	9
FIGURE 3: FULL CITY INTERSECTION CONNECTIVITY.....	19
FIGURE 4: COUPLED AGENT-BASED BEHAVIOR.....	24
FIGURE 5: INDIVIDUAL MATLAB SIMULATION INTERSECTION.....	25
FIGURE 6: LANE LENGTH REFERENCE.....	28
FIGURE 7: HONDA ACCORD ADAPTIVE CRUISE CONTROL OPTIONS.....	33
FIGURE 8: DISTANCE ACCELERATION PROCESS.....	34
FIGURE 9: DISTANCE FROM INTERSECTION TO LIKELY PROCEED THROUGH BASED ON DESIRED TURN.....	43
FIGURE 10: 2 X 3 INTERSECTION GRID.....	53
FIGURE 11: COORDINATED INTELLIGENT INTERSECTION SETUP.....	58
FIGURE 12: COORDINATED TRAFFIC SIGNAL LOGIC.....	59
FIGURE 13: INDIVIDUAL TRAFFIC SIGNAL STATUS.....	59
FIGURE 14: 2X2 FIXED TIMING SIGNAL SEQUENCE.....	64
FIGURE 15: FIXED TIMING, 5 VEHICLES.....	65
FIGURE 16: FIXED TIMING, 10 VEHICLES.....	65
FIGURE 17: FIXED TIMING, 20 VEHICLES.....	66
FIGURE 18: FIXED TIMING, 40 VEHICLES.....	66
FIGURE 19: FIXED TIMING, 60 VEHICLES.....	66
FIGURE 20: INEFFICIENCY FIXED TIMING, 5 VEHICLES.....	67
FIGURE 21: INEFFICIENCY FIXED TIMING, 10 VEHICLES.....	67
FIGURE 22: INEFFICIENCY FIXED TIMING, 20 VEHICLES.....	67
FIGURE 23: INEFFICIENCY FIXED TIMING, 40 VEHICLES.....	67

FIGURE 24: INEFFICIENCY FIXED TIMING, 60 VEHICLES.....	67
FIGURE 25: AVERAGE TIME, FIXED, 5 VEHICLES.....	68
FIGURE 26: AVERAGE TIME, FIXED, 10 VEHICLES.....	68
FIGURE 27: AVERAGE TIME, FIXED, 20 VEHICLES.....	68
FIGURE 28: AVERAGE TIME, FIXED, 40 VEHICLES.....	68
FIGURE 29: AVERAGE TIME, FIXED, 60 VEHICLES	69
FIGURE 30: INEFFICIENCY, FIXED, 5 VEHICLES.....	69
FIGURE 31: INEFFICIENCY, FIXED, 10 VEHICLES.....	69
FIGURE 32: INEFFICIENCY, FIXED, 20 VEHICLES.....	70
FIGURE 33: INEFFICIENCY, FIXED, 40 VEHICLES.....	70
FIGURE 34: INEFFICIENCY, FIXED, 60 VEHICLES	70
FIGURE 35: AVERAGE VEHICLE TIME FOR 2X2 INTERSECTION, FIXED SIGNAL.....	72
FIGURE 36: AVERAGE VEHICLE TIME FOR 2X3 INTERSECTION, FIXED SIGNAL.....	72
FIGURE 37: AVERAGE VEHICLE TIME FOR 3X3 INTERSECTION, FIXED SIGNAL.....	73
FIGURE 38: COORDINATED SIGNAL LIGHT SEQUENCE	73
FIGURE 39: AVERAGE TIME, COORDINATED, 5 VEHICLES.....	74
FIGURE 40: INEFFICIENCY, COORDINATED, 5 VEHICLES.....	74
FIGURE 41: AVERAGE TIME, COORDINATED, 10 VEHICLES.....	75
FIGURE 42: INEFFICIENCY, COORDINATED, 10 VEHICLES.....	75
FIGURE 43: AVERAGE TIME, COORDINATED, 20 VEHICLES.....	75
FIGURE 44: INEFFICIENCY, COORDINATED, 20 VEHICLES.....	75
FIGURE 45: AVERAGE TIME, COORDINATED, 40 VEHICLES.....	75
FIGURE 46: INEFFICIENCY, COORDINATED, 40 VEHICLES.....	75
FIGURE 47: AVERAGE TIME, COORDINATED, 60 VEHICLES.....	76
FIGURE 48: INEFFICIENCY, COORDINATED, 60 VEHICLES.....	76

FIGURE 49: 2x2 INTERSECTION LIGHT NUMBER LOCATIONS76
FIGURE 50: VISUAL STATUS REPRESENTATION OF COORDINATED TRAFFIC SIGNALS77
FIGURE 51: AVERAGE VEHICLE TIME FOR 2x2 INTERSECTION, COORDINATED SIGNAL.....	.79
FIGURE 52: AVERAGE VEHICLE TIME FOR 2x3 INTERSECTION, COORDINATED SIGNAL.....	.79
FIGURE 53: AVERAGE VEHICLE TIME FOR 3x3 INTERSECTION, COORDINATED SIGNAL.....	.80
FIGURE 54: AVERAGE TIME, ADAPTIVE, 5 VEHICLES81
FIGURE 55: INEFFICIENCY, ADAPTIVE, 5 VEHICLES81
FIGURE 56: AVERAGE TIME, ADAPTIVE, 10 VEHICLES.....	.81
FIGURE 57: INEFFICIENCY, ADAPTIVE, 10 VEHICLES.....	.81
FIGURE 58: AVERAGE TIME, ADAPTIVE, 20 VEHICLES.....	.82
FIGURE 59: INEFFICIENCY, ADAPTIVE, 20 VEHICLES.....	.82
FIGURE 60: AVERAGE TIME, ADAPTIVE, 40 VEHICLES.....	.82
FIGURE 61: INEFFICIENCY, ADAPTIVE 40 VEHICLES.....	.82
FIGURE 62: AVERAGE TIME, ADAPTIVE, 60 VEHICLES.....	.82
FIGURE 63: INEFFICIENCY, ADAPTIVE, 60 VEHICLES.....	.82
FIGURE 64: VISUAL STATUS REPRESENTATION OF ADAPTIVE TRAFFIC SIGNALS83
FIGURE 65: OVERALL COMPARISON OF AVERAGE VEHICLE TIME FROM ADAPTIVE SIGNAL IMPLEMENTATION87
FIGURE 66: VARIOUS SIGNAL SEQUENCE COMPARISON OF AVERAGE VEHICLE TIME.....	.88
FIGURE 67: VARIOUS SIGNAL SEQUENCE COMPARISON OF INEFFICIENCY.....	.88
FIGURE 68: TRENDLINE COMPARISON OF VARIOUS SIGNAL SEQUENCES.....	.88

1. Introduction

1.1 Introduction

Connected autonomous intelligent agents (AIA) with enhanced decision making through machine learning can improve intersection performance and resilience for the transportation infrastructure. An agent is an autonomous decision maker whose decision making is determined internally but may be altered by interactions with the environment or other agents. Implementing agent-based modeling techniques to advance communication for more appropriate decision making will provide great benefits to autonomous vehicle technology.

In general, swarm robotic decision making has been optimized using a central master controller that relays information to other slave robots. This technique may be difficult to utilize when many subsystems must be controlled. This is due to the main controller functioning as the only input to one robot; these robots do not consider the input of neighboring robots. This creates difficulty in synchronizing the control of all subsystems. Therefore, agent-based communication is a beneficial alternative to improve autonomous vehicle decision making. Advancing this form of technology will improve traffic flow as well as create a safer and more resilient environment for the transportation infrastructure.

To improve overall communication within a group of agents, it is beneficial to allow the robots to communicate individually as opposed to having one centralized controller.

This will create a more resilient environment and allow each agent to communicate their own status to neighboring agents. Success of this method will ensure the system stays more up to date as time continues. Furthermore, rather than having a centralized controller gathering all information, each individual robot will gather data that can be interpreted and used for individual decision making. The ability for individual agents to gather data to provide information to nearby agents will allow the system to function more as a realistic intersection model.

1.2 Research Problem and Motivation

With the emerging autonomous vehicle technology, it is important to study the positive and negative effects that may occur throughout a realistic connected vehicle/city environment. Furthermore, how can the newer communication technology be used to improve performance through intersections? Enhanced safety, traffic flow, and resilience are all beneficial to vehicle transportation and an agent-based approach will ultimately create a more positive outcome.

This technique can be applied to autonomous vehicle transportation aspects. The ability for each car to gather data through its own sensors as well as pull data from other local cars can be a huge technology improvement. If automobiles can relay information about current locations to nearby vehicles as well as traffic lights, the intersection can apply the received data to the immediate situation. For example, an individual intersection may be overcrowded from a high number of approaching vehicles. The improved communication between the status of intersections will allow traffic lights to

communicate to provide alternative light signal times. This is just one example that displays the benefits of improved vehicle communication and decision making. This knowledge will ultimately improve traffic flow, safety and overall agent behavior.

Intelligence is a systems ability to act appropriately in an uncertain environment, where an appropriate action increases the probability of success, and success is the achievement of behavioral sub-goals that support the system’s main goal [Likhachev 2009]. To improve autonomous vehicle intelligence, this Autonomous Intelligent Agent

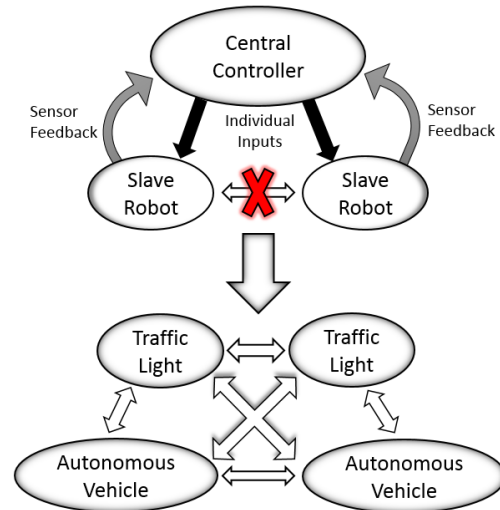


Figure 1: Agent Based Modeling Approach.

(AIA) technique can be applied to the automotive transportation system. Individual vehicles can recognize and communicate status to other nearby vehicles regarding position, velocity and upcoming desired direction. The ability for these agents to communicate will allow for more detailed traffic data and the interpretation of this data can result in an improvement of resilience for the transportation infrastructure.

Furthermore, to ultimately improve roadway intersection performance, it is beneficial to consider traffic signal controllers as agents as well. Figure 1 displays this utilization of the connected agent technology. Automobiles can relay information about their current speed and location to the intelligent traffic signal controllers whom can then interpret this data and apply it appropriately. This ideally will allow the traffic signal

controllers to make a status decision based on the abundance or lack of vehicles approaching from a specific direction. It will be demonstrated how relaying individual status can ultimately improve traffic flow at intersection scenarios.

1.3 Thesis Statement

A new algorithm is proposed to improve the decision-making process of autonomous vehicles and intelligent traffic signals, specifically at city like intersections. This is completed by understanding vehicle to vehicle (V2V), vehicle to infrastructure (V2I), and infrastructure to infrastructure (I2I) communication and using gathered data to ensure these agents make more appropriate decisions given the circumstances. Working with more complex and realistic situations for intersections is a new study and overall, the connection of the vehicles and traffic lights will ultimately allow for the ability to solve an ideal traffic signal optimization problem. This concept will be proven by modeling city intersections while considering vehicle distances from an intersection to determine light change probability. The performance will be evaluated, and the information gathered through the agent-based communication at each intersection will be used to improve the decision making of the traffic signals at individual intersections.

With improved decision making, we can create more accurate models of common intersections. These vehicles and signals are modeled to adapt to the common traffic flow of the intersection and ultimately find an optimum flow that will decrease average vehicle time through each intersection. Considering each light and vehicle as an agent and utilizing communication between these agents will enable opportunity for data

transmission. As data is gathered, each agent can make safer, time dependent decisions to benefit the intersection in that city. **Improving agent-based I2I communication and decision making will provide performance benefits to traffic flow capacities.** Furthermore, advancing this form of communication to allow for self-configuring systems will improve traffic flow as well as create a safer environment for the transportation infrastructure.

1.4 Research Outline

This section will highlight the details of each chapter to provide insight on what can be expected.

1.4.1 *Chapter 2: Literature Review*

A literature review has been completed on different approaches to modeling autonomous vehicle behavior and on the potential to connect intersections in a smart city environment. The initial improvement of autonomous vehicles is a new technological advancement and studies to show potential upgrades using this technology for more advanced communication abilities are the main areas of focus for this literature review. Simulating a connected agent environment is a new study but literature reviews on many aspects of autonomous vehicles can demonstrate the need for agent-based behavior in this specific environment.

1.4.2 *Chapter 3: Simulation Method*

The MATLAB model demonstrating the improved traffic flow will be described. The logic and equations used to calculate the appropriate behavior for each vehicle and traffic

signal will be explained. The various techniques and methods used throughout the code are related to the kinematic performance of vehicles and the data gathered will be used to prove that this method is of use to the transportation infrastructure.

1.4.3 Chapter 4: Simulation Results

The results from the various simulations run will be displayed and discussed. Numerous tests were carried out based on overall car count in the simulation, maximum green signal light time per intersection, and car load at a specific intersection. Basic intersection models with fixed timing signals were initially evaluated and ultimately compared to the adaptive timing signal network. The average vehicle time through each intersection is evaluated to determine if overall traffic flow is improved.

1.4.4 Chapter 5: Conclusions and Future Work

Traffic flow improvement is an engineering problem that has been discussed for many years. With emerging autonomous vehicle technology, a new type of solution can be utilized. This autonomous agent technique can eventually be applied for all dynamic components that may be considered in a city environment.

2. Literature Review

The individual details of each component in the transportation infrastructure are complex, therefore highlights of current, new, and potential improvements will be discussed. Overall, there is a need and potential for improved communication between vehicles and traffic signals given emerging autonomous vehicle technology.

2.1 Current Infrastructure Technology

General traffic lights operate on a fixed timing schedule typically only allowing adjustments to the sequence based on a sensor to detect vehicles at the intersection and through expected volumes of traffic based on daily traffic routines. These common approaches are solely based on the detection of nearby vehicles. A benefit of detection devices present in a fixed light sequence traffic signal is the option to alter traffic flow directions through nearby vehicle detection given the light has not reached its maximum green light time display. Another benefit is the option to skip certain cycles if no vehicle is present in a specific direction at that intersection. This will result in an improved flow of traffic in the opposite direction. Another common technique for traffic flow optimization is the use of the concept rolling horizon [Goodall 2013]. A traffic control algorithm will optimize an objective function over a short period of time to estimate the position of vehicles over future cycles. This approach again only allows for estimation of a vehicle location as opposed to a precise recognition. With emerging autonomous vehicle technology, intersection performance can greatly be enhanced.

Furthermore, a rolling horizon quadratic programming approach was used for signal control [Aboudolas 2010]. They investigated recently developed signal control and discovered new ways to improve real-time network control in large-scale networks. The traffic responsive urban control (TUC) method was used and is based on a linear quadratic multivariable regulator which considers minimum green time constraints and cycle time. Two different strategies of first and second class were created. First class considers undersaturated traffic conditions while second class considers oversaturated traffic conditions. Overall optimization for network wide signal control of traffic was proven effective through efficiency improvement.

2.2 Recent Autonomous Vehicle Technology Advancements

For over a decade, there have been several attempts to develop approaches for improving operations of self-driving vehicles through signalized intersections [Mladenovic 2014]. One main concentration for improvement has been the cooperation of the vehicles to improve safety. About 96% of traffic engineers recognize the importance of safety at intersections, while identifying the concern for respect and morality. Crashes that occur generally are due to human error. Therefore, to implement autonomous vehicles and ensure citizens are content with this improvement, a safer environment throughout the automotive transportation must be proven successful.

A wide range of approaches to improve decision making of autonomous vehicles have been carried out. Although certain problems may be anticipated, methods in the literature neglect uncertainty on the future states of other nearby vehicles [Petti 2005].

Some approaches assume a dynamic model of a detected obstacle and propagate its state using standard filtering techniques such as the extended Kalman filter [Fulgenzi 2008]. However, these experiments have resulted in unrealistic models as these approaches have led to conservative and partially unrealistic data due to too many assumptions of current and future states of nearby vehicles.

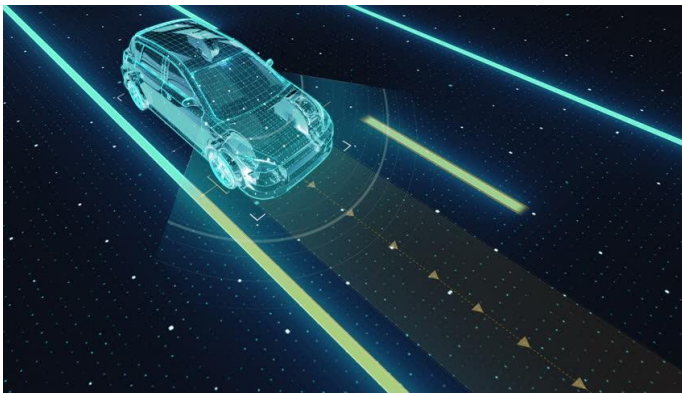


Figure 2: Vehicle Recognition Technology.

Autonomous vehicles have also seen instances of decision-making approaches in traffic situations from the 2007 DARPA Urban Challenge [Darpa 2007].

Generally, decision making was performed for a variety of solutions that ranged from finite state machines [Montemerlo 2008], to decision trees [Miller 2008], to several heuristics [Urmson 2008]. However, some approaches have attempted to solve the decision-making problem for autonomous driving through the lens of trajectory optimization [Ferguson 2008]. These approaches were beneficial in demonstrating the use of this technology. Newer studies have gathered information from these experiments to improve the ability of autonomous vehicles in different environments.

Decision-making for autonomous driving can be challenging because of uncertainty and the continuous state of nearby vehicles [Galceran 2015]. Majority of autonomous vehicle decision making algorithms assume full knowledge of each dynamic component

which can lead to incorrect perception of the road. Galceran et al [2015] discusses the development of a new algorithm with an improved method of tracking dynamic objects on the road by using model-based estimation. This algorithm is used to improve reasoning in occluded regions and in passing, merging, or intersection handling situations that autonomous vehicles may encounter. The overall method in tracking full dynamic behavior of certain components can be implemented in an appropriate agent-based model algorithm.

Galceran et al [2015] also created a method to allow nearby autonomous vehicles to evaluate consequences of potential actions given possible decisions of that vehicle. The history of common dynamic states of a nearby vehicle is first evaluated to create a likely outcome policy for the nearby vehicle. The intentions of the initial vehicle are observed as well. Given the two behaviors of the vehicles, a closed loop interaction maximizes the reward given the direct scenario. These outcomes were then evaluated to prove this anticipated decision-making approach is beneficial. This decision-making approach can be used to improve the overall safety in an agent-based environment as nearby vehicles will be able to track and predict upcoming states of nearby vehicles

Cunningham et al [2015] also created a multipolicy approach for improved decision making in uncertain environments. Considering the future states of other agents has resulted in an ability to scale the model to more complex traffic scenarios. A real-time policy from nearby vehicles as well as the selected vehicle is evaluated, and the algorithm ultimately selects the best outcome for the controlled vehicle. Furthermore, different

driving preferences are considered which creates opportunity for an improved human experience based on driver preference. The experiment for this approach was demonstrated using a real-world autonomous vehicle to justify the need for this algorithm. Overall, implementing these autonomous systems into our infrastructure requires a delicate balance of new technologies, but the improved performance in an agent-based environment is evident.

2.3 Partially Observable Markov Decision Process (POMDP)

A common method that has been used to solve decision optimization problems is the Partially Observable Markov Decision Process (POMDP). This model provides a mathematically rigorous formalization of the decision-making problem in uncertain dynamic scenarios [Galceran 2015]. In general, these problems have resulted in very complex computations which can take several hours to converge even while considering an extremely basic non-real-world scenario. Although solutions have been considered lengthy, POMDP methods have still been formulated to arrive at appropriate solutions for these decision-making scenarios.

Bandyopadhyay et al [2013] performed a POMDP approach that considers motion planning through the possibilities of the human intention. The model uses a finite set of possible human intentions and an algorithm for an autonomous robot is developed to recognize these intentions. The method consists of an autonomous vehicle agent interacting with a human agent. Given the possibilities of the human intention, the autonomous vehicle agent will observe the behavior and establish a decision in advance

to act optimally in that specific scenario. Given the different scenarios in this type of environment, a safer decision can be determined to create a safer environment. Agent based modeling can be used to improve interactions between automobiles and humans.

A point-based Markov decision process for single-lane driving and merging was performed by [Wei 2011] and [Ulbrich 2013] and was applied to a POMPD formulation that considered highway changes. [Brechtel 2014] also performed an experiment using continuous state space reasoning about objects that may potentially be hidden while considering observation uncertainty. Ideas using these methods have been performed but due to the complex nature of these problems, it is generally not extremely beneficial to include a POMDP problem in this research.

2.4 Connected Vehicle Technology

Although quality decision making is important in the improvement of safety, connectivity between vehicles adds an extra component to improve the traffic flow and overall safety of the vehicle. Talebpour and Mahmassani [2016] performed a study demonstrating the influence of connected autonomous vehicles and the impact on traffic flow. It was proven that connected vehicle technology can provide real-time information about nearby traffic and ultimately can increase efficiency and reliability.

In the same article published by Talebpour and Mahmassani [2016], the type of communication that can occur in an autonomous environment was discussed. Active Vehicle-to-Vehicle (V2V) communication is the ability for one vehicle to maintain an appropriate distance behind another. This is typically based on desired spacing,

comfortable acceleration or deceleration, and the relative velocity between the vehicles. This specific type of communication is like adaptive cruise control (ACC) which allows a user to specify a top speed which may be reduced based on the distance behind and speed of a vehicle in front. Vehicle to Infrastructure (V2I) is also an important level of communication. Active V2I communications allow real-time data to be transmitted regarding speeds of multiple vehicles. The signal can then update an appropriate speed limit to allow the connected autonomous vehicles to work in harmony. It is concluded that the general autonomous vehicle will calculate the appropriate acceleration based on all inputs to the system from nearby vehicles and infrastructure signals. This calculation is important as the basic behavior of a vehicle begins with the ability to accelerate and decelerate appropriately.

Smith et al [2010] created a decentralized innovative traffic signal algorithm that utilizes IntelliDrive technologies to improve the efficacy of traffic signals. This traffic control algorithm determined the optimal point to terminate the green phase in one direction based on the present traffic pattern. Furthermore, data gathering strategies for changes in acceleration, network connectivity, and road conditions were implemented. This knowledge allowed traffic control algorithms to be created that would determine the strength in the connectivity of connected vehicles. Eventually, an interface between MATLAB and VISSIM was used to implement the algorithm and real-world performance was evaluated in the Washington DC metro area. Dedicated short range communication technology (DSRC) was used to implement this communication. The proportion of

vehicles passing through the intersection was compared with real world data and demonstrates that a decentralized adaptive traffic signal was beneficial.

More recent research relating to the topic of connecting autonomous vehicles and traffic signals is through [Feng 2015] paper titled “A real-time adaptive signal control in a connected vehicle environment.” Common traffic signals have been optimized to improve traffic flow based on real-time traffic conditions. Adaptive signal controls design signal time and phasing on-the-fly based on real-time traffic demand as well as predicted traffic demand. Furthermore, they can use sensors embedded in the pavement or non-intrusive sensors, like video detectors. However, this traffic flow can be improved with advances in wireless communication technology as vehicles can communicate with each other and with the infrastructure in the emerging connected vehicle system [Feng 2015]. There have been many advances in Vehicle to Vehicle (V2V) communication as well as Vehicle to Infrastructure (V2I) communication. These technologies use dedicated short-range communication (DSRC) and this technology can be used to gather data for these specific communication scenarios.

This study considered both autonomous and non-autonomous vehicles. Applications utilizing V2I communication enable the intersection to acquire a more complete picture of the nearby vehicle states. Data from connected vehicles provide real-time vehicle location, speed, acceleration, and other status-based vehicle data. From this new source of data, traffic controllers should be able to make “smarter” decisions [Feng 2015]. This author has presented a real-time adaptive traffic control algorithm by utilizing data from

connected vehicles. Algorithms for this study utilize arrival time, estimation for traffic signal timing, and phasing decision at the traffic controller.

To improve light signal timing, Goodall et al [2013] created an algorithm to control traffic signals with connected vehicles. Instead of relying on point detectors to recognize vehicles at a fixed location, traffic signals can use data transmitted from a vehicle through DSRC to gain access to previously estimated measures such as vehicle speed, position, arrival time, acceleration rates, and queue lengths [Goodall 2013]. The predictive microscopic simulation algorithm (PMSA) was then created to improve state of the practice performance by responding to real time demands while eliminating the ability to reidentify records of an individual vehicle to protect driver privacy. The algorithm initially receives data regarding the position and speed within a 300-meter distance of the light. Assuming a minimum green light signal time of 5 seconds and a maximum of 15 seconds, the most appropriate green light signal timing is determined by the time required to clear vehicles in that direction.

Similar connected vehicle and infrastructure research was also completed for situational awareness for a connected autonomous vehicle (CAV) making a left turn at a signalized intersection [Khan 2019]. Video cameras as well as lidar and radar sensors are placed at the intersection to recognize upcoming vehicles traveling in the opposite direction of the vehicle intending to make a yielding left turn. The intersection will predict the arrival time to the intersection of the opposite direction vehicles. If the maneuver can be completed safely, the intersection sensors will notify the CAV (I2V) that it may proceed

through the intersection. Furthermore, given a two-lane road, the autonomous vehicle control system can recognize behind vehicles to determine if a safe maneuver to the left turn lane can be completed [Khan 2019]. This study was completed given an aggressive non-CAV driver which is important to consider because not all vehicles on the road today are autonomous. Overall, the ability for the traffic signal to recognize upcoming vehicles from a distance was proven effective.

2.5 Multi-Intersection and Adaptive Signal Control for Traffic Optimization

SCOOT and SCAT traffic signal techniques have been used widely throughout traffic control for many decades. SCOOT is an optimization technique that incorporates a centralized system that measure traffic loads continuously [Luk 1984]. These measurements of traffic volumes adjust signal timings to minimize the average vehicle queue in specific areas per intersections [Stevanovic 2009]. Multiple details of the overall optimization include split timing, offset, and cycle length which provide smaller individual details for queue minimization. SCAT is an automated real time traffic responsive signal control strategy that incorporates local and regional computers [Stevanovic 2009]. Information from vehicle detectors regarding location is used to adjust signal timing based on the variation in traffic demand. Software program VISSIM is often used with this method and overall, signal timing is adjusted based on change in traffic flow which is monitored from the heuristic feedback system.

A connected vehicle research study based on an adaptive traffic signal in a mixed traffic stream was also completed [Khan 2019]. Connected vehicles (CV) are considered

mobile nodes that communicate with nearby vehicles (connected road users) and infrastructure traffic signals. The intersection signals use an algorithm to optimize the traffic flow and adapt the timing based on vehicle load through the intersection. Initially, traffic signal timing is estimated based on the number of connected vehicles at the intersection. As vehicles travel through the intersection, dynamic offsets based on the initial signal timing can be implemented from the vehicle data load [Khan 2019]. Finally, the green time interval can be adjusted from the queue load of vehicles in the red direction. Overall, the time a vehicle is stopped at the intersection (stopped delay) can be reduced through adaptive signal timing.

Reinforcement learning (RL) for adaptive traffic signal control was also an important addition to traffic signal technology. Reinforcement learning involves an agent that finds new ways to achieve a goal by interacting dynamically with the environment [Abdulhai 2003]. The agent will consider different situations and evaluate performance to determine the overall best sequence of actions to achieve the ideal goal in the most appropriate manner. Feedback signals aid the agent in determining the level of contribution for each situation. The research uses a Q-learning technique [Watkins 1989] to determine appropriate relationships between states, actions, and rewards given the interaction with the environment.

Reinforcement learning was also applied in a multi-agent system [Arel 2010]. In this research, two types of agents were considered. Outbound agents adjust traffic signals by considering the length of the queue at individual intersections. This is determined using

the longest queue first algorithm (LQF). Central agents are also considered which use a value function to alter status which is driven by local neighboring traffic conditions. Overall, a machine learning technique is implemented to approximate and determine the optimal decision [Arel 2010] through interactions between the different agents.

Finally, multi intersection autonomous vehicle interactions have been simulated based on distributed mixed integer linear programming (MILP) to enhance traffic flow at signalized intersections [Ashtiani 2018]. Using connected autonomous vehicles (CAV), intersections solve their own optimizations given vehicle information and communicate decisions to other autonomous vehicles. Using time for a vehicle to proceed through an intersection and distance to the intersection, the controller can create a list of subscribed vehicles to neighboring intersection to find the desired access time. Overall traffic flow is optimized given these calculations.

This research was also incorporated using optimal schedule of autonomous vehicle arrivals at intelligent intersections [Fayazi 2017]. Using the mixed integer linear programming technique (MILP), a live picture of traffic conditions can be created. Notifications per vehicle can be communicated to the upcoming intersections to determine arrival time of that vehicle. Considering all subscribed vehicles to the upcoming intersection, an optimal schedule for light time can be determined to minimize intersection delay while ensuring safety. The access distance of the vehicle to the intersection allows for further calculations of the desired arrival time to ensure vehicles

do not face extreme delays. Furthermore, safety is more improved through ensuring vehicles travel safely behind vehicles ahead given autonomous vehicle reaction time.

2.6 Potential for Communication Improvement

In general, studies have been completed through connecting vehicles to determine common traffic flow. This data is used to ultimately improve the traffic signal patterns. However, there is not a significant amount of research considering the communication

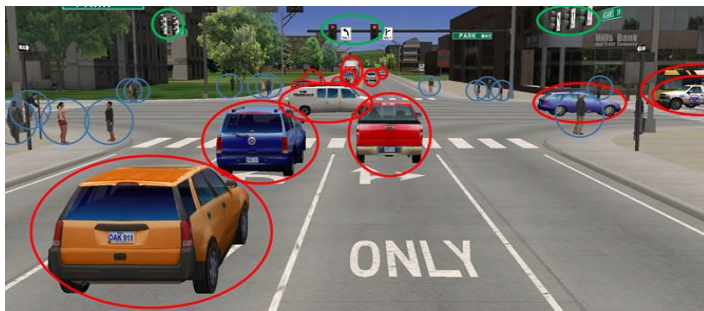


Figure 3: Full City Intersection Connectivity.

between both vehicles and traffic signals and treating each as an individual intelligent agent. This further includes a

lack in research of

communication between adjacent intersections (I2I). Including this newer form of communication can create improved traffic flow across a wider range of roads. Connecting vehicle flow through multiple intersections allows for more accurate status updates that can be used to improve both vehicle and intersection status decisions. Considering previous research regarding connected vehicle behavior and implementing optimization algorithms for the addition of connected traffic signals will allow for further improvement of intersection performance. A full network of agent-based communication between autonomous vehicles and intelligent traffic signals is a new study that will be discussed and proven to be advantageous to the transportation infrastructure. Eventually, this communication can be improved to full dynamic component connectivity

in a city intersection as shown in Figure 3 however, this research will only consider communication between autonomous vehicles and intelligent traffic signals.

3. Simulation Method

3.1 Research Campaign

The following tasks have been completed to demonstrate improved intersection performance from agent-based communication between autonomous vehicles and intelligent traffic signals:

- Introduce Autonomous Vehicles and Intelligent Traffic Signals as Agents;
- Create a MATLAB model of a realistic single fixed timing signal intersection;
- Create a grid of intersection situations with fixed timing signals and gather data of vehicle behavior to establish a baseline for improvement;
- Implement coordinated traffic signals to alter light status due to the status of neighboring intersections; and
- Improve overall intelligence through adaptive light signal timing based on nearby intersection light status and upcoming vehicle load to individual intersections.

Individual details about the tasks will be explained throughout the following chapter. Methods using AIA (Autonomous Intelligent Agents) are introduced and evaluations are completed to demonstrate improvement of intersection efficiency and safety.

3.2 Introduction of Autonomous Vehicle and Intelligent Traffic Signal Agents

3.2.1 *Defining an Agent Based Autonomous Vehicle*

The initial task consists of ultimately defining what is an agent based autonomous vehicle. The look, behavior, and interaction are qualities that must be addressed and

defined as these autonomous agents can interact with other vehicle agents, nearby intelligent traffic signal agents, as well as the environment. Each autonomous agent will have its own set of rules and the goal is to model the behavior of these agents to simulate potential interaction between autonomous vehicles to ultimately determine if these connected vehicles do improve safety and timing of traffic in a city.

Inputs, outputs, and individual behaviors for autonomous agents will be discussed. These qualities are important for determining a desired goal of an agent through an intersection and to provide output details to communicate these goals and current status. Environmental inputs to an agent are the heading direction to an intersection (d) and the desired exit direction (O). Upcoming vehicle locations (x_F) will also be considered for proper yielding. The uncertainty variables that are calculated per autonomous agent iteration are based on the vehicle instantaneous velocity (v_i) and distance from the intersection (x_i). This data will be used to predict the amount of time (t_{id}) a vehicle will take to proceed through the entirety of the intersection. Previous research emphasizes the importance of reaction time when considering braking times per distance to an intersection [McGehee 2000]. However, for this simulation, reaction time will be neglected as the autonomous agent velocity will be calculated in real time to simulate a connected vehicle environment. Technology with instantaneous feedback is more efficient in recognizing upcoming vehicle updates than the common human reaction time. Finally, an initial calculation of time to an upcoming intersection will be used as an input to the intelligent agent traffic signal.

3.2.2 Defining an Agent Based Intelligent Traffic Signal

The intelligent traffic signal agent is beneficial for gathering status data from nearby autonomous vehicles and neighboring light agents. An intelligent agent is also designed to communicate individual status to other nearby agents. The signal agents are designed to work together to improve traffic flow through individual intersections. With these goals in mind, qualities of traffic signal agents can be determined to ultimately lead to the creation of a world model for intersection improvement.

The initial control input parameter for the intelligent lights will be a fixed timing per light signal status. These timing values in seconds consist of the common green (t_G), yellow (t_Y), and red (t_R) light signals that are present today. The initial timing per green light will be adjusted for different simulations to evaluate traffic flow of a common intersection. For all simulations, the individual intersection setup consists of a one lane input and output per direction. Directions are limited to north, south, east, and west. The nearby light status (n) depending on which intersection it is receiving data from, will change with time based on the status of these adjacent intersections. This variable will only be used when multiple intersections are considered. The current light status (c) is a dependent output from an individual light that will be considered an input to both nearby vehicles and traffic signals. By considering the status of adjacent intersections, the status of a current traffic signal may be adjusted for improved traffic flow.

3.2.3 Coupled System of Autonomous Vehicles and Intelligent Traffic Signals

The main goal is to model an intersection that connects both autonomous vehicle and intelligent traffic signal agents. Figure 4 displays the ideal coupled system at an individual intersection when considering both an autonomous vehicle and an intelligent traffic signal each as an agent. Realistically, numerous nearby vehicles and traffic signals

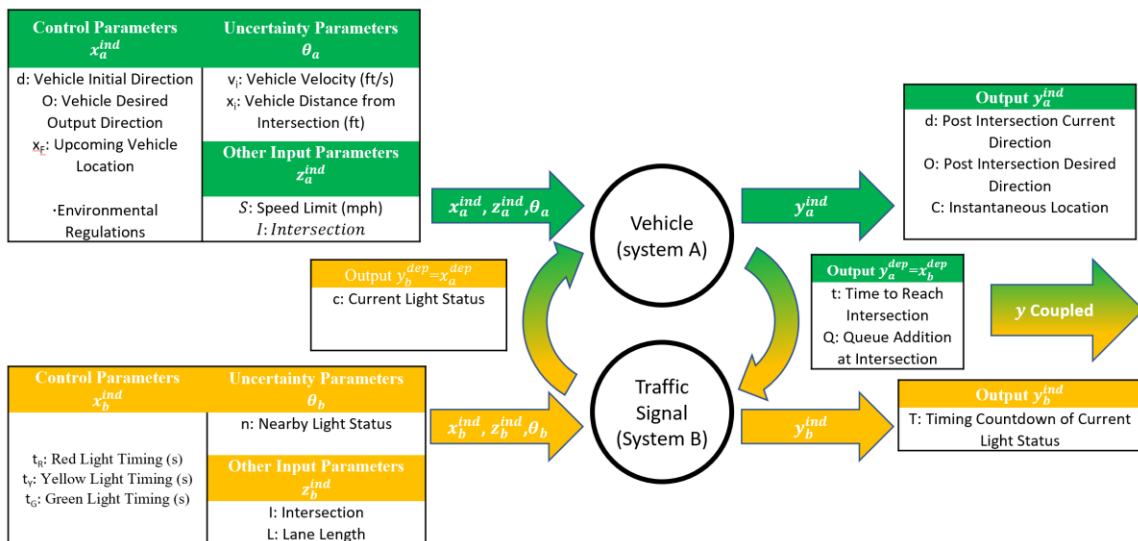


Figure 4: Coupled Agent-Based Behavior.

will be in simultaneous communication. However, for simplicity, initially only the interaction between one vehicle and light are considered.

The importance of this experiment is to ensure the autonomous vehicle agents and intelligent traffic signal agents are working in harmony. As displayed in Figure 4, autonomous agents can communicate approach time to an intersection when necessary. The intelligent light can then put that autonomous car in queue and determine if a potential light status change is necessary based on the load of vehicles currently waiting

at the light. The traffic signal will continuously gather nearby vehicle data to determine if a light status change is necessary. Furthermore, to allow two-way communication, intelligent traffic signals can relay light status to upcoming vehicles to ensure common traffic laws are obeyed.

3.3 The MATLAB Model

The following section will describe the process for constructing the MATLAB model and provide detailed explanations regarding individual code sections.

3.3.1 Modeling a Single Fixed Timing Signal Intersection

The MATLAB model has been created to run specific simulation scenarios on the behavior of autonomous vehicles in a smart city environment. The initial model was

created to demonstrate traffic flow at one individual intersection. Figure 5 displays the individual intersection model labeled with specific directional values for facilitated reference throughout this section. The direction number is based on the

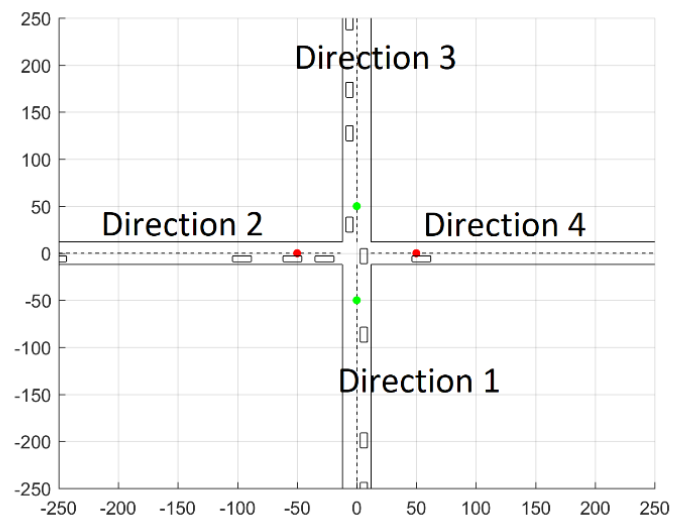


Figure 5: Individual MATLAB Simulation Intersection.

input or output location relative to the center of the intersection and it is assumed that all cars will travel on the right side of the road. The overall model is a fixed time step iteration-based code which calculates the desired acceleration of each individual vehicle

for appropriate movement throughout the simulation. This model is based on the behavior of a realistic vehicle. As time passes, drivers change positions relative to the traveled road in the desired direction. Vehicles will continue to move throughout the simulation until their desired destination is reached.

3.3.1.1 Important Intersection Inputs for Desired Behavior

The basic intersection was developed to allow for different input values into the simulation. The important inputs are displayed in Table 1 which include the MATLAB model variable name for reference, a range of potential values that can be chosen for common intersection performance, and the units of that value.

The number of vehicles in the simulation refers to the fixed number of vehicles that will always be active in the simulation. With a fixed number of vehicles per intersection, the departure of one vehicle will automatically place a new vehicle at an entrance point in the simulation. That new vehicle will have a different driver behavior compared to the exiting vehicle. The timestep is critical as the iteration for the movement of each vehicle is based on that specific time. Based on the given time in the table, each vehicle update will be completed 88 times to simulate 1 second in real time. The time to run the simulation is based on the desired length of running time in seconds. This number can be compared with the maximum number of cars to run through the simulation. The length of the simulation can either be based on the total number of vehicles that pass through the intersection or the desired length of time depending on which value is reached first.

Table 1: Important Simulation Inputs.

Input Variable	MATLAB Name	Common Value	Units
Overall Simulation Inputs			
Number of Vehicles in Simulation	CarsPerInt	5 – 60	-
Time of Simulation	TotalTime	30 - 300	seconds
Maximum Cars Through Simulation	MaxCars	500	-
Timestep for Vehicle Iterations	timestep	1/88	seconds
Individual Intersection Properties			
Speed Limit	SpeedLimit	20 – 60	mph
Lane Length to Intersection	LaneLength	400 – 1200	feet
Display Window	Window	250	feet
Light Timing Sequences			
Red Light Cycle Time	Red	0.5 – 2	seconds
Yellow Light Cycle Time	Yellow	2 – 5	seconds
Green Light Cycle Time	Green	10 – 60	seconds

The individual intersection properties are based on situations that may occur in everyday life. The lane length refers to the number of feet a vehicle must travel to an intersection center after indicating that it will be soon be arriving at that center. The lane length value (L) is also the location where new vehicles will be placed. Figure 6 displays a sample lane length value. The speed limit value can be chosen based on the desired simulation. In real world scenarios, the speed limit is determined from safe traffic

conditions. Typically, a shorter lane length will be paired with a slower speed limit due to cars having less distance to accelerate to higher speeds.

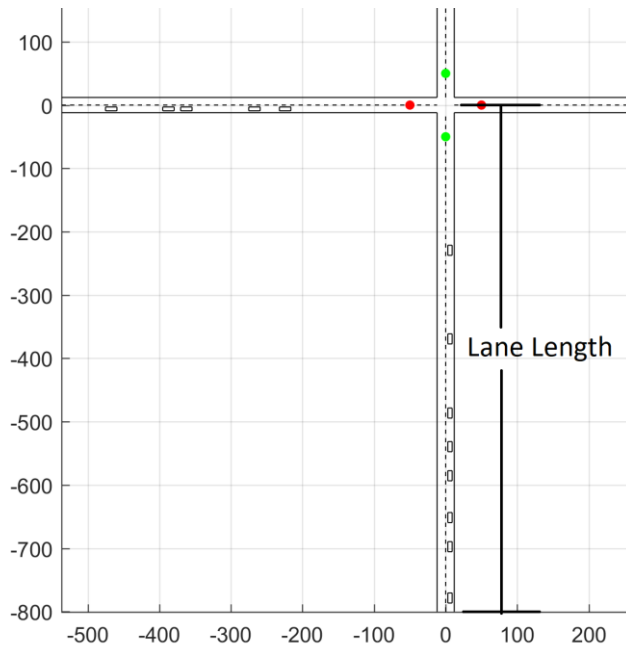


Figure 6: Lane Length Reference.

The light timing sequences are important for overall intersection performance evaluation. Based on this simulation, the red, yellow, and green light times make up the time in seconds given to a specific direction per intersection. Referring to Figure 5, directions 1 and 3 will have simultaneous green lights for the

given input time with 2 and 4 being held to red. The intersection behavior will then switch allowing vehicles from directions 2 and 4 to travel through for the duration of that common green light time. The yellow light time is generally determined by the speed limit. A common method for evaluating the appropriate yellow light duration is by dividing the speed limit in mph by 10. This will allow for a general approximation in seconds. Furthermore, the appropriate red-light time can be adjusted to account for vehicles that may arrive at the intersection slightly after their individual light changes from yellow to red (running a red light). This will ensure vehicles in the direction perpendicular to a previous yellow light will not proceed immediately through the intersection as some vehicles in the opposite direction may be clearing the intersection at higher speeds.

Finally, the red-light time can be adjusted to lengthier times to account for scenarios where all traffic is stopped to allow pedestrians to proceed through safely. Overall, these scenarios are not considered in this research.

3.3.1.2 Creating Vehicles with Random Behavior and Location Placement

To ensure few prior assumptions are made that could improve data outcome, individual vehicle behavior and initial placement in the simulation are randomized. This section is evaluated for the number of vehicles in the intersection. The process provides a random initial input and output direction combined with a random intensity rating per vehicle.

To compare a realistic example of a vehicle scenario that may occur, the directions from Figure 5 will be considered. Given directions 1 – 4, a random input direction is initially determined with the output direction being a random value in that same range but neglecting the previously determined input. U turns are not considered in this scenario. From the input and output direction, it can be determined the type of turn that will be made by the individual vehicle (ex. 1 to 4 is a right turn, 3 to 1 is straight, etc.). Further calculations can be completed from knowing the upcoming direction desired.

The random intensity rating is important as it determines the type of individual driver behavior. Throughout the entire driving population, there is a wide range of various driving behaviors that are present on roads today. For simplicity, only 10 potential but common options are considered. The intensity rating parameter determines the driver desired speed of travel relative to the speed limit, the desired acceleration or deceleration

given more intense drivers tend to change velocities at a more rapid rate, and the desired headway a vehicle traveling directly in front. Less headway time accounts for a driver whom is more likely to tailgate. The potential intensity ratings that can be implemented on a scale from 1 to 10 are displayed in Table 2. An intensity rating of 5 is a driver whom is considered neither too cautious nor aggressive.

Table 2: Intensity Rating Behavior.

Intensity Rating	Desired Speed (mph)	Acceleration (ft/s)	Headway (s)
1	Speed Limit – 4	6	3.0
2	Speed Limit – 3	7	2.8
3	Speed Limit – 2	8	2.6
4	Speed Limit – 1	9	2.4
5	Speed Limit	10	2.2
6	Speed Limit + 1	12	2.0
7	Speed Limit + 3	14	1.9
8	Speed Limit + 5	16	1.6
9	Speed Limit + 7	18	1.3
10	Speed Limit + 10	20	1.0

Individual drivers will travel at their own desired speed which may be altered due to vehicles in front traveling at a lower speed. In this scenario, the vehicle will react to the slower speed and adjust to the desired time behind given the intensity rating. A simple kinematics equation is used to determine the headway of a vehicle in seconds (t) given the distance from the upcoming vehicle (d) and the velocity of the current vehicle (v).

$$t = \frac{d}{v} \quad (3.1)$$

After receiving data regarding the initial input direction of the vehicle to the intersection and the intensity rating, a random placement in that direction is determined.

If no vehicles have been placed in a specific direction, a vehicle can be placed randomly throughout a range of 200 feet from the center of the intersection to the lane length. Finally, to ensure vehicles are not placed in similar locations, a random location placement is found in that overall range neglecting a smaller range of ± 20 feet from an originally placed vehicle.

3.3.1.3 Vehicle Acceleration Calculation

Updated behavior parameters calculated from the previous iteration to be used for the current iteration include the current position (x_i) and velocity (v_i) of an individual vehicle. The main objective of each iteration per car is to calculate the appropriate acceleration given the situation. Four main types of accelerations are calculated per iteration and the most appropriate acceleration is implemented in the final car movement calculation. The accelerations are highlighted in Table 3 and explained in detail throughout this section.

Table 3: Individual Iteration Acceleration Options

Distance Acceleration	Maintaining an appropriate following distance behind a car given the desired headway time.
Light Status Acceleration	Determining the appropriate acceleration given no cars ahead, the current light status, and an intent to proceed straight through the intersection
Right Turn Acceleration	Calculated instantaneous acceleration given no cars to impede upcoming progress and a desire to make a right turn at the upcoming intersection
Left Turn Acceleration	Calculated instantaneous acceleration given no cars to impede upcoming progress and a desire to make a left turn at the upcoming intersection

Distance Acceleration

It is critical for a vehicle to have the ability to accelerate and decelerate when a vehicle in front is present. Maintaining a safe distance is extremely important to ensure the following vehicle can slow down appropriately to avoid contact with the vehicle in front in an emergency stop situation. Therefore, based on the intensity rating assigned to an individual vehicle, the headway of a car in front will be maintained based on the actions of the preceding vehicle.

Today, many vehicles are equipped with adaptive cruise control. This technological improvement allows the following car to maintain a safe distance from the lead vehicle given a selected following distance. The 2018 Honda Accord has the option to select four following distances when using this feature [Honda 2018]. This is like the intensity rating headway feature as more aggressive drivers will tend to choose a following distance that is closer compared to a cautious driver. Figure 7 displays the different distance/headway selection options from the 2018 Honda Accord owner's manual [Honda 2018]. The Accord options are consistent with the headway values from the intensity rating.





Following Interval		When the Set Speed is:	
		50 mph (80 km/h)	65 mph (104 km/h)
Short		83 feet 25 meters 1.1 sec	100 feet 31 meters 1.1 sec
Middle		110 feet 33 meters 1.5 sec	137 feet 42 meters 1.5 sec
Long		154 feet 47 meters 2.1 sec	200 feet 61 meters 2.1 sec
Extra Long		207 feet 63 meters 2.9 sec	272 feet 83 meters 2.9 sec

Figure 7: Honda Accord Adaptive Cruise Control Options.

Headway time is used to evaluate the appropriate following distance due to consistency given different velocities of a vehicle. From the figure, it is evident that the following distance in meters is greater for higher speeds but the time behind remains the same. This is due to a fixed ratio from equation 3.1. Rearranging the equation to solve for distance ($d = vt$) gives the product of the instantaneous velocity and the desired headway time to determine the appropriate following distance. The radar sensor on the vehicle can then detect the instantaneous distance from the vehicle in front and adjust the velocity of the vehicle to minimize the error between the ideal and actual distance.

This research uses the series of steps displayed in Figure 8 to ultimately arrive at the appropriate distance acceleration. Eventually, if statements in the time domain are used to calculate the appropriate acceleration per iteration given the instantaneous headway. The current and initial slowing down time are first evaluated. The current time (t_c) behind is

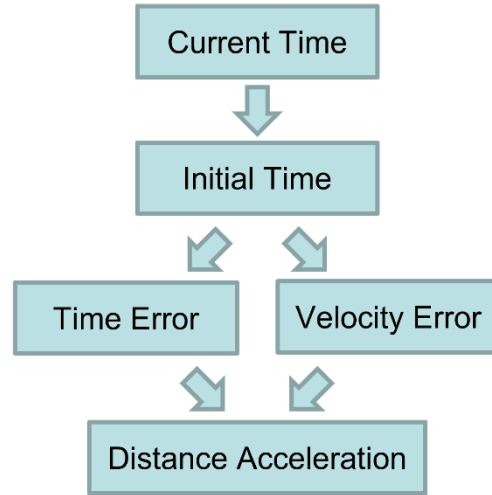


Figure 8: Distance Acceleration Process.

measured as described in equation 3.1 relating the distance and velocity. The initial time (t_i) is calculated as the appropriate headway time (current time) to begin slowing down at the desired acceleration to reach the desired time behind at a similar velocity of the preceding vehicle. For consistent units, all velocities are calculated in ft/s and accelerations in ft/s^2 . The real time (t_{eq}) it takes for the behind vehicle to reach the same speed as the front vehicle given the front vehicle velocity (v_{Fi}), the initial ideal speed of the behind vehicle (v_{Bideal}), and the ideal acceleration of the behind vehicle (a_{Bid}) is calculated as follows:

$$t_{eq} = \frac{v_{Fi} - v_{Bideal}}{-a_{Bid}} \quad (3.2)$$

The final position of the front car (x_{Ff}) is then found using the front vehicle initial position (x_{Fi}) and the time (t_{eq}) value previously calculated:

$$x_{Ff} = x_{Fi} + v_{Fi} * t_{eq} \quad (3.3)$$

The final position of the rear vehicle (x_{Bf}) is then found relative to the front vehicle position x_{Ff} using the desired final time behind (t_f) and the desired final speed which is equivalent to the front vehicle assuming no velocity change.

$$x_{Bf} = x_{Ff} - v_{Fi} * t_f \quad (3.4)$$

The real vehicle initial position for slowing down (x_{Bi}) is finally calculated.

$$x_{Bi} = x_{Bf} - v_{Bideal} * t_{eq} + \frac{1}{2} * a_{Bid} * t_{eq}^2 \quad (3.5)$$

The initial time behind (t_i) to begin slowing down is then found by using the quadratic formula by relating the positions between both vehicles, the initial velocity of the rear vehicle, and the ideal acceleration.

$$0 = x_{Bi} - x_{Fi} + v_{Bideal} * t_i + \frac{1}{2} * a_{Bid} * t_i^2 \quad (3.6)$$

Two solutions for t_i will be presented after using this formula but the most appropriate answer is selected. The error between the desired time (t_f) and current time (t_{ci}) is evaluated as well as the error between the velocity of the rear (v_{Bi}) and front car. The equations are used in the if statement logic to determine the most appropriate acceleration.

$$t_{error} = t_{ci} - t_f \quad (3.7)$$

$$v_{error} = v_{Bi} - v_{Fi} \quad (3.8)$$

A potential acceleration (a_p) value is calculated to arrive at the appropriate t_f based on relative positions between the two cars and the current velocity. This acceleration is generally used when $t_{ci} \approx t_f$ or $t_{ci} - t_{ci-1} \approx 0$ but relative velocities are still

different. The answer will be accurate based on the final sign (acceleration > 0, deceleration < 0),

$$a_p = \frac{2 * (x_{Fi} - x_{Bi}) - v_{Bi} * t_f}{t_f^2} \quad (3.9)$$

Finally, the appropriate distance acceleration is determined. The logic to determine the correct decision is displayed in Table 4. Previous calculations of t_{ci} are used to understand how the relative headway time is changing between iterations. One g is equal to gravitational acceleration (32.2 ft/s²).

Table 4: Distance Acceleration Decision Logic.

Current < Initial	Current > Final	Back Vel > Front Vel	Back Vel < Ideal Vel	Gaining Time	Acceleration Choice
$t_i - t_c$	$t_{ci} - t_f$	$v_{Bi} - v_{Fi}$	$v_{Bideal} - v_{Bi}$	$t_{ci-1} - t_{ci}$	Acceleration
+	+	+	+	± 0.0001	a_p
+	+	-	+	± 0.0001	$-a_p$
+	+	+	+	+	$-a_{Bid}$
+	+	+	+	-	$-a_p$
+	+	-	+	-	a_{Bid}
+	± 0.01	+	+	± 0.0001	a_p
+	± 0.01	-	+	± 0.0001	$-a_p$
+	± 0.01	+	+	+	$-a_{Bid}$
+	± 0.01	+	+	-	$-a_{Bid}$
+	± 0.01	-	+	-	a_{Bid}
$-\infty$	∞		+		a_{Bid}
-	+		+		a_{Bid}
+	-		+		$-2 * a_{Bid}$
+	$\approx < -0.5$		+	+	$-1g$
-	+		± 0.01		0

This acceleration is considered the distance acceleration throughout the code. It is solely based on a vehicle in front of the currently evaluated vehicle. This acceleration is used frequently throughout the simulation.

Light Status Acceleration

The light acceleration calculation is directly formed from the status of the upcoming traffic signal. This light only has the option of being green, yellow, or red. Therefore, calculations regarding the light acceleration are based on these three status. Variables discussed in the distance acceleration section used B and F to refer to the behind and front vehicle. New variables will neglect the capitalized letters as only individual vehicles will be considered.

The green light acceleration calculation is extremely simple. Considering a goal to arrive at the light without traffic conditions impeding progress, the vehicle will move at its desired speed until it is in a certain range from the upcoming intersection. Details regarding this range will be explained in upcoming paragraphs. Overall, for a green light scenario, the light acceleration will only be the ideal acceleration (a_{id}) given the vehicle is traveling slower than its desired speed. A vehicle can travel through the light at its desired speed without having to consider a slower pace for an upcoming turn.

The main calculation for this type of acceleration is finding the appropriate distance from the intersection that a vehicle should begin to slow down if necessary. This calculation is based on the number of cars between a specific vehicle and the intersection

(C_i), the current velocity (v_i), and the ideal acceleration rate (a_{id}). A vehicle is considered through the intersection if the front of the vehicle has entered the intersection.

The following equation is used to determine the ideal straight slow down point for a vehicle (x_s) heading towards an intersection. In this given equation, the vehicle is traveling in the positive direction 2 (west to east) therefore, the slow down point value will be numerically less than the center coordinate of the intersection (0,0). The fixed car length throughout the simulation is 16 feet therefore, the constant 25 is used to approximate a 9-foot distance between stopped vehicles waiting for a green light. The exact coordinate point of the vehicle in direction 2 is measured from the back of the vehicle and the limit line for the intersection entrance is at $x = -12$. Given the fixed ending point of -35 ft from the back of the vehicle, the vehicle front coordinate will be -7 ft from the entrance of the intersection. Given these initial values, the ideal deceleration point can be calculated. Vehicles will only consider the light acceleration option if the current position of the car (x_i) is greater than point x_s .

$$x_s = -35 - 25 * C_i - \frac{v_i^2}{2 * a_{id}} \quad (3.10)$$

Situations may occur where a vehicle is determined to decide if it can proceed through an intersection from a changed yellow light. Those scenarios will be discussed in the following section. Only two scenarios may occur regarding a yellow light status assuming the vehicle has not decided to proceed rapidly through the intersection. First, if the vehicle position is of a lower value compared to location x_s , it will continue to travel

with its current behavior until $x_i \geq x_s$. At this point, the light acceleration will equal a_{id} until the vehicle has come to a complete stop. This same equation can be applied to a red light at the upcoming intersection. The second scenario is if a vehicle has decided not to proceed through the intersection but, has a position greater than x_s . A new equation is used to determine the appropriate acceleration regarding the straight light status (a_s) given previously discussed parameters.

$$a_s = \frac{-v_i^2}{2(-35 - 25 * C_l - x_i)} \quad (3.11)$$

Right Turn Acceleration

The right turn acceleration option is calculated assuming a vehicle wants to turn right at the upcoming intersection. To simplify the scenario, no right turns on a red light are allowed throughout the simulation. A vehicle may only proceed right on a green or yellow light. A red-light scenario for this acceleration is like the previously discussed light status acceleration.

Given a green light at the upcoming intersection with the desire to make a right turn, the vehicle must decelerate to an appropriate velocity to avoid vehicle roll during the turn. This turning velocity (v_T) has a fixed value of 10 mph throughout the simulation regardless of the vehicle intensity rating. For consistency, the same direction 2 will be considered. Using similar values from the light status acceleration combined with the turning velocity of 10 mph (14.667 ft/s) being reached at the limit line of the intersection, the ideal right turn slowing point (x_R) is calculated using equation 3.12.

$$x_R = -28 - \frac{v_i^2 - v_T^2}{2 * a_{id}} \quad (3.12)$$

The same equation can also be used for a yellow light given the car can proceed through the light with normal behavior due to no interactions with other vehicles. If this basic scenario arises, the acceleration due to a right turn will be $a_R = a_{id}$.

Left Turn Acceleration

The left turn acceleration is like the right turn acceleration. The main differences are the position where v_T begins and the potential to yield to vehicles traveling in the opposite direction. The calculation for the left turn initial starting point given ideal acceleration (x_L) is based on the front of the vehicle having traveled 8 ft into the intersection ($x = -4$). To consider the exact location of the vehicle measured from the back, the ideal arrival coordinate with the velocity reaching v_T is at -20. The calculation for x_L uses the following equation:

$$x_L = -20 - \frac{v_i^2 - v_T^2}{2 * a_{id}} \quad (3.13)$$

The same equation can also be used for a yellow light given the car is able to proceed through the light with normal behavior due to no interactions with other vehicles or having to rush through the intersection to beat the red light. If this basic scenario arises, the acceleration due to a left turn will be $a_L = a_{id}$.

Furthermore, vehicles that intend to turn left are required to yield to oncoming traffic traveling in the opposite direction. The criteria for a vehicle to proceed across the oncoming traffic lane is based on the opposite vehicle ability to proceed through the

intersection given an immediate green to yellow light change. The calculations to determine if a vehicle can proceed through the intersection without altering the behavior given a sudden yellow light will be discussed in the next section.

As previously discussed, the ideal turn speed v_T of 10 mph and front of the vehicle at $x = -4$ ft from the center of the intersection is the goal of a vehicle with intentions to turn left. However, in some cases, the vehicle may need to completely stop due to oncoming traffic in the opposite direction. The x coordinate where the vehicle will officially begin the left turn is when the front of the vehicle reaches $x = 6$. This allows the vehicle about 10 ft to perform a velocity change of -10 mph. In the event of a more rapid stop due to a last second decision to not proceed through the oncoming traffic lane, an acceleration is calculated for a vehicle to reduce its speed to 0 mph at a location slightly before the front of the vehicle reaches the turning point of $x = 6$ (ex. $x = 5$) or through measuring from the rear of the vehicle at $x = -11$. This equation will ensure the vehicle reaches 0 velocity to safely wait for oncoming traffic to pass through the intersection before completing the turn.

$$a_L = \frac{v_i^2}{2(x_i + 11)} \quad (3.14)$$

3.3.1.4 Through Intersection Calculations

Calculations in the previous section were based on direct positions of the vehicles relative to the intersection and the desired final velocities. The following equations based on the desired turn are calculated to determine if the vehicle at the current speed will

make it through the intersection if the light were to change immediately from green to yellow. Considering the same direction 2, if the current x_i position is greater than the through intersection calculation position, it is highly likely the vehicle will make it through given an immediate yellow light change. The equations are evaluated only with a green light at the upcoming intersection. An additional value used is the yellow light time (t_Y). The following equations calculate the passing position (through point) based on ideal behavior for a straight (x_{PS}), right (x_{PR}), and left (x_{PL}) turns.

$$x_{PS} = -28 - v_{id} * t_Y \quad (3.15)$$

$$x_{PR} = -22 - v_{id} * t_Y + \frac{1}{2} * a_{id} * t_Y^2 \quad (3.16)$$

$$x_{PL} = -16 - v_{id} * t_Y + \frac{1}{2} * a_{id} * t_Y^2 \quad (3.17)$$

These equations are used to determine if a vehicle will pass through the intersection and to provide opposite direction vehicle status to confirm if a vehicle can proceed through an unprotected left turn. If no cars in that opposite direction have reached the through point, the yielding vehicle can proceed safely across the lanes of traffic. Figure 9 displays a comparison between the straight (x_{PS}), right (x_{PR}), and left (x_{PL}) turns values for a vehicle with an intensity rating of 5. It is clear from the graph that a vehicle traveling straight can be the furthest distance away from the center of the intersection but still make the light given a potential yellow light change. This is because no deceleration is required for a vehicle proceeding straight assuming no nearby vehicles are impeding the progress. Vehicles turning left and right are required to slow down to complete the turn safely.

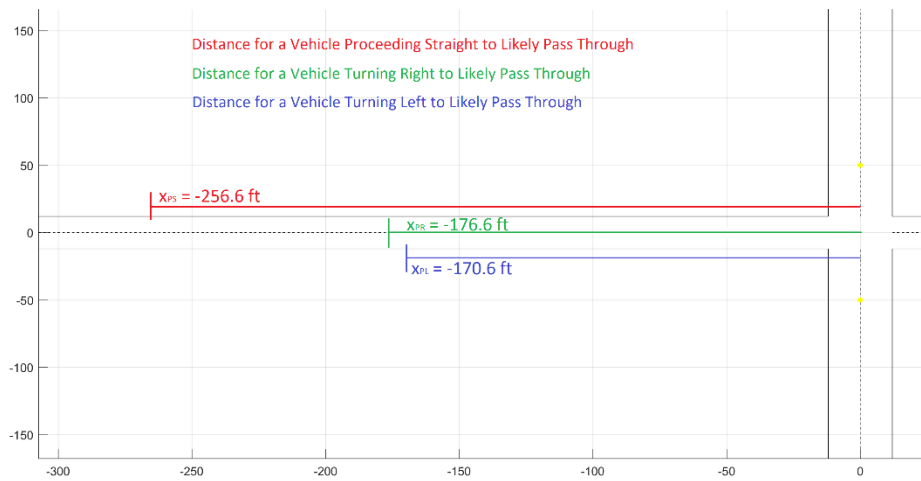


Figure 9: Distance from Intersection to Likely Proceed Through Based on Desired Turn.

3.3.1.5 Yellow Light Decision Logic

In realistic scenarios, drivers are required to make split second decisions when a nearby upcoming light immediately becomes yellow. To improve the decision making of an individual vehicle, calculations are performed using the instantaneous vehicle position, velocity, desired acceleration, and the fixed yellow light time. There are three main decision options for a vehicle. These options are briefly displayed in Table 5 and will be explained in detail throughout this section.

Table 5: Yellow Light Decision Options.

Vehicle Options	Options Description
Sustained Behavior	The vehicle will maintain its original behavior as if there was no yellow light update.
Boosted Behavior	Based on the upcoming turn, the vehicle will accelerate or decelerate more rapidly than normal to avoid running the red light.
Missed Light	The vehicle will slow down behind the limit line due to no possibility of proceeding through safely and legally.

Sustained Behavior

A vehicle will sustain its initial plan upon making this decision. This decision is based on the vehicle being close enough to the upcoming intersection that it can travel through with ease. The following calculations detail the logic for this decision based on post intersection behavior.

For a vehicle proceeding straight, a common kinematics equation is used to determine if the vehicle will clear the intersection based on the current velocity and distance from the limit line. All calculations assume the car is traveling from direction 2 west to east and the absolute position of the vehicle is measured from the rear. If the vehicle position is greater than this calculated value, the vehicle can proceed through by maintaining this current speed. The equation to determine the minimum position relative to the intersection for proceeding straight at normal behavior (x_{SS}) is as follows:

$$x_{SS} = -28 - v_i * t_Y \quad (3.18)$$

The calculations considering a vehicle making a right or left turn at an intersection are different for the necessity of slowing down. The decision logic is similar when turning with the only exception being the numerical target line approach for the appropriate turning velocity therefore, only the right turn scenario will be discussed in detail.

The main criteria to determine if a vehicle can maintain its current path upon encountering a yellow light is its ability to reduce speed to make a safe turn in the time it takes for the light to turn red. Therefore, a timing calculation (t_1) is required based on the

initial velocity, the desired turning speed (10 mph), and the desired deceleration rate from the initial intensity rating.

$$t_1 = \frac{v_i - v_T}{a_{id}} \quad (3.19)$$

This time is then evaluated against the fixed yellow light time at that intersection. This time value is required to be less than the yellow light time or the vehicle cannot realistically complete the maneuver. This evaluation will determine how much extra time is available (t_{1e}) when relating the yellow light time. A positive answer is required to continue with the procedure.

$$t_{1e} = t_Y - t_1, \quad t_{1e} \geq 0 \quad (3.20)$$

With a positive evaluation of variable t_{1e} , the minimum position of the vehicle relative to the intersection can now be calculated. The current velocity, desired acceleration, and yellow light time are used. With direction 2 being the input, the current position of the vehicle must be greater than the calculated position to maintain the current vehicle behavior through the intersection. The calculated position to sustain behavior (x_{SR}) for a right turn is as follows:

$$x_{SR} = -28 - v_i * t_Y + \frac{1}{2} * a_{id} * t_1^2 \quad (3.21)$$

In this equation, the velocity term uses the light time as a timing reference while the acceleration term uses the t_1 term calculated previously. This is to allow a further maximum distance from the intersection due to the shorter amount of time it will take to slow down. If a vehicle required more overall time to slow down to the turning speed v_T ,

the allowable minimum distance to the intersection would be smaller as the vehicle would be traveling at a slower speed throughout a longer duration of the deceleration process. This equation indicates that if the current position of the vehicle was perfectly at the minimum position to proceed through the light ($x_i = x_{SR}$), the vehicle would decelerate throughout the total length of the yellow light barely proceeding through to avoid running the red light. If the yellow light time is greater than the time taken to slow down to the turn speed $t_{1e} > 0$, the vehicle could travel at its initial velocity for the length of time t_{1e} and begin the deceleration at a specific position related to the distance required to change velocities based on the desired deceleration rate. This specific decision point (x_{SD}) for the sustained behavior is calculated using the following equation.

$$x_{SD} = -28 - \frac{v_i^2 - v_T^2}{2 * a_{id}} \quad (3.22)$$

For a vehicle making a right turn, this position is directly related to variable x_R which also determines the point at which the vehicle begins to decelerate given the desired deceleration rate. The initial time evaluation $t_{1e} \geq 0$ simply checks that this maneuver can be completed in an appropriate amount of time. The distance equations and process for a left turn are like the right with an adjustment solely based on the final position relative to the intersections. The equations are shown below.

$$x_{SL} = -16 - v_i * t_Y + \frac{1}{2} * a_{id} * t_1^2 \quad (3.23)$$

$$x_{SD} = -16 - \frac{v_i^2 - v_T^2}{2 * a_{id}} \quad (3.24)$$

Boosted Behavior

If the yellow light time is shorter than the required time for the vehicle to decelerate to the turning speed $t_{1e} < 0$ or if the current location of the vehicle is less than the minimum distance to the intersection assuming direction 2 and a right turn scenario ($x_i < x_{SR}$), the boosted acceleration/deceleration option is considered. A common driver determined to proceed through the intersection upon encountering a yellow light generally will perform a more rapid maneuver. To simulate this scenario, a boosted behavior for a vehicle proceeding straight will be explained first.

The minimum position for a vehicle proceeding straight is based on the initial velocity of a vehicle and the ideal acceleration. Generally, a vehicles acceleration ability is lower at higher speeds due to a decreased amount of torque. To account for this mechanical disadvantage, half of the ideal acceleration will be applied. The equation to determine the minimum position value for a vehicle to proceed straight through the intersection with a boosted acceleration is as follows:

$$x_{BS} = -28 - v_i * t_Y - \frac{1}{4} * a_{id} * t_Y^2 \quad (3.25)$$

If the vehicle position is greater than this boosted value calculation but less than the maintained position value, ($x_{BS} \leq x_i \leq x_{SS}$), an acceleration half of the ideal value will be applied to the vehicle until it clears the intersection.

Furthermore, the boosted behavior for a turning vehicle must be considered. Similar to the maintained vehicle behavior, the right and left turning values have a similar process therefore, only the right turn process will be described. The main addition to the

upgraded driving intensity is the ability to decelerate at a more rapid rate than normal. Given a typical driver deceleration rate based on the intensity rating, the improved rate is 3 ft/s² faster than the initial. This allows for more time at higher speeds and for a quicker velocity change reducing to the appropriate turn speed.

Similar to the maintained behavior, a timing calculation (t_2) is required to determine how long it takes for a vehicle to reduce its current speed to the desired turning speed (10 mph). The equation is as follows:

$$t_2 = \frac{v_i - v_T}{a_{id} + 3} \quad (3.26)$$

In the boosted calculation, the velocity difference is the same however, the deceleration is numerically 3 more ft/s² than the ideal. This new time value can be compared to the yellow light signal time.

$$t_{2e} = t_Y - t_2, \quad t_{2e} \geq 0 \quad (3.27)$$

The numerical value t_{2e} must also be smaller than t_Y to avoid proceeding through the intersection on a red light. With a positive evaluation of variable t_{2e} , the minimum position of the vehicle relative to the intersection to make the light can now be calculated. Referencing direction 2, the current position must be greater than the calculated decision position to allow opportunity for the improved deceleration through the intersection. The calculation to determine the minimum location (x_{BR}) for a boosted right turn is as follows:

$$x_{BR} = -28 - v_i * t_Y + \frac{1}{2} * (a_{id} + 3) * t_2^2 \quad (3.28)$$

If $x_{SR} \geq x_i \geq x_{BR}$, and $t_{2e} \geq 0$, the vehicle will perform the boosted deceleration maneuver. Due to a quicker deceleration process that requires less time to complete, the vehicle can maintain the higher initial speed for a longer duration of time. This will ultimately allow for a further minimum point compared to the sustained behavior option.

Finally, given a more rapid deceleration rate, a calculation to determine the location where the decision is made to begin slowing down (x_{BD}) is carried out.

$$x_{BD} = -28 - \frac{v_i^2 - v_T^2}{2 * (a_{id} + 3)} \quad (3.29)$$

The vehicle will initially maintain speed (v_i) until the rear part of the car reaches coordinate x_{BD} . The improved deceleration rate will then be implemented to reduce the initial velocity down to the safe turning speed.

The equations for making a boosted left turn are like that of a right. They are listed as the following:

$$x_{BL} = -16 - v_i * t_Y + \frac{1}{2} * (a_{id} + 3) * t_2^2 \quad (3.30)$$

$$x_{BD} = -16 - \frac{v_i^2 - v_T^2}{2 * (a_{id} + 3)} \quad (3.31)$$

Missed Light

If neither the sustained nor boosted behavior can be implemented based on the distance from the intersection or from the lengthy time required to slow down, the vehicle will miss the light and be prohibited from proceeding through the intersection. At this stage, the vehicle will perform the previously discussed light deceleration process to ultimately arrive at stopping point x_s which is determined by the number of vehicles in

front. The overall criteria for the yellow light decision logic regarding the 3 options is displayed in Table 6.

Table 6: Yellow Light Decision Logic.

	Straight	Right	Left
Sustained	$X_i \geq X_{SS}$	$X_i \geq X_{SR}$	$X_i \geq X_{SL}$
Boosted	$X_{SS} > X_i \geq X_{BS}$	$X_{SR} > X_i \geq X_{BR}$	$X_{SL} > X_i \geq X_{BL}$
Missed	$X_i < X_{BS}$	$X_i < X_{BR}$	$X_i < X_{BL}$

3.3.1.6 Vehicle Acceleration Decision and Instantaneous Behavior Update

After calculating all potential acceleration options of a vehicle, the most appropriate decision can be made given the circumstances. This logic is the main evaluation for an autonomous agent. The agent will continue to travel at its desired behavior ($v_i = v_{id}$, $a_i = 0$) until it either recognizes an upcoming autonomous agent or if it is approaching an upcoming traffic signal.

Overall in any scenario, if a vehicle is following another, it will always yield appropriately based on the headway. This is a specific type of interaction with another agent. This detail is fixed due to only one lane of traffic per direction. Until a front vehicle is removed from that direction through a different turning agenda, an autonomous agent will continue to follow at the fixed headway value.

Alternative scenarios include yielding to the environment which considers upcoming turns or engaging yellow and red lights. While referencing direction 2, the ultimate acceleration decision per vehicle based on the current scenario is displayed in Table 7. The criteria for the decision is based on the current time behind a

Table 7: Appropriate Acceleration Decision.

Headway Time	Vehicle Position	Vehicle Velocity	Light Status	Vehicle Output	Acceleration Decision
$t_c \approx t_f$	$x_i \geq x_d$	$v_i < v_{id}$	G/Y/R	S/R/L	-
Light Status and Vehicle Output Direction Not Considered					
True	True	True	-	-	Distance
True	False	True	-	-	Distance
True	False	False	-	-	Distance
False	False	False	-	-	Distance
False	False	True	-	-	Distance
Green Light Scenarios					
False	True	False	Green	Straight	Distance
False	True	False	Green	Right	Right
False	True	False	Green	Left	Left
False	True	True	Green	Straight	Distance
False	True	True	Green	Right	Right
False	True	True	Green	Left	Left
Yellow Light Scenarios					
False	True	False	Yellow	Straight	Light
False	True	False	Yellow	Right	Right
False	True	False	Yellow	Left	Left
False	True	True	Yellow	Straight	Light
False	True	True	Yellow	Right	Right
False	True	True	Yellow	Left	Left
Red Light Scenarios					
False	True	False	Red	Straight	Light
False	True	False	Red	Right	Light
False	True	False	Red	Left	Light
False	True	True	Red	Straight	Light
False	True	True	Red	Right	Light
False	True	True	Red	Left	Light

vehicle (t_c), current vehicle speed (v_i), current vehicle position (x_i), vehicle output (O), and the upcoming light status (c). For simplicity, (x_d) will be the variable related to the point at which a vehicle will begin interacting with the upcoming intersection regardless of the desired directional output. The light status table abbreviation considers G/Y/R as Green,

Yellow, and Red. The vehicle output table abbreviation S/R/L refers to Straight, Right, and Left turns. The acceleration options are referred to as distance acceleration (Distance), Straight Light Status Acceleration (Light), Right Turn Acceleration (Right), Left Turn Acceleration (Left). At this stage of the MATLAB code, all numerical values have already been calculated based on the previous equations. The logic here is to identify the most appropriate acceleration choice.

After determining the appropriate acceleration (a_i) for the current iteration, given known input position (x_i) and velocity (v_i) parameters, kinematic equations can be used to determine the new position (x_{i+1}) and velocity (v_{i+1}). The kinematic equations are

$$x_{i+1} = x_i + v_i * t + \frac{1}{2} * a_i * t^2 \quad (3.32)$$

$$v_{i+1} = v_i + a_i * t \quad (3.33)$$

where the timestep (t) is equal to 1/88 seconds for this simulation. These newly calculated position and velocity values are used for the same vehicle in the next iteration and the entire process beginning at section 3.3.1.3 is repeated.

3.3.2 Intersection Grid with Fixed Timing Signals

Upon completion of the basic intersection model, individual intersections are then connected to create a city environment. The initial evaluation of traffic flow through the intersection is evaluated using fixed timing signals. This objective is critical for understanding the status and behavior of previously constructed intersection or city models. Once the behavior of a basic city environment can be evaluated, approaches to

improve the efficiency of these models while ensuring a safer environment for the transportation infrastructure can be determined.

3.3.2.1 Background Evaluation of Basic City Intersection Behavior

Initially, basic intersection modeling was carried out. A simple intersection model has been created with vehicles passing through the intersection based on light status. Next, the basic intersection model was scaled to simulate a city environment with fixed timing signals. This is important so the efficiency of the grid setup and an average vehicle time through individual intersections can be calculated. Overall, these parameters will be calculated given fixed green light signal times to establish a baseline for improvement in each city grid scenario.

3.3.2.2 Additional MATLAB Modeling for Connected Intersections

The initial MATLAB code was created to simulate vehicles traveling through one specific intersection. To consider a city scenario, individual intersections are placed in new locations throughout a mapped area with one lane road transitions connecting each adjacent intersection. An example of a 2x3 intersection setup is shown in Figure 10. The center of each individual intersection was

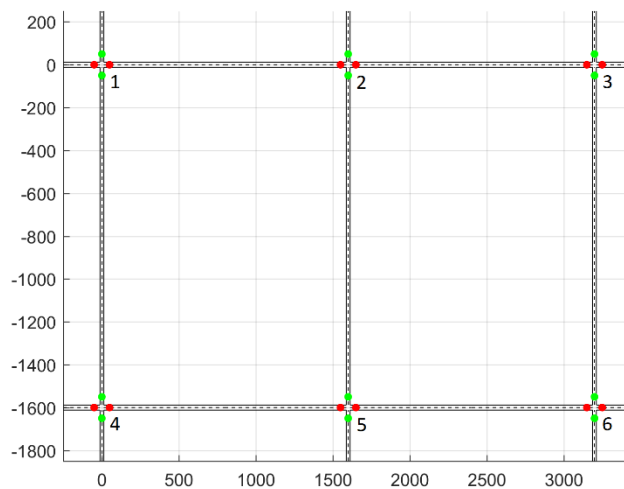


Figure 10: 2 x 3 Intersection Grid.

placed at a specific (x_i, y_i) coordinate on the map based on the lane length (L) of the intersection, the intersection grid row (R) count and intersection column (C) count. Furthermore, the intersection number (I) is established as well as the direction (d) per intersection. The single intersection 1 used direction numbers 1 – 4 to establish north, south, east, and west surrounding the intersection; intersection 2 will possess directions 5 – 8. This numbering process will continue for the total amount of intersections in the grid $(N = R * C)$.

The center location of the individual intersection is based on the lane length (L) . In Figure 10, $L = 800$ ft. Due to the intersections all possessing the same length, the location of each intersection center must be exactly $L * 2$ ft away from an adjacent intersection to represent a square setup. The grid setup in Figure 10 is not displayed to scale.

The iteration process per vehicle is very similar compared to the single intersection acceleration determination process. The main difference is the overall location evaluation per vehicle. Individual calculations regarding position are all based on the center location of the intersection. For example, the through intersection equations from section 3.3.1.4 now include an additional x_i center location term. Furthermore, the specific direction number 2 from the individual intersection cannot always be referenced however, the west direction relative to an individual intersection will still be considered. The new through intersection evaluation equations are listed as

$$x_{PS} = x_I - 28 - v_{id} * t_Y \quad (3.34)$$

$$x_{PR} = x_I - 22 - v_{id} * t_Y + \frac{1}{2} * a_{id} * t_Y^2 \quad (3.35)$$

$$x_{PL} = x_I - 16 - v_{id} * t_Y + \frac{1}{2} * a_{id} * t_Y^2 \quad (3.36)$$

which still directly relate to the individual intersection equations. All equations from section 3.3.1 can be used for any direction west of an intersection given the center x coordinate. This consistency highlights the scalable ability of the individual intersection to a grid city model.

3.3.2.3 Evaluation of Fixed Timing Intersection Performance

The initial evaluation of intersection performance will begin by modeling a select few basic intersection networks to build a framework for the experiment. To evaluate the performance of this intersection, we will consider the average time for each car to proceed through an intersection. Individual vehicle timing can be evaluated using the actual simulation start time (t_s) of the vehicle when it is located at the beginning of the intersection and the final intersection departure time (t_d). The actual time through the simulation (t_a) can be found using the equation

$$t_a = t_d - t_s \quad (3.37)$$

To compare the quality of this value, we can study how long it may take for each individual car to pass through based on the typical behavior of that driver (desired speed, following time, desired acceleration). We can determine the time it would take for an individual driver to pass through this intersection given a green light and no other cars to impede the progress. This calculation will be used as an ideal time (t_{id}) scenario per car.

The ideal time is based on the output direction relative to the input direction. Given different output directions of straight, right, or left, the following three equations for the ideal time (t_{id}) can be determined.

$$t_{ids} = \frac{2L}{v_{id}} \quad (3.38)$$

The right and left turn equations require more detail regarding deceleration and acceleration time as it is necessary to slow down to complete a turn safely.

$$t_{idR} = 2 \left[\frac{-28 - v_{id} \left(\frac{v_{id} - v_T}{a_{id}} \right) + \frac{1}{2}(a_{id}) \left(\frac{v_{id} - v_T}{a_{id}} \right)^2 + L}{v_{id}} \right] + 2 \left(\frac{v_{id} - v_T}{a_{id}} \right) + \frac{6}{v_T} \quad (3.39)$$

$$t_{idL} = 2 \left[\frac{-20 - v_{id} \left(\frac{v_{id} - v_T}{a_{id}} \right) + \frac{1}{2}(a_{id}) \left(\frac{v_{id} - v_T}{a_{id}} \right)^2 + L}{v_{id}} \right] + 2 \left(\frac{v_{id} - v_T}{a_{id}} \right) + \frac{10}{v_T} \quad (3.40)$$

Eventually, an overall real average time per car is evaluated to determine the efficiency of the intersection given the average ideal calculation.

Given details from the MATLAB model from a fixed timing intersection, there is room for improvement. Autonomous vehicles are moving in a positive direction possessing new vehicle recognition and 5g technology. This creates the ability for vehicles to communicate. Building on this initial model and improving intersection performance by light and vehicle communication is a challenging problem. These initial basic steps will help determine the best approach for intersection evaluation.

3.3.3 Implementing Coordinated Traffic Signals

After evaluating and modeling multiple basic intersections, agents will be introduced in traffic signals. As previously stated, an agent is an autonomous decision maker whose decision making is determined internally but may be altered by interactions with the environment or other agents. Therefore, to ensure a proper functioning network, individual traffic signal agents will have their own internal behavior function and will gather data from nearby intelligent intersections to build and improve the model from section 3.3.2. These coordinated traffic signals communicate light status to optimize traffic flow for safety and resilience improvement for the transportation network.

3.3.3.1 Additional MATLAB Modeling for Coordinated Intelligent Traffic Signals

Given the basic intersection model for city grid scenarios, intelligent traffic signals are now implemented. Nearby intersections view the status of adjacent intersections and adapt their own status based on average time for a vehicle to travel between intersections. The average time for a vehicle to travel between intersections (t_b) is

$$t_b = \frac{2L}{S} \quad (3.41)$$

where L is the lane length and S is the speed limit. This is the approximated time a vehicle will take to travel from the center of one intersection to another. Given fixed values of $L = 800$ and $S = 40$ mph (58.6667 ft/s) for all simulations run, the average vehicle time between intersection $t_b = 27.27$ s. This calculation considers vehicles traveling at the full speed limit throughout the transition. Realistically, a range of drivers will travel above and

below this value however, this is used as a reasonable approximation for this scenario.

This value is used to initialize the adjacent intersection green light countdown.

In this example, the 2 x 3 intersection setup will be used.

Figure 11 displays a sampled view of the coordinated traffic signal intersection setup. All traffic is initially routed to pass through each intersection in the north and south directions

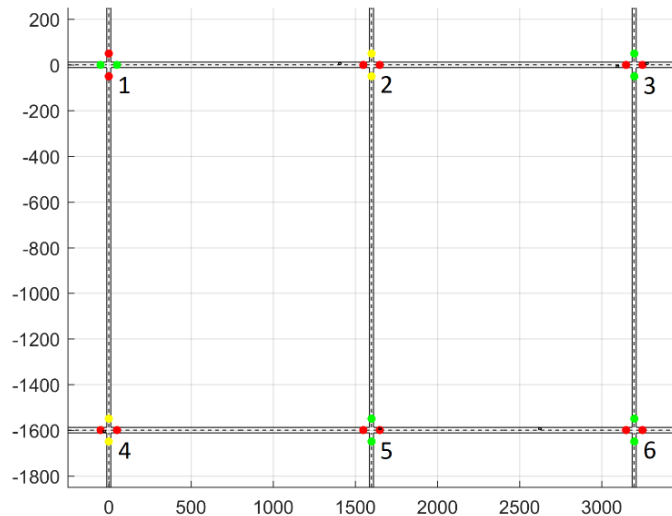


Figure 11: Coordinated Intelligent Intersection Setup.

(Figure 10). Note the intersection numbers displayed in Figure 11. For this coordinated traffic scenario, intersection 1 has the leading fixed signal. This intersection behaves similarly to the fixed timing lights from section 3.3.2 based on the maximum green light time. All other intersections in this simulation will adapt to the nearby intersections relative to the west or north depending on location.

Based on the overall mapping, the direction of coordinated traffic flow will be in the southeast direction. Intersections 2 and 4 are informed when intersection 1 changes state to a yellow light and will immediately start a countdown for their individual light change generally based on the average time for a vehicle to travel between intersections ($t_b = 27.27s$). Referring to the display in Figure 11, intersection 1 has already allowed traffic to pass through in the east and west directions while the north and south traffic is

held at a red light. Intersections 2 and 4 have switched to a yellow light which has started a countdown for intersections 3 and 5 to adjust to their yellow light. Figure 12 shows the coordinated logic for a 2x3 intersection example.

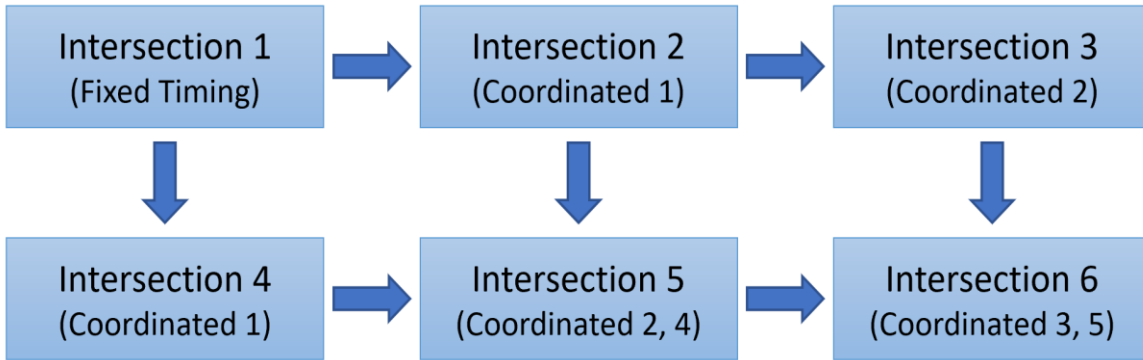


Figure 12: Coordinated Traffic Signal Logic.

3.3.3.2 Coordinated Traffic Signals

Overall, various maximum green light times were run to simulate alternative traffic scenarios. To coordinate traffic appropriately, the intersections receive the instant yellow light change and create a countdown for their individual light change based on either the maximum green light time (t_G) or the average time for a vehicle to travel between intersections (t_b). The appropriate choice is determined by which numerical value is smaller.

Light #	Status	Countdown	Intersection
1	0	0	1
2	2	4.5341	1
3	0	0	1
4	2	4.5341	1
5	1	3.5	2
6	0	0	2
7	1	3.5	2
8	0	0	2
9	2	9.5	3
10	0	0	3
11	2	9.5	3
12	0	0	3
13	1	3.5	4
14	0	0	4
15	1	3.5	4
16	0	0	4
17	2	9.5	5
18	0	0	5
19	2	9.5	5
20	0	0	5
21	2	79.4773	6
22	0	0	6
23	2	79.4773	6
24	0	0	6

Figure 13: Individual Traffic Signal Status.

Figure 13 displays a section of the Traffic Light matrix from the MATLAB code which displays the light number, the status of the light, the countdown of that specific light

status, and the intersection at which the light is placed. For the status, green = 2, yellow = 1, and red = 0. For reference, the individual intersection setup in Figure 5 can be directly labeled as intersection 1. Intersections 2 – 6 contain the same relative direction numbers as intersection 1. The traffic light status in Figure 13 can be directly related to the visual light status representation in Figure 11. An equation relating light status countdown from coordinated intersections (3 coordinate from 2) is important to consider.

$$t_Y - T_5 = t_G - T_9 \quad (3.42)$$

From Figure 13, given a yellow light time of 4 seconds ($t_Y = 4$) and a maximum green light time of 10 seconds ($t_G = 10$), it can be observed that intersection 2, which contains lights 5 and 7 has communicated information to intersections 3 (lights 9 and 11) and 5 about the recent light status change from green to yellow (0.5 seconds have passed since the light change). Therefore, the simulation logic is accurate as the countdown time that has passed in intersections 3 and 5 is exactly 0.5 seconds less than the maximum green light time. This logic is repeated for the duration of the simulation and the average vehicle time through individual intersections is evaluated.

3.3.4 Implementing Adaptive Traffic Signals

Finally, autonomous vehicles will be implemented into the current coordinated traffic signal environment through vehicle recognition ability. This will give lights at an intersection the ability to change status according to the number of vehicles waiting in a red-light direction. This ability will ultimately improve traffic flow through individual

intersections as individual light countdowns may be altered due to a potential higher level of traffic in that specific direction.

3.3.4.1 Additional MATLAB Modeling for Adaptive Traffic Signals

The main addition to the MATLAB model is the ability for a traffic light to recognize the level of vehicles waiting in a specific direction at the intersection. This is referred to as the queue (Q) of the light. For each experiment, the queue of each individual intersection throughout the overall simulation is a fixed value. Each intersection will add up the number of vehicles waiting in the red light direction and when the current queue value reaches the maximum ($Q \geq Q_{\max}$) set in the simulation, a new countdown for the green light direction may be applied.

Overall, the intelligent coordinated traffic signal environment is still implemented. However, the status may be altered due to a large queue. For consistency, the basic intersection directions from Figure 5 will be referenced in this scenario. If directions 1 and 3 (north and south) currently display a green light, directions 2 and 4 (east and west) will display red to avoid intersection collisions. As the countdown to a yellow light continues, traffic from the direction with a red light will build up. A car will be officially added to the queue count when it is completely stopped at the intersection ($v_i = 0$) while waiting for the light to change. When the Q_{\max} value is reached in either direction 2 or 4, a calculation to determine the amount of time it will take for the furthest vehicle in a green light direction (1 or 3) to reach the intersection (t_v) is carried out.

Initially, the code determines the furthest vehicle in each direction from the intersection that is within the lane length range. A situation may occur where no upcoming vehicles are present in the green light directions. In this scenario, the green light countdown will automatically be reduced to 1. This will ensure the vehicles in the red-light direction are not waiting more time when no traffic is present for a green light.

The time each vehicle will take to reach the intersection in each direction (t_{v1} and t_{v3}) will be evaluated. For simplicity, only the evaluation for direction 1 will be explained. First, a calculation is completed to determine the minimum distance the individual vehicle can be from the intersection to make it through given ideal behavior in the event of an immediate change to a yellow light. These calculations are the same compared to the through intersection calculations in section 3.3.1.4. They are shown here again as equations 3.15, 3.16, and 3.17 and are based on the desired output direction straight (x_{PS}), right (x_{PR}), or left(x_{PL}).

$$x_{PS} = -28 - v_{id} * t_Y \quad (3.15)$$

$$x_{PR} = -22 - v_{id} * t_Y + \frac{1}{2} * a_{id} * t_Y^2 \quad (3.16)$$

$$x_{PL} = -16 - v_{id} * t_Y + \frac{1}{2} * a_{id} * t_Y^2 \quad (3.17)$$

Next a calculation is completed to determine how long it will take the vehicle to reach this specific point. Realistically, depending on the true output direction of the vehicle, any of the three above equations could be used. For this scenario, the displayed calculation assumes the upcoming vehicle will make a right turn therefore, variable (x_{PR}) will be used. The time for the vehicle in direction 1 (t_{v1}) is calculated as

$$t_{v1} = \frac{x_{PR} - x_{1i}}{v_{1i}} \quad (3.43)$$

given current vehicle speed (v_{1i}) and position(x_{1i}).

A comparison for calculated values t_{v1} and t_{v3} is completed. The greater value will be used as the ultimate t_v value which is then compared to the light countdown. In this scenario, $t_{v1} > t_{v3}$ is assumed therefore, $t_v = t_{v1}$. The individual light countdown in direction one (T_1) is then compared to the calculated value t_v . If $t_v < T_1$, the new countdown value will be t_v . This will allow all vehicles in each green light direction to proceed through the light and ensures that once this happens, vehicles in the red-light direction will not have to wait unnecessarily. Alternatively, if $t_v > T_1$, the light will continue its normal countdown (T_1) to keep traffic continuously moving. In this research, no additional time will be given to a green light countdown.

3.4 Simulation Goals of the MATLAB Model

The MATLAB code can run simulations regarding either the fixed, coordinated, or adaptive timing signal network. Furthermore, based on details explained throughout chapter 3, a variety of inputs can be adjusted to vary the intersection simulation (car load, intersection grid setup, etc.) These varying parameters allow a user to determine how efficient an intersection may be based on the input details and the signal performance choice. The overall evaluation and explanation regarding the performance difference of each scenario will be explained throughout chapter 4. The appendix contains the main MATLAB code for an adaptive traffic signal setup.

4. Simulation Results

To determine if this adaptive approach improves intersection performance in city environments, evaluations per car will be completed. These evaluations will be completed by comparing overall distance traveled through the entirety of the intersection and time it takes to reach the destination from the initial starting point. Evaluations were initially completed from the fixed timing intersection model to establish a baseline and to confirm there is potential for improvement. The evaluations of the fixed timing signal are compared to the coordinated signal setup as well as the adaptive signal setup. Comparison criteria will be the intersection grid setup, the number of vehicles in the setup, and the queue count for the various adaptive signal options.

4.1 Fixed Timing Evaluation Results

Timing is the most important factor that will be considered during this evaluation. Performance will be evaluated for overall time through intersections. The key factor in

this research is reducing wait times at intersections

(stopped delay).

Figure 14 displays

the fixed timing

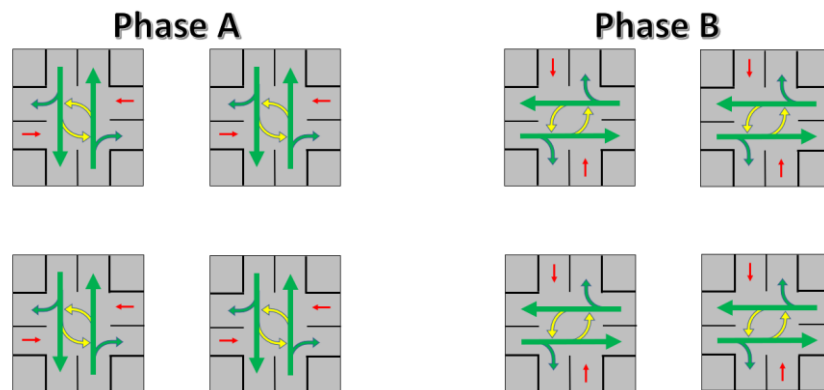


Figure 14: 2x2 Fixed Timing Signal Sequence

intersection sequence in two phases given a 2x2 intersection setup. Due to the fixed

timing evaluation, the only varying parameters are the intersection setup, the number of vehicles on the road, and the maximum green light time. The main comparisons are the intersection setup and the number of vehicles present on the road. The green light time varies and average traffic flow time and overall efficiency through an intersection is evaluated. The maximum green light ranges from a time of 10 – 50 seconds and the number of vehicles on the road per intersection ranges from 5 – 60 cars. The intersection setups evaluated for the fixed timing signal are 1x1, 2x2, 2x3, and 3x3.

4.1.1 1x1 Intersection Setup

The initial baseline evaluation was determined from a single intersection with varied light times and number of vehicles on the road. The graphs represent the varying number of vehicles comparing the average amount of time it takes for one vehicle to

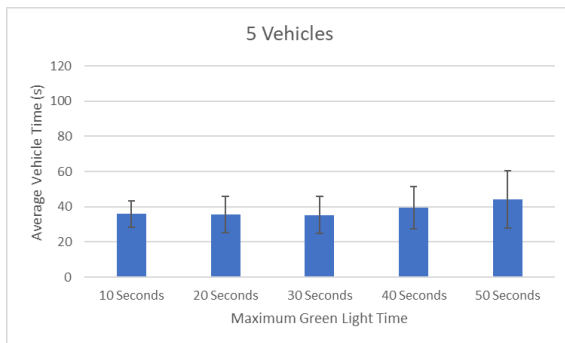


Figure 15: Fixed Timing, 5 Vehicles

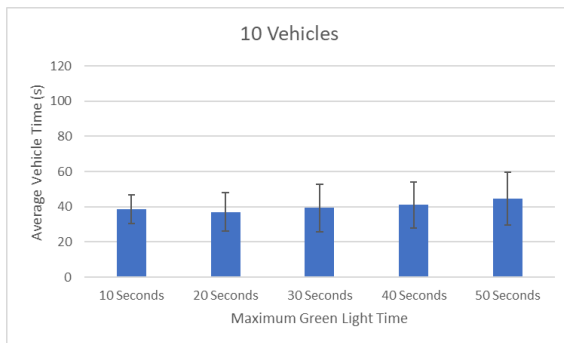


Figure 16: Fixed Timing, 10 Vehicles

proceed through the intersection based on the initial input direction and final output direction. As shown through the display of the graphs, the average time for a vehicle to proceed through the intersection is much larger with a higher number of vehicles on the

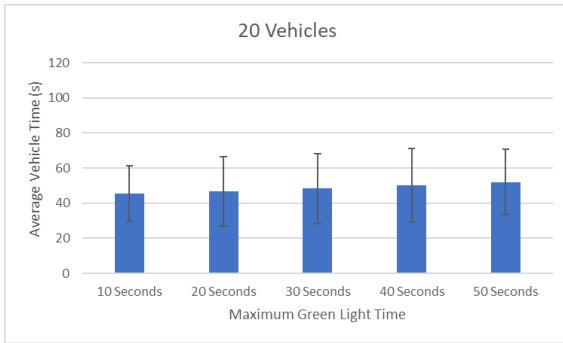


Figure 17: Fixed Timing, 20 Vehicles

road. The average vehicle travel time generally takes longer given the maximum green light time. However, few cases are the opposite with more traffic on the road due to it being more efficient to allow more vehicles through in one cycle. This is like a

scenario where an intersection signal on a

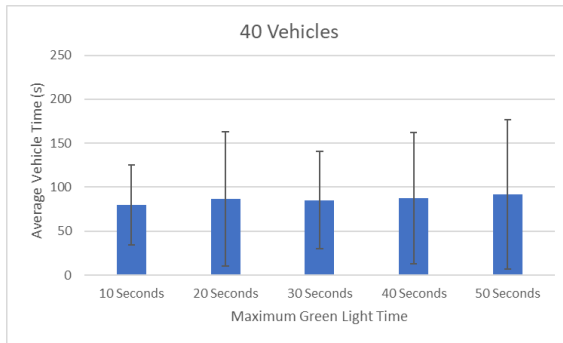


Figure 18: Fixed Timing, 40 Vehicles

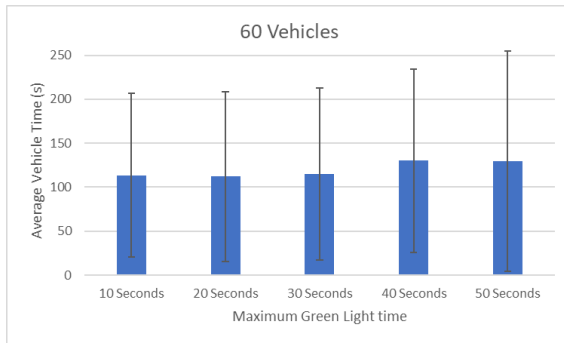


Figure 19: Fixed Timing, 60 Vehicles

main road has broken down resulting in a 4 way stop. Traffic flow through an intersection in this scenario typically is slower therefore, the 10 second light results given a large amount of traffic can be compared.

Efficiencies per scenario are also compared. To maintain consistency for minimizing parameters, the inefficiency will be measured and displayed using the following equation:

$$I_{eff} = 1 - \%E = \left(1 - \frac{t_{id}}{t_a}\right) \quad (4.1)$$

where t_{id} is the average ideal time and t_a is the average actual time for a vehicle to proceed through the intersection.

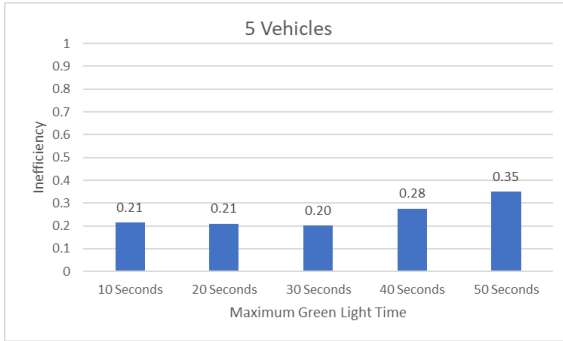


Figure 20: Inefficiency Fixed Timing, 5 Vehicles

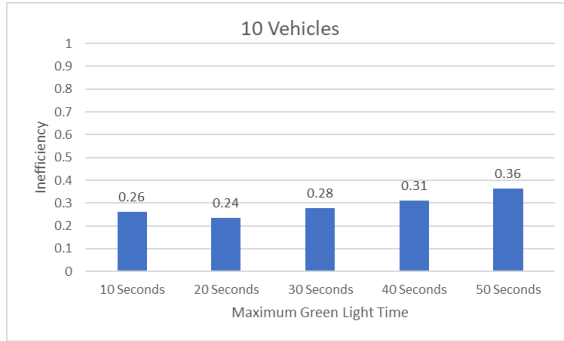


Figure 21: Inefficiency Fixed Timing, 10 Vehicles

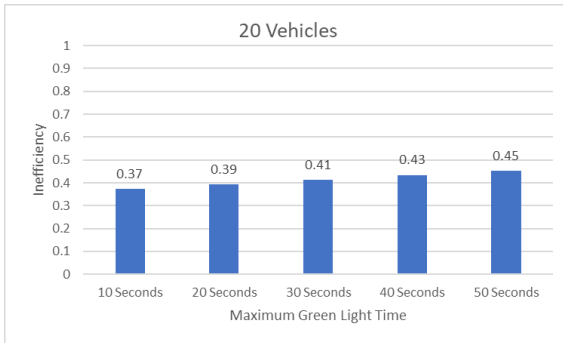


Figure 22: Inefficiency Fixed Timing, 20 Vehicles

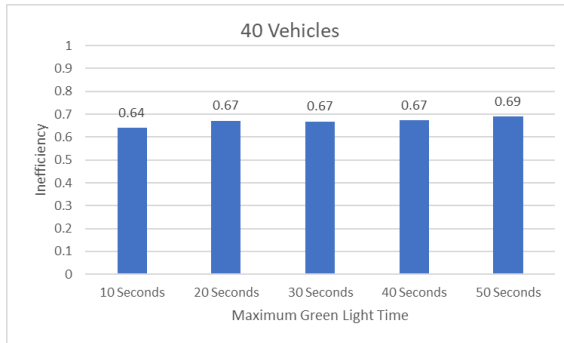


Figure 23: Inefficiency Fixed Timing, 40 Vehicles

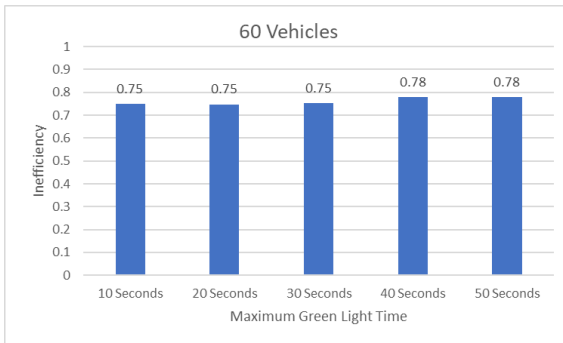


Figure 24: Inefficiency Fixed Timing, 60 Vehicles

For an individual fixed timing signal, based on the figures displayed on this page, it is evident the inefficiency of the intersection rises based on the overall traffic level. This was the baseline data recorded and efforts to improve the efficiency of the intersection

were completed to prove that an adaptive intersection can be beneficial to the transportation infrastructure.

4.1.2 2x2 Intersection Setup

To compare all evaluation results, the 2x2 intersection will be used. From the timing evaluation sequence displayed in Figure 14, the following data regarding the average vehicle time through the intersection and the inefficiency is evaluated. Like the 1x1 intersection setup, the overall goal is to minimize both parameters. The minimization of these parameters will demonstrate an improvement in intersection performance. The 2x2 fixed performance evaluation is the base level to improve the network.

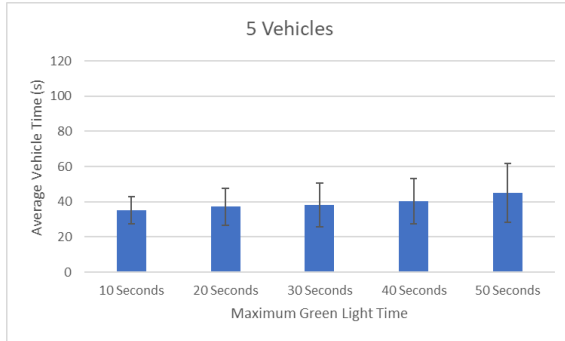


Figure 25: Average Time, Fixed, 5 Vehicles

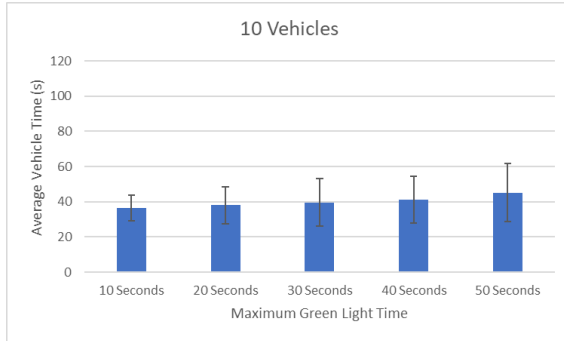


Figure 26: Average Time, Fixed, 10 Vehicles

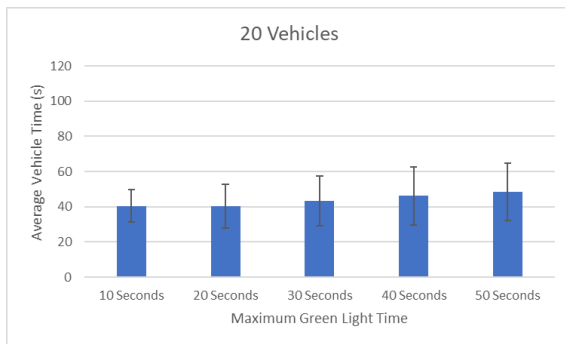


Figure 27: Average Time, Fixed, 20 Vehicles

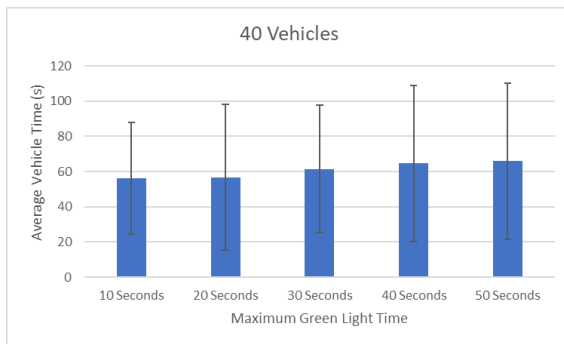


Figure 28: Average Time, Fixed, 40 Vehicles

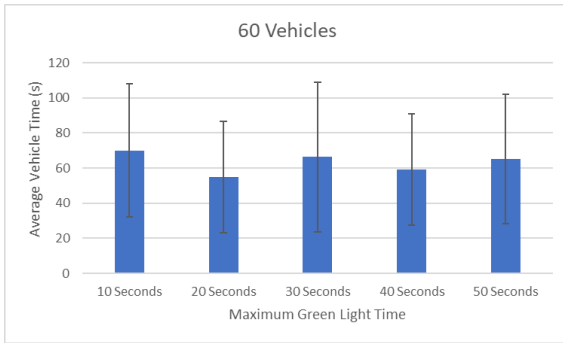


Figure 29: Average Time, Fixed, 60 Vehicles

The 2x2 intersection grid setup behaves similarly to the 1x1 setup. Overall, the average time it takes for one vehicle to proceed through the light increases as there is more traffic on the road. Similarly, the scenario with 60 vehicles on the road

creates a nonlinear situation where the longer green light creates average shorter vehicle times through the intersection. However, the general trend is consistent in allowing more traffic through per cycle like the 1x1 intersection.

The inefficiency scenarios are also consistent with the 1x1 as the efficiency/inefficiency is directly related to the actual vehicle time through the intersection. Given these baseline parameters, the improved traffic signals will be improved and compared.

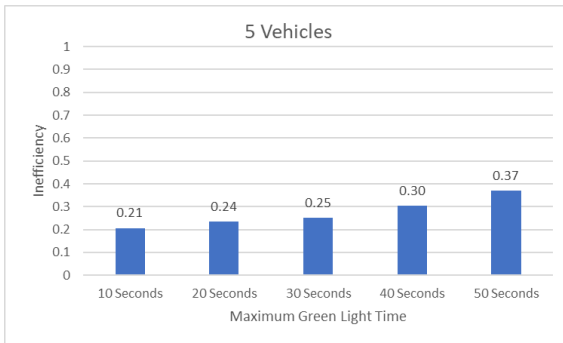


Figure 30: Inefficiency, Fixed, 5 Vehicles

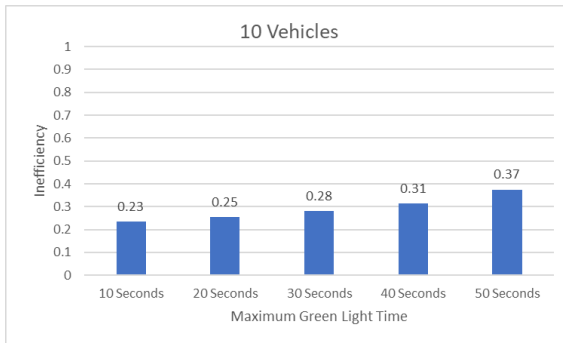


Figure 31: Inefficiency, Fixed, 10 Vehicles

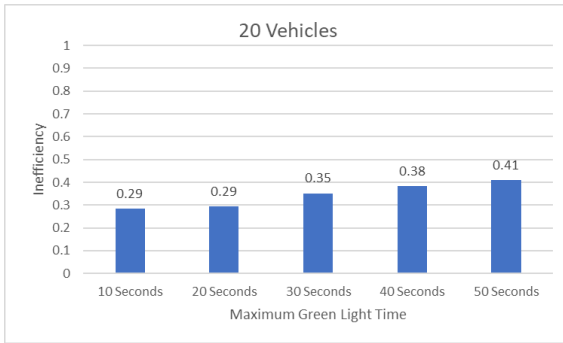


Figure 32: Inefficiency, Fixed, 20 Vehicles

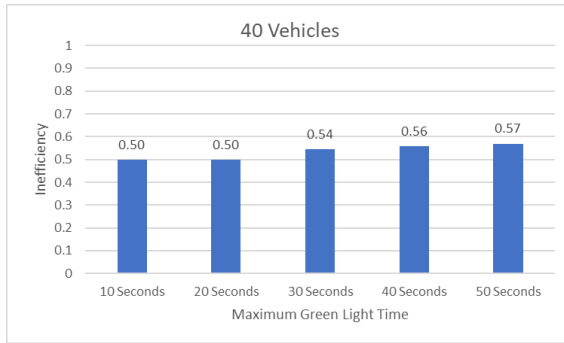


Figure 33: Inefficiency, Fixed, 40 Vehicles

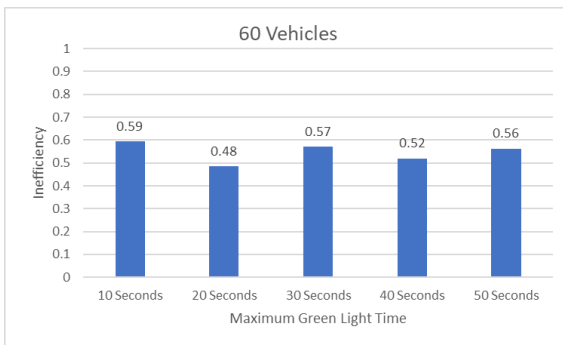


Figure 34: Inefficiency, Fixed, 60 Vehicles

4.1.3 Overall Fixed Timing Results

The overall details for the basic intersection setup regarding the numerical details per intersection are displayed in Table 8. The detailed comparisons show the difference in vehicle time through each intersection based on the vehicle load, the green light times per direction, and the intersection setup. It is evident that the intersection setup does not increase the average time per vehicle. It is overall based on the longer light time as well as the increased number of vehicles on the road.

Table 8: Fixed Signal Timing Results

2x2 Intersection											
	Average Vehicle Time (s)						Inefficiency				
	Maximum Green Light						Maximum Green Light				
Cars	10	20	30	40	50		10	20	30	40	50
5	35.2	37.1	38.2	40.3	45.1		0.20	0.24	0.25	0.30	0.37
10	36.4	38.0	39.6	41.2	45.2		0.23	0.25	0.28	0.31	0.37
20	40.4	40.4	43.5	46.1	48.4		0.29	0.29	0.35	0.38	0.41
40	56.1	56.7	61.6	64.6	66.0		0.50	0.50	0.54	0.56	0.57
60	70.1	55.0	66.3	59.2	65.2		0.59	0.48	0.57	0.52	0.56
2x3 Intersection											
	Average Vehicle Time (s)						Inefficiency				
Cars	10	20	30	40	50		10	20	30	40	50
5	35.8	38.1	36.6	40.0	44.6		0.20	0.26	0.23	0.28	0.36
10	36.5	37.4	39.5	39.9	44.2		0.22	0.24	0.27	0.29	0.36
20	40.8	40.5	43.8	45.1	47.6		0.30	0.29	0.34	0.37	0.40
40	54.4	52.7	53.8	55.4	58.4		0.48	0.45	0.47	0.48	0.51
60	70.3	67.8	59.0	62.8	71.3		0.59	0.58	0.51	0.55	0.60
3x3 Intersection											
	Average Vehicle Time (s)						Inefficiency				
Cars	10	20	30	40	50		10	20	30	40	50
5	35.9	36.3	37.8	39.4	44.6		0.20	0.22	0.25	0.28	0.36
10	36.6	35.9	39.1	40.9	43.9		0.24	0.20	0.27	0.30	0.35
20	39.5	39.9	44.1	43.7	47.6		0.28	0.29	0.36	0.35	0.41
40	52.5	49.6	54.3	54.1	53.0		0.46	0.43	0.47	0.47	0.46
60	67.6	62.5	59.9	62.6	63.4		0.58	0.54	0.53	0.54	0.55

Graphs to summarize the data can be viewed in Figure 35, Figure 36, and Figure 37. Trends generally show that increasing the signal timing and the number of vehicles on the road will yield longer times through the intersection. 60 vehicles on the road has unique trends most likely based on the number of vehicles at an individual intersection. It can be seen though that longer green light times are more beneficial for a larger vehicle count.

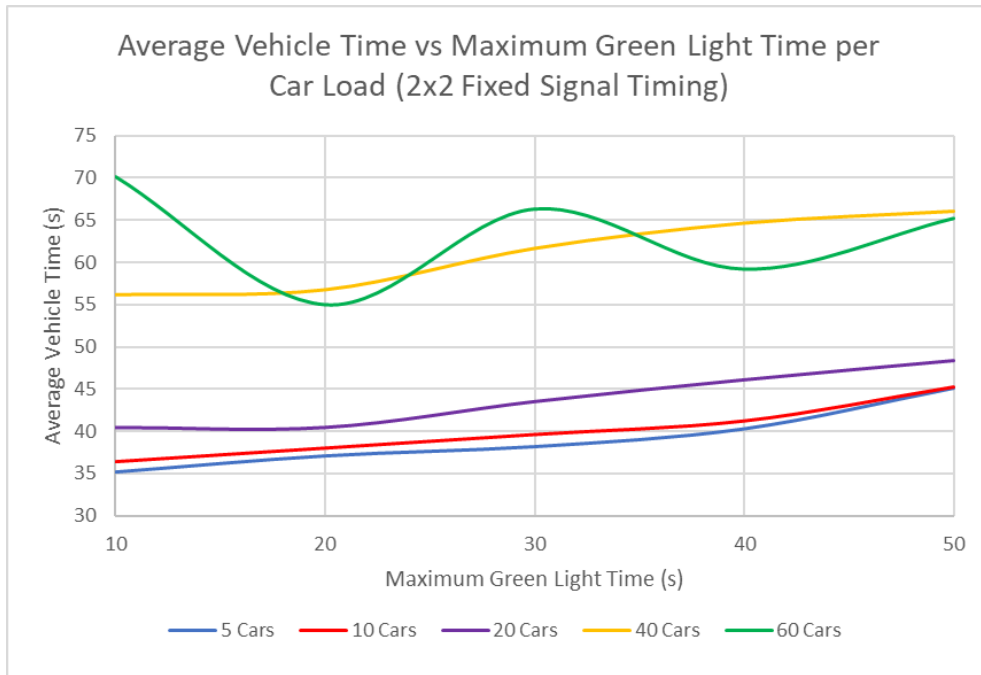


Figure 35: Average Vehicle Time for 2x2 Intersection, Fixed Signal

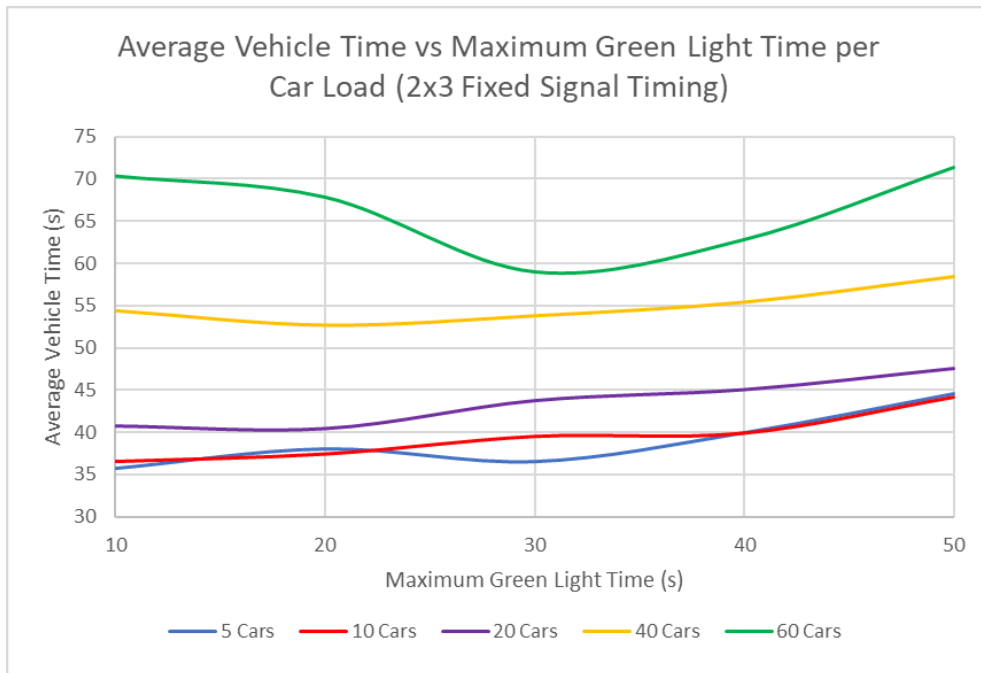


Figure 36: Average Vehicle Time for 2x3 Intersection, Fixed Signal

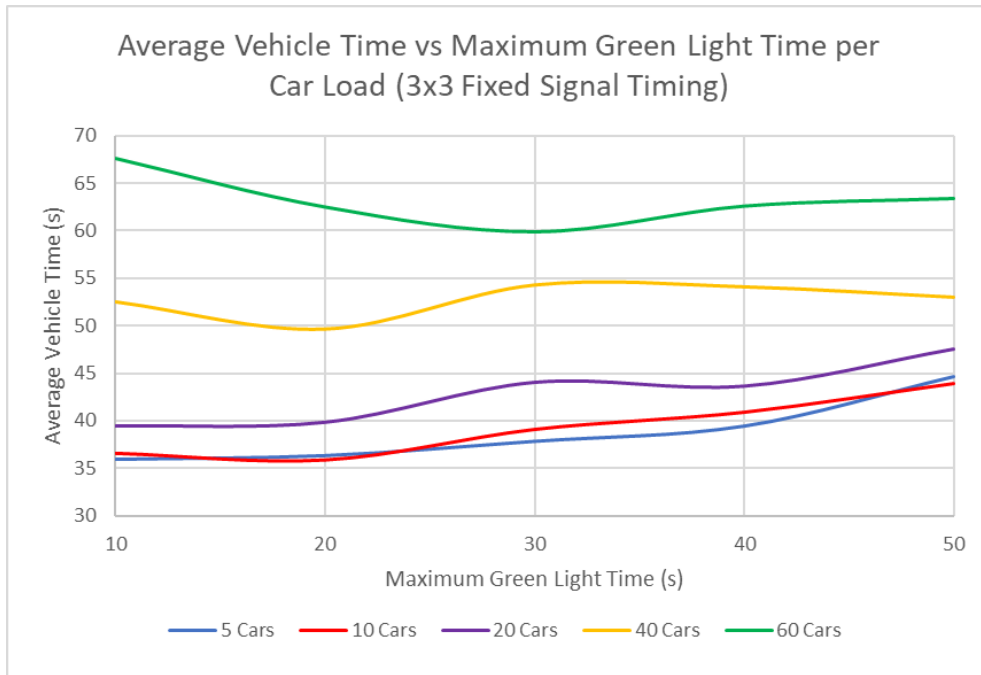


Figure 37: Average Vehicle Time for 3x3 Intersection, Fixed Signal

4.2 Coordinated Signal Evaluation Results

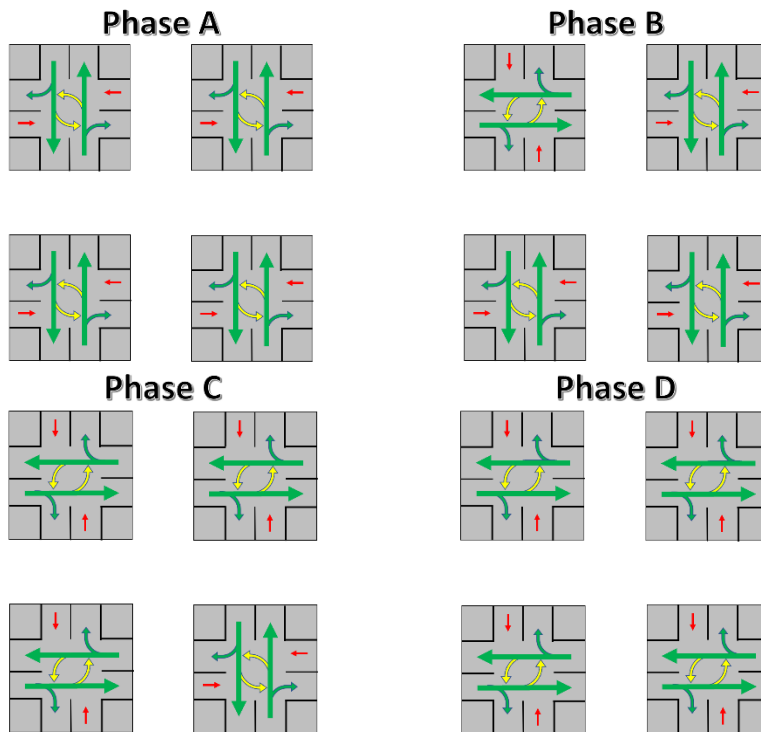


Figure 38: Coordinated Signal Light Sequence

The coordinated traffic signal is the first step for improving the intersection performance. As discussed in chapter 3, the nearby light sequences are now adaptable based on nearby intersection signal changes which includes rudimentary communication. A

representation of this light sequence in ongoing phases for a 2x2 intersection is displayed in Figure 38. The exact phase changes may not be represented by the exact figure but overall, the upper left intersection will change first to allow the opposite directional traffic to flow. The next phase includes the adjacent intersections compared to the initial. Finally, the bottom right intersection will adjust the direction. Other situations could occur where the initial intersection may change back to allow north and south traffic to flow before the last intersection has the option for a change. This ultimately will depend on the maximum green light time.

4.2.1 Graphic Displays of Coordinated Light Sequence

The data displayed for each graph will also be for a 2x2 intersection grid setup for consistency. More traffic creates an overall longer wait time. However, the wait times are more consistent given the longer green light time in scenarios with a higher number of vehicles. This is also consistent for the inefficiency of the intersection.

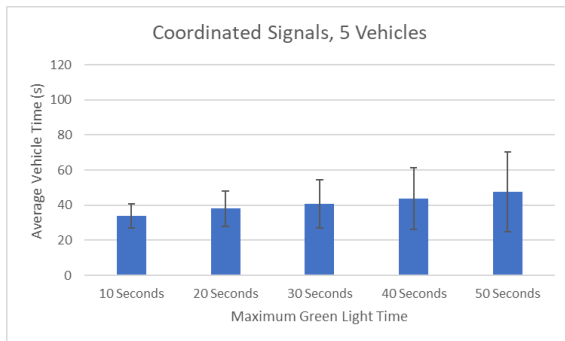


Figure 39: Average Time, Coordinated, 5 Vehicles

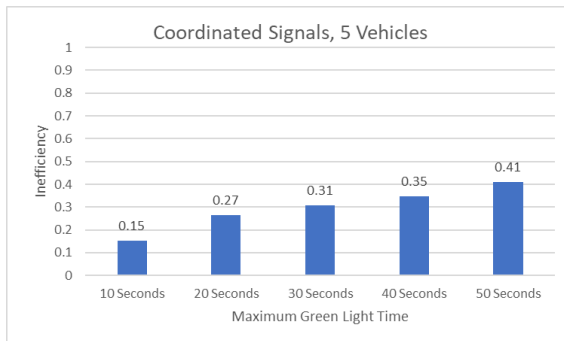


Figure 40: Inefficiency, Coordinated, 5 Vehicles

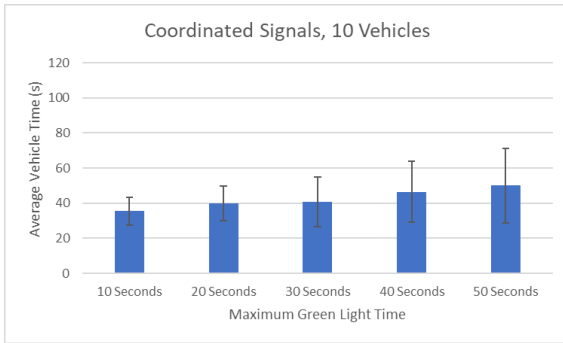


Figure 41: Average Time, Coordinated, 10 Vehicles

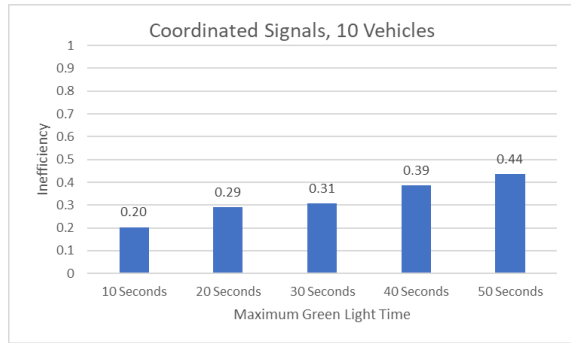


Figure 42: Inefficiency, Coordinated, 10 Vehicles

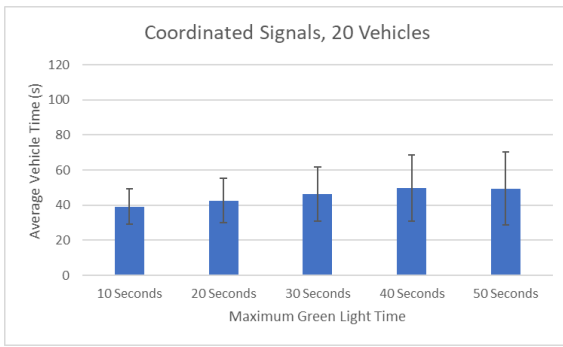


Figure 43: Average Time, Coordinated, 20 Vehicles

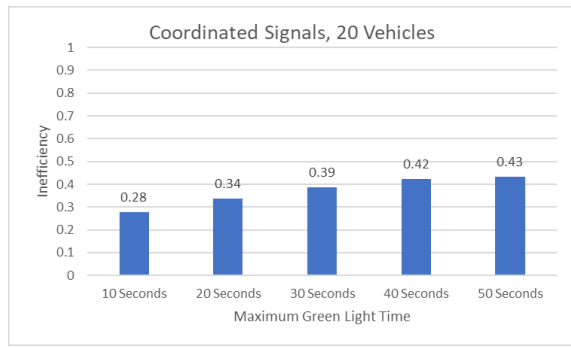


Figure 44: Inefficiency, Coordinated, 20 Vehicles

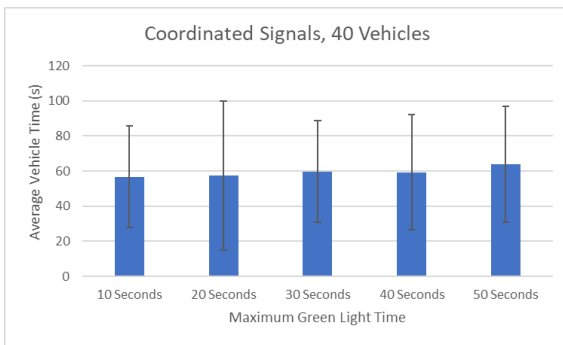


Figure 45: Average Time, Coordinated, 40 Vehicles

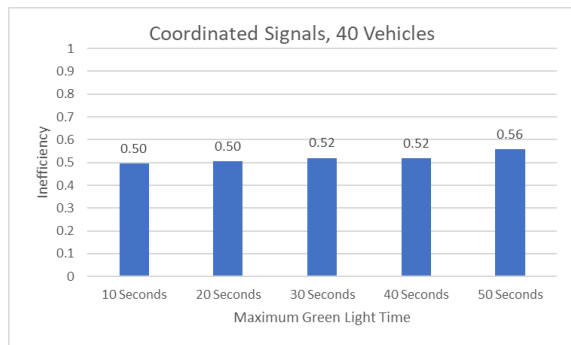


Figure 46: Inefficiency, Coordinated, 40 Vehicles

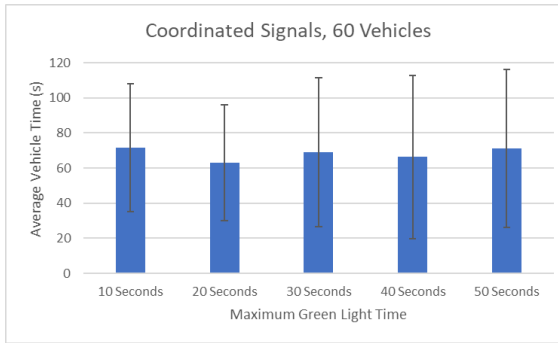


Figure 47: Average Time, Coordinated, 60 Vehicles

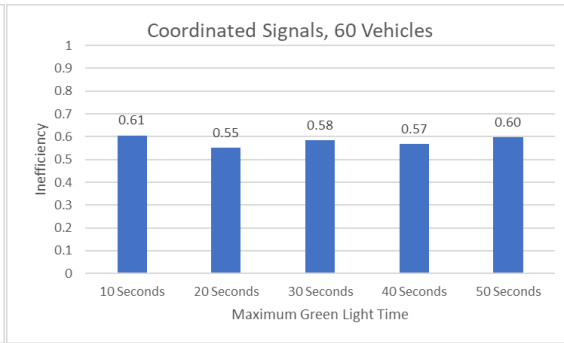
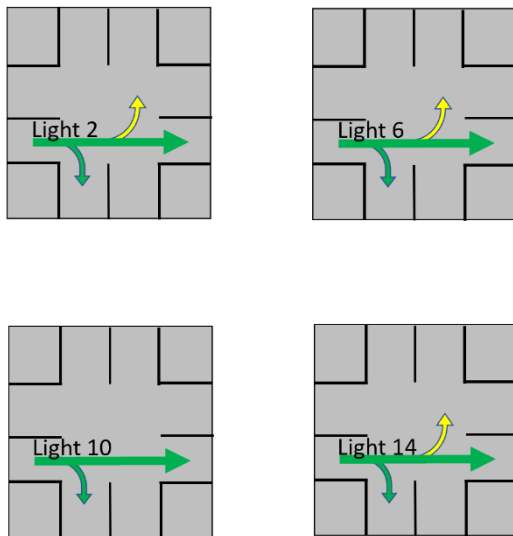


Figure 48: Inefficiency, Coordinated, 60 Vehicles

To demonstrate the importance of the light signals, Figure 50 shows the light status as run through a 2x2 simulation with 20 vehicles on the road per intersection (80 total). These values specifically are identified as the west directional light statuses per



intersection in the 2x2 setup. Light 2 is the upper left intersection, light 6 is for the upper right, light 10 references lower left, and light 14 refers to the lower right intersection. Figure 49

shows the light locations for a 2x2 intersection setup. The light patterns can be compared, and it is evident that as light 2 changes to a yellow,

Figure 49: 2x2 Intersection Light Number Locations

it is within a certain amount of time that lights

6 and 10 will alter their status. It is furthermore clear that lights 6 and 10 are on the exact same track as they both alter their status based on light 2.

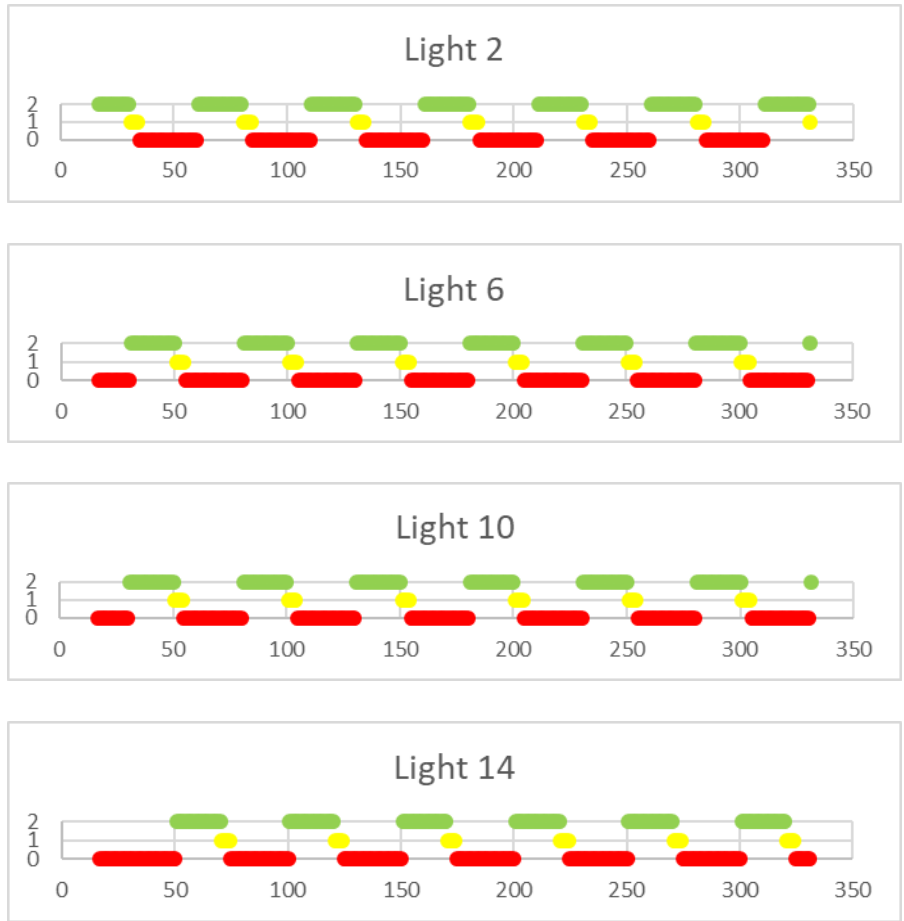


Figure 50: Visual Status Representation of Coordinated Traffic Signals

4.2.2 Overall Coordinated Light Sequence Results

The overall details for the basic intersection setup regarding the numerical details per intersection are displayed in Table 9. As the vehicle load increases, the longer green light times seem to stay nearly as efficiency as the lower light times. This is due to the coordinated lights syncing up appropriately to allow vehicles to make it through the upcoming light given it is traveling in a direction towards the new signal changed intersection.

Table 9: Coordinated Signal Timing Results

2x2 Intersection											
	Average Vehicle Time (s)						Inefficiency				
	Maximum Green Light						Maximum Green Light				
Cars	10	20	30	40	50		10	20	30	40	50
5	33.9	38.1	40.7	43.6	47.4		0.15	0.27	0.31	0.35	0.41
10	35.4	39.8	40.8	46.5	50.1		0.20	0.29	0.31	0.39	0.44
20	39.1	42.5	46.3	49.7	49.4		0.28	0.34	0.39	0.42	0.43
40	56.8	57.5	59.7	59.3	63.9		0.50	0.50	0.52	0.52	0.56
60	71.8	63.2	69.2	66.3	71.0		0.61	0.55	0.58	0.57	0.60
2x3 Intersection											
	Average Vehicle Time (s)						Inefficiency				
Cars	10	20	30	40	50		10	20	30	40	50
5	33.7	38.1	40.3	43.8	49.0		0.16	0.26	0.29	0.36	0.42
10	34.8	38.8	43.5	43.8	51.3		0.18	0.28	0.35	0.35	0.45
20	40.0	44.5	47.9	48.1	50.6		0.30	0.36	0.41	0.41	0.44
40	51.4	55.3	59.1	60.2	64.2		0.45	0.48	0.52	0.53	0.56
60	73.1	62.3	67.9	70.4	68.4		0.61	0.55	0.58	0.60	0.58
3x3 Intersection											
	Average Vehicle Time (s)						Inefficiency				
Cars	10	20	30	40	50		10	20	30	40	50
5	33.9	40.0	41.3	44.6	47.2		0.16	0.30	0.31	0.36	0.40
10	34.6	40.8	45.6	45.3	50.5		0.19	0.30	0.37	0.38	0.43
20	38.4	45.9	51.0	50.5	51.6		0.26	0.38	0.44	0.44	0.44
40	53.4	58.7	57.1	62.2	59.2		0.46	0.51	0.50	0.54	0.52
60	70.6	61.5	61.2	65.5	60.2		0.59	0.54	0.53	0.57	0.53

Trendlines can again be formed from the data in Table 9. These are shown in Figure 51, Figure 52, and Figure 53. Again, like the fixed signal timing, it is evident that longer wait times occur with more vehicles on the road and with longer green signal times. Higher loads of vehicles contain trends where longer green light times improve the overall efficiency. This is due to more vehicles allowed through the intersection in one cycle.

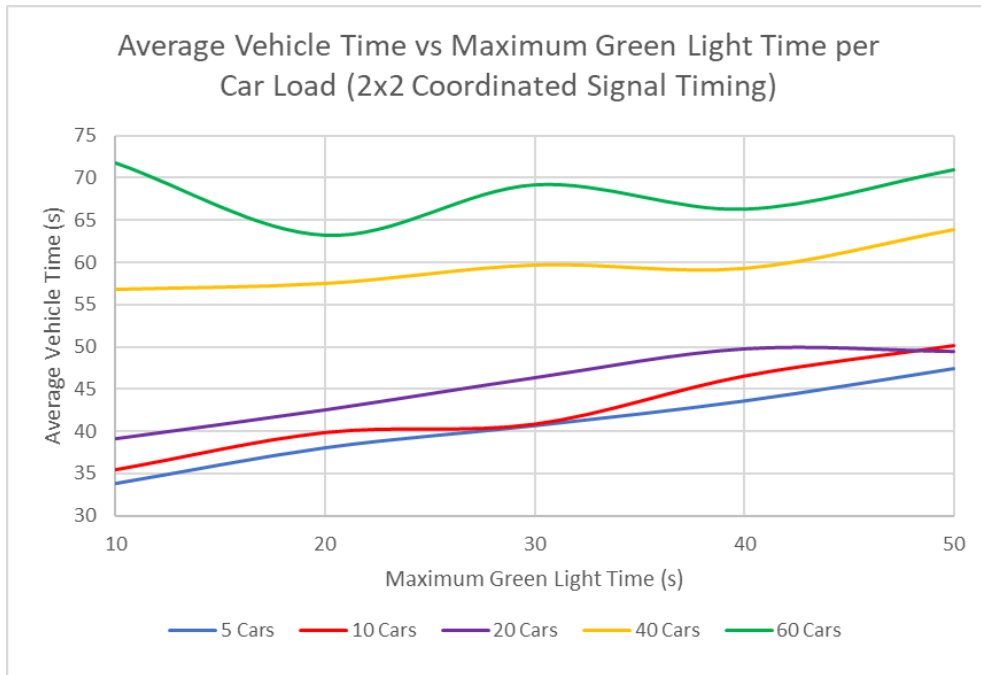


Figure 51: Average Vehicle Time for 2x2 Intersection, Coordinated Signal

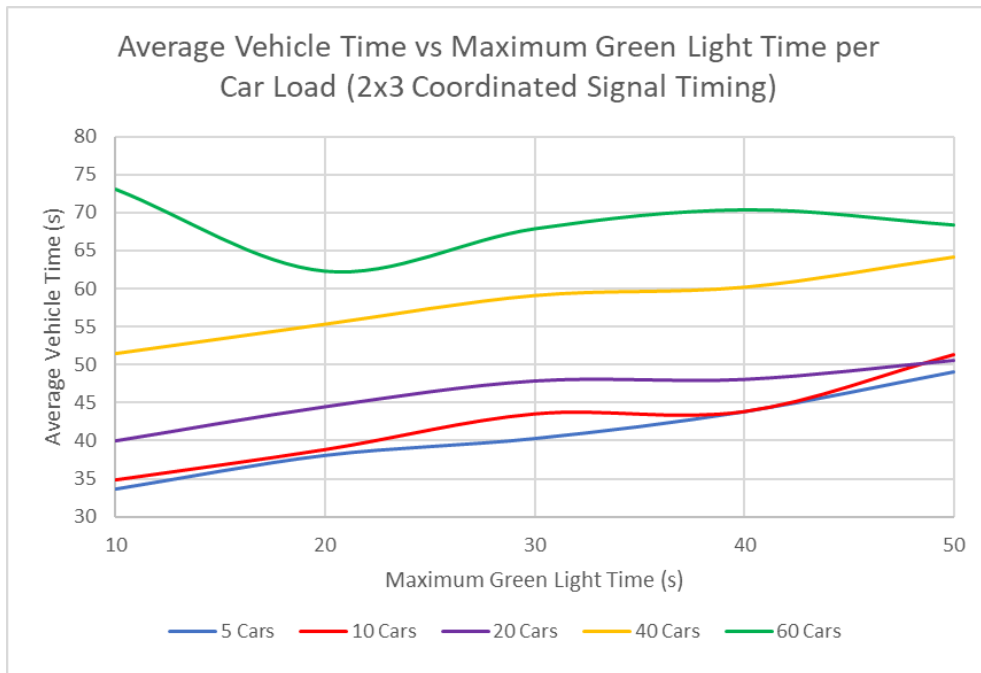


Figure 52: Average Vehicle Time for 2x3 Intersection, Coordinated Signal

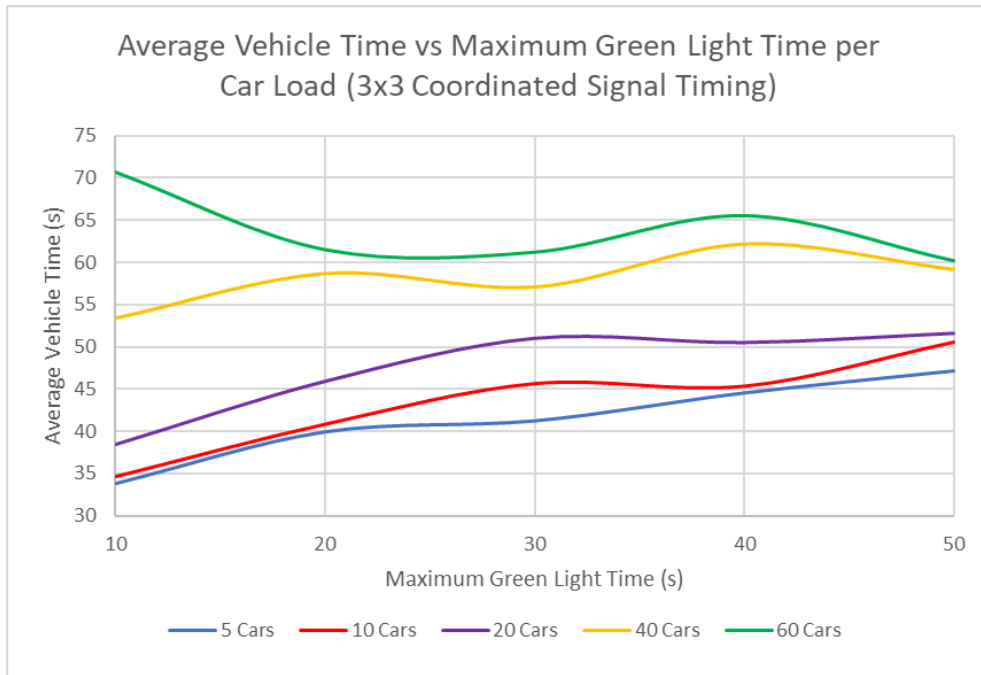


Figure 53: Average Vehicle Time for 3x3 Intersection, Coordinated Signal

4.3 Adaptive Signal Results

Overall, the best improvement for the vehicles to proceed through the intersection is the addition of the adaptive signal. This will allow the lights to adjust their light signal from neighboring traffic light status as well as the vehicle queue at a specific intersection. There is no specific phase diagram for this sequence. The initial light changes are based on the phases from Figure 38, but may be altered based on the queue size of cars at each intersection in the direction of the red light. In this research, the queue is not adjusted per simulation.

4.3.1 Graphic Displays of Adaptive Light Sequence

There have been simulations run for queue lineup numbers set for 1, 3 and 10 vehicles. Future research will enable a smart queue factor however, to display the initial benefits of this adaptive signal, a queue value of 3 vehicles will be used. All simulations displayed are based on a 2x2 intersection scenario.

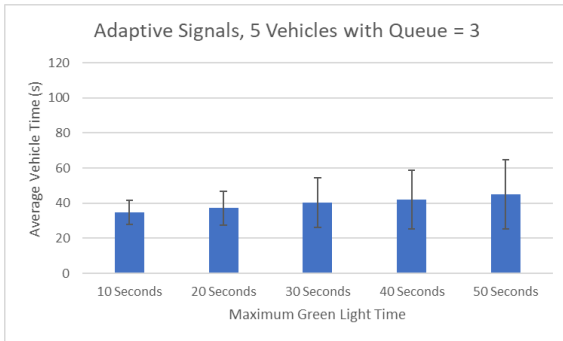


Figure 54: Average time, Adaptive, 5 Vehicles

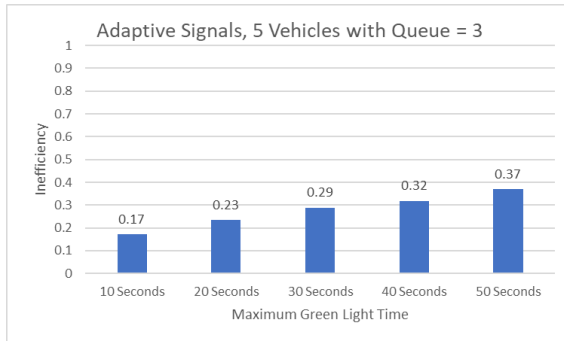


Figure 55: Inefficiency, Adaptive, 5 Vehicles

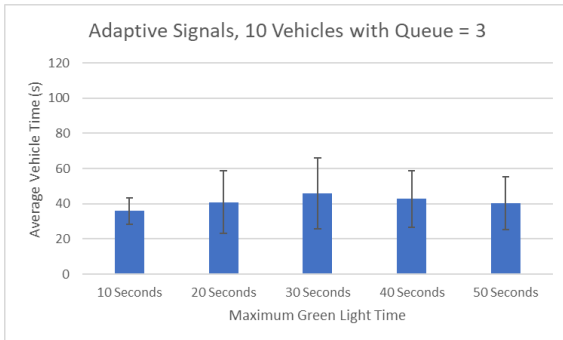


Figure 56: Average Time, Adaptive, 10 Vehicles

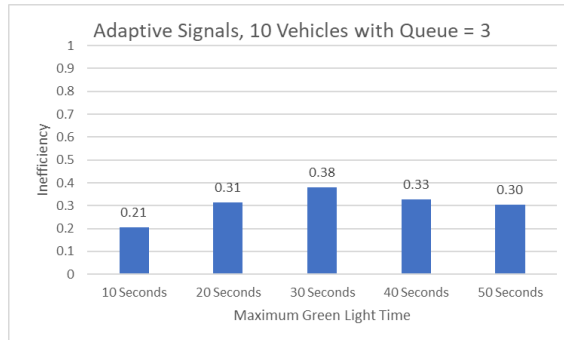


Figure 57: Inefficiency, Adaptive, 10 Vehicles

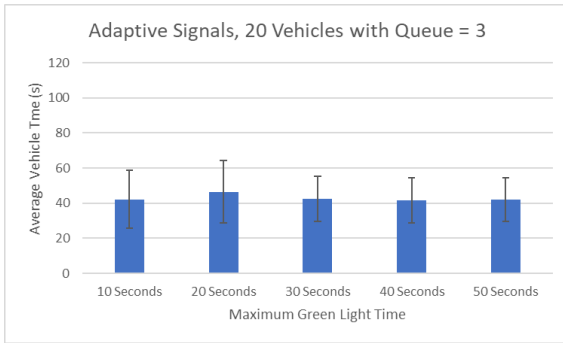


Figure 58: Average Time, Adaptive, 20 Vehicles

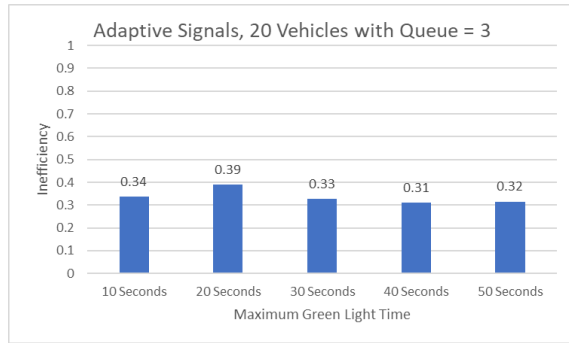


Figure 59: Inefficiency, Adaptive, 20 Vehicles

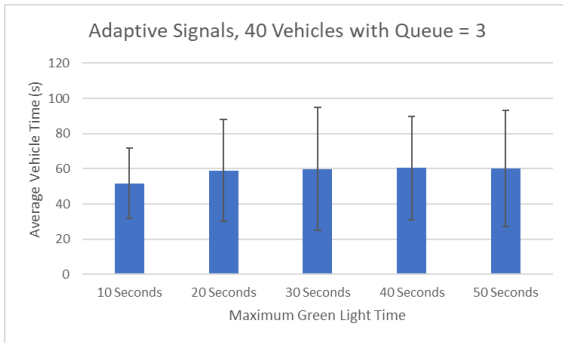


Figure 60: Average Time, Adaptive, 40 Vehicles

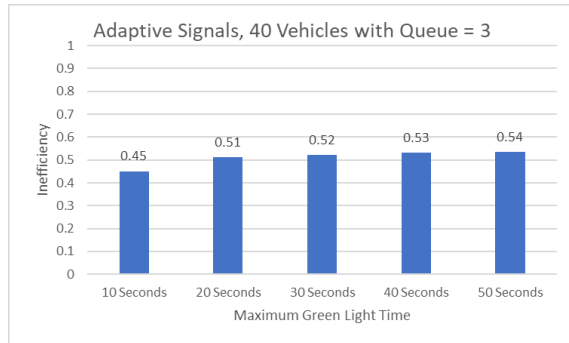


Figure 61: Inefficiency, Adaptive 40 Vehicles

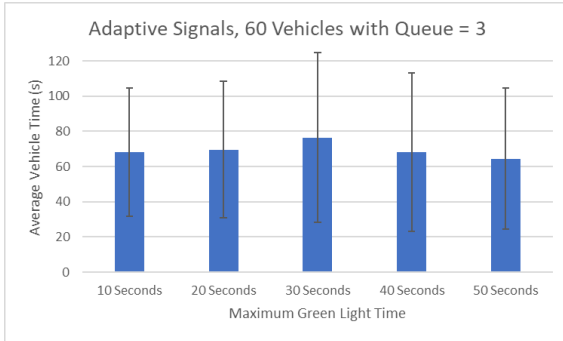


Figure 62: Average Time, Adaptive, 60 Vehicles

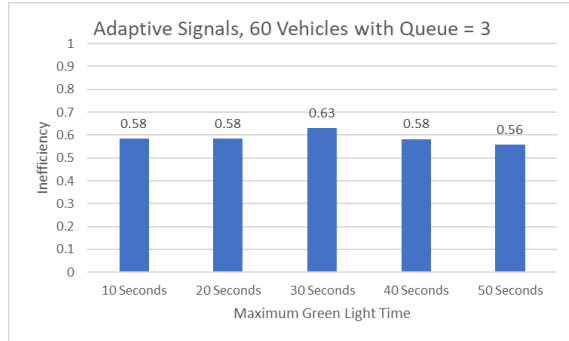


Figure 63: Inefficiency, Adaptive, 60 Vehicles

Furthermore, the light sequence status can be viewed as well. Like the coordinated signal setup, these lights are all based on the west direction from each intersection as displayed in Figure 49. Individual lights now have their own specific agenda based on the queue count but can also be altered from the neighboring light status. Light 14 in Figure 64 has a red-light section that lasts a very large amount of time. This scenario

may occur when no vehicles are present in a specific direction. In this case, no vehicles are waiting for the lights at the intersection approaching from both the east and west input directions. This allows for the north and south signals to be green for an extended period as it is unnecessary to alter the status for no upcoming vehicles.

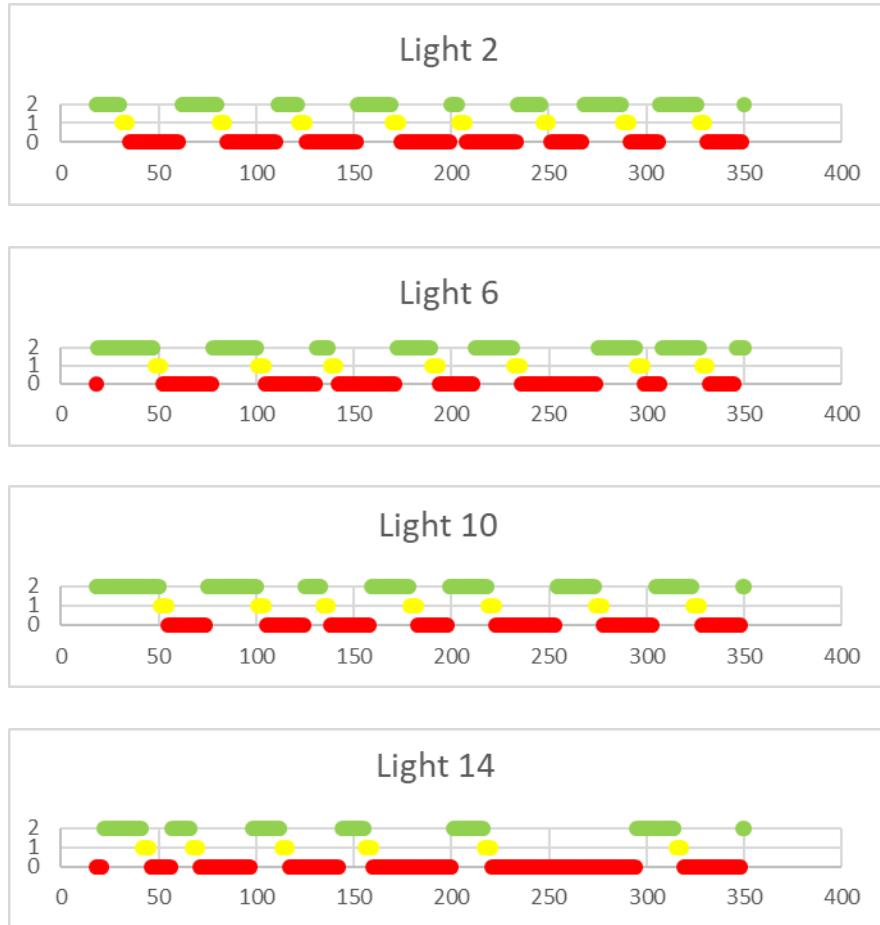


Figure 64: Visual Status Representation of Adaptive Traffic Signals

4.3.2 Overall Adaptive Light Sequence Results

The results from the adaptive light sequences are displayed in the following tables. They are separated by the maximum queue values (1, 3, 10) for each intersection grid setup. Note that in some scenarios, the average time per vehicles is reduced more given

a higher queue with a higher number of vehicles on the road. Based on the data from the tables and from the previous detail, there is an ideal queue value that can be chosen based on the number of vehicles on the road. This queue size can be made adaptable in future problems regarding this research.

Table 10: Adaptive Signal Timing Results, Queue 1

2x2 Intersection											
	Average Vehicle Time (s)						Inefficiency				
	Maximum Green Light						Maximum Green Light				
Cars	10	20	30	40	50		10	20	30	40	50
5	35.9	41.5	35.2	34.9	35.4		0.20	0.31	0.19	0.18	0.19
10	34.8	37.7	36.2	37.6	39.8		0.19	0.25	0.22	0.25	0.28
20	42.1	45.0	41.1	43.1	44.1		0.32	0.37	0.31	0.35	0.34
40	60.1	59.4	61.0	64.6	67.2		0.53	0.51	0.53	0.56	0.57
60	78.8	77.2	62.8	64.3	71.6		0.64	0.63	0.55	0.55	0.60
2x3 Intersection											
	Average Vehicle Time (s)						Inefficiency				
Cars	10	20	30	40	50		10	20	30	40	50
5	35.0	36.9	41.5	35.3	35.8		0.20	0.23	0.31	0.19	0.22
10	36.6	36.9	35.4	36.4	39.3		0.23	0.23	0.20	0.22	0.27
20	39.3	39.3	41.5	42.5	43.8		0.27	0.27	0.31	0.33	0.34
40	57.2	53.4	57.4	51.1	56.6		0.50	0.47	0.50	0.44	0.49
60	73.6	70.5	72.2	71.6	72.9		0.61	0.59	0.60	0.60	0.61
3x3 Intersection											
	Average Vehicle Time (s)						Inefficiency				
Cars	10	20	30	40	50		10	20	30	40	50
5	35.1	36.6	36.7	35.8	34.2		0.20	0.23	0.22	0.20	0.18
10	36.5	36.9	37.4	37.2	37.1		0.21	0.23	0.24	0.25	0.23
20	39.5	39.0	42.1	42.8	39.7		0.28	0.26	0.33	0.35	0.29
40	53.4	55.6	53.1	59.3	56.5		0.47	0.49	0.46	0.52	0.50
60	54.3	60.4	66.6	59.5	61.6		0.48	0.53	0.57	0.52	0.53

Table 11: Adaptive Signal Timing Results, Queue 3

2x2 Intersection											
	Average Vehicle Time (s)						Inefficiency				
	Maximum Green Light						Maximum Green Light				
Cars	10	20	30	40	50		10	20	30	40	50
5	34.8	37.1	40.4	41.9	45.0		0.17	0.23	0.29	0.32	0.37
10	35.8	40.9	45.8	42.7	40.3		0.21	0.31	0.38	0.33	0.30
20	42.1	46.4	42.3	41.7	41.9		0.34	0.39	0.33	0.31	0.32
40	51.8	59.0	59.8	60.5	60.3		0.45	0.51	0.52	0.53	0.54
60	68.1	69.6	76.5	68.1	64.5		0.58	0.58	0.63	0.58	0.56
2x3 Intersection											
	Average Vehicle Time (s)						Inefficiency				
Cars	10	20	30	40	50		10	20	30	40	50
5	34.1	38.3	40.6	42.5	44.4		0.17	0.26	0.30	0.34	0.37
10	35.0	40.9	42.3	43.0	41.1		0.18	0.30	0.33	0.34	0.31
20	41.1	42.7	43.3	42.3	43.1		0.31	0.34	0.34	0.32	0.33
40	57.1	53.5	56.3	56.5	58.9		0.50	0.47	0.49	0.49	0.52
60	66.2	69.9	65.5	71.2	63.8		0.57	0.59	0.57	0.60	0.55
3x3 Intersection											
	Average Vehicle Time (s)						Inefficiency				
Cars	10	20	30	40	50		10	20	30	40	50
5	34.1	37.0	41.1	42.6	41.7		0.18	0.24	0.31	0.34	0.34
10	35.6	40.1	44.5	41.9	41.6		0.21	0.29	0.37	0.33	0.33
20	42.3	41.1	42.0	40.5	41.4		0.33	0.31	0.32	0.30	0.32
40	57.0	52.4	54.8	52.1	53.2		0.50	0.46	0.48	0.46	0.46
60	64.3	61.4	68.9	68.1	63.4		0.56	0.54	0.59	0.58	0.55

Table 12: Adaptive Signal Timing Results, Queue 10

2x2 Intersection											
	Average Vehicle Time (s)						Inefficiency				
	Maximum Green Light						Maximum Green Light				
Cars	10	20	30	40	50		10	20	30	40	50
5	34.6	39.3	40.5	44.4	45.9		0.18	0.28	0.30	0.37	0.39
10	35.4	39.9	42.6	45.0	51.5		0.20	0.29	0.33	0.37	0.45
20	39.8	44.8	46.4	48.8	51.6		0.29	0.36	0.39	0.41	0.45
40	52.6	60.0	64.6	63.4	61.8		0.46	0.53	0.56	0.55	0.54
60	69.1	64.6	79.6	61.8	68.6		0.59	0.55	0.64	0.54	0.58
2x3 Intersection											
	Average Vehicle Time (s)						Inefficiency				
Cars	10	20	30	40	50		10	20	30	40	50
5	33.7	39.2	41.9	42.5	49.0		0.15	0.28	0.32	0.33	0.42
10	35.3	40.5	44.8	46.5	48.8		0.18	0.29	0.36	0.38	0.42
20	39.2	44.6	48.8	49.3	54.6		0.27	0.36	0.43	0.42	0.48
40	55.8	58.5	58.6	58.3	61.7		0.50	0.51	0.51	0.51	0.54
60	73.5	69.6	69.0	71.3	68.5		0.61	0.59	0.58	0.60	0.58
3x3 Intersection											
	Average Vehicle Time (s)						Inefficiency				
Cars	10	20	30	40	50		10	20	30	40	50
5	33.9	37.4	41.5	45.7	49.6		0.17	0.26	0.31	0.38	0.42
10	35.3	40.5	44.0	45.0	49.1		0.20	0.30	0.35	0.37	0.42
20	39.2	45.2	47.4	48.1	50.2		0.28	0.36	0.40	0.40	0.44
40	54.0	54.1	54.9	58.0	58.1		0.47	0.48	0.48	0.51	0.51
60	64.6	62.9	65.6	63.4	65.5		0.56	0.54	0.57	0.55	0.56

The overall comparisons from the different intersection setups compared to the queue values can be evaluated. Note that a lower count of vehicles on the road has better performance with a smaller queue as the lights need to adapt more recently. The larger queue though is more beneficial for a higher number of cars on the road and the trendline is steadier for the varying light times. By using this data appropriately, an adaptive queue can be implemented in future work.

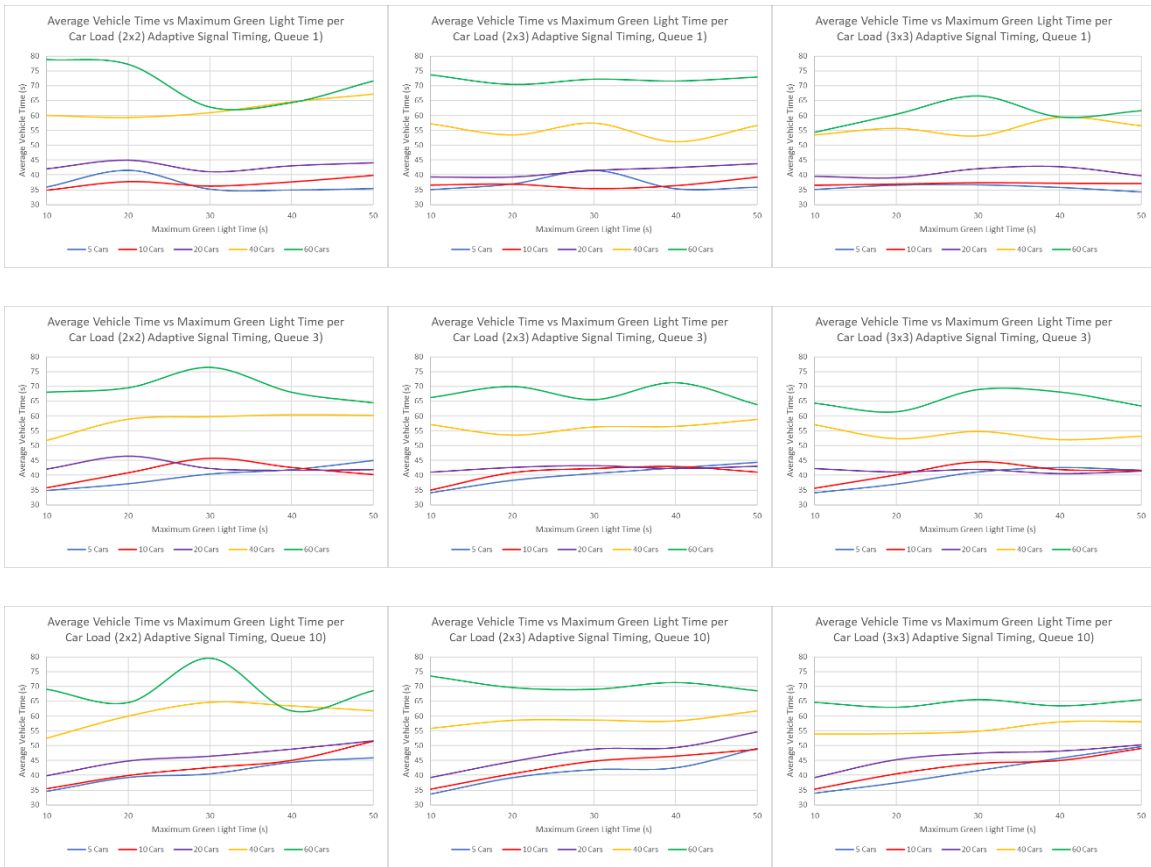


Figure 65: Overall Comparison of Average Vehicle Time from Adaptive Signal Implementation

4.4 Light Sequence Comparison

Furthermore, to consider the benefits of the adaptive traffic signal, details of each traffic light in a 2x2 scenario can be observed based on the gathered data. For consistency, the 2x2 intersections graphs will be compared while 40 vehicles are on the road. Figure 68 displays trendlines of the average vehicle time by comparing all car counts and signal types for a 2x2 intersection setup.

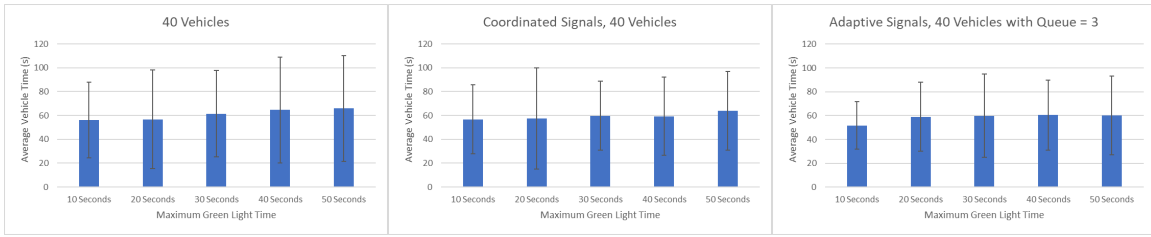


Figure 66: Various Signal Sequence Comparison of Average Vehicle Time

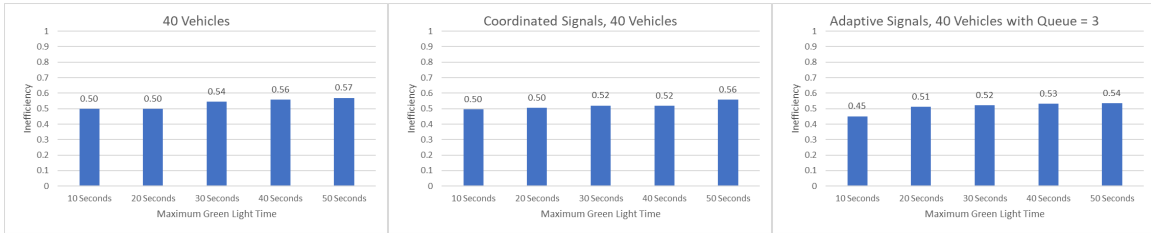


Figure 67: Various Signal Sequence Comparison of Inefficiency

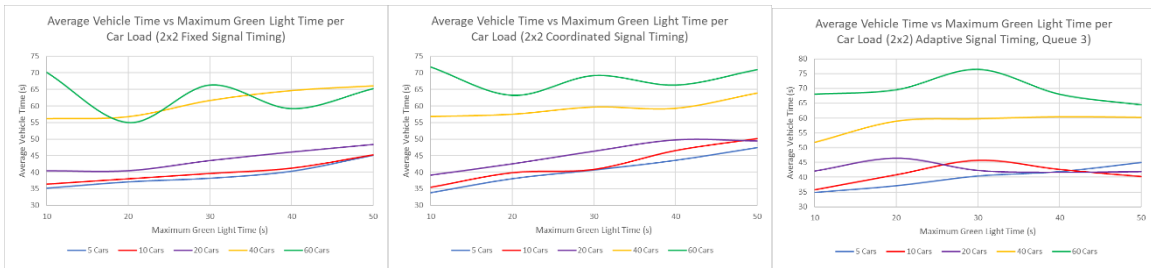


Figure 68: Trendline Comparison of Various Signal Sequences

As can be seen here, by comparing the basic light timing sequence to the coordinated and adaptive sequences, the overall time per vehicle is reduced as well as the inefficiency especially with 40 vehicles on the road. In some cases, the coordinated signal may be more beneficial compared to the adaptive but that is generally based on the maximum queue value. As previously mentioned, this research only considers a fixed queue value per simulation and overall, a lower vehicle load will require a smaller queue value to improve traffic flow. It will eventually be beneficial to consider an adaptive queue for greater intersection improvement.

5. Conclusions and Future Work

5.1 Research Conclusion

All in all, the data outlining the comparisons between the different signal abilities proved that the adaptive signal does increase intersection performance with respect to the metrics considered. Using this data and methods appropriately coded for different scenarios shows that this new type of communication can be implemented into the real world. The MATLAB method allows for different types of inputs that would be compared to vehicle behavior and an intersection setup seen today. The various details that can be implemented allow for a numerous amount of simulations to be carried out.

5.2 Future Work

Future work for this research can be taken in several directions. First, in this specific research, the queue size was fixed for different scenarios. A new research method would be to implement an adaptive queue size for the number of vehicles on the road or the duration of vehicles in the queue. The data already gathered from this research can be used to create a linear or quadratic maximum queue count for individual intersections or the overall intersection setup. This could depend on the number of vehicles proceeding through one intersection which may require that specific intersection to allow for an adaptive queue size.

Another opportunity for future work would be the individual intersection setup. Common intersections today have 2 or more lanes approaching from an individual

direction. Many intersections also include a designated left turn lane which may assist with traffic flow improvement as well. Allowing these different types of intersections to communicate with each other (I2I) as well as with nearby vehicles (I2V) adds complexity on a new level. Queue sizes will need to then be adjusted potentially per number of lanes and for a potential left turn lane. Given the wide variety of intersection setups that are seen today, the possibilities are endless.

A final opportunity for future work is related to the different types of connected agents. In a real intersection, more types of dynamic components are found throughout. Examples of more components may include but are not limited to pedestrians, bikers, electric scooters, and pets. For improved safety, it will be beneficial to consider these components as agents as well. This will ensure autonomous vehicles will know one of these components is nearby regardless of camera technology ability. These extra possibilities that can be considered will add more complexity to the system but the ability to model this will be beneficial for improving safety.

With overall implementation of this future work, intersection performance can be evaluated and improved regarding average vehicle time, resilience, and safety. The newer technologies for individual autonomous vehicles allow for connected vehicles in city intersections to be implemented. The addition of agent-based communication for improved performance will greatly enhance the transportation infrastructure.

5.3 Thesis Reflection

All in all, the use of agent-based communication for improved decision making has been proven effective. Referring to the design statement, it has been shown that improving agent-based Infrastructure to Infrastructure (I2I) communication and decision making does provide performance benefits to traffic flow capacities.

The initial communication of queue size from the vehicle to the traffic light (V2I) allows the intersection to make an appropriate decision for the status based on the load of traffic. This change of state based on the queue is then communicated to nearby traffic signals. Next, the addition of agent communication between traffic signals allows for further improved decision making. The I2I addition is the main area of improvement for the transportation infrastructure. This improved ability allows intersections to communicate status effectively and the coordinated approach demonstrates success of this improvement. From the individual light status change based on the level of traffic to the communication between intersections, a level of connectivity is created between vehicles and traffic lights that are at different intersections. The traffic flow is then further optimized as the intelligent signals communicate and adjust individual status based on nearby intersection signal updates.

The decision-making process and improved communication through intersections (I2I) is proven effective and can be implemented throughout the real world as overall vehicle technology improves. This has been proven through multiple scenario simulations regarding various city intersection setups, load of traffic present throughout the

simulation, and for alternative maximum green signal times. The overall ability to reduce average vehicle time through an intersection and reduce inefficiency is possible through adaptive signals and nearby intersection communication.

6. References

Abdulhai, B., Pringle, R., & Karakoulas, G. J. (2003). Reinforcement learning for true adaptive traffic signal control. *Journal of Transportation Engineering*, 129(3), 278-285.

Aboudolas, K., Papageorgiou, M., Kouvelas, A., & Kosmatopoulos, E. (2010). A rolling-horizon quadratic-programming approach to the signal control problem in large-scale congested urban road networks. *Transportation Research Part C: Emerging Technologies*, 18(5), 680-694.

Arel, I., Liu, C., Urbanik, T., & Kohls, A. G. (2010). Reinforcement learning-based multi-agent system for network traffic signal control. *IET Intelligent Transport Systems*, 4(2), 128-135.

Ashtiani, F., Fayazi, S. A., & Vahidi, A. (2018, June). Multi-Intersection Traffic Management for Autonomous Vehicles via Distributed Mixed Integer Linear Programming. In *2018 Annual American Control Conference (ACC)* (pp. 6341-6346). IEEE.

Bandyopadhyay, T., Won, K. S., Frazzoli, E., Hsu, D., Lee, W. S., & Rus, D. (2013). Intention-aware motion planning. In *Algorithmic Foundations of Robotics X* (pp. 475-491). Springer, Berlin, Heidelberg.

Brechtel, S., Gindele, T., & Dillmann, R. (2011, October). Probabilistic MDP-behavior planning for cars. In *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on* (pp. 1537-1542). IEEE.

Brechtel, S., Gindele, T., & Dillmann, R. (2014, October). Probabilistic decision-making under uncertainty for autonomous driving using continuous POMDPs. In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on* (pp. 392-399). IEEE.

Cunningham, A. G., Galceran, E., Eustice, R. M., & Olson, E. (2015, May). MPDM: Multipolicy decision-making in dynamic, uncertain environments for autonomous driving. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on* (pp. 1670-1677). IEEE.

DARPA (2007) DARPA Urban Challenge. <http://archive.darpa.mil/grandchallenge/>

Dolgov, D., Thrun, S., Montemerlo, M., & Diebel, J. (2010). Path planning for autonomous vehicles in unknown semi-structured environments. *The International Journal of Robotics Research*, 29(5), 485-501.

Fayazi, S. A., Vahidi, A., & Luckow, A. (2017, May). Optimal scheduling of autonomous vehicle arrivals at intelligent intersections via MILP. In *2017 American control conference (ACC)* (pp. 4920-4925). IEEE.

Fayazi, S. A., & Vahidi, A. (2017, August). Vehicle-in-the-loop (VIL) verification of a smart city intersection control scheme for autonomous vehicles. In *2017 IEEE Conference on Control Technology and Applications (CCTA)* (pp. 1575-1580). IEEE.

Feng, Y., Head, K. L., Khoshmashgham, S., & Zamanipour, M. (2015). A real-time adaptive signal control in a connected vehicle environment. *Transportation Research Part C: Emerging Technologies*, *55*, 460-473.

Ferguson, D., Darms, M., Urmson, C., & Kolski, S. (2008, June). Detection, prediction, and avoidance of dynamic obstacles in urban environments. In *Intelligent Vehicles Symposium, 2008 IEEE* (pp. 1149-1154). IEEE.

Ferguson, D., Howard, T. M., & Likhachev, M. (2008). Motion planning in urban environments. *Journal of Field Robotics*, *25*(11-12), 939-960.

Francis, R., & Bekera, B. (2014). A metric and frameworks for resilience analysis of engineered and infrastructure systems. *Reliability Engineering & System Safety*, *121*, 90-103.

Fulgenzi, C., Tay, C., Spalanzani, A., & Laugier, C. (2008, September). Probabilistic navigation in dynamic environment using rapidly-exploring random trees and gaussian processes. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on* (pp. 1056-1062). IEEE.

Galceran, E., Olson, E., & Eustice, R. M. (2015, September). Augmented vehicle tracking under occlusions for decision-making in autonomous driving. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on* (pp. 3559-3565). IEEE.

Galceran, E., Cunningham, A. G., Eustice, R. M., & Olson, E. (2015, July). Multipolicy Decision-Making for Autonomous Driving via Change-point-based Behavior Prediction. In *Robotics: Science and Systems* (pp. 2290-2297).

Goodall, N. J., Smith, B. L., & Park, B. (2013). Traffic signal control with connected vehicles. *Transportation Research Record*, *2381*(1), 65-72.

Hunt, P. B., Robertson, D. I., Bretherton, R. D., & Royle, M. C. (1982). The SCOOT on-line traffic signal optimisation technique. *Traffic Engineering & Control*, *23*(4).

Honda. (2018). *Honda Accord: Owner's Manual*.

Khan, S.M. and Chowdhury, M., 2019. Connected Vehicle Supported Adaptive Traffic Control for Near-congested Condition in a Mixed Traffic Stream. *arXiv preprint arXiv:1907.07243*.

Khan, S.M. and Chowdhury, M., 2019. Situation-Aware Left-Turning Connected and Automated Vehicle Operation at Signalized Intersections. *arXiv preprint arXiv:1908.00981*.

Lee, J., & Park, B. (2012). Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment. *IEEE Transactions on Intelligent Transportation Systems*, 13(1), 81-90.

Likhachev, M., & Ferguson, D. (2009). Planning long dynamically feasible maneuvers for autonomous vehicles. *The International Journal of Robotics Research*, 28(8), 933-945.

Luk, J. Y. K. (1984). Two traffic-responsive area traffic control methods: SCAT and SCOOT. *Traffic engineering & control*, 25(1).

McGehee, D. V., Mazzae, E. N., & Baldwin, G. S. (2000, July). Driver reaction time in crash avoidance research: validation of a driving simulator study on a test track. In *Proceedings of the human factors and ergonomics society annual meeting* (Vol. 44, No. 20, pp. 3-320). Sage CA: Los Angeles, CA: SAGE Publications.

Mladenovic, M. N., & Abbas, M. (2014, November). Priority-based intersection control framework for self-driving vehicles: Agent-based model development and evaluation. In *Connected Vehicles and Expo (ICCVE), 2014 International Conference on* (pp. 377-384). IEEE.

Miller, I., Campbell, M., Huttenlocher, D., Kline, F. R., Nathan, A., Lupashin, S., ... & Garcia, E. (2008). Team Cornell's Skynet: Robust perception and planning in an urban environment. *Journal of Field Robotics*, 25(8), 493-527.

Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D., Ettinger, S., ... & Johnston, D. (2008). Junior: The stanford entry in the urban challenge. *Journal of field Robotics*, 25(9), 569-597.

Petti, S., & Fraichard, T. (2005, August). Safe motion planning in dynamic environments. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on* (pp. 2210-2215). IEEE.

Smith, B. L., Venkatanarayana, R., Park, H., Goodall, N., Datesh, J., & Skerit, C. (2010). IntelliDriveSM traffic signal control algorithms. *University of Virginia*.

Stevanovic, A., Kergaye, C., & Martin, P. T. (2009, January). Scoot and scats: A closer look into their operations. In *88th Annual Meeting of the Transportation Research Board. Washington DC*.

Talebpour, A., & Mahmassani, H. S. (2016). Influence of connected and autonomous vehicles on traffic flow stability and throughput. *Transportation Research Part C: Emerging Technologies*, 71, 143-163.

Ulbrich, S., & Maurer, M. (2013, October). Probabilistic online POMDP decision making for lane changes in fully automated driving. In *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*(pp. 2063-2067). IEEE.

Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Clark, M. N., ... & Gittleman, M. (2008). Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8), 425-466.

Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Clark, M. N., ... & Gittleman, M. (2009). Autonomous driving in urban environments: Boss and the urban challenge. In *The DARPA Urban Challenge* (pp. 1-59). Springer, Berlin, Heidelberg.

Veres, S. M., Molnar, L., Lincoln, N. K., & Morice, C. P. (2011). Autonomous vehicle control systems—a review of decision making. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 225(2), 155-195.

Wei, J., Dolan, J. M., Snider, J. M., & Litkouhi, B. (2011, May). A point-based mdp for robust single-lane autonomous driving behavior under uncertainties. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on* (pp. 2586-2592). IEEE.

7. Appendix

```
%% Important Inputs to the Simulation
CarsPerInt = 10; % Maximum number of cars in the simulation per intersection
TotalTime = 100000; % Total time in seconds to run the simulation, large for MaxCar
variable
MaxCars = 500; % How many total cars will run through the simulation
timestep = 1/88; % Time step accuracy for cars (DO NOT CHANGE!)

% Properties of each Individual Intersection
SpeedLimit = 40; % mph
LaneLength = 800; % 800 feet normally
Window = 250; % display window

% Light Sequence Time in seconds
Red = 1; % Normal Range from 0.5 - 2.0 Seconds
Yellow = SpeedLimit/10; % Approximation based on Speed Limit
Greens = [10 20 30 40 50];

% Other Important Properties
CarLength = 16; % Constant Throughout Simulation
MaxQueue = 10; % Check for Light Change Based on Number of Stopped Vehicles

% Intersection Grid Setup
Rows = 3; % Max 10
Columns = 3; % Max 10

Cars = CarsPerInt*Rows*Columns;

% Simulation On/Off 1 = On, 0 = Off
Simulation = 0;
%% Multi Intersection Setup

Intersections = zeros(Rows*Columns,8);
IntNumber = 1; % IntersectionNumber

for R = 1:Rows
    for C = 1:Columns
        % Intersection Number
        Intersections(IntNumber,1) = IntNumber;

        % Intersection Center Location
        if C == 1
            Intersections(IntNumber,2) = 0;
        else
            Intersections(IntNumber,2) = (2*C*LaneLength) - (2*LaneLength); % X Center
        Location
        end

        if R == 1
            Intersections(IntNumber,3) = 0;
        else
            Intersections(IntNumber,3) = (-2*R*LaneLength) + (2*LaneLength); % Y Center
        Location
        end

        %Intersection Directions
        Intersections(IntNumber,4) = (IntNumber*4) - 3; % South
        Intersections(IntNumber,5) = (IntNumber*4) - 2; % West
        Intersections(IntNumber,6) = (IntNumber*4) - 1; % North
        Intersections(IntNumber,7) = (IntNumber*4); % East

        % New IntNumber for Next Iteration
```

```

        IntNumber = IntNumber + 1;

    end
end

%% Plotting the intersection (Used for visualization of the simulation)

% Multiple figures to display different intersections
% for I = 1:size(Intersections,1)
%
% % Figure Number
% figure(I + 1)
%
% hold on
% Line1 = plot([Intersections(I,2)-LaneLength,Intersections(I,2)-12],
[Intersections(I,3),Intersections(I,3)], '--');
% Line2 = plot([Intersections(I,2)-LaneLength,Intersections(I,2)-12],
[Intersections(I,3)-12,Intersections(I,3)-12]);
% Line3 = plot([Intersections(I,2)-LaneLength,Intersections(I,2)-12],
[Intersections(I,3)+12,Intersections(I,3)+12]);
% Line4 = plot([Intersections(I,2)+12,Intersections(I,2)+LaneLength],
[Intersections(I,3),Intersections(I,3)], '--');
% Line5 = plot([Intersections(I,2)+12,Intersections(I,2)+LaneLength],
[Intersections(I,3)-12,Intersections(I,3)-12]);
% Line6 = plot([Intersections(I,2)+12,Intersections(I,2)+LaneLength],
[Intersections(I,3)+12,Intersections(I,3)+12]);
%
% Line7 = plot([Intersections(I,2),Intersections(I,2)], [Intersections(I,3)-
LaneLength,Intersections(I,3)-12], '--');
% Line8 = plot([Intersections(I,2)-12,Intersections(I,2)-12], [Intersections(I,3)-
LaneLength,Intersections(I,3)-12]);
% Line9 = plot([Intersections(I,2)+12,Intersections(I,2)+12], [Intersections(I,3)-
LaneLength,Intersections(I,3)-12]);
% Line10 = plot([Intersections(I,2),Intersections(I,2)],
[Intersections(I,3)+12,Intersections(I,3)+LaneLength], '--');
% Line11 = plot([Intersections(I,2)-12,Intersections(I,2)-12],
[Intersections(I,3)+12,Intersections(I,3)+LaneLength]);
% Line12 = plot([Intersections(I,2)+12,Intersections(I,2)+12],
[Intersections(I,3)+12,Intersections(I,3)+LaneLength]);
%
%
% set(Line1,'color','black')
% set(Line2,'color','black')
% set(Line3,'color','black')
% set(Line4,'color','black')
% set(Line5,'color','black')
% set(Line6,'color','black')
% set(Line7,'color','black')
% set(Line8,'color','black')
% set(Line9,'color','black')
% set(Line10,'color','black')
% set(Line11,'color','black')
% set(Line12,'color','black')
% axis([Intersections(I,2)-Window Intersections(I,2)+Window Intersections(I,3)-Window
Intersections(I,3)+Window])
% grid on
% hold off
%
%
% end

% Total Grid of All Intersections
for I = 1:size(Intersections,1)

% Figure Number
figure(1) % figure(size(Intersections,1) + 1) for other simulations

```

```

hold on
Line1 = plot([Intersections(I,2)-LaneLength,Intersections(I,2)-12],
[Intersections(I,3),Intersections(I,3)], '--');
Line2 = plot([Intersections(I,2)-LaneLength,Intersections(I,2)-12], [Intersections(I,3)-
12,Intersections(I,3)-12]);
Line3 = plot([Intersections(I,2)-LaneLength,Intersections(I,2)-12],
[Intersections(I,3)+12,Intersections(I,3)+12]);
Line4 = plot([Intersections(I,2)+12,Intersections(I,2)+LaneLength],
[Intersections(I,3),Intersections(I,3)], '--');
Line5 = plot([Intersections(I,2)+12,Intersections(I,2)+LaneLength], [Intersections(I,3)-
12,Intersections(I,3)-12]);
Line6 = plot([Intersections(I,2)+12,Intersections(I,2)+LaneLength],
[Intersections(I,3)+12,Intersections(I,3)+12]);

Line7 = plot([Intersections(I,2),Intersections(I,2)], [Intersections(I,3)-
LaneLength,Intersections(I,3)-12], '--');
Line8 = plot([Intersections(I,2)-12,Intersections(I,2)-12], [Intersections(I,3)-
LaneLength,Intersections(I,3)-12]);
Line9 = plot([Intersections(I,2)+12,Intersections(I,2)+12], [Intersections(I,3)-
LaneLength,Intersections(I,3)-12]);
Line10 = plot([Intersections(I,2),Intersections(I,2)],
[Intersections(I,3)+12,Intersections(I,3)+LaneLength], '--');
Line11 = plot([Intersections(I,2)-12,Intersections(I,2)-12],
[Intersections(I,3)+12,Intersections(I,3)+LaneLength]);
Line12 = plot([Intersections(I,2)+12,Intersections(I,2)+12],
[Intersections(I,3)+12,Intersections(I,3)+LaneLength]);

% Intersection Line Features
set(Line1,'color','black')
set(Line2,'color','black')
set(Line3,'color','black')
set(Line4,'color','black')
set(Line5,'color','black')
set(Line6,'color','black')
set(Line7,'color','black')
set(Line8,'color','black')
set(Line9,'color','black')
set(Line10,'color','black')
set(Line11,'color','black')
set(Line12,'color','black')

end

axis([0-Window Intersections(Columns,2)+Window Intersections((Columns*Rows)-(Columns-
1),3)-Window 0+Window]);
grid on

%% Creating Cars

for Green = Greens(1):10:Greens(5)

% Creating Matrices for Car Evaluations
Locations = zeros(Cars,9);
CarBehavior = zeros(Cars,5);
Positions = zeros(Cars,10);
Time = zeros(Cars,6);
Light = zeros(Cars,9);
Observation = zeros(Cars,9);
TimeEvaluation = zeros(Cars, 12);
PositionEvaluation = zeros(1,Cars*2 + 1);
CarChange = zeros(Cars,2);

% Possibly Irrelevant
Overlap = zeros(Cars - 1, 9);

```

```

i = 1;
Exclude = 0;
TimeIdeal = 0;

while i <= Cars

% Choose random direction placement and intensity rating
Direction = randi([1,Intersections(size(Intersections,1),size(Intersections,2) -
2)],1,1);
% d = randsample(setdiff(1:3, 2), 1); % For testing intersections 1 and 3
Intensity = randi([1 10],1,1);
[Speedy, Accel, Timing] = IntensityRating(Intensity,SpeedLimit);

% Choosing a Random Output Direction Based on Intersection
[IntNumber,d] = find(Intersections(:,4:7) == Direction); % Finding row and column
Out = randsample(setdiff(Intersections(IntNumber,4):Intersections(IntNumber,7),
Direction), 1);

% Intersection X and Y Position
IntX = Intersections(IntNumber,2);
IntY = Intersections(IntNumber,3);

% Check if placement location is the same, if not go to original
k = find(Locations(:,5) == Direction); % Returns index or empty matrix
v = isempty(k); % determines if matrix is empty or not

if v == 0 % if the matrix is not empty (there is a match in direction)
% For a similar direction
% Car Number   X Position   Y Position   Ouput   Intersection   MPH

if d == 1
Option = Locations(k,3); % Other starting car locations
Exclude = zeros(length(Option)*41,1); % Initial matrix of excluding values

for e = 1:length(Option) % Run amount of times as other cars in that direction
Z = Option(e)-20:Option(e)+20; % Create 41 points that cannot be used

for f = 1:41
Exclude(41*(e-1)+f) = Z(f); % Put 5 points into exclude matrix
end

end

elseif d == 2
Option = Locations(k,2); % Other starting car locations
Exclude = zeros(length(Option)*41,1); % Initial matrix of excluding values

for e = 1:length(Option) % Run amount of times as other cars in that direction
Z = Option(e)-20:Option(e)+20; % Create 5 points that cannot be used
for f = 1:41
Exclude(41*(e-1)+f) = Z(f); % Put 5 points into exclude matrix
end

end

y = IntY - 9;
x = randsample(setdiff(IntX - LaneLength:IntX - 200, Exclude), 1);

elseif d == 3
Option = Locations(k,3); % Other starting car locations
Exclude = zeros(length(Option)*41,1); % Initial matrix of excluding values

for e = 1:length(Option) % Run amount of times as other cars in that direction
Z = Option(e)-20:Option(e)+20; % Create 5 points that cannot be used

```

```

        for f = 1:41
            Exclude(41*(e-1)+f) = Z(f); % Put 5 points into exclude matrix
        end
    end

    else
        Option = Locations(k,2); % Other starting car locations
        Exclude = zeros(length(Option)*41,1); % Initial matrix of excluding values

        for e = 1:length(Option) % Run amount of times as other cars in that direction
            Z = Option(e)-20:Option(e)+20; % Create 5 points that cannot be used
            for f = 1:41
                Exclude(41*(e-1)+f) = Z(f); % Put 5 points into exclude matrix
            end
        end
    end

end

else

% Option if a car has not yet been placed in this direction
if d == 1
    x = IntX + 3;
    y = randi([(IntY - LaneLength) (IntY - 200)],1,1);
elseif d == 2
    x = randi([(IntX - LaneLength) (IntX - 200)],1,1);
    y = IntY - 9;
elseif d == 3
    x = IntX - 9;
    y = randi([(IntY + 200 - CarLength) (IntY + LaneLength - CarLength)],1,1);
else
    x = randi([(IntX + 200 - CarLength) (IntX + LaneLength - CarLength)],1,1);
    y = IntY + 3;
end

end

if v == 1

elseif length(Exclude) > length(IntX + 200 - CarLength: IntX + LaneLength - CarLength)
    % If the random direction is full, place the car elsewhere
    i = i - 1;
else
    if d == 1
        x = IntX + 3;
        y = randsample(setdiff(IntY - LaneLength: IntY - 200, Exclude), 1); % choose placement
        that has not been occupied
    elseif d == 2
        y = IntY - 9;
        x = randsample(setdiff(IntX - LaneLength: IntX - 200, Exclude), 1);
    elseif d == 3
        x = IntX - 9;
        y = randsample(setdiff(IntY + 200 - CarLength: IntY + LaneLength - CarLength,
        Exclude), 1);
    else
        y = IntY + 3;
        x = randsample(setdiff(IntX + 200 - CarLength: IntX + LaneLength - CarLength,
        Exclude), 1);
    end
end

end

% Input Values for the Location Matrix

```

```

% Car Number(1)   X Position(2)   Y Position(3)   OutDirection(4)   InDirection(5)   MPH(6)
Locations(i,1) = i;
Locations(i,2) = x;
Locations(i,3) = y;

% Output Direction
[IntNumber,0] = find(Intersections(:,4:7) == Out); % Finding row and column

Locations(i,4) = Out; % output direction (Out)
Locations(i,5) = Direction;
Locations(i,6) = Speedy;
Locations(i,7) = IntNumber;
Locations(i,8) = d;
Locations(i,9) = 0;

% Values for the Car Behavior Matrix
CarBehavior(i,1) = i;
CarBehavior(i,2) = Intensity;
CarBehavior(i,3) = Speedy; % Normally based on intensity rating
CarBehavior(i,4) = Accel; % Normally based on intensity rating
CarBehavior(i,5) = Timing; % Normally based on intensity rating

% Values for Positions Matrix
Positions(i,1) = i;
Positions(i,5) = ToFPS(Speedy);
Positions(i,6) = ToFPS(Speedy);
Positions(i,7) = ToFPS(Speedy);

if d == 1
    Positions(i,2) = y;
    Positions(i,3) = (y - ToFPS(Speedy)*timestep);
    Positions(i,4) = (y - ToFPS(Speedy)*2*timestep);
elseif d == 2
    Positions(i,2) = x;
    Positions(i,3) = (x - ToFPS(Speedy)*timestep);
    Positions(i,4) = (x - ToFPS(Speedy)*2*timestep);
elseif d == 3
    Positions(i,2) = y;
    Positions(i,3) = (y + ToFPS(Speedy)*timestep);
    Positions(i,4) = (y + ToFPS(Speedy)*2*timestep);
else
    Positions(i,2) = x;
    Positions(i,3) = (x + ToFPS(Speedy)*timestep);
    Positions(i,4) = (x + ToFPS(Speedy)*2*timestep);
end

% Values for Time Matrix
Time(i,1) = i;
Time(i,6) = Timing;

% Values for Light Matrix
Light(i,1) = i;
Light(i,2) = 4;

% Initial Slowing Down Point for the Light
if d == 1
    Light(i,5) = IntY - 35 - (Positions(i,5))^2/(2*CarBehavior(i,4));
elseif d == 2
    Light(i,5) = IntX - 35 - (Positions(i,5))^2/(2*CarBehavior(i,4));
elseif d == 3
    Light(i,5) = IntY + 19 + (Positions(i,5))^2/(2*CarBehavior(i,4));
else

```

```

    Light(i,5) = IntX + 19 + (Positions(i,5))^2/(2*CarBehavior(i,4));
end

if d == 2 || d == 4
    Light(i,2) = 4;
    FC = i;
end

% Initial Time Evaluation
TimeEvaluation(i,1) = i;
TimeEvaluation(i,2) = Locations(i,5); % Input Direction
TimeEvaluation(i,3) = Locations(i,2); % Input X Location
TimeEvaluation(i,4) = Locations(i,3); % Input Y Location
TimeEvaluation(i,5) = 0;
TimeEvaluation(i,6) = Locations(i,4); % Output Direction

[TimeIdeal] = IdealTime(d,O,Speedy,Accel,LaneLength);
TimeEvaluation(i,11) = TimeIdeal;
TimeEvaluation(i,12) = Intensity;

% CarChange Matrix
CarChange(i,1) = i;

% Updated Car Value
i = i + 1;

end

%% Moving Cars Light Status

% Lights

jj = 0;

% Acceleration Initial Values
DistanceAccel = 0;

% Creating Traffic Light Matrix
TrafficLight = zeros(Rows*Columns*4,9);
LightStatus = zeros(100000,Rows*Columns*4 + 1);

for traf = 1:size(TrafficLight,1)
    TrafficLight(traf,1) = traf;
    LightStatus(1,traf + 1) = traf;
end

% Initial Light Status (Green Light in North and South Directions)
for L = 1:size(TrafficLight,1)

    if L == 1 || L == 3
        TrafficLight(L,2) = 2;
        TrafficLight(L,7) = 10;
    elseif mod(L,2) == 1 % Odd Number (North and South)
        TrafficLight(L,2) = 2;
        TrafficLight(L,7) = 100;
    else
        TrafficLight(L,2) = 0;
        TrafficLight(L,7) = 0;
    end
end

% Direction Transition Added to Traffic LightMatrix
Transition = zeros(size(TrafficLight,1),2);

```



```

% Transition Directions
for a = 1:Rows
    for b = 1:Columns - 1
        D = Intersections((a*Columns) - Columns + b,7);
        TrafficLight(D,3) = D + 2;
        TrafficLight(D+2,3) = D;
    end
end

for a = 1:Rows - 1
    for b = 1:Columns
        D = Intersections((a*Columns) - Columns + b,4);
        O = Intersections((a*Columns) + b,6);

        % Placing Appropriate Output Directions
        TrafficLight(D,3) = O;
        TrafficLight(O,3) = D;
    end
end

% Creating Countdown Values
for TLight = 1:size(TrafficLight,1)
    if TrafficLight(TLight,2) == 2
        Indicator = 1;
        Switch = 1;
    else
        Countdown = 0;
        Indicator = 0;
        Switch = 0;
    end

    TrafficLight(TLight,8) = Indicator;
    TrafficLight(TLight,9) = Switch;
end

DD = 1;

% Intersection Details Per Light
for TLight = 1:size(TrafficLight,1)
    if DD == 1
        TrafficLight(TLight,4) = TLight + 1; % Left Direction
        TrafficLight(TLight,5) = TLight + 2; % Straight Direction
        TrafficLight(TLight,6) = TLight + 3; % Right Direction
    elseif DD == 2
        TrafficLight(TLight,4) = TLight + 1; % Left Direction
        TrafficLight(TLight,5) = TLight + 2; % Straight Direction
        TrafficLight(TLight,6) = TLight - 1; % Right Direction
    elseif DD == 3
        TrafficLight(TLight,4) = TLight + 1; % Left Direction
        TrafficLight(TLight,5) = TLight - 2; % Straight Direction
        TrafficLight(TLight,6) = TLight - 1; % Right Direction
    else
        TrafficLight(TLight,4) = TLight - 3; % Left Direction
        TrafficLight(TLight,5) = TLight - 2; % Straight Direction
        TrafficLight(TLight,6) = TLight - 1; % Right Direction
    end

    DD = DD + 1;
    if DD > 4
        DD = 1;
    else
        end
end
end

```

```

% Intersection Number for Light
for II = 1:size(Intersections,1)
    TrafficLight(II*4 - 3,10) = II;
    TrafficLight(II*4 - 2,10) = II;
    TrafficLight(II*4 - 1,10) = II;
    TrafficLight(II*4 ,10) = II;

end

% Starting Direction for Intersections (Green North and South)
Intersections(:,8) = 2;
Intersections(:,9) = 1;

% Traffic Signal Graphic Values
for TLight = 1:size(TrafficLight,1)
    if TrafficLight(TLight,2) == 2
        if Intersections(TrafficLight(TLight,10),4) == TLight

plot(Intersections(TrafficLight(TLight,10),2),Intersections(TrafficLight(TLight,10),3) -
50, '.g', 'MarkerSize',15)
        elseif Intersections(TrafficLight(TLight,10),5) == TLight
            plot(Intersections(TrafficLight(TLight,10),2) -
50,Intersections(TrafficLight(TLight,10),3), '.g', 'MarkerSize',15)
            elseif Intersections(TrafficLight(TLight,10),6) == TLight

plot(Intersections(TrafficLight(TLight,10),2),Intersections(TrafficLight(TLight,10),3) +
50, '.g', 'MarkerSize',15)
                else
                    plot(Intersections(TrafficLight(TLight,10),2) +
50,Intersections(TrafficLight(TLight,10),3), '.g', 'MarkerSize',15)
                end
            else
                if Intersections(TrafficLight(TLight,10),4) == TLight

plot(Intersections(TrafficLight(TLight,10),2),Intersections(TrafficLight(TLight,10),3) -
50, '.r', 'MarkerSize',15)
                    elseif Intersections(TrafficLight(TLight,10),5) == TLight
                        plot(Intersections(TrafficLight(TLight,10),2) -
50,Intersections(TrafficLight(TLight,10),3), '.r', 'MarkerSize',15)
                        elseif Intersections(TrafficLight(TLight,10),6) == TLight

plot(Intersections(TrafficLight(TLight,10),2),Intersections(TrafficLight(TLight,10),3) +
50, '.r', 'MarkerSize',15)
                            else
                                plot(Intersections(TrafficLight(TLight,10),2) +
50,Intersections(TrafficLight(TLight,10),3), '.r', 'MarkerSize',15)
                            end
                        end
                    end
                end
            end
        end

% Time to run the simulation
% v = VideoWriter('Intersection.avi');
% open(v);

% TimeMatrix = zeros(88*TotalTime,1);
% BehindMatrix = zeros(88*TotalTime,1);
% SmallMatrix = zeros(88*TotalTime,1);
% VelocityMatrix = zeros(88*TotalTime,1);

% Intersection Transition Time (Based on Lane Length and Speed Limit)

```

```

TransitionTime = (LaneLength*2)/ToFPS(SpeedLimit);

Allie = 1;

for j = 0:(1/timestep)*TotalTime % Total Time to Run the Simulation
%% Updating Traffic Lights

% Adaptive Traffic Signal Calculations

for III = 1:size(Intersections,1)

% Adaptive Signal from Queue Overload
if Intersections(III,8) == 2 || Intersections(III,8) == -2
    for G = 1:4
        IntChoice = Intersections(III,G + 3); % Current Direction
        DirectionIndex = find(Locations(:,5) == IntChoice);
        Stopped = zeros(length(DirectionIndex),1);

        if isempty(Stopped)
        else
        for E = 1:length(Stopped)
            if Locations(DirectionIndex(E),6) == 0
                Stopped(E) = 1;
            else
                Stopped(E) = 0;
            end
        end
        end
        end

        if sum(Stopped) >= MaxQueue && Intersections(III,9) == 1;
            % Need to Switch the Light to a New Countdown
            % Remember to switch to the new light then countdown, dont keep
            % switching back to the new light time

            if G == 1 % Count Opposite Directions from this G

                % Right Side Furthest Car Time Calculation
                RightSide = find(Locations(:,5) == Intersections(III,7)); % Find Cars
Numbers to the Right (Direction 4)

                if isempty(RightSide)
                    RightTime = 1;
                else
                    [DistanceR,indexR] = max(Positions(RightSide,2));
                    RightCar = RightSide(indexR,1);

                    if Locations(RightCar,9) == 2 % Straight Output
                        RightDistance = Intersections(III,2) + 12 +
ToFPS(CarBehavior(RightCar,3))*Yellow;
                    elseif Locations(RightCar,9) == 3 % Right Output
                        RightDistance = Intersections(III,2) + 6 +
ToFPS(CarBehavior(RightCar,3))*Yellow - (1/2)*CarBehavior(RightCar,4)*Yellow^2;
                    elseif Locations(RightCar,9) == 1 % Left Output
                        RightDistance = Intersections(III,2) + 0 +
ToFPS(CarBehavior(RightCar,3))*Yellow - (1/2)*CarBehavior(RightCar,4)*Yellow^2;
                    end

                    RightTime = (Positions(RightCar,2) -
RightDistance)/Positions(RightCar,5);
                    end

                    % Left Side Furthest Car Time Calculation
                    LeftSide = find(Locations(:,5) == Intersections(III,5)); % Find Cars
Numbers to the Left (Direction 2)

                    if isempty(LeftSide)

```

```

        LeftTime = 1;
    else
        [DistanceL,indexL] = min(Positions(LeftSide,2));
        LeftCar = LeftSide(indexL,1);

        if Locations(LeftCar,9) == 2 % Straight Output
            LeftDistance = Intersections(III,2) - 28 -
ToFPS(CarBehavior(LeftCar,3))*Yellow;
        elseif Locations(LeftCar,9) == 3 % Right Output
            LeftDistance = Intersections(III,2) - 22 -
ToFPS(CarBehavior(LeftCar,3))*Yellow + (1/2)*CarBehavior(LeftCar,4)*Yellow^2;
        elseif Locations(LeftCar,9) == 1 % Left Output
            LeftDistance = Intersections(III,2) - 16 -
ToFPS(CarBehavior(LeftCar,3))*Yellow + (1/2)*CarBehavior(LeftCar,4)*Yellow^2;
        end

        LeftTime = (LeftDistance - Positions(LeftCar,2))/Positions(LeftCar,5);
    end

    if isempty(RightTime) && isempty(LeftTime)
        LeftTime = 1;
        RightTime = 1;
    elseif isempty(RightTime)
        RightTime = 1;
    elseif isempty(LeftTime)
        LeftTime = 1;
    end

    % New Light Countdown Time (Max of Right or Left)
    NewTime = max(RightTime,LeftTime);

    if NewTime < TrafficLight(Intersections(III,5),7) || NewTime <
TrafficLight(Intersections(III,7),7)
        TrafficLight(Intersections(III,5),7) = NewTime;
        TrafficLight(Intersections(III,7),7) = NewTime;

        % If Intersection can update Load Calculation
        Intersections(III,9) = 0;
    else
        Intersections(III,9) = 0;
    end

elseif G == 2

    % Right Side Furthest Car Time Calculation
    RightSide = find(Locations(:,5) == Intersections(III,4)); % Find Cars
Numbers to the Right (Direction 1)

    if isempty(RightSide)
        RightTime = 1;
    else

        [DistanceR,indexR] = min(Positions(RightSide,2));
        RightCar = RightSide(indexR,1);

        if Locations(RightCar,9) == 3 % Straight Output
            RightDistance = Intersections(III,3) - 28 -
ToFPS(CarBehavior(RightCar,3))*Yellow;
        elseif Locations(RightCar,9) == 4 % Right Output
            RightDistance = Intersections(III,3) - 22 -
ToFPS(CarBehavior(RightCar,3))*Yellow + (1/2)*CarBehavior(RightCar,4)*Yellow^2;
        elseif Locations(RightCar,9) == 2 % Left Output
            RightDistance = Intersections(III,3) - 16 -
ToFPS(CarBehavior(RightCar,3))*Yellow + (1/2)*CarBehavior(RightCar,4)*Yellow^2;
        end

```

```

        RightTime = (RightDistance -
Positions(RightCar,2))/Positions(RightCar,5);
        end

        % Left Side Furthest Car Time Calculation
        LeftSide = find(Locations(:,5) == Intersections(III,6)); % Find Cars
Numbers to the Left (Direction 3)

        if isempty(LeftSide)
            LeftTime = 1;
        else
            [DistanceL,indexL] = max(Positions(LeftSide,2));
            LeftCar = LeftSide(indexL,1);

            if Locations(LeftCar,9) == 1 % Straight Output
                LeftDistance = Intersections(III,3) + 12 +
ToFPS(CarBehavior(LeftCar,3))*Yellow;
            elseif Locations(LeftCar,9) == 2 % Right Output
                LeftDistance = Intersections(III,3) + 6 +
ToFPS(CarBehavior(LeftCar,3))*Yellow - (1/2)*CarBehavior(LeftCar,4)*Yellow^2;
            elseif Locations(LeftCar,9) == 4 % Left Output
                LeftDistance = Intersections(III,3) + 0 +
ToFPS(CarBehavior(LeftCar,3))*Yellow - (1/2)*CarBehavior(LeftCar,4)*Yellow^2;
            end

            LeftTime = (Positions(LeftCar,2) - LeftDistance)/Positions(LeftCar,5);
        end

        if isempty(RightTime) && isempty(LeftTime)
            LeftTime = 1;
            RightTime = 1;
        elseif isempty(RightTime)
            RightTime = 1;
        elseif isempty(LeftTime)
            LeftTime = 1;
        end

        % New Light Countdown Time (Max of Right or Left)
        NewTime = max(RightTime,LeftTime);

        if NewTime < TrafficLight(Intersections(III,4),7) || NewTime <
TrafficLight(Intersections(III,6),7)
            TrafficLight(Intersections(III,4),7) = NewTime;
            TrafficLight(Intersections(III,6),7) = NewTime;

            % If Intersection can update Load Calculation
            Intersections(III,9) = 0;
        else
            Intersections(III,9) = 0;
        end

    elseif G == 3

        % Right Side Furthest Car Time Calculation
        RightSide = find(Locations(:,5) == Intersections(III,5)); % Find Cars
Numbers to the Right (Direction 2)

        if isempty(RightSide)
            RightTime = 1;
        else
            [DistanceR,indexR] = min(Positions(RightSide,2));
            RightCar = RightSide(indexR,1);

            if Locations(RightCar,9) == 4 % Straight Output
                RightDistance = Intersections(III,2) - 28 -
ToFPS(CarBehavior(RightCar,3))*Yellow;

```

```

        elseif Locations(RightCar,9) == 1 % Right Output
            RightDistance = Intersections(III,2) - 22 -
ToFPS(CarBehavior(RightCar,3))*Yellow + (1/2)*CarBehavior(RightCar,4)*Yellow^2;
        elseif Locations(RightCar,9) == 3 % Left Output
            RightDistance = Intersections(III,2) - 16 -
ToFPS(CarBehavior(RightCar,3))*Yellow + (1/2)*CarBehavior(RightCar,4)*Yellow^2;
        end

        RightTime = (RightDistance -
Positions(RightCar,2))/Positions(RightCar,5);
        end

        % Left Side Furthest Car Time Calculation
        LeftSide = find(Locations(:,5) == Intersections(III,7)); % Find Cars
Numbers to the Left (Direction 4)

        if isempty(LeftSide)
            LeftTime = 1;
        else
            [DistanceL,indexL] = max(Positions(LeftSide,2));
            LeftCar = LeftSide(indexL,1);

            if Locations(LeftCar,9) == 2 % Straight Output
                LeftDistance = Intersections(III,2) + 12 +
ToFPS(CarBehavior(LeftCar,3))*Yellow;
            elseif Locations(LeftCar,9) == 3 % Right Output
                LeftDistance = Intersections(III,2) + 6 +
ToFPS(CarBehavior(LeftCar,3))*Yellow - (1/2)*CarBehavior(LeftCar,4)*Yellow^2;
            elseif Locations(LeftCar,9) == 1 % Left Output
                LeftDistance = Intersections(III,2) + 0 +
ToFPS(CarBehavior(LeftCar,3))*Yellow - (1/2)*CarBehavior(LeftCar,4)*Yellow^2;
            end

            LeftTime = (Positions(LeftCar,2) - LeftDistance)/Positions(LeftCar,5);
        end

        if isempty(RightTime) && isempty(LeftTime)
            LeftTime = 1;
            RightTime = 1;
        elseif isempty(RightTime)
            RightTime = 1;
        elseif isempty(LeftTime)
            LeftTime = 1;
        end

        % New Light Countdown Time (Max of Right or Left)
        NewTime = max(RightTime,LeftTime);

        if NewTime < TrafficLight(Intersections(III,5),7) || NewTime <
TrafficLight(Intersections(III,7),7)
            TrafficLight(Intersections(III,5),7) = NewTime;
            TrafficLight(Intersections(III,7),7) = NewTime;

            % If Intersection can update Load Calculation
            Intersections(III,9) = 0;
        else
            Intersections(III,9) = 0;
        end
    end

else

        % Right Side Furthest Car Time Calculation
        RightSide = find(Locations(:,5) == Intersections(III,6)); % Find Cars
Numbers to the Right (Direction 3)

        if isempty(RightSide)
            RightTime = 1;

```

```

else
    [DistanceR,indexR] = max(Positions(RightSide,2));
    RightCar = RightSide(indexR,1);

    if Locations(RightCar,9) == 1 % Straight Output
        RightDistance = Intersections(III,3) + 12 +
ToFPS(CarBehavior(RightCar,3))*Yellow;
    elseif Locations(RightCar,9) == 2 % Right Output
        RightDistance = Intersections(III,3) + 6 +
ToFPS(CarBehavior(RightCar,3))*Yellow - (1/2)*CarBehavior(RightCar,4)*Yellow^2;
    elseif Locations(RightCar,9) == 4 % Left Output
        RightDistance = Intersections(III,3) + 0 +
ToFPS(CarBehavior(RightCar,3))*Yellow - (1/2)*CarBehavior(RightCar,4)*Yellow^2;
    end

    RightTime = (Positions(RightCar,2) -
RightDistance)/Positions(RightCar,5);
    end

    % Left Side Furthest Car Time Calculation
    LeftSide = find(Locations(:,5) == Intersections(III,4)); % Find Cars
Numbers to the Left (Direction 1)

    if isempty(LeftSide)
        LeftTime = 1;
    else
        [DistanceL,indexL] = min(Positions(LeftSide,2));
        LeftCar = LeftSide(indexL,1);

        if Locations(LeftCar,9) == 3 % Straight Output
            LeftDistance = Intersections(III,3) - 28 -
ToFPS(CarBehavior(LeftCar,3))*Yellow;
        elseif Locations(LeftCar,9) == 4 % Right Output
            LeftDistance = Intersections(III,3) - 22 -
ToFPS(CarBehavior(LeftCar,3))*Yellow + (1/2)*CarBehavior(LeftCar,4)*Yellow^2;
        elseif Locations(LeftCar,9) == 2 % Left Output
            LeftDistance = Intersections(III,3) - 16 -
ToFPS(CarBehavior(LeftCar,3))*Yellow + (1/2)*CarBehavior(LeftCar,4)*Yellow^2;
        end

        LeftTime = (LeftDistance - Positions(LeftCar,2))/Positions(LeftCar,5);
    end

    if isempty(RightTime) && isempty(LeftTime)
        LeftTime = 1;
        RightTime = 1;
    elseif isempty(RightTime)
        RightTime = 1;
    elseif isempty(LeftTime)
        LeftTime = 1;
    end

    % New Light Countdown Time (Max of Right or Left)
    NewTime = max(RightTime,LeftTime);

    if NewTime < TrafficLight(Intersections(III,4),7) || NewTime <
TrafficLight(Intersections(III,6),7)
        TrafficLight(Intersections(III,4),7) = NewTime;
        TrafficLight(Intersections(III,6),7) = NewTime;

        % If Intersection can update Load Calculation
        Intersections(III,9) = 0;
    else
        Intersections(III,9) = 0;
    end
end
end
end
end
end

```

```

end
end

% Intelligent Traffic Signal Calculations from Neighboring Lights
% (Only update if signal timing is bigger than the actual car load time calculation)

if Intersections(III,9) == 1

    if Intersections(III,8) == 1 && TrafficLight(Intersections(III,4),7) == Yellow
        if Intersections(III,2) < max(Intersections(:,2)) && Intersections(III,3) >
min(Intersections(:,3))
            if Green > TransitionTime
                TrafficLight(Intersections(III + 1,4),7) = TransitionTime;
                TrafficLight(Intersections(III + 1,6),7) = TransitionTime;
                TrafficLight(Intersections(III + Columns,4),7) = TransitionTime;
                TrafficLight(Intersections(III + Columns,6),7) = TransitionTime;
            else
                TrafficLight(Intersections(III + 1,4),7) = Green;
                TrafficLight(Intersections(III + 1,6),7) = Green;
                TrafficLight(Intersections(III + Columns,4),7) = Green;
                TrafficLight(Intersections(III + Columns,6),7) = Green;
            end
        end

        elseif Intersections(III,2) == max(Intersections(:,2)) && Intersections(III,3) ==
min(Intersections(:,3))
            % Do Nothing
        elseif Intersections(III,2) == max(Intersections(:,2)) || Intersections(III,3) ==
min(Intersections(:,3))
            if Intersections(III,2) == max(Intersections(:,2))
                if Green > TransitionTime
                    TrafficLight(Intersections(III + Columns,4),7) = TransitionTime;
                    TrafficLight(Intersections(III + Columns,6),7) = TransitionTime;
                else
                    TrafficLight(Intersections(III + Columns,4),7) = Green;
                    TrafficLight(Intersections(III + Columns,6),7) = Green;
                end
            end
        else
            if Green > TransitionTime
                TrafficLight(Intersections(III + 1,4),7) = TransitionTime;
                TrafficLight(Intersections(III + 1,6),7) = TransitionTime;
            else
                TrafficLight(Intersections(III + 1,4),7) = Green;
                TrafficLight(Intersections(III + 1,6),7) = Green;
            end
        end
    end
end

% West and East Direction Light Configuration
elseif Intersections(III,8) == -1 && TrafficLight(Intersections(III,5),7) == Yellow
    if Intersections(III,2) < max(Intersections(:,2)) && Intersections(III,3) >
min(Intersections(:,3))
        if Green > TransitionTime
            TrafficLight(Intersections(III + 1,5),7) = TransitionTime;
            TrafficLight(Intersections(III + 1,7),7) = TransitionTime;
            TrafficLight(Intersections(III + Columns,5),7) = TransitionTime;
            TrafficLight(Intersections(III + Columns,7),7) = TransitionTime;
        else
            TrafficLight(Intersections(III + 1,5),7) = Green;
            TrafficLight(Intersections(III + 1,7),7) = Green;
            TrafficLight(Intersections(III + Columns,5),7) = Green;
            TrafficLight(Intersections(III + Columns,7),7) = Green;
        end
    end
end

```



```

        elseif Intersections(III,2) == max(Intersections(:,2)) && Intersections(III,3) ==
min(Intersections(:,3))
            % Do Nothing
        elseif Intersections(III,2) == max(Intersections(:,2)) || Intersections(III,3) ==
min(Intersections(:,3))
            if Intersections(III,2) == max(Intersections(:,2))
                if Green > TransitionTime
                    TrafficLight(Intersections(III + Columns,5),7) = TransitionTime;
                    TrafficLight(Intersections(III + Columns,7),7) = TransitionTime;
                else
                    TrafficLight(Intersections(III + Columns,5),7) = Green;
                    TrafficLight(Intersections(III + Columns,7),7) = Green;
                end
            end
        else
            if Green > TransitionTime
                TrafficLight(Intersections(III + 1,5),7) = TransitionTime;
                TrafficLight(Intersections(III + 1,7),7) = TransitionTime;
            else
                TrafficLight(Intersections(III + 1,5),7) = Green;
                TrafficLight(Intersections(III + 1,7),7) = Green;
            end
        end
    end
end

else
end

end

% Traffic Light Operation From Intelligent Calculations
for T = 1:size(TrafficLight,1)

    if TrafficLight(T,7) >= 0 % Counting down Green Light
        TrafficLight(T,7) = TrafficLight(T,7) - timestep;
    elseif TrafficLight(T,2) == 2 && TrafficLight(T,7) <= 0 % Changing Green to Yellow
        TrafficLight(T,2) = 1;
        TrafficLight(T,7) = Yellow;

    if Simulation == 1
        if Intersections(TrafficLight(T,10),4) == T
            plot(Intersections(TrafficLight(T,10),2),Intersections(TrafficLight(T,10),3)
- 50, '.y', 'MarkerSize',15)
        elseif Intersections(TrafficLight(T,10),5) == T
            plot(Intersections(TrafficLight(T,10),2) -
50,Intersections(TrafficLight(T,10),3), '.y', 'MarkerSize',15)
        elseif Intersections(TrafficLight(T,10),6) == T
            plot(Intersections(TrafficLight(T,10),2),Intersections(TrafficLight(T,10),3)
+ 50, '.y', 'MarkerSize',15)
        else
            plot(Intersections(TrafficLight(T,10),2) +
50,Intersections(TrafficLight(T,10),3), '.y', 'MarkerSize',15)
        end
    end

    elseif TrafficLight(T,2) == 1 && TrafficLight(T,7) <= 0 && TrafficLight(T,8) == 1 %
Changing Yellow to Red
        TrafficLight(T,2) = 0;
        TrafficLight(T,7) = 0;
        TrafficLight(T,8) = 0;
    end
end

```

```

        TrafficLight(TrafficLight(T,4),8) = 1; % Light to left Indicator at 1
if Simulation == 1
    if Intersections(TrafficLight(T,10),4) == T
        plot(Intersections(TrafficLight(T,10),2),Intersections(TrafficLight(T,10),3)
- 50, '.r', 'MarkerSize',15)
    elseif Intersections(TrafficLight(T,10),5) == T
        plot(Intersections(TrafficLight(T,10),2) -
50,Intersections(TrafficLight(T,10),3), '.r', 'MarkerSize',15)
    elseif Intersections(TrafficLight(T,10),6) == T
        plot(Intersections(TrafficLight(T,10),2),Intersections(TrafficLight(T,10),3)
+ 50, '.r', 'MarkerSize',15)
    else
        plot(Intersections(TrafficLight(T,10),2) +
50,Intersections(TrafficLight(T,10),3), '.r', 'MarkerSize',15)
    end
end

elseif T > TrafficLight(T,5) && TrafficLight(T,2) == 0 && TrafficLight(T,7) <= 0 &&
TrafficLight(T,8) == 1 && TrafficLight(T,9) == 0 && TrafficLight(TrafficLight(T,5),8) ==
1
    TrafficLight(T,7) = Red;
    TrafficLight(T,9) = 1;

    TrafficLight(TrafficLight(T,5),7) = Red;
    TrafficLight(TrafficLight(T,5),9) = 1;

    TrafficLight(TrafficLight(T,4),9) = 0;
    TrafficLight(TrafficLight(T,6),9) = 0;

elseif T > TrafficLight(T,5) && TrafficLight(T,2) == 0 && TrafficLight(T,7) <= 0 &&
TrafficLight(T,8) == 1 && TrafficLight(T,9) == 1 && TrafficLight(TrafficLight(T,5),9) ==
1
    TrafficLight(T,2) = 2;
    TrafficLight(T,7) = Green;

    TrafficLight(TrafficLight(T,5),2) = 2;
    TrafficLight(TrafficLight(T,5),7) = Green;
    Intersections(TrafficLight(T,10),9) = 1;

if Simulation == 1
    if Intersections(TrafficLight(T,10),4) == T
        plot(Intersections(TrafficLight(T,10),2),Intersections(TrafficLight(T,10),3)
- 50, '.g', 'MarkerSize',15)
    elseif Intersections(TrafficLight(T,10),5) == T
        plot(Intersections(TrafficLight(T,10),2) -
50,Intersections(TrafficLight(T,10),3), '.g', 'MarkerSize',15)
    elseif Intersections(TrafficLight(T,10),6) == T
        plot(Intersections(TrafficLight(T,10),2),Intersections(TrafficLight(T,10),3)
+ 50, '.g', 'MarkerSize',15)
    else
        plot(Intersections(TrafficLight(T,10),2) +
50,Intersections(TrafficLight(T,10),3), '.g', 'MarkerSize',15)
    end

    if Intersections(TrafficLight(T - 2,10),4) == T - 2
        plot(Intersections(TrafficLight(T - 2,10),2),Intersections(TrafficLight(T -
2,10),3) - 50, '.g', 'MarkerSize',15)
    elseif Intersections(TrafficLight(T - 2,10),5) == T - 2
        plot(Intersections(TrafficLight(T - 2,10),2) -
50,Intersections(TrafficLight(T - 2,10),3), '.g', 'MarkerSize',15)
    elseif Intersections(TrafficLight(T - 2,10),6) == T - 2
        plot(Intersections(TrafficLight(T - 2,10),2),Intersections(TrafficLight(T -
2,10),3) + 50, '.g', 'MarkerSize',15)

```

```

        else
            plot(Intersections(TrafficLight(T - 2,10),2) +
50,Intersections(TrafficLight(T - 2,10),3), '.g', 'MarkerSize',15)
            end

end

end

% Filling in Light Status Matrix
LightStatus(j + 2,1) = j*timestep;
LightStatus(j + 2,T + 1) = TrafficLight(T,2);

end

% Updating Intersection Direction Change
for III = 1:size(Intersections,1)

    if TrafficLight(Intersections(III,4),2) == 2 || TrafficLight(Intersections(III,6),2)
== 2
        Intersections(III,8) = 2;
        elseif TrafficLight(Intersections(III,4),2) == 1 ||
TrafficLight(Intersections(III,6),2) == 1
            Intersections(III,8) = 1;
        elseif TrafficLight(Intersections(III,5),2) == 2 ||
TrafficLight(Intersections(III,7),2) == 2
            Intersections(III,8) = -2;
        elseif TrafficLight(Intersections(III,5),2) == 1 ||
TrafficLight(Intersections(III,7),2) == 1
            Intersections(III,8) = -1;
        else
            Intersections(III,8) = 0;
        end
end

end

TrafficLight;
Intersections;

for k = 1:Cars
%% Transition when Changing Direction

% Intersection Location
IntNumber = Locations(k,7);
IntX = Intersections(IntNumber,2);
IntY = Intersections(IntNumber,3);

if Locations(k,8) == 1 && Locations(k,9) ~= 3

    if Locations(k,9) == 4 && Positions(k,2) >= IntY - 22
        Positions(k,2) = IntX + 5;
        Locations(k,3) = IntY - 9;
        Locations(k,5) = Intersections(IntNumber,5);
        Locations(k,8) = 2;
        Locations(k,9) = 4;
        Light(k,2) = 5;

    elseif Locations(k,9) == 2 && Positions(k,2) >= IntY - 10
        Positions(k,2) = IntX - 11;

```

```

        Locations(k,3) = IntY + 3;
        Locations(k,5) = Intersections(IntNumber,7);
        Locations(k,8) = 4;
        Locations(k,9) = 2;
        Light(k,2) = 5;
    end

elseif Locations(k,8) == 2 && Locations(k,9) ~= 4

    if Locations(k,9) == 1 && Positions(k,2) >= IntX - 22
        Positions(k,2) = IntY - 21;
        Locations(k,2) = IntX - 9;
        Locations(k,5) = Intersections(IntNumber,6);
        Locations(k,8) = 3;
        Locations(k,9) = 1;
        Light(k,2) = 5;
    elseif Locations(k,9) == 3 && Positions(k,2) >= IntX - 10
        Positions(k,2) = IntY - 5;
        Locations(k,2) = IntX + 3;
        Locations(k,5) = Intersections(IntNumber,4);
        Locations(k,8) = 1;
        Locations(k,9) = 3;
        Light(k,2) = 5;
    end

elseif Locations(k,8) == 3 && Locations(k,9) ~= 1

    if Locations(k,9) == 2 && Positions(k,2) <= IntY + 6
        Positions(k,2) = IntX - 21;
        Locations(k,3) = IntY + 3;
        Locations(k,5) = Intersections(IntNumber,7);
        Locations(k,8) = 4;
        Locations(k,9) = 2;
        Light(k,2) = 5;
    elseif Locations(k,9) == 4 && Positions(k,2) <= IntY - 6
        Positions(k,2) = IntX - 5;
        Locations(k,3) = IntY - 9;
        Locations(k,5) = Intersections(IntNumber,5);
        Locations(k,8) = 2;
        Locations(k,9) = 4;
        Light(k,2) = 5;
    end

elseif Locations(k,8) == 4 && Locations(k,9) ~= 2

    if Locations(k,9) == 3 && Positions(k,2) <= IntX + 6
        Positions(k,2) = IntY + 5;
        Locations(k,2) = IntX + 3;
        Locations(k,5) = Intersections(IntNumber,4);
        Locations(k,8) = 1;
        Locations(k,9) = 3;
        Light(k,2) = 5;
    elseif Locations(k,9) == 1 && Positions(k,2) <= IntX - 6
        Positions(k,2) = IntY - 11;
        Locations(k,2) = IntX - 9;
        Locations(k,5) = Intersections(IntNumber,6);
        Locations(k,8) = 3;
        Locations(k,9) = 1;
        Light(k,2) = 5;
    end

end
end

```

```

%% Ensuring one car does not run into the car directly in front
% Find the location where the directions are the same
Direct = find(Locations(:,5) == Locations(k,5));
% Create the new matrix for same direction data
Check = zeros(length(Direct),9);
Check(1,:) = Locations(k,:); % Plot the first line

% Delete Location where it is the same as the first line
X = find(Direct(:) == Locations(k,1));
Direct(X, :) = [];

% Place rest of similar direction data into the check matrix
for z = 1:length(Direct)
    Check(z + 1,:) = Locations(Direct(z),:);
end

% Find which car is directly in front based on initial direction
% and have it slow down to the speed of front car if necessary
if Check(1,8) == 1
    % Create initial matrix to fill in distance values
    Distance = zeros(length(Direct),1);

    for m = 1:length(Direct)
        % Check the distance of each car relative to the first
        Distance(m) = Check(m + 1,3) - Check(1,3);
    end

    % Make sure the values is only considered as greater than 0 but
    % that the index considers all locations
    Small = min(Distance(Distance > 0));

elseif Check(1,8) == 2
    % Create initial matrix to fill in distance values
    Distance = zeros(length(Direct),1);

    for m = 1:length(Direct)
        % Check the distance of each car relative to the first
        Distance(m) = Check(m + 1,2) - Check(1,2);
    end

    % Make sure the values is only considered as greater than 0 but
    % that the index considers all locations
    Small = min(Distance(Distance > 0));

elseif Check(1,8) == 3
    % Create initial matrix to fill in distance values
    Distance = zeros(length(Direct),1);

    for m = 1:length(Direct)
        % Check the distance of each car relative to the first
        Distance(m) = Check(1,3) - Check(m + 1,3);
    end

    % Make sure the values is only considered as greater than 0 but
    % that the index considers all locations
    Small = min(Distance(Distance > 0));

else
    % Create initial matrix to fill in distance values
    Distance = zeros(length(Direct),1);

```

```

for m = 1:length(Direct)
    % Check the distance of each car relative to the first
    Distance(m) = Check(1,2) - Check(m + 1,2);
end

% Make sure the values is only considered as greater than 0 but
% that the index considers all locations
Small = min(Distance(Distance > 0));

end

if isempty(Small) % No cars in front of the current car
    if Positions(k,5) < ToFPS(CarBehavior(k,3)) && Positions(k,5) >= 0
        DistanceAccel = CarBehavior(k,4);
        Time(k,2) = 0;
    else
        DistanceAccel = 0;
        Time(k,2) = Inf;
    end

else

Index = find(Distance == Small);

% Front and Back Car Numbers
FC = Check(Index+1,1); % Number of the car in front
BC = Check(1,1); % Number of the back car (current car) = k

if k > FC

% Calculation for Current Time Behind and Desired Initial Time
[Current,Initial] = BehindTime(Positions(FC,3),Positions(k,2),Positions(FC,6),...
Positions(k,5),ToFPS(CarBehavior(k,3)),CarBehavior(k,4),CarBehavior(k,5),CarLength,Locations(k,5));

Time(k,2) = Current;
Time(k,5) = Initial;

% Appropriate Acceleration Calculation
Accel =
Acceleration(Positions(FC,3),Positions(k,2),Positions(k,5),CarBehavior(k,5),Locations(k,5));

% Desired Velocity and Time Errors
TimeError = abs(Current - CarBehavior(k,5));
VelocityError = Positions(k,5) - Positions(FC,6);

% Acceleration Calculation
[DistanceAccel] = CarInFront(Positions(FC,6),Positions(k,5),...
CarBehavior(k,4),Current,Time(k,3),Initial,CarBehavior(k,5),...
ToFPS(CarBehavior(k,3)),TimeError,VelocityError,Accel);

else % If the Back Car is a lower number than the front car

% Calculation for Current Time Behind and Desired Initial Time
[Current,Initial] = BehindTime(Positions(FC,2),Positions(k,2),Positions(FC,5),...
Positions(k,5),ToFPS(CarBehavior(k,3)),CarBehavior(k,4),CarBehavior(k,5),CarLength,Locations(k,5));

```

```

Time(k,2) = Current;
Time(k,5) = Initial;

% Appropriate Acceleration Calculation
Accel =
Acceleration(Positions(FC,2), Positions(k,2), Positions(k,5), CarBehavior(k,5), Locations(k,5)
);

% Desired Velocity and Time Errors
TimeError = abs(Current - CarBehavior(k,5));
VelocityError = Positions(k,5) - Positions(FC,5);

% Acceleration Calculation
[DistanceAccel] = CarInFront(Positions(FC,5), Positions(k,5), ...
    CarBehavior(k,4), Current, Time(k,3), Initial, CarBehavior(k,5), ...
    ToFPS(CarBehavior(k,3)), TimeError, VelocityError, Accel);

end

end

%% Decision at Instant Occurance of Yellow Light

if TrafficLight(Locations(k,5),2) == 1 && TrafficLight(Locations(k,5),7) == Yellow
    [Decision,xii] =
    LightTime(Positions(k,2), Locations(k,8), Positions(k,5), Yellow, CarBehavior(k,4), Locations(k,9), IntX, IntY);
    Light(k,2) = Decision;
    Light(k,9) = xii;
end

%     if Locations(k,8) == 1
%         [Decision,xii] =
LightTime(Positions(k,2), Locations(k,8), Positions(k,5), Yellow, CarBehavior(k,4), Locations(k,9), IntX, IntY);
%         Light(k,2) = Decision;
%         Light(k,9) = xii;
%     elseif Locations(k,8) == 3
%         [Decision,xii] =
LightTime(Positions(k,2), Locations(k,8), Positions(k,5), Yellow, CarBehavior(k,4), Locations(k,9), IntX, IntY);
%         Light(k,2) = Decision;
%         Light(k,9) = xii;
%     end
%
%
%
%
%     if Locations(k,8) == 2
%         [Decision,xii] =
LightTime(Positions(k,2), Locations(k,8), Positions(k,5), Yellow, CarBehavior(k,4), Locations(k,9), IntX, IntY);
%         Light(k,2) = Decision;
%         Light(k,9) = xii;
%     elseif Locations(k,8) == 4
%         [Decision,xii] =
LightTime(Positions(k,2), Locations(k,8), Positions(k,5), Yellow, CarBehavior(k,4), Locations(k,9), IntX, IntY);
%         Light(k,2) = Decision;
%         Light(k,9) = xii;
%     end

```

```

%% Straight Proceed Light Status Acceleration

% Calculate initial slowing down position

if Locations(k,8) == 1

    xi = Light(k,6) - (Positions(k,5))^2/(2*CarBehavior(k,4));
    Light(k,5) = xi; % Starting position for slowing down before the light

    if TrafficLight(Locations(k,5),2) == 0 % Red Light
        if Light(k,2) == 4 && Positions(k,5) < 0.5 && Positions(k,2) <= Light(k,6) + 1 &&
Positions(k,2) >= Light(k,6) - 1
            LightAccel = 0;
            Positions(k,5) = 0;
        elseif Positions(k,2) >= Light(k,5) && Light(k,2) == 4
            LightAccel = -Positions(k,5)^2/(2*(Light(k,6) - Positions(k,2)));
        elseif Positions(k,2) < Light(k,5) && Positions(k,5) < ToFPS(CarBehavior(k,3)) &&
Light(k,2) == 4
            LightAccel = CarBehavior(k,4);
        end

    elseif TrafficLight(Locations(k,5),2) == 1 % Yellow Light
        if Light(k,2) == 4 && Positions(k,5) < 0.5 && Positions(k,2) <= Light(k,6) + 1 &&
Positions(k,2) >= Light(k,6) - 1
            LightAccel = 0;
            Positions(k,5) = 0;
        elseif Light(k,2) == 4 && Positions(k,2) >= Light(k,5)
            LightAccel = -Positions(k,5)^2/(2*(Light(k,6) - Positions(k,2)));
        elseif Light(k,2) == 3
            LightAccel = 0; % Adjust for Faster Acceleration
        elseif Light(k,2) == 2
            LightAccel = 0;
        end

    else % Green Light
        if Time(k,2) == Inf || Time(k,2) < 0 || Time(k,5) == 0
            LightAccel = CarBehavior(k,4);
        end

    end

elseif Locations(k,8) == 2

    xi = Light(k,6) - (Positions(k,5))^2/(2*CarBehavior(k,4));
    Light(k,5) = xi; % Starting position for slowing down before the light

    if TrafficLight(Locations(k,5),2) == 0 % Red Light
        if Light(k,2) == 4 && Positions(k,5) < 0.5 && Positions(k,2) <= Light(k,6) + 1 &&
Positions(k,2) >= Light(k,6) - 1
            LightAccel = 0;
            Positions(k,5) = 0;
        elseif Positions(k,2) >= Light(k,5) && Light(k,2) == 4
            LightAccel = -Positions(k,5)^2/(2*(Light(k,6) - Positions(k,2)));
        elseif Positions(k,2) < Light(k,5) && Positions(k,5) < ToFPS(CarBehavior(k,3)) &&
Light(k,2) == 4
            LightAccel = CarBehavior(k,4);
        end

    elseif TrafficLight(Locations(k,5),2) == 1 % Yellow Light
        if Light(k,2) == 4 && Positions(k,5) < 0.5 && Positions(k,2) <= Light(k,6) + 1 &&
Positions(k,2) >= Light(k,6) - 1
            LightAccel = 0;
            Positions(k,5) = 0;
        elseif Light(k,2) == 4 && Positions(k,2) >= Light(k,5)

```



```

        LightAccel = -Positions(k,5)^2/(2*(Light(k,6) - Positions(k,2)));
    elseif Light(k,2) == 3
        LightAccel = 0; % Adjust for Faster Acceleration
    elseif Light(k,2) == 2
        LightAccel = 0;
    end

else % Green Light
    if Time(k,2) == Inf || Time(k,2) < 0 || Time(k,5) == 0
        LightAccel = CarBehavior(k,4);
    end

end

elseif Locations(k,8) == 3
    xi = Light(k,6) + (Positions(k,5))^2/(2*CarBehavior(k,4));
    Light(k,5) = xi; % Starting position for slowing down before the light

    if TrafficLight(Locations(k,5),2) == 0 % Red Light
        if Light(k,2) == 4 && Positions(k,5) < 0.5 && Positions(k,2) <= Light(k,6) + 1 &&
Positions(k,2) >= Light(k,6) - 1
            LightAccel = 0;
            Positions(k,5) = 0;
        elseif Positions(k,2) <= Light(k,5) && Light(k,2) == 4
            LightAccel = -Positions(k,5)^2/(2*(-Light(k,6) + Positions(k,2)));
        elseif Positions(k,2) > Light(k,5) && Positions(k,5) < ToFPS(CarBehavior(k,3)) &&
Light(k,2) == 4
            LightAccel = CarBehavior(k,4);
        end

    elseif TrafficLight(Locations(k,5),2) == 1 % Yellow Light
        if Light(k,2) == 4 && Positions(k,5) < 0.5 && Positions(k,2) <= Light(k,6) + 1 &&
Positions(k,2) >= Light(k,6) - 1
            LightAccel = 0;
            Positions(k,5) = 0;
        elseif Light(k,2) == 4 && Positions(k,2) <= Light(k,5)
            LightAccel = -Positions(k,5)^2/(2*(-Light(k,6) + Positions(k,2)));
        elseif Light(k,2) == 3
            LightAccel = 0; % Adjust for Faster Acceleration
        elseif Light(k,2) == 2
            LightAccel = 0;
        end

    else % Green Light
        if Time(k,2) == Inf || Time(k,2) < 0 || Time(k,5) == 0
            LightAccel = CarBehavior(k,4);
        end

    end

else
    xi = Light(k,6) + (Positions(k,5))^2/(2*CarBehavior(k,4));
    Light(k,5) = xi; % Starting position for slowing down before the light

    if TrafficLight(Locations(k,5),2) == 0 % Red Light
        if Light(k,2) == 4 && Positions(k,5) < 0.5 && Positions(k,2) <= Light(k,6) + 1 &&
Positions(k,2) >= Light(k,6) - 1
            LightAccel = 0;
            Positions(k,5) = 0;
        elseif Positions(k,2) <= Light(k,5) && Light(k,2) == 4
            LightAccel = -Positions(k,5)^2/(2*(-Light(k,6) + Positions(k,2)));
        elseif Positions(k,2) > Light(k,5) && Positions(k,5) < ToFPS(CarBehavior(k,3)) &&
Light(k,2) == 4
            LightAccel = CarBehavior(k,4);
        end
    end
end

```

```

elseif TrafficLight(Locations(k,5),2) == 1 % Yellow Light
    if Light(k,2) == 4 && Positions(k,5) < 0.5 && Positions(k,2) <= Light(k,6) + 1 &&
Positions(k,2) >= Light(k,6) - 1
        LightAccel = 0;
        Positions(k,5) = 0;
    elseif Light(k,2) == 4 && Positions(k,2) <= Light(k,5)
        LightAccel = -Positions(k,5)^2/(2*(-Light(k,6) + Positions(k,2)));
    elseif Light(k,2) == 3
        LightAccel = 0; % Adjust for Faster Acceleration
    elseif Light(k,2) == 2
        LightAccel = 0;
    end

else % Green Light
    if Time(k,2) == Inf || Time(k,2) < 0 || Time(k,5) == 0
        LightAccel = CarBehavior(k,4);
    end

end

end

%% Changing Directions at the Intersection (Right Turn Acceleration)
if Locations(k,8) == 1
    if TrafficLight(Locations(k,5),2) == 2 || TrafficLight(Locations(k,5),2) == 1 %
Green/Yellow Light
        xi = IntY - 28 - ((Positions(k,5)^2 - ToFPS(10)^2)/(2*CarBehavior(k,4))); %
Look at this calculation
        Light(k,7) = xi; % Starting position for slowing down before the light

        if Light(k,2) == 3 && Positions(k,2) < Light(k,9)
            TurnAccel = 0; % Stay Fast until Rushed Deceleration
        elseif Light(k,2) == 3 && Positions(k,2) >= Light(k,9)
            TurnAccel = -CarBehavior(k,4) - 3; % Rushed Deceleration
            if j == 5;
                end
        elseif Positions(k,2) >= Light(k,7) && Positions(k,2) < IntY - 28
            TurnAccel = -CarBehavior(k,4);
        else
            TurnAccel = 0;
        end
    else
        TurnAccel = (0 - Positions(k,5)^2)/(2*(Light(k,6) - Positions(k,2))); % Red
Light
    end
elseif Locations(k,8) == 2
    if TrafficLight(Locations(k,5),2) == 2 || TrafficLight(Locations(k,5),2) == 1 %
Green/Yellow Light
        xi = IntX - 28 - ((Positions(k,5)^2 - ToFPS(10)^2)/(2*CarBehavior(k,4))); %
Look at this calculation
        Light(k,7) = xi; % Starting position for slowing down before the light

        if Light(k,2) == 3 && Positions(k,2) < Light(k,9)
            TurnAccel = 0; % Stay Fast until Rushed Deceleration
        elseif Light(k,2) == 3 && Positions(k,2) >= Light(k,9)
            TurnAccel = -CarBehavior(k,4) - 3; % Rushed Deceleration
        elseif Positions(k,2) >= Light(k,7) && Positions(k,2) < IntX - 28
            TurnAccel = -CarBehavior(k,4);
        else
            TurnAccel = 0;
        end
    else
        TurnAccel = (0 - Positions(k,5)^2)/(2*(Light(k,6) - Positions(k,2))); % Red
Light

```

```

end

elseif Locations(k,8) == 3
    if TrafficLight(Locations(k,5),2) == 2 || TrafficLight(Locations(k,5),2) == 1 %
Green/Yellow Light
        xi = IntY + 12 + ((Positions(k,5)^2 - ToFPS(10)^2)/(2*CarBehavior(k,4))); %
Look at this calculation
        Light(k,7) = xi; % Starting position for slowing down before the light

        if Light(k,2) == 3 && Positions(k,2) > Light(k,9)
            TurnAccel = 0; % Stay Fast until Rushed Deceleration
        elseif Light(k,2) == 3 && Positions(k,2) <= Light(k,9)
            TurnAccel = -CarBehavior(k,4) - 3; % Rushed Deceleration
        elseif Positions(k,2) <= Light(k,7) && Positions(k,2) > IntY + 12
            TurnAccel = -CarBehavior(k,4);
        else
            TurnAccel = 0;
        end
    end
else
    TurnAccel = (0 - Positions(k,5)^2)/(2*(-Light(k,6) + Positions(k,2))); % Red
Light
end

else
    if TrafficLight(Locations(k,5),2) == 2 || TrafficLight(Locations(k,5),2) == 1 %
Green/Yellow Light
        xi = IntX + 12 + ((Positions(k,5)^2 - ToFPS(10)^2)/(2*CarBehavior(k,4))); %
Look at this calculation
        Light(k,7) = xi; % Starting position for slowing down before the light

        if Light(k,2) == 3 && Positions(k,2) > Light(k,9)
            TurnAccel = 0; % Stay Fast until Rushed Deceleration
        elseif Light(k,2) == 3 && Positions(k,2) <= Light(k,9)
            TurnAccel = -CarBehavior(k,4) - 3; % Rushed Deceleration
        elseif Positions(k,2) <= Light(k,7) && Positions(k,2) > IntX + 12
            TurnAccel = -CarBehavior(k,4);
        else
            TurnAccel = 0;
        end
    end
else
    TurnAccel = (0 - Positions(k,5)^2)/(2*(-Light(k,6) + Positions(k,2))); % Red
Light
end

end

%% Yielding at the Intersection (Left Turn Acceleration)

if Locations(k,8) == 1

    if TrafficLight(Locations(k,5),2) == 2 || TrafficLight(Locations(k,5),2) == 1 ||
TrafficLight(Locations(k,5),2) == 0
        xi = IntY - 20 - (Positions(k,5)^2 - ToFPS(10)^2)/(2*CarBehavior(k,4));
        Light(k,8) = xi; % Starting position for slowing down before the light
        % Adjust the xi for other directions

        if Light(k,2) == 3 && Positions(k,2) < Light(k,9)
            YieldAccel = 0; % Stay Fast until Rushed Deceleration
        elseif Light(k,2) == 3 && Positions(k,2) >= Light(k,9)
            YieldAccel = -CarBehavior(k,4) - 3; % Rushed Deceleration
            if j == 5;
                end
        end

        elseif Positions(k,2) >= IntY - 20 && Positions(k,2) < IntY - 10 % Yielding
to Opposing Traffic

```

```

% Check Traffic in Opposite Direction

% Find the location where the direction is opposite
Opposite = find(Locations(:,5) == Intersections(IntNumber,6));
% Create the new matrix for opposite direction data
Yield = zeros(length(Opposite),9);
% Yield(1,:) = Light(k,:); % Plot the first line

% Delete Location where it is the same as the first line
Y = find(Opposite(:) == Light(k,1));
Opposite(Y, :) = [];

% Check if there are cars near the intersection in the
% opposite direction
if isempty(Opposite)
    pp = 0;
else
    pp = 0;
    for z = 1:length(Opposite)
        if Light(Opposite(z),2) == 4 || Light(Opposite(z),2) == 5
            pp = pp + 0;
        else
            pp = pp + 1;
        end
    end
end

if pp == 0 && Positions(k,2) >= IntY - 20 && Positions(k,2) < IntY - 11
    && Positions(k,5) >= ToFPS(10)
        YieldAccel = 0;
    elseif pp >= 1 && Positions(k,2) >= IntY - 20 && Positions(k,2) < IntY -
11
        YieldAccel = (Positions(k,5)^2)/(2*(Positions(k,2) + IntY + 11));
    elseif pp == 0 && Positions(k,2) >= IntY - 20 && Positions(k,2) < IntY -
11 && Positions(k,5) < ToFPS(10)
        YieldAccel = CarBehavior(k,4);
    elseif pp >= 1 && Positions(k,2) > IntY - 12 && Positions(k,2) < IntY -
11
        YieldAccel = 0;
        Positions(k,5) = 0;
    elseif pp == 0 && Positions(k,2) > IntY - 11
        if Positions(k,5) > 1 && Positions(k,5) <= 2
            YieldAccel = 0;
        elseif Positions(k,5) > 2
            YieldAccel = -CarBehavior(k,4);
        else
            YieldAccel = 0;
            Positions(k,5) = 1;
        end
    elseif pp >= 1 && Positions(k,2) > IntY - 11 && Locations(k,4) ==
Intersections(IntNumber,5); % Watch for unsolvable Yield Accel
        YieldAccel = 0;
        Positions(k,5) = 2;
    end

elseif Positions(k,2) >= Light(k,8) && Positions(k,2) < IntY - 20
    YieldAccel = -CarBehavior(k,4);
else
    YieldAccel = 0;
end

end

elseif Locations(k,8) == 2

```

```

%         if TrafficLight(2,2) == 2 || TrafficLight(2,2) == 1
%             xi = -20 - (Positions(k,5)^2 - ToFPS(10)^2)/(2*CarBehavior(k,4));
%             Light(k,8) = xi; % Starting position for slowing down before the light
%         % Adjust the xi for other directions
%
%         if Light(k,2) == 3 && Positions(k,2) < Light(k,9)
%             YieldAccel = 0; % Stay Fast until Rushed Deceleration
%         elseif Light(k,2) == 3 && Positions(k,2) >= Light(k,9)
%             YieldAccel = -CarBehavior(k,4) - 3; % Rushed Deceleration
%             if j == 5;
%                 end
%
%         elseif Positions(k,2) >= -20 && Positions(k,2) < -10 % Yielding to Opposing
Traffic
%             % Check Traffic in Opposite Direction
%
%             % Find the location where the direction is opposite
%             Opposite = find(Locations(:,5) == 3);
%             % Create the new matrix for opposite direction data
%             Yield = zeros(length(Opposite),9);
%             % Yield(1,:) = Light(k,:); % Plot the first line
%
%             % Delete Location where it is the same as the first line
%             Y = find(Opposite(:) == Light(k,1));
%             Opposite(Y, :) = [];
%
%             if isempty(Opposite)
%                 pp = 0;
%             else
%                 pp = 0;
%                 for z = 1:length(Opposite)
%
%                     if Positions(Opposite(z),2) > 100 || Positions(Opposite(z),2)
< 0
%                         pp = pp + 0;
%                     else
%                         pp = pp + 1;
%                     end
%                 end
%             end
%
%             if pp == 0 && Positions(k,5) < 14 % If none of the opposite cars,
proceed with the yield
%                 YieldAccel = CarBehavior(k,4);
%             elseif pp == 0
%                 YieldAccel = 0;
%             elseif pp >= 1 && Positions(k,2) < -11 % If there is an opposite car
about to pass through intersection
%                 % Have cars that passed through turn to 4 so they aren't
%                 % counted This deceleration is kinda fast
%                 YieldAccel = (Positions(k,5)^2)/(2*(Positions(k,2) + 11));
%             else
%                 YieldAccel = 0;
%                 Positions(k,5) = 0;
%             end
%
%             if j == 0
%                 end
%
%         elseif Positions(k,2) >= Light(k,8) && Positions(k,2) < -20
%             YieldAccel = -CarBehavior(k,4);

```

```

%
%     else
%         YieldAccel = 0;
%     end
%
%     else
%         % Find the location where the direction is opposite
%         Opposite = find(Locations(:,5) == 3);
%         % Create the new matrix for opposite direction data
%         Yield = zeros(length(Opposite),9);
%         % Yield(1,:) = Light(k,:); % Plot the first line
%
%         % Delete Location where it is the same as the first line
%         Y = find(Opposite(:) == Light(k,1));
%         Opposite(Y, :) = [];
%
%         % Place rest of similar direction data into the check matrix
%
%         if isempty(Opposite)
%             pp = 0;
%         else
%             pp = 0;
%             for z = 1:length(Opposite)
%
%                 if Positions(Opposite(z),2) > 12
%                     pp = pp + 0;
%                 else
%                     pp = pp + 1;
%                 end
%             end
%         end
%     end
%
%     end
%
%     if pp == 0 && Positions(k,5) < 14 % If none of the opposite cars,
proceed with the yield
%         YieldAccel = CarBehavior(k,4);
%     elseif pp == 0
%         YieldAccel = 0;
%     elseif pp >= 1 && Positions(k,2) < -11 % If there is an opposite car
about to pass through intersection
%         % Have cars that passed through turn to 4 so they aren't
%         counted This deceleration is kinda fast
%         YieldAccel = (Positions(k,5)^2)/(2*(Positions(k,2) + 11));
%     else
%         YieldAccel = 0;
%         Positions(k,5) = 0;
%     end
%
%     end
%
%     if j == 0
%         end

    if TrafficLight(Locations(k,5),2) == 2 || TrafficLight(Locations(k,5),2) == 1 ||
TrafficLight(Locations(k,5),2) == 0
        xi = IntX - 18 - (Positions(k,5)^2 - ToFPS(10)^2)/(2*CarBehavior(k,4));
        Light(k,8) = xi; % Starting position for slowing down before the light
        % Adjust the xi for other directions

        if Light(k,2) == 3 && Positions(k,2) < Light(k,9)
            YieldAccel = 0; % Stay Fast until Rushed Deceleration
        elseif Light(k,2) == 3 && Positions(k,2) >= Light(k,9)
            YieldAccel = -CarBehavior(k,4) - 3; % Rushed Deceleration

        elseif Positions(k,2) >= IntX - 20 && Positions(k,2) < IntX - 10 % Yielding
to Opposing Traffic
            % Check Traffic in Opposite Direction

            % Find the location where the direction is opposite

```

```

Opposite = find(Locations(:,5) == Intersections(IntNumber,7));
% Create the new matrix for opposite direction data
Yield = zeros(length(Opposite),9);
% Yield(1,:) = Light(k,:); % Plot the first line

% Delete Location where it is the same as the first line
Y = find(Opposite(:) == Light(k,1));
Opposite(Y, :) = [];

% Check if there are cars near the intersection in the
% opposite direction
if isempty(Opposite)
    pp = 0;
else
    pp = 0;
    for z = 1:length(Opposite)

        if Light(Opposite(z),2) == 4 || Light(Opposite(z),2) == 5
            pp = pp + 0;
        else
            pp = pp + 1;
        end
    end
end

if pp == 0 && Positions(k,2) >= IntX - 20 && Positions(k,2) < IntX - 11
&& Positions(k,5) >= ToFPS(10)
    YieldAccel = 0;
elseif pp >= 1 && Positions(k,2) >= IntX - 20 && Positions(k,2) < IntX -
11
    YieldAccel = (Positions(k,5)^2)/(2*(Positions(k,2) + IntX + 11));
elseif pp == 0 && Positions(k,2) >= IntX - 20 && Positions(k,2) < IntX -
11 && Positions(k,5) < ToFPS(10)
    YieldAccel = CarBehavior(k,4);
elseif pp >= 1 && Positions(k,2) > IntX - 12 && Positions(k,2) < IntX -
11
    YieldAccel = 0;
    Positions(k,5) = 0;
elseif pp == 0 && Positions(k,2) > IntX - 11
    if Positions(k,5) > 1 && Positions(k,5) <= 2
        YieldAccel = 0;
    elseif Positions(k,5) > 2
        YieldAccel = -CarBehavior(k,4);
    else
        YieldAccel = 0;
        Positions(k,5) = 1;
    end
elseif pp >= 1 && Positions(k,2) > IntX - 11 && Locations(k,4) ==
Intersections(IntNumber,6); % Watch for unsolvable Yield Accel
    YieldAccel = 0;
    Positions(k,5) = 2;
end

elseif Positions(k,2) >= Light(k,8) && Positions(k,2) < IntX - 20
    YieldAccel = -CarBehavior(k,4);
else
    YieldAccel = 0;
end

end

elseif Locations(k,8) == 3

```

```

        if TrafficLight(Locations(k,5),2) == 2 || TrafficLight(Locations(k,5),2) == 1 ||
TrafficLight(Locations(k,5),2) == 0
            xi = IntY + 4 + (Positions(k,5)^2 - ToFPS(10)^2)/(2*CarBehavior(k,4));
            Light(k,8) = xi; % Starting position for slowing down before the light

        if Light(k,2) == 3 && Positions(k,2) > Light(k,9)
            YieldAccel = 0; % Stay Fast until Rushed Deceleration
        elseif Light(k,2) == 3 && Positions(k,2) <= Light(k,9)
            YieldAccel = -CarBehavior(k,4) - 3; % Rushed Deceleration

        elseif Positions(k,2) <= IntY + 4 && Positions(k,2) > IntY - 6 % Yielding to
Opposing Traffic
            % Check Traffic in Opposite Direction

            % Find the location where the direction is opposite
            Opposite = find(Locations(:,5) == Intersections(IntNumber,7));
            % Create the new matrix for opposite direction data
            Yield = zeros(length(Opposite),9);
            % Yield(1,:) = Light(k,:); % Plot the first line

            % Delete Location where it is the same as the first line
            Y = find(Opposite(:) == Light(k,1));
            Opposite(Y, :) = [];

        if isempty(Opposite)
            pp = 0;
        else
            pp = 0;
            for z = 1:length(Opposite)

                if Light(Opposite(z),2) == 4 || Light(Opposite(z),2) == 5
                    pp = pp + 0;
                else
                    pp = pp + 1;
                end
            end
        end

        if pp == 0 && Positions(k,2) <= IntY + 4 && Positions(k,2) > IntY - 5 &&
Positions(k,5) >= ToFPS(10)
            YieldAccel = 0;
        elseif pp >= 1 && Positions(k,2) <= IntY + 4 && Positions(k,2) > IntY - 5
            YieldAccel = (Positions(k,5)^2)/(2*(Positions(k,2) + IntY + 6)); %
Make sure this is accurate
        elseif pp == 0 && Positions(k,2) <= IntY + 4 && Positions(k,2) > IntY - 5
&& Positions(k,5) < ToFPS(10)
            YieldAccel = CarBehavior(k,4);
        elseif pp >= 1 && Positions(k,2) < IntY - 4 && Positions(k,2) > IntY - 5
            YieldAccel = 0;
            Positions(k,5) = 0;
        elseif pp == 0 && Positions(k,2) < IntY - 5
            if Positions(k,5) > 1 && Positions(k,5) <= 2
                YieldAccel = 0;
            elseif Positions(k,5) > 2
                YieldAccel = -CarBehavior(k,4);
            else
                YieldAccel = 0;
            end
        end
    end
end

```



```

        Positions(k,5) = 1;
    end
    elseif pp >= 1 && Positions(k,2) < IntY - 5 && Locations(k,4) ==
Intersections(IntNumber,7); % Watch for unsolvable Yield Accel
        YieldAccel = 0;
        Positions(k,5) = 2;
    end

    elseif Positions(k,2) <= Light(k,8) && Positions(k,2) > IntY + 4
        YieldAccel = -CarBehavior(k,4);
    else
        YieldAccel = 0;
    end

end

%
% else
%     % Find the location where the direction is opposite
%     Opposite = find(Locations(:,5) == 1);
%     % Create the new matrix for opposite direction data
%     Yield = zeros(length(Opposite),9);
%     % Yield(1,:) = Light(k,:); % Plot the first line
%
%     % Delete Location where it is the same as the first line
%     Y = find(Opposite(:) == Light(k,1));
%     Opposite(Y, :) = [];
%
%     % Place rest of similar direction data into the check matrix
%     for z = 1:length(Opposite)
%         Yield(z,:) = Light(Opposite(z),:);
%     end
%
%     True = Yield(:,2) ~= 4;
%     v = nonzeros(True);
%
%     if isempty(v) && Positions(k,5) == 0
%         YieldAccel = CarBehavior(k,4);
%     elseif Positions(k,5) == 0
%         YieldAccel = 0;
%     else
%         YieldAccel = (0 - Positions(k,5)^2)/(2*(-Light(k,6) + Positions(k,2)));
%     end
%
% end

else
    if TrafficLight(Locations(k,5),2) == 2 || TrafficLight(Locations(k,5),2) == 1 ||
TrafficLight(Locations(k,5),2) == 0
        xi = IntX + 4 + (Positions(k,5)^2 - ToFPS(10)^2)/(2*CarBehavior(k,4));
        Light(k,8) = xi; % Starting position for slowing down before the light

    if Light(k,2) == 3 && Positions(k,2) > Light(k,9)
        YieldAccel = 0; % Stay Fast until Rushed Deceleration
    elseif Light(k,2) == 3 && Positions(k,2) <= Light(k,9)
        YieldAccel = -CarBehavior(k,4) - 3; % Rushed Deceleration
    end
end

```

```

        elseif Positions(k,2) <= IntX + 4 && Positions(k,2) > IntX - 6 % Yielding to
Opposing Traffic
            % Check Traffic in Opposite Direction

            % Find the location where the direction is opposite
Opposite = find(Locations(:,5) == Intersections(IntNumber,5));
            % Create the new matrix for opposite direction data
Yield = zeros(length(Opposite),9);
            % Yield(1,:) = Light(k,:); % Plot the first line

            % Delete Location where it is the same as the first line
Y = find(Opposite(:) == Light(k,1));
Opposite(Y, :) = [];

            if isempty(Opposite)
                pp = 0;
            else
                pp = 0;
                for z = 1:length(Opposite)

                    if Light(Opposite(z),2) == 4 || Light(Opposite(z),2) == 5
                        pp = pp + 0;
                    else
                        pp = pp + 1;
                    end
                end
            end

            if pp == 0 && Positions(k,2) <= IntX + 4 && Positions(k,2) > IntX - 5 &&
Positions(k,5) >= ToFPS(10)
                YieldAccel = 0;
            elseif pp >= 1 && Positions(k,2) <= IntX + 4 && Positions(k,2) > IntX - 5
                YieldAccel = (Positions(k,5)^2)/(2*(Positions(k,2) + IntX + 6)); %
Make sure this is accurate
            elseif pp == 0 && Positions(k,2) <= IntX + 4 && Positions(k,2) > IntX - 5
&& Positions(k,5) < ToFPS(10)
                YieldAccel = CarBehavior(k,4);
            elseif pp >= 1 && Positions(k,2) < IntX - 4 && Positions(k,2) > IntX - 5
                YieldAccel = 0;
                Positions(k,5) = 0;
            elseif pp == 0 && Positions(k,2) < IntX - 5
                if Positions(k,5) > 1 && Positions(k,5) <= 2
                    YieldAccel = 0;
                elseif Positions(k,5) > 2
                    YieldAccel = -CarBehavior(k,4);
                else
                    YieldAccel = 0;
                    Positions(k,5) = 1;
                end
            elseif pp >= 1 && Positions(k,2) < IntX - 5 && Locations(k,4) ==
Intersections(IntNumber,7); % Watch for unsolvable Yield Accel
                YieldAccel = 0;
                Positions(k,5) = 2;
            end

            elseif Positions(k,2) <= Light(k,8) && Positions(k,2) > IntX + 4
                YieldAccel = -CarBehavior(k,4);
            else
                YieldAccel = 0;
            end
        end

    end
end

```

```

end

%% Transition to New Intersection or Random Reset Location
if Locations(k,9) == 3 && Locations(k,3) >= IntY + LaneLength % If car runs out of bounds

% Final Time Evaluation Update
if TimeEvaluation(k,10) == 0
    TimeEvaluation(k,7) = Locations(k,2);
    TimeEvaluation(k,8) = Locations(k,3);
    TimeEvaluation(k,9) = j*timestep;
    TimeEvaluation(k,10) = size(TimeEvaluation,1) + 1;

    % Update Car Change Matrix
    CarChange(k,2) = TimeEvaluation(k,10);
    CarNumber = k;

else
    Count = 0;
    index = 1;
    CarNumber = TimeEvaluation(k,10);
    while index > 0
        if TimeEvaluation(CarNumber,10) == 0
            index = 0;
        else
            CarNumber = TimeEvaluation(CarNumber,10);
            index = 1;
        end
        Count = Count + 1;
    end

    TimeEvaluation(CarNumber,7) = Locations(k,2);
    TimeEvaluation(CarNumber,8) = Locations(k,3);
    TimeEvaluation(CarNumber,9) = j*timestep;
    TimeEvaluation(CarNumber,10) = size(TimeEvaluation,1) + 1;

    % Update Car Change Matrix
    CarChange(k,Count + 2) = TimeEvaluation(CarNumber,10);

end

% Configure Newly Created Car
if TrafficLight(Locations(k,4),3) == 0 % If Car Exits the Simulation

% Random New Location and Position
indices = find(TrafficLight(:,3) == 0);
Entrance = datasample(indices,1);

[IntNumber,d] = find(Intersections(:,4:7) == Entrance); % Finding row and column
Out = randsample(setdiff(Intersections(IntNumber,4):Intersections(IntNumber,7),
Entrance), 1);
[IntNumber,O] = find(Intersections(:,4:7) == Out); % Finding row and column

IntX = Intersections(IntNumber,2);
IntY = Intersections(IntNumber,3);

% New Position Location
Locations(k,4) = Out;
Locations(k,5) = Entrance; % Adjust to make random entrance into the simulation
Locations(k,7) = IntNumber;
Locations(k,8) = d;
Locations(k,9) = 0;

```

```

if d == 1
    Locations(k,2) = IntX + 3;
    Locations(k,3) = IntY - LaneLength;
    Positions(k,2) = Locations(k,3);
elseif d == 2
    Locations(k,2) = IntX - LaneLength;
    Locations(k,3) = IntY - 9;
    Positions(k,2) = Locations(k,2);
elseif d == 3
    Locations(k,2) = IntX - 9;
    Locations(k,3) = IntY + LaneLength - CarLength;
    Positions(k,2) = Locations(k,3);
else
    Locations(k,2) = IntX + LaneLength - CarLength;
    Locations(k,3) = IntY + 3;
    Positions(k,2) = Locations(k,2);
end

% New Car Behavior
Intensity = randi([1 10],1,1);
[Speedy, Accel, Timing] = IntensityRating(Intensity,SpeedLimit);

CarBehavior(k,2) = Intensity;
CarBehavior(k,3) = Speedy;
CarBehavior(k,4) = Accel;
CarBehavior(k,5) = Timing;

% Updated Position Matrix
Positions(k,5) = ToFPS(Speedy);
Positions(k,8) = 0;

Time(k,6) = Timing;
Light(k,2) = 4;

% New Car Evaluation
TimeEvaluation(end + 1,1) = size(TimeEvaluation,1) + 1; %#ok<SAGROW>
TimeEvaluation(end,2) = Locations(k,5);
TimeEvaluation(end,3) = Locations(k,2);
TimeEvaluation(end,4) = Locations(k,3);
TimeEvaluation(end,5) = j*timestep;
TimeEvaluation(end,6) = Locations(k,4);
TimeEvaluation(end,12) = Intensity;

% Ideal Time Calculation
[TimeIdeal] = IdealTime(d,0,Speedy,Accel,LaneLength);
TimeEvaluation(end,11) = TimeIdeal;

else % If Car is Transitioning Within the Simulation

% New Position Directions
Direction = TrafficLight(Locations(k,4),3); % New Direction Input
[IntNumber,d] = find(Intersections(:,4:7) == Direction); % Finding row and column

Out = randsample(setdiff(Intersections(IntNumber,4):Intersections(IntNumber,7),
Direction), 1);
[IntNumber,O] = find(Intersections(:,4:7) == Out); % Finding row and column

IntX = Intersections(IntNumber,2);
IntY = Intersections(IntNumber,3);

% Adding Values to Locations Matrix
Locations(k,4) = Out;
Locations(k,5) = Direction;

```

```

Locations(k,7) = IntNumber;
Locations(k,8) = d;
Locations(k,9) = 0;

% Light Matrix
Light(k,2) = 4;

% New Car Evaluation
TimeEvaluation(end + 1,1) = size(TimeEvaluation,1) + 1; %#ok<SAGROW>
TimeEvaluation(end,2) = Locations(k,5);
TimeEvaluation(end,3) = Locations(k,2);
TimeEvaluation(end,4) = Locations(k,3);
TimeEvaluation(end,5) = j*timestep;
TimeEvaluation(end,6) = Locations(k,4);
TimeEvaluation(end,11) = TimeEvaluation(CarNumber,11);
TimeEvaluation(end,12) = TimeEvaluation(CarNumber,12);

end

elseif Locations(k,9) == 4 && Locations(k,2) >= IntX + LaneLength

% Final Time Evaluation Update
if TimeEvaluation(k,10) == 0
    TimeEvaluation(k,7) = Locations(k,2);
    TimeEvaluation(k,8) = Locations(k,3);
    TimeEvaluation(k,9) = j*timestep;
    TimeEvaluation(k,10) = size(TimeEvaluation,1) + 1;

    % Update Car Change Matrix
    CarChange(k,2) = TimeEvaluation(k,10);
    CarNumber = k;

else
    Count = 0;
    index = 1;
    CarNumber = TimeEvaluation(k,10);
    while index > 0
        if TimeEvaluation(CarNumber,10) == 0
            index = 0;
        else
            CarNumber = TimeEvaluation(CarNumber,10);
            index = 1;
        end
        Count = Count + 1;
    end

    TimeEvaluation(CarNumber,7) = Locations(k,2);
    TimeEvaluation(CarNumber,8) = Locations(k,3);
    TimeEvaluation(CarNumber,9) = j*timestep;
    TimeEvaluation(CarNumber,10) = size(TimeEvaluation,1) + 1;

    % Update Car Change Matrix
    CarChange(k,Count + 2) = TimeEvaluation(CarNumber,10);

end

% Configure Newly Created Car
if TrafficLight(Locations(k,4),3) == 0 % If Car Exits the Simulation

% Random New Location and Position
indices = find(TrafficLight(:,3) == 0);
Entrance = datasample(indices,1);

[IntNumber,d] = find(Intersections(:,4:7) == Entrance); % Finding row and column

```

```

Out = randsample(setdiff(Intersections(IntNumber,4):Intersections(IntNumber,7),
Entrance), 1);
[IntNumber,0] = find(Intersections(:,4:7) == Out); % Finding row and column

IntX = Intersections(IntNumber,2);
IntY = Intersections(IntNumber,3);

% New Position Location
Locations(k,4) = Out;
Locations(k,5) = Entrance; % Adjust to make random entrance into the simulation
Locations(k,7) = IntNumber;
Locations(k,8) = d;
Locations(k,9) = 0;

if d == 1
    Locations(k,2) = IntX + 3;
    Locations(k,3) = IntY - LaneLength;
    Positions(k,2) = Locations(k,3);
elseif d == 2
    Locations(k,2) = IntX - LaneLength;
    Locations(k,3) = IntY - 9;
    Positions(k,2) = Locations(k,2);
elseif d == 3
    Locations(k,2) = IntX - 9;
    Locations(k,3) = IntY + LaneLength - CarLength;
    Positions(k,2) = Locations(k,3);
else
    Locations(k,2) = IntX + LaneLength - CarLength;
    Locations(k,3) = IntY + 3;
    Positions(k,2) = Locations(k,2);
end

% New Car Behavior
Intensity = randi([1 10],1,1);
[Speedy, Accel, Timing] = IntensityRating(Intensity,SpeedLimit);

CarBehavior(k,2) = Intensity;
CarBehavior(k,3) = Speedy;
CarBehavior(k,4) = Accel;
CarBehavior(k,5) = Timing;

% Updated Position Matrix
Positions(k,5) = ToFPS(Speedy);
Positions(k,8) = 0;

Time(k,6) = Timing;
Light(k,2) = 4;

% New Car Evaluation
TimeEvaluation(end + 1,1) = size(TimeEvaluation,1) + 1; %#ok<SAGROW>
TimeEvaluation(end,2) = Locations(k,5);
TimeEvaluation(end,3) = Locations(k,2);
TimeEvaluation(end,4) = Locations(k,3);
TimeEvaluation(end,5) = j*timestep;
TimeEvaluation(end,6) = Locations(k,4);
TimeEvaluation(end,12) = Intensity;

% Ideal Time Calculation
[TimeIdeal] = IdealTime(d,0,Speedy,Accel,LaneLength);
TimeEvaluation(end,11) = TimeIdeal;

else % If Car is Transitioning Within the Simulation

```

```

% New Position Directions
Direction = TrafficLight(Locations(k,4),3); % New Direction Input
[IntNumber,d] = find(Intersections(:,4:7) == Direction); % Finding row and column

Out = randsample(setdiff(Intersections(IntNumber,4):Intersections(IntNumber,7),
Direction), 1);
[IntNumber,O] = find(Intersections(:,4:7) == Out); % Finding row and column

IntX = Intersections(IntNumber,2);
IntY = Intersections(IntNumber,3);

% Adding Values to Locations Matrix
Locations(k,4) = Out;
Locations(k,5) = Direction;
Locations(k,7) = IntNumber;
Locations(k,8) = d;
Locations(k,9) = 0;

% Light Matrix
Light(k,2) = 4;

% New Car Evaluation
TimeEvaluation(end + 1,1) = size(TimeEvaluation,1) + 1; %#ok<SAGROW>
TimeEvaluation(end,2) = Locations(k,5);
TimeEvaluation(end,3) = Locations(k,2);
TimeEvaluation(end,4) = Locations(k,3);
TimeEvaluation(end,5) = j*timestep;
TimeEvaluation(end,6) = Locations(k,4);
TimeEvaluation(end,11) = TimeEvaluation(CarNumber,11);
TimeEvaluation(end,12) = TimeEvaluation(CarNumber,12);

end

elseif Locations(k,9) == 1 && Locations(k,3) <= IntY - LaneLength - CarLength

% Final Time Evaluation Update
if TimeEvaluation(k,10) == 0
    TimeEvaluation(k,7) = Locations(k,2);
    TimeEvaluation(k,8) = Locations(k,3);
    TimeEvaluation(k,9) = j*timestep;
    TimeEvaluation(k,10) = size(TimeEvaluation,1) + 1;

    % Update Car Change Matrix
    CarChange(k,2) = TimeEvaluation(k,10);
    CarNumber = k;

else
    Count = 0;
    index = 1;
    CarNumber = TimeEvaluation(k,10);
    while index > 0
        if TimeEvaluation(CarNumber,10) == 0
            index = 0;
        else
            CarNumber = TimeEvaluation(CarNumber,10);
            index = 1;
        end
        Count = Count + 1;
    end

    TimeEvaluation(CarNumber,7) = Locations(k,2);
    TimeEvaluation(CarNumber,8) = Locations(k,3);
    TimeEvaluation(CarNumber,9) = j*timestep;

```

```

    TimeEvaluation(CarNumber,10) = size(TimeEvaluation,1) + 1;

    % Update Car Change Matrix
    CarChange(k,Count + 2) = TimeEvaluation(CarNumber,10);

end

% Configure Newly Created Car
if TrafficLight(Locations(k,4),3) == 0 % If Car Exits the Simulation

% Random New Location and Position
indices = find(TrafficLight(:,3) == 0);
Entrance = datasample(indices,1);

[IntNumber,d] = find(Intersections(:,4:7) == Entrance); % Finding row and column
Out = randsample(setdiff(Intersections(IntNumber,4):Intersections(IntNumber,7),
Entrance), 1);
[IntNumber,0] = find(Intersections(:,4:7) == Out); % Finding row and column

IntX = Intersections(IntNumber,2);
IntY = Intersections(IntNumber,3);

% New Position Location
Locations(k,4) = Out;
Locations(k,5) = Entrance; % Adjust to make random entrance into the simulation
Locations(k,7) = IntNumber;
Locations(k,8) = d;
Locations(k,9) = 0;

if d == 1
    Locations(k,2) = IntX + 3;
    Locations(k,3) = IntY - LaneLength;
    Positions(k,2) = Locations(k,3);
elseif d == 2
    Locations(k,2) = IntX - LaneLength;
    Locations(k,3) = IntY - 9;
    Positions(k,2) = Locations(k,2);
elseif d == 3
    Locations(k,2) = IntX - 9;
    Locations(k,3) = IntY + LaneLength - CarLength;
    Positions(k,2) = Locations(k,3);
else
    Locations(k,2) = IntX + LaneLength - CarLength;
    Locations(k,3) = IntY + 3;
    Positions(k,2) = Locations(k,2);
end

% New Car Behavior
Intensity = randi([1 10],1,1);
[Speedy, Accel, Timing] = IntensityRating(Intensity,SpeedLimit);

CarBehavior(k,2) = Intensity;
CarBehavior(k,3) = Speedy;
CarBehavior(k,4) = Accel;
CarBehavior(k,5) = Timing;

% Updated Position Matrix
Positions(k,5) = ToFPS(Speedy);
Positions(k,8) = 0;

Time(k,6) = Timing;
Light(k,2) = 4;

% New Car Evaluation

```



```

TimeEvaluation(end + 1,1) = size(TimeEvaluation,1) + 1; %#ok<SAGROW>
TimeEvaluation(end,2) = Locations(k,5);
TimeEvaluation(end,3) = Locations(k,2);
TimeEvaluation(end,4) = Locations(k,3);
TimeEvaluation(end,5) = j*timestep;
TimeEvaluation(end,6) = Locations(k,4);
TimeEvaluation(end,12) = Intensity;

% Ideal Time Calculation
[TimeIdeal] = IdealTime(d,O,Speedy,Accel,LaneLength);
TimeEvaluation(end,11) = TimeIdeal;

else % If Car is Transitioning Within the Simulation

% New Position Directions
Direction = TrafficLight(Locations(k,4),3); % New Direction Input
[IntNumber,d] = find(Intersections(:,4:7) == Direction); % Finding row and column

Out = randsample(setdiff(Intersections(IntNumber,4):Intersections(IntNumber,7),
Direction), 1);
[IntNumber,O] = find(Intersections(:,4:7) == Out); % Finding row and column

IntX = Intersections(IntNumber,2);
IntY = Intersections(IntNumber,3);

% Adding Values to Locations Matrix
Locations(k,4) = Out;
Locations(k,5) = Direction;
Locations(k,7) = IntNumber;
Locations(k,8) = d;
Locations(k,9) = O;

% Light Matrix
Light(k,2) = 4;

% New Car Evaluation
TimeEvaluation(end + 1,1) = size(TimeEvaluation,1) + 1; %#ok<SAGROW>
TimeEvaluation(end,2) = Locations(k,5);
TimeEvaluation(end,3) = Locations(k,2);
TimeEvaluation(end,4) = Locations(k,3);
TimeEvaluation(end,5) = j*timestep;
TimeEvaluation(end,6) = Locations(k,4);
TimeEvaluation(end,11) = TimeEvaluation(CarNumber,11);
TimeEvaluation(end,12) = TimeEvaluation(CarNumber,12);

end

elseif Locations(k,9) == 2 && Locations(k,2) <= IntX - LaneLength - CarLength

if TimeEvaluation(k,10) == 0
    TimeEvaluation(k,7) = Locations(k,2);
    TimeEvaluation(k,8) = Locations(k,3);
    TimeEvaluation(k,9) = j*timestep;
    TimeEvaluation(k,10) = size(TimeEvaluation,1) + 1;

    % Update Car Change Matrix
    CarChange(k,2) = TimeEvaluation(k,10);
    CarNumber = k;

else

```

```

Count = 0;
index = 1;
CarNumber = TimeEvaluation(k,10);
while index > 0
    if TimeEvaluation(CarNumber,10) == 0
        index = 0;
    else
        CarNumber = TimeEvaluation(CarNumber,10);
        index = 1;
    end
    Count = Count + 1;
end

TimeEvaluation(CarNumber,7) = Locations(k,2);
TimeEvaluation(CarNumber,8) = Locations(k,3);
TimeEvaluation(CarNumber,9) = j*timestep;
TimeEvaluation(CarNumber,10) = size(TimeEvaluation,1) + 1;

% Update Car Change Matrix
CarChange(k,Count + 2) = TimeEvaluation(CarNumber,10);

end

% Configure Newly Created Car
if TrafficLight(Locations(k,4),3) == 0 % If Car Exits the Simulation

% Random New Location and Position
indices = find(TrafficLight(:,3) == 0);
Entrance = datasample(indices,1);

[IntNumber,d] = find(Intersections(:,4:7) == Entrance); % Finding row and column
Out = randsample(setdiff(Intersections(IntNumber,4):Intersections(IntNumber,7),
Entrance), 1);
[IntNumber,0] = find(Intersections(:,4:7) == Out); % Finding row and column

IntX = Intersections(IntNumber,2);
IntY = Intersections(IntNumber,3);

% New Position Location
Locations(k,4) = Out;
Locations(k,5) = Entrance; % Adjust to make random entrance into the simulation
Locations(k,7) = IntNumber;
Locations(k,8) = d;
Locations(k,9) = 0;

if d == 1
    Locations(k,2) = IntX + 3;
    Locations(k,3) = IntY - LaneLength;
    Positions(k,2) = Locations(k,3);
elseif d == 2
    Locations(k,2) = IntX - LaneLength;
    Locations(k,3) = IntY - 9;
    Positions(k,2) = Locations(k,2);
elseif d == 3
    Locations(k,2) = IntX - 9;
    Locations(k,3) = IntY + LaneLength - CarLength;
    Positions(k,2) = Locations(k,3);
else
    Locations(k,2) = IntX + LaneLength - CarLength;
    Locations(k,3) = IntY + 3;
    Positions(k,2) = Locations(k,2);
end

% New Car Behavior
Intensity = randi([1 10],1,1);

```

```

[Speedy, Accel, Timing] = IntensityRating(Intensity,SpeedLimit);

CarBehavior(k,2) = Intensity;
CarBehavior(k,3) = Speedy;
CarBehavior(k,4) = Accel;
CarBehavior(k,5) = Timing;

% Updated Position Matrix
Positions(k,5) = ToFPS(Speedy);
Positions(k,8) = 0;

Time(k,6) = Timing;
Light(k,2) = 4;

% New Car Evaluation
TimeEvaluation(end + 1,1) = size(TimeEvaluation,1) + 1; %#ok<SAGROW>
TimeEvaluation(end,2) = Locations(k,5);
TimeEvaluation(end,3) = Locations(k,2);
TimeEvaluation(end,4) = Locations(k,3);
TimeEvaluation(end,5) = j*timestep;
TimeEvaluation(end,6) = Locations(k,4);
TimeEvaluation(end,12) = Intensity;

% Ideal Time Calculation
[TimeIdeal] = IdealTime(d,0,Speedy,Accel,LaneLength);
TimeEvaluation(end,11) = TimeIdeal;

else % If Car is Transitioning Within the Simulation

% New Position Directions
Direction = TrafficLight(Locations(k,4),3); % New Direction Input
[IntNumber,d] = find(Intersections(:,4:7) == Direction); % Finding row and column

Out = randsample(setdiff(Intersections(IntNumber,4):Intersections(IntNumber,7),
Direction), 1);
[IntNumber,O] = find(Intersections(:,4:7) == Out); % Finding row and column

IntX = Intersections(IntNumber,2);
IntY = Intersections(IntNumber,3);

% Adding Values to Locations Matrix
Locations(k,4) = Out;
Locations(k,5) = Direction;
Locations(k,7) = IntNumber;
Locations(k,8) = d;
Locations(k,9) = 0;

% Light Matrix
Light(k,2) = 4;

% New Car Evaluation
TimeEvaluation(end + 1,1) = size(TimeEvaluation,1) + 1; %#ok<SAGROW>
TimeEvaluation(end,2) = Locations(k,5);
TimeEvaluation(end,3) = Locations(k,2);
TimeEvaluation(end,4) = Locations(k,3);
TimeEvaluation(end,5) = j*timestep;
TimeEvaluation(end,6) = Locations(k,4);
TimeEvaluation(end,11) = TimeEvaluation(CarNumber,11);
TimeEvaluation(end,12) = TimeEvaluation(CarNumber,12);

end

```

end

```
%% Choosing the Appropriate Acceleration

% Do you thing, CasualAccel
% Following another car, FollowAccel
% Inching closer, InchAccel
% No Acceleration, Stopped

if Locations(k,8) == 1
    if TrafficLight(Locations(k,5),2) == 0 % Red Light
        if Light(k,2) == 4 && Positions(k,2) >= Light(k,5)
            FinalAcceleration = LightAccel;
            Light(k,3) = 2;
            if Time(k,2) < 0.5 && Time(k,2) > 0
                FinalAcceleration = -32;
                Light(k,3) = 1;
            end
        elseif Positions(k,2) > IntY - 20 && Light(k,2) ~= 4 && Locations(k,9) == 2
            FinalAcceleration = YieldAccel;
            Light(k,3) = 4;
        elseif Light(k,2) == 4 && Positions(k,5) < 0.5 && Positions(k,2) <= Light(k,6) +
1 && Positions(k,2) >= Light(k,6) - 1
            FinalAcceleration = LightAccel;
            Light(k,3) = 2;
        elseif Light(k,2) == 5
            FinalAcceleration = DistanceAccel;
            Light(k,3) = 1;
        elseif Light(k,2) == 4 && Light(k,4) == 0 && Positions(k,2) >= Light(k,5)
            FinalAcceleration = LightAccel;
            Light(k,3) = 2;
        elseif Light(k,2) == 4 && Positions(FC,5) < 5
            FinalAcceleration = LightAccel;
            Light(k,3) = 2;
        else
            FinalAcceleration = DistanceAccel;
            Light(k,3) = 1;
        end

    elseif TrafficLight(Locations(k,5),2) == 1 % Yellow Light
        if Light(k,2) == 4 && Positions(k,2) >= Light(k,5) % Stop Before Light
            FinalAcceleration = LightAccel;
            Light(k,3) = 2;
            if Time(k,2) < 0.5 && Time(k,2) > 0
                FinalAcceleration = -32;
                Light(k,3) = 1;
            end
        elseif Light(k,2) == 3 || Light(k,2) == 2
            if Light(k,4) == 0
                if Locations(k,9) == 3
                    FinalAcceleration = LightAccel;
                    Light(k,3) = 2;
                elseif Locations(k,9) == 4
                    FinalAcceleration = TurnAccel;
                    Light(k,3) = 3;
                elseif Locations(k,9) == 2
                    FinalAcceleration = YieldAccel;
                    Light(k,3) = 4;
                else
                    FinalAcceleration = DistanceAccel;
                end
            end
        end
    end
end
```

```

        Light(k,3) = 1;
    end
    else % Edit this for multiple cars making through decisions
        FinalAcceleration = DistanceAccel;
        Light(k,3) = 1;
    end
    else % if Light(k,2) == 1 or 5
        FinalAcceleration = DistanceAccel;
        Light(k,3) = 1;
    end
end

else % Green Light
    if Locations(k,9) == 3 % Proceeding Straight
        ToIntersection = IntY - 28 - ToFPS(CarBehavior(k,3))*Yellow;
        if Positions(k,2) >= ToIntersection && Positions(k,2) < IntY - 10
            Light(k,2) = 1;
        elseif Positions(k,2) < ToIntersection
            Light(k,2) = 4;
        else
            Light(k,2) = 5;
        end
        % On a Green Light Proceeding Straight, always DistanceAccel
        FinalAcceleration = DistanceAccel;
        Light(k,3) = 1;

        elseif Locations(k,9) == 4 % Right Turn
            ToIntersection = IntY - 22 - ToFPS(CarBehavior(k,3))*Yellow +
(1/2)*CarBehavior(k,4)*Yellow^2;
            if Positions(k,2) >= ToIntersection && Positions(k,2) < IntY - 12
                Light(k,2) = 1;
            elseif Positions(k,2) < ToIntersection
                Light(k,2) = 4;
            else
                Light(k,2) = 5;
            end

            if Positions(k,2) >= Light(k,7) && Light(k,4) == 0
                FinalAcceleration = TurnAccel;
                Light(k,3) = 3;
            else
                FinalAcceleration = DistanceAccel;
                Light(k,3) = 1;
            end
            elseif Locations(k,9) == 2 % Left Turn
                ToIntersection = IntY - 16 - ToFPS(CarBehavior(k,3))*Yellow +
(1/2)*CarBehavior(k,4)*Yellow^2;
                if Positions(k,2) >= ToIntersection && Positions(k,2) < IntY - 10
                    Light(k,2) = 1;
                elseif Positions(k,2) < ToIntersection
                    Light(k,2) = 4;
                else
                    Light(k,2) = 5;
                end

                if Positions(k,2) >= Light(k,8) && Light(k,4) == 0
                    FinalAcceleration = YieldAccel;
                    Light(k,3) = 4;
                elseif Positions(k,2) >= IntY - 20 && Positions(k,2) < IntY - 10 &&
Light(k,4) == 0
                    FinalAcceleration = YieldAccel;
                    Light(k,3) = 4;
                else
                    FinalAcceleration = DistanceAccel;
                    Light(k,3) = 1;
                end
            end
        end
    end
end
end

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
elseif Locations(k,8) == 2
    if TrafficLight(Locations(k,5),2) == 0 % Red Light
        if Light(k,2) == 4 && Positions(k,2) >= Light(k,5)
            FinalAcceleration = LightAccel;
            Light(k,3) = 2;
            if Time(k,2) < 0.5 && Time(k,2) > 0
                FinalAcceleration = -32;
                Light(k,3) = 1;
            end
        elseif Positions(k,2) > IntX - 20 && Light(k,2) ~= 4 && Locations(k,9) == 3
            FinalAcceleration = YieldAccel;
            Light(k,3) = 4;
        elseif Light(k,2) == 4 && Positions(k,5) < 0.5 && Positions(k,2) <= Light(k,6) +
1 && Positions(k,2) >= Light(k,6) - 1
            FinalAcceleration = LightAccel;
            Light(k,3) = 2;
        elseif Light(k,2) == 5
            FinalAcceleration = DistanceAccel;
            Light(k,3) = 1;
        elseif Light(k,2) == 4 && Light(k,4) == 0 && Positions(k,2) >= Light(k,5)
            FinalAcceleration = LightAccel;
            Light(k,3) = 2;
        elseif Light(k,2) == 4 && Positions(FC,5) < 5
            FinalAcceleration = LightAccel;
            Light(k,3) = 2;
        else
            FinalAcceleration = DistanceAccel;
            Light(k,3) = 1;
        end

    elseif TrafficLight(Locations(k,5),2) == 1 % Yellow Light
        if Light(k,2) == 4 && Positions(k,2) >= Light(k,5) % Stop Before Light
            FinalAcceleration = LightAccel;
            Light(k,3) = 2;
            if Time(k,2) < 0.5 && Time(k,2) > 0
                FinalAcceleration = -32;
                Light(k,3) = 1;
            end
        elseif Light(k,2) == 3 || Light(k,2) == 2
            if Light(k,4) == 0
                if Locations(k,9) == 4
                    FinalAcceleration = LightAccel;
                    Light(k,3) = 2;
                elseif Locations(k,9) == 1
                    FinalAcceleration = TurnAccel;
                    Light(k,3) = 3;
                elseif Locations(k,9) == 3
                    FinalAcceleration = YieldAccel;
                    Light(k,3) = 4;
                else
                    FinalAcceleration = DistanceAccel;
                    Light(k,3) = 1;
                end
            end
            else % Edit this for multiple cars making through decisions
                FinalAcceleration = DistanceAccel;
                Light(k,3) = 1;
            end
        else % if Light(k,2) == 1 or 5
            FinalAcceleration = DistanceAccel;
            Light(k,3) = 1;
        end

    else % Green Light

```

```

if Locations(k,9) == 4 % Proceeding Straight
    ToIntersection = IntX - 28 - ToFPS(CarBehavior(k,3))*Yellow;
    if Positions(k,2) >= ToIntersection && Positions(k,2) < IntX - 10
        Light(k,2) = 1;
    elseif Positions(k,2) < ToIntersection
        Light(k,2) = 4;
    else
        Light(k,2) = 5;
    end
    % On a Green Light Proceeding Straight, always DistanceAccel
    FinalAcceleration = DistanceAccel;
    Light(k,3) = 1;

elseif Locations(k,9) == 1 % Right Turn
    ToIntersection = IntX - 22 - ToFPS(CarBehavior(k,3))*Yellow +
(1/2)*CarBehavior(k,4)*Yellow^2;
    if Positions(k,2) >= ToIntersection && Positions(k,2) < IntX - 12
        Light(k,2) = 1;
    elseif Positions(k,2) < ToIntersection
        Light(k,2) = 4;
    else
        Light(k,2) = 5;
    end

    if Positions(k,2) >= Light(k,7) && Light(k,4) == 0
        FinalAcceleration = TurnAccel;
        Light(k,3) = 3;
    else
        FinalAcceleration = DistanceAccel;
        Light(k,3) = 1;
    end

elseif Locations(k,9) == 3 % Left Turn
    ToIntersection = IntX - 16 - ToFPS(CarBehavior(k,3))*Yellow +
(1/2)*CarBehavior(k,4)*Yellow^2;
    if Positions(k,2) >= ToIntersection && Positions(k,2) < IntX - 10
        Light(k,2) = 1;
    elseif Positions(k,2) < ToIntersection
        Light(k,2) = 4;
    else
        Light(k,2) = 5;
    end

    if Positions(k,2) >= Light(k,8) && Light(k,4) == 0
        FinalAcceleration = YieldAccel;
        Light(k,3) = 4;
    elseif Positions(k,2) >= IntX - 20 && Positions(k,2) < IntX - 10 &&
Light(k,4) == 0
        FinalAcceleration = YieldAccel;
        Light(k,3) = 4;
    else
        FinalAcceleration = DistanceAccel;
        Light(k,3) = 1;
    end
end
end
end

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
elseif Locations(k,8) == 3

    if TrafficLight(Locations(k,5),2) == 0 % Red Light

        if Light(k,2) == 4 && Positions(k,2) <= Light(k,5)
            FinalAcceleration = LightAccel;
            Light(k,3) = 2;
            if Time(k,2) < 0.5 && Time(k,2) > 0
                FinalAcceleration = -32;
                Light(k,3) = 1;
            end
        elseif Positions(k,2) < IntY + 4 && Light(k,2) ~= 4 && Locations(k,9) == 4
            FinalAcceleration = YieldAccel;
            Light(k,3) = 4;
        elseif Light(k,2) == 4 && Positions(k,5) < 0.5 && Positions(k,2) <= Light(k,6) +
1 && Positions(k,2) >= Light(k,6) - 1
            FinalAcceleration = LightAccel;
            Light(k,3) = 2;
        elseif Light(k,2) == 5
            FinalAcceleration = DistanceAccel;
            Light(k,3) = 1;
        elseif Light(k,2) == 4 && Light(k,4) == 0 && Positions(k,2) <= Light(k,5)
            FinalAcceleration = LightAccel;
            Light(k,3) = 2;
        elseif Light(k,2) == 4 && Positions(FC,5) < 5
            FinalAcceleration = LightAccel;
            Light(k,3) = 2;
        else
            FinalAcceleration = DistanceAccel;
            Light(k,3) = 1;
        end

    elseif TrafficLight(Locations(k,5),2) == 1 % Yellow Light
        if Light(k,2) == 4 && Positions(k,2) <= Light(k,5) % Stop Before Light
            FinalAcceleration = LightAccel;
            Light(k,3) = 2;
            if Time(k,2) < 0.5 && Time(k,2) > 0
                FinalAcceleration = -32;
                Light(k,3) = 1;
            end
        elseif Light(k,2) == 3 || Light(k,2) == 2
            if Light(k,4) == 0
                if Locations(k,9) == 1
                    FinalAcceleration = LightAccel;
                    Light(k,3) = 2;
                elseif Locations(k,9) == 2
                    FinalAcceleration = TurnAccel;
                    Light(k,3) = 3;
                elseif Locations(k,9) == 4
                    FinalAcceleration = YieldAccel;
                    Light(k,3) = 4;
                else
                    FinalAcceleration = DistanceAccel;
                    Light(k,3) = 1;
                end
            end
        else % Edit this for multiple cars making through decisions
            FinalAcceleration = DistanceAccel;
            Light(k,3) = 1;
        end
    end
end

```



```

        end
    else % if Light(k,2) == 1 or 5
        FinalAcceleration = DistanceAccel;
        Light(k,3) = 1;
    end

    else % Green Light
    if Locations(k,9) == 1 % Proceeding Straight
        ToIntersection = IntY + 12 + ToFPS(CarBehavior(k,3))*Yellow;
        if Positions(k,2) <= ToIntersection && Positions(k,2) > IntY - 6
            Light(k,2) = 1;
        elseif Positions(k,2) > ToIntersection
            Light(k,2) = 4;
        else
            Light(k,2) = 5;
        end
        % On a Green Light Proceeding Straight, always DistanceAccel
        FinalAcceleration = DistanceAccel;
        Light(k,3) = 1;

        elseif Locations(k,9) == 2 % Right Turn
            ToIntersection = IntY + 6 + ToFPS(CarBehavior(k,3))*Yellow -
(1/2)*CarBehavior(k,4)*Yellow^2;
            if Positions(k,2) <= ToIntersection && Positions(k,2) > IntY - 4
                Light(k,2) = 1;
            elseif Positions(k,2) > ToIntersection
                Light(k,2) = 4;
            else
                Light(k,2) = 5;
            end

            if Positions(k,2) <= Light(k,7) && Light(k,4) == 0
                FinalAcceleration = TurnAccel;
                Light(k,3) = 3;
            else
                FinalAcceleration = DistanceAccel;
                Light(k,3) = 1;
            end
            elseif Locations(k,9) == 4 % Left Turn
                ToIntersection = IntY + 0 + ToFPS(CarBehavior(k,3))*Yellow -
(1/2)*CarBehavior(k,4)*Yellow^2;
                if Positions(k,2) <= ToIntersection && Positions(k,2) > IntY - 6
                    Light(k,2) = 1;
                elseif Positions(k,2) > ToIntersection
                    Light(k,2) = 4;
                else
                    Light(k,2) = 5;
                end

                if Positions(k,2) <= Light(k,8) && Light(k,4) == 0
                    FinalAcceleration = YieldAccel;
                    Light(k,3) = 4;
                elseif Positions(k,2) <= IntY + 4 && Positions(k,2) > IntY - 6 && Light(k,4)
== 0
                    FinalAcceleration = YieldAccel;
                    Light(k,3) = 4;
                else
                    FinalAcceleration = DistanceAccel;
                    Light(k,3) = 1;
                end
            end
        end
    end
end
end

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
else
    if TrafficLight(Locations(k,5),2) == 0 % Red Light
        if Light(k,2) == 4 && Positions(k,2) <= Light(k,5)
            FinalAcceleration = LightAccel;
            Light(k,3) = 2;
            if Time(k,2) < 0.5 && Time(k,2) > 0
                FinalAcceleration = -32;
                Light(k,3) = 1;
            end
        elseif Positions(k,2) < IntX + 4 && Light(k,2) ~= 4 && Locations(k,9) == 1
            FinalAcceleration = YieldAccel;
            Light(k,3) = 4;
        elseif Light(k,2) == 4 && Positions(k,5) < 0.5 && Positions(k,2) <= Light(k,6) +
1 && Positions(k,2) >= Light(k,6) - 1
            FinalAcceleration = LightAccel;
            Light(k,3) = 2;
        elseif Light(k,2) == 5
            FinalAcceleration = DistanceAccel;
            Light(k,3) = 1;
        elseif Light(k,2) == 4 && Light(k,4) == 0 && Positions(k,2) <= Light(k,5)
            FinalAcceleration = LightAccel;
            Light(k,3) = 2;
        elseif Light(k,2) == 4 && Positions(FC,5) < 5
            FinalAcceleration = LightAccel;
            Light(k,3) = 2;
        else
            FinalAcceleration = DistanceAccel;
            Light(k,3) = 1;
        end

    elseif TrafficLight(Locations(k,5),2) == 1 % Yellow Light
        if Light(k,2) == 4 && Positions(k,2) <= Light(k,5) % Stop Before Light
            FinalAcceleration = LightAccel;
            Light(k,3) = 2;
            if Time(k,2) < 0.5 && Time(k,2) > 0
                FinalAcceleration = -32;
                Light(k,3) = 1;
            end
        elseif Light(k,2) == 3 || Light(k,2) == 2
            if Light(k,4) == 0
                if Locations(k,9) == 2
                    FinalAcceleration = LightAccel;
                    Light(k,3) = 2;
                elseif Locations(k,9) == 3
                    FinalAcceleration = TurnAccel;
                    Light(k,3) = 3;
                elseif Locations(k,9) == 1
                    FinalAcceleration = YieldAccel;
                    Light(k,3) = 4;
                else
                    FinalAcceleration = DistanceAccel;
                    Light(k,3) = 1;
                end
            else % Edit this for multiple cars making through decisions
                FinalAcceleration = DistanceAccel;
                Light(k,3) = 1;
            end
        else % if Light(k,2) == 1 or 5
            FinalAcceleration = DistanceAccel;
            Light(k,3) = 1;
        end
    end
end

```

```

else % Green Light
    if Locations(k,9) == 2 % Proceeding Straight
        ToIntersection = IntX + 12 + ToFPS(CarBehavior(k,3))*Yellow;
        if Positions(k,2) <= ToIntersection && Positions(k,2) > IntX + -6
            Light(k,2) = 1;
        elseif Positions(k,2) > ToIntersection
            Light(k,2) = 4;
        else
            Light(k,2) = 5;
        end
        % On a Green Light Proceeding Straight, always DistanceAccel
        FinalAcceleration = DistanceAccel;
        Light(k,3) = 1;

        elseif Locations(k,9) == 3 % Right Turn
            ToIntersection = IntX + 6 + ToFPS(CarBehavior(k,3))*Yellow -
(1/2)*CarBehavior(k,4)*Yellow^2;
            if Positions(k,2) <= ToIntersection && Positions(k,2) > IntX - 4
                Light(k,2) = 1;
            elseif Positions(k,2) > ToIntersection
                Light(k,2) = 4;
            else
                Light(k,2) = 5;
            end

            if Positions(k,2) <= Light(k,7) && Light(k,4) == 0
                FinalAcceleration = TurnAccel;
                Light(k,3) = 3;
            else
                FinalAcceleration = DistanceAccel;
                Light(k,3) = 1;
            end
        elseif Locations(k,9) == 1 % Left Turn
            ToIntersection = IntX + 0 + ToFPS(CarBehavior(k,3))*Yellow -
(1/2)*CarBehavior(k,4)*Yellow^2;
            if Positions(k,2) <= ToIntersection && Positions(k,2) > IntX - 6
                Light(k,2) = 1;
            elseif Positions(k,2) > ToIntersection
                Light(k,2) = 4;
            else
                Light(k,2) = 5;
            end

            if Positions(k,2) <= Light(k,8) && Light(k,4) == 0
                FinalAcceleration = YieldAccel;
                Light(k,3) = 4;
            elseif Positions(k,2) <= 4 && Positions(k,2) > IntX - 6 && Light(k,4) == 0
                FinalAcceleration = YieldAccel;
                Light(k,3) = 4;
            else
                FinalAcceleration = DistanceAccel;
                Light(k,3) = 1;
            end
        end
    end
end

Positions(k,8) = FinalAcceleration;

    if Positions(k,5) < 0
        Positions(k,5) = 0;
        Positions(k,8) = 0;
    end

end

% If Acceleration is infinity or -infinity do not plot it, choose another

```

```

%% If statement for moving the car based on the location and direction
    if Locations(k,8) == 1
        x = Locations(k,2);
y = Positions(k,2)+(Positions(k,5)*timestep)+((1/2)*Positions(k,8)*(timestep)^2);
        a = 6;
        b = 16;
    elseif Locations(k,8) == 2
x = Positions(k,2)+(Positions(k,5)*timestep)+((1/2)*(Positions(k,8))*(timestep)^2);
        y = Locations(k,3);
        a = 16;
        b = 6;
    elseif Locations(k,8) == 3
        x = Locations(k,2);
y = Positions(k,2)-(Positions(k,5)*timestep)-((1/2)*(Positions(k,8))*(timestep)^2);
        a = 6;
        b = 16;
    else
x = Positions(k,2)-(Positions(k,5)*timestep)-((1/2)*(Positions(k,8))*(timestep)^2);
        y = Locations(k,3);
        a = 16;
        b = 6;
    end

% Plotting the car based on position and direction

if Simulation == 1
    figure(1)
        Car(k) = rectangle('Position',[x y a b],'Curvature',0.3);
        Locations(k,2) = x;
        Locations(k,3) = y;
elseif Simulation == 0
        Locations(k,2) = x;
        Locations(k,3) = y;
end

% Filling in the Position Evaluation Matrix
PositionEvaluation(Allie,1) = j*timestep;
PositionEvaluation(Allie,k*2) = x;
PositionEvaluation(Allie,k*2 + 1) = y;

% Plotting the appropriate number next to the car
% if Locations(k,8) == 1
%     b = num2str(k);
%     c = cellstr(b);
%     Txt(k) = text(x+12,y+8,c);
% elseif Locations(k,8) == 2
%     if k < 10
%         b = num2str(k);
%         c = cellstr(b);
%         Txt(k) = text(x+5,y-12,c);
%     else
%         b = num2str(k);
%         c = cellstr(b);
%         Txt(k) = text(x+0.5,y-12,c);
%     end
% elseif Locations(k,8) == 3
%     if k < 10
%         b = num2str(k);
%         c = cellstr(b);

```

```

%         Txt(k) = text(x-12,y+8,c);
%     else
%         b = num2str(k);
%         c = cellstr(b);
%         Txt(k) = text(x-20,y+8,c);
%     end
% else
%     if k < 10
%         b = num2str(k);
%         c = cellstr(b);
%         Txt(k) = text(x+4,y+20,c);
%     else
%         b = num2str(k);
%         c = cellstr(b);
%         Txt(k) = text(x+0.5,y+20,c);
%     end
% end
% end

% Updating the Positions Matrix after each car has individually moved

% Position
Positions(k,4) = Positions(k,3);
Positions(k,3) = Positions(k,2);

if Locations(k,8) == 1
    Positions(k,2) = y;
elseif Locations(k,8) == 2
    Positions(k,2) = x;
elseif Locations(k,8) == 3
    Positions(k,2) = y;
else
    Positions(k,2) = x;
end

% Velocity
Positions(k,7) = Positions(k,6);
Positions(k,6) = Positions(k,5);
Positions(k,5) = Positions(k,6) + Positions(k,8)*timestep;
Locations(k,6) = ToMPH(Positions(k,5));

% Acceleration
Positions(k,10) = Positions(k,9);
Positions(k,9) = Positions(k,8);
% Find new value for Positions(k,8) next iteration (Acceleration)

% Time
Time(k,4) = Time(k,3);
Time(k,3) = Time(k,2);

% Observation
Observation(k,1) = k;
Observation(k,2) = Light(k,2);
Observation(k,3) = Light(k,3);
Observation(k,4) = Locations(k,5);
Observation(k,5) = Locations(k,4);
Observation(k,6) = Positions(k,2);
Observation(k,7) = Locations(k,6);
Observation(k,8) = FinalAcceleration;
Observation(k,9) = Time(k,2);

%% Car Direction and Location Evaluation

if Locations(k,8) == 1

```

```

        if Positions(k,2) >= IntY - 10
            Light(k,2) = 5;
        end
    elseif Locations(k,8) == 2
        if Positions(k,2) >= IntX - 10
            Light(k,2) = 5;
        end
    elseif Locations(k,8) == 3
        if Positions(k,2) <= IntY - 6
            Light(k,2) = 5;
        end
    elseif Locations(k,8) == 4
        if Positions(k,2) <= IntX - 6
            Light(k,2) = 5;
        end
    end
end

% if Light(k,3) ~= 1 && Locations(k,5) == 1
%     Light(k,3)
% end

% for temp = 1:size(TimeEvaluation,1)
%     if TimeEvaluation(temp,9) == 0
%     %
%     %     else
%     %         if (TimeEvaluation(temp,9) - TimeEvaluation(temp,5)) < TimeEvaluation(temp,11)
%     %             TimeEvaluation(temp,11) - (TimeEvaluation(temp,9) -
TimeEvaluation(temp,5));
%     %
%     %         if j == 0
%     %             end
%     %         else
%     %             end
%     %     end
%     end
% end

end

% change to 4 when passing through
% Calculate Number of Cars in Front of Intersection
for xx = 1:Cars

    % Intersection Location
    IntNumber = Locations(xx,7);
    IntX = Intersections(IntNumber,2);
    IntY = Intersections(IntNumber,3);

    % Find the location where the directions are the same
    Direct = find(Locations(:,5) == Locations(xx,5));
    % Create the new matrix for same direction data
    Check = zeros(length(Direct),9);
    Check(1,:) = Locations(xx,:); % Plot the first line

    % Delete Location where it is the same as the first line
    X = find(Direct(:) == Locations(xx,1));
    Direct(X, :) = [];

    % Place rest of similar direction data into the check matrix
    for z = 1:length(Direct)
        Check(z + 1,:) = Locations(Direct(z),:);
    end
end

```

```

        end

Front = 0;

if Locations(xx,8) == 1
for h = 1:length(Check(:,1)) - 1
    if Check(1,3) < Check(h+1,3)
        index = Check(h+1,1);

        if Light(index,2) ~= 5 && Positions(index,2) < IntY - 10
            Front = Front + 1;
        end
    end
end

if Front == 0 && Light(xx,2) ~= 4
    Light(xx,6) = IntY;
else
    Light(xx,6) = IntY - 35 - 25*Front;
end

elseif Locations(xx,8) == 2
for h = 1:length(Check(:,1)) - 1
    if Check(1,2) < Check(h+1,2)
        index = Check(h+1,1);

        if Light(index,2) ~= 5 && Positions(index,2) < IntX - 10
            Front = Front + 1;
        end
    end
end

if Front == 0 && Light(xx,2) ~= 4
    Light(xx,6) = IntX;
else
    Light(xx,6) = IntX - 35 - 25*Front;
end

elseif Locations(xx,8) == 3
for h = 1:length(Check(:,1)) - 1
    if Check(1,3) > Check(h+1,3)
        index = Check(h+1,1);

        if Light(index,2) ~= 5 && Positions(index,2) > IntY - 6
            Front = Front + 1;
        end
    end
end

if Front == 0 && Light(xx,2) ~= 4
    Light(xx,6) = IntY;
else
    Light(xx,6) = IntY + 19 + 25*Front;
end

elseif Locations(xx,8) == 4
for h = 1:length(Check(:,1)) - 1
    if Check(1,2) > Check(h+1,2)
        index = Check(h+1,1);

```

```

        if Light(index,2) ~= 5 && Positions(index,2) > IntX - 6
            Front = Front + 1;
        end

    end

    end

    if Front == 0 && Light(xx,2) ~= 4
        Light(xx,6) = IntX;
    else
        Light(xx,6) = IntX + 19 + 25*Front;
    end

    end

    end

    Light(xx,4) = Front;

end

if Simulation == 1
    pause(timestep) % Pausing the simulation to display dynamic change
elseif Simulation == 0
end

% frame = getframe(gcf);
% writeVideo(v,frame);

% Deleting the previous car that was displayed to update overall position
if j == (1/timestep)*TotalTime % To display final position of all cars

    else

Time;
Positions;
Light;
Locations;
Observation;
PositionEvaluation;
TimeEvaluation;

if Simulation == 1
    Observation
elseif Simulation == 0
    size(TimeEvaluation,1)
end

% Row and Column Names for Observation Matrix
% colNames =
{'Car','Decision','AccelType','Input','Output','Position','Velocity','Acceleration','Time
Behind'};
% ObservationTable = array2table(Observation,'VariableNames',colNames);

if Simulation == 1
    for l = 1:Cars
        delete(Car(l))
    %         delete(Txt(l))
    end
elseif Simulation == 0
end
end

```



```

end

% frame = getframe(gcf);
% writeVideo(v,frame);

jj = jj + 1;

if jj > 2*(Green+Yellow+Red)/timestep
    jj = 0;
end

% Counting Variable
Allie = Allie + 1;

if size(TimeEvaluation,1) >= MaxCars + (Cars*2)
    break
end

end

%% Evaluation

% Revised Data Compilation
TimeEvaluationRevised = TimeEvaluation(Cars+1:size(TimeEvaluation,1),:);
indices = find(TimeEvaluationRevised(:,10) == 0);
TimeEvaluationRevised(indices,:) = [];

% Time Evaluations Through Simulation
FinalTime = TimeEvaluationRevised(:,9);
InitialTime = TimeEvaluationRevised(:,5);
ITime = TimeEvaluationRevised(:,11);

% Efficiency Calculation
ActualTime = FinalTime - InitialTime;
Average_Time = mean(ActualTime);
Ideal_Average = mean(ITime);

Average_Efficiency = Ideal_Average/Average_Time*100;

ExtraTime = mean(ActualTime - ITime); % Extra Average Seconds per Car

% 3D Positions Matrix
CarPositions3D = zeros(Cars,3);

% Incorrect Calculations
Difference = ActualTime - ITime;
Indices = find(Difference(:) < 0);

Incorrect = TimeEvaluationRevised(Indices,:);
IDiff = (Incorrect(:,9) - Incorrect(:,5)) - Incorrect(:,11);

% Updated Time Evaluation Matrix
TimeEvaluationRevised(:,13) = ActualTime;
Standard = std(ActualTime);

% Directions
Left = 0;

```

```

Straight = 0;
Right = 0;

for ii = 1:length(TimeEvaluationRevised)

    In = TimeEvaluationRevised(ii,2);
    Out = TimeEvaluationRevised(ii,6);

    if Out - 1 == In || Out + 3 == In
        Left = Left + 1;
    elseif Out + 1 == In || Out - 3 == In
        Right = Right + 1;
    else
        Straight = Straight + 1;
    end

end

Left
Straight
Right

% Signal Timing Details
LightStatusRevised = LightStatus(TimeEvaluationRevised(1,5)/timestep +
2:TimeEvaluationRevised(size(TimeEvaluationRevised,1),9)/timestep + 2,:);

%% Saving Evaluated Matrices to Files

% Excel
filename = ('Cars60AdaptiveQ10.xlsx');
sheet = (Green/10) + 1;

xlswrite(filename,TimeEvaluationRevised,sheet,'A2') % Evaluation Matrix
xlswrite(filename,Average_Time,sheet,'O2') % Average Time
xlswrite(filename,Ideal_Average,sheet,'O3') % Ideal Average Time
xlswrite(filename,Average_Efficiency,sheet,'O4')
xlswrite(filename,ExtraTime,sheet,'O5')
xlswrite(filename,Standard,sheet,'O6')
xlswrite(filename,Left,sheet,'O7')
xlswrite(filename,Straight,sheet,'O8')
xlswrite(filename,Right,sheet,'O9')

% Notepad
dlmwrite(strcat(sprintf('60_Cars_Q10_%d',Green),'_Green_Adaptive.txt'),PositionEvaluation
,'newline','pc','delimiter','\t','precision',7);
dlmwrite(strcat(sprintf('60_Cars_Q10_%d',Green),'_Green_LightStatus.txt'),LightStatusRevised,
'newline','pc','delimiter','\t','precision',7);

end

% close(v)

% Can begin with cars at fixed positions, or record the random locations
% and initial speeds and behavior

% Based on positions and behavior, can determine how long it would take to
% reach destination without a light or other cars? This would be based on

```

```
% only allowing one intersection to run at a time no cars either

% Can factor in the light and determine the amount of time with a fixed
% light

% vary number of cars as well

%

% Add in the adaptive light and attempt to make light arrival time close
% to the ideal case

TotalMinutes = j*timestep/60
TotalCars = length(TimeEvaluation)
```