Navigating Intersections Through Mutual Consensus: An Approach For Autonomous Vehicles

Todd Macdonald[1], Ashwin Sreenivas[1], and Jessica Zhao[1]

[1]Department of Computer Science, Stanford University, Stanford, CA
{*tmacd, ashwinsr, jesszhao*}*@stanford.edu*

ABSTRACT

Smart cities of the future will be full of autonomous cars, and therefore, roads that are adapted to autonomous behavior. One feature of current roads that will soon be obsolete is the traffic light. With our Mutual Consensus Interaction Management Model (MCIM), we explore how autonomous vehicles interact with intersections in the absence of a centralized authority. Our model determines traffic flow by assigning each vehicle an acceleration or deceleration at each time step, determined by a look-ahead strategy and a cost function that penalizes collisions, time delays, and rapid changes in a vehicle's speed. We additionally simulate a traditional intersection governed by traffic lights as a baseline to compare average time delays and utilities and find significant improvements across each measure.

Keywords: Autonomous driving, consensus algorithms, intersection navigation

Github Repository: github.com/jessicae5/cs238-final-project

1. INTRODUCTION

Advancements in autonomous vehicles have inspired discussion on how roads will adapt to these agents. One such vision is the demise of human-centered navigation signals such as the traffic light. Traditional traffic light systems are designed for human drivers and are the source of several inefficiencies. A 2015 study by the Texas A & M Transportation Institute found that Americans annually spend 3 billion gallons of fuel and 7 billion hours idling due to traffic congestion. Nationally, that amounts to $160 billion in costs and is only projected to increase to $192 billion by 2020. However, a future with autonomous vehicles may drastically reduce these costs. We are interested in exploring how autonomous vehicles can interact with intersections if they no longer have a centralized authority such as a traffic light to tell them what to do. An analogy to illustrate what we hope to achieve is TCAS for cars: Can cars decide amongst themselves how to safely cross an intersection without a collision while minimizing their delay?

Assuming a fixed trajectory, an autonomous vehicle has one key decision to make when approaching an intersection: how much to accelerate or decelerate at the current point in time. This decision must incorporate several utility functions and restraints, however. The vehicle will want to keep the ride comfortable for its passengers, so the acceleration should not be abrupt. In addition, the vehicle will want to reach its destination as quickly as possible, while also minimally delaying other cars and not breaking traffic laws. Lastly and most importantly, the autonomous vehicle will need to minimize the chance of a collision.

This problem has many sources of uncertainty. Since there is no central authority, other autonomous cars arriving at the intersection will have uncertain behavior and be operating under their own set of priorities. A given vehicle at an intersection will need to predict the current and future speed of other cars, their current locations, and their future trajectories.

2. RELATED WORK

The promise of autonomous vehicles combined with the legacy infrastructure of traffic lights yields a space ripe for innovative solutions to tackle these inefficiencies. Many researchers have imagined alternate systems with a variety of environmental variables, from a mixed world of human and autonomous drivers to a complete world of fully autonomous vehicles. We look at a few studies that focus on a world with universally autonomous vehicles.

In their work on multiagent intersection management, Dresner and Stone assume that each vehicle acts as an autonomous agent with the goal of exceeding the efficiency of traffic lights. However, in place of the traffic light, they introduce a substitute centralized authority that manages reservations from incoming autonomous vehicles requesting to cross the intersection. Akin to booking a hotel reservation, vehicles may call ahead and book an intersection crossing. This requires the overhead of a smart agent for every single intersection, which can be costly. We seek to improve this model by removing the need for a centralized authority and having vehicles self-sufficiently communicate amongst themselves.

A study by Carlino, Boyles, and Stone also imagines a world of fully autonomous vehicles and introduces a decentralized system governed by market forces to handle intersection congestion. Each vehicle makes a bid to cross the intersection and can submit higher bids if they have greater wealth or greater urgency to get to their destination (for example, if the vehicle's passenger is late). The auction results in an orderly queue of traffic, which is individually based. However, we want to allow for simultaneous movement across the intersection that considers the incoming vehicles collectively and supports compatible crossing trajectories. Their model focuses on financial bids as the arbiter of intersection flow, while we seek to incorporate vehicle acceleration, deceleration, and crossing time in determining flow.

Another paper by Bashiri, Jafarzadeh, and Fleming similarly proposes an intersection management system for autonomous vehicles. Their decentralized model uses a cost function focused on reducing time delay, a measure we also incorporate in ours. However, they singularly focus on time, whereas we hope to not only minimize time across the intersection, but also minimize acceleration and deceleration for a more natural flow and comfortable passenger experience.

3. THE MODEL

Problem: Determine a policy $\pi(s)$ for each car i, which is the acceleration that car i should take from a given state s.

State and Action Space: To discretize our state space, we imagine an $r \times c$ grid with cars travelling in the four cardinal directions (north, south, east, west), where each direction of travel has l lanes. Each car has a position (x,y) in the grid, a speed from $v = \{0, 1, 2, 3, 4, 5\}$, and a direction of travel that remains constant (ie, for simplicity, we assume that cars cannot turn). The action space is defined by accelerations from $a = \{-2, -1, 0, 1, 2\}$.

Initialization of State Space

1. Initialize $rows \times columns$ grid with one set of perpendicular roads in the middle of the grid. Define the roads to have $lanes$ number of lanes in each direction.
2. Randomly place $numCars$ cars on the grid, such that the cars are evenly distributed between the 4 cardinal directions and that the cars are distance $d \sim Uniform[r/4, r/2]$ or $d \sim Uniform[c/4, c/2]$ away from the intersection, depending on whether the cars are facing north/south or east/west, respectively.
3. Set all cars to have an initial speed $s_v = v_{max}$

In our final simulations, we use $rows = columns = 10$, $lanes = 1$, and $numCars = 10$

Transitions: At each time step, the position of each car will advance deterministically according to its speed, direction, and chosen acceleration. For cars moving in the east and west directions, the transition function is as follows, where s is the state, sp is the future state, s_x is the x position value, sy is the y position value, and s_v is the current speed:

$$\begin{cases} T(s, sp, a) = 1 & sp_x = s_x + s_v + \frac{a}{2}, sp_y = s_y, sp_v = s_v + a \\ T(s, sp, a) = 0 & otherwise \end{cases}$$

For cars moving in the north as south directions, the transition function is similarly:

$$\begin{cases} T(s, sp, a) = 1 & sp_x = s_x, sp_y = s_y + s_v + \frac{a}{2}, sp_v = s_v + a \\ T(s, sp, a) = 0 & otherwise \end{cases}$$

These transition equations are identical to the mechanics equations in Newtonian physics.

Rewards: We define our reward by the following, where s_x indicates the x-position of the state, s_y indicates the y-position of the state, $c_c = -10000$ indicates the collision cost, $c_a = -10$ indicates the acceleration cost, and $c_t = -5$ indicates the time cost:

$$Reward(s, a) = numCars(s_x, s_y) * c_c + |a| * c_a + |v_{max} - s_v| * c_t$$

We use $numCars(s_x, s_y)$ in the reward equation to make the collision cost proportional to the number of cars in the collision; in addition, the magnitude of c_c is very large compared to c_a and c_t to make collisions highly discouraged. We chose c_a and c_t values such that the maximum possible time cost $||v_{max} - s_v| * c_t|$ at a given timestep t is roughly the same value as the maximum possible acceleration cost $||a| * c_a|$.

4. THE STOPLIGHT IMPLEMENTATION

To serve as a baseline model and a point of comparison to our mutual consensus intersection management model (MCIM), we implemented a simulation of an intersection using stoplights. The stoplight model is designed such that we can compare the expected utility and travel time of a car under the stoplight model of an intersection compared to that of a car with our MCIM model of an intersection. In this stoplight implementation, the transition function, state space, and action space are identical to those in our mutual consensus model.

The reward function is nearly identical to that in MCIM, except that we underestimate the time cost and collision cost. We underestimate the time cost to be 0 in all instances except when a car is waiting at a stoplight, and we underestimate the collision cost by assuming that collisions are impossible under the stoplight model.

We see that under these assumptions, the reward in MCIM for the same state and action pair is always lower than the reward in the stoplight model, where s_x and s_y are the x and y positions, s_v is the current speed, a is the acceleration, $c_c = -10000$ is the collision cost, $c_t = -5$ is the time cost, and $c_a = -10$ is the acceleration cost:

$$\begin{aligned} Reward_{MCIM}(s, a) &= numCars(s_x, s_y) * c_c + |v_{max} - s_v| * c_t + |a| * c_a \\ &<= [|v_{max} - s_v| * c_t] I_{s_v=0} + |a| * c_a \\ &= Reward_{stoplight}(s, a) <= 0 \end{aligned}$$

Therefore, if the expected utility of cars in the stoplight model is lower than the expected utility of cars in MCIM, we are able to conclude that MCIM implementation is strictly more optimal.

To simulate red and green lights, we assume that at any given timestep, only cars moving in the north and south directions or in the east and west directions can go through the intersection. Every $t_{interval}$ seconds, the stoplight will change and allow cars in the other two directions to proceed through the intersection. In the initial state for the simulation, the seconds remaining before a stoplight change, to either red or green, is a random integer between 0 and $t_{interval}$, inclusive.

All other simulation parameters, such as the initialization of car locations, are identical to our MCIM model.

5. THE CONSENSUS ALGORITHM

Given that there aren't central authorities, such as traffic lights, to coordinate among the autonomous cars, it is imperative that any algorithm coordinating their actions rely upon consensus. Computing the optimal policy that each car follows (to optimize some global utility metric), however, is computationally unfeasible. Since the state space described is exceptionally large, solving the optimally policy for a large number of cars, in a

large number of lanes, in a large intersection will be extremely computationally expensive. Furthermore, this is complicated by the stochastic arrival and departure of new cars in the intersection the arrival of any new car will immediately change the optimal policy and require a full re-computation of the optimal policy. Finally, it is important to realize that time is of the essence here since all these cars are likely travelling at high speeds, quick computation and coordination is of the essence for safety.

Given that an optimal solution is unfeasible, we turned our attention to heuristics that could approximate near optimal behavior; given the deeply practical need that such an algorithm would fulfill, we believe that any such algorithm must also have the following properties:

1. Fairness: any algorithm that requires wide adoption must contain some guarantees on fairness. This is not to say that everyone must have equal utility at all times, but rather that the algorithm will try to ensure that no participant is excessively penalized for the greater good. Under such constraints, if the optimal solution (to maximize global utility) was to have one car slow to a crawl while everyone else sped by, the algorithm would not pick this solution; rather a solution in which everyone got through the intersection in a moderate amount of time would be chosen.

2. Consensus: since there is no centralized authority to dictate policy, it is vital that all cars be able to agree on how they would collectively navigate the intersection.

3. Scalability: in a practical scenario, for safety considerations, we would need this to work for a large number of cars simultaneously (≥ 30). Thus, any solution in which the runtime of the algorithm is exponential in the number of cars is simply unacceptable.

4. Speed: Since all these cars are constantly moving, and at risk of crashing into each other, we would need every decision to be arrived at in under 0.1 seconds.

Below, we will describe the consensus algorithm that we designed, that satisfies all the properties above.

5.1 Description

The consensus algorithm works by repeatedly allowing the lowest utility participant to selfishly modify their own actions until we arrive at convergence. The algorithm begins by assuming that every participant takes no action for the next $LOOKAHEAD$ steps. Assuming that there are no crashes, this is the unique action that maximizes both individual and collective utility (since taking any other action can only give you negative utility). Thus, if there are no crashes, the algorithm immediately terminates, and each car simply continues at its current speed. However, in the case that a crash is detected in the next $LOOKAHEAD$ steps, the iterative section of the algorithm begins.

The iterative section of the algorithm works as follows. First every car calculates the utility of every other car (assuming that each car takes no action). Next, the car that suffers from the lowest utility is allowed to change its action vector for any of the next $LOOKAHEAD$ steps it modifies this in such a way that maximizes its own utility. Since the car with lowest utility could only have received this initial negative utility because of a detected crash, the new action (of accelerating or decelerating) could only be to avoid a crash. However, this small change could result in other cars being forced to crash. Hence, every car recomputes every other cars utility with this new action vector, and now the new car with the lowest utility is allowed to selfishly maximize its own actions. This process of identifying the lowest utility car, allowing it to maximize its own utility, and then recomputing the utility of every car, is repeated until no further changes can be made. At this point, a consensus is deemed to have been achieved, and each car executes the action that it agreed upon. Then, at the next timestep, the algorithm is run again.

5.2 Pseudocode

As we can see, the GetBestActions method forms the core of the Consensus Algorithm. Line 2 initializes the initial action vector to 0 for all cars. Lines 4 - 7 then repeatedly update the action vector, breaking out of the loop on line 6 if convergence has been detected.

Algorithm 1 The Consensus Algorithm

```
1: procedure GetBestActions
2:     actions ← [0 ... number of cars] × LOOKAHEAD
3:     loop
4:         utils ← CalculateAllUtilities(actions)
5:         newActions ← UpdateActions(utils, actions)
6:         if newActions == actions then break
7:         actions ← newActions
8:     return actions
```

```
1: procedure UpdateActions(utils, actions)
2:     minUtilityCar ← argmin(utils)
3:     possibleActions ← []
4:     for timestep in range(LOOKAHEAD) do
5:         for singleAction in actionSet do
6:             if singleAction is not valid then continue
7:             possibleAction ← singleAction in position timestep
8:             utility ← CalculateAllUtilities(possibleAction)
9:             possibleActions append (utility, possibleAction)
10:    return possibleAction in possibleActions with largest utility
```

UpdateActions is a subfunction for one iteration of the algorithm. Line 2 identifies the car with the current lowest utility, and then lines 4 - 9 iterate over every possible action that it could take for each timestep, and calculates the utility the car would enjoy if it chose that action. Line 10 finally picks the action that resulted in the highest utility.

6. RESULTS AND DISCUSSION

intersection type	avg time per car	avg utility per car	c_a	c_t	c_c	vehicle length
MCIM	18.074	-1.575	-10	-5	-10000	1
MCIM	17.995	-1.3	-5	-10	-10000	1
MCIM	18.182	-169.06	-10	-5	-10000	3
MCIM	18.1	-128.59	-5	-10	-10000	3
traffic lights	19.153	-112.5	-10	-5	-10000	1
traffic lights	19.15	-150.0	-5	-10	-10000	1
traffic lights	19.153	-112.5	-10	-5	-10000	3
traffic lights	19.15	-150.0	-5	-10	-10000	3

From our results, we see that when a vehicle is interpreted as the length of half of a 2×2 grid intersection (ie, similar to the length of a car) the MCIM model performs drastically better than the stoplight model for both average trip time and average utility per car, with respective average utilities of -1.575 and -112.5 and trip times per car of 18.074 and 19.153, for the case where $c_a = -10$ and $c_t = -5$. The reason that the utility is significantly lower with MCIM in this case is that the cars can apply minimal acceleration to slightly change their speed and narrowly avoid a collision; in comparison, the cars in the stoplight model cars that hit a red light have no choice but to completely decelerate to a speed of 0, incurring both a time and acceleration cost. For the same reason, cars in the stoplight model have a greater trip time compared to those in the MCIM model.

When vehicles are treated as having a significant length, such as 1.5 times the length of a 2×2 grid intersection (ie, similar to the length of a long RV), the MCIM model does worse than the stoplight model in terms of utility but still better in terms of average trip time per vehicle. For $c_a = -10$ and $c_t = -5$, MCIM had an average utility and trip time of -169.06 and 18.182, compared to -112.5 and 19.153 for vehicles in the stoplight model.

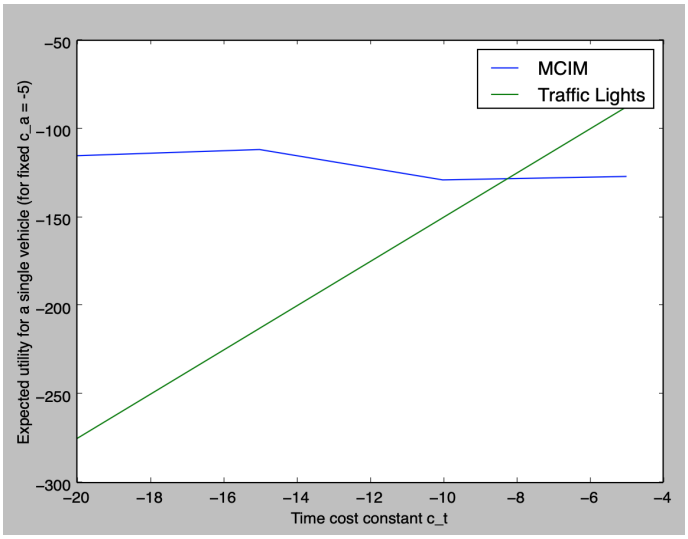

Figure 1. Expected utility of a single vehicle using the MCIM and traffic light models, for varying time cost constant c_t. The acceleration cost constant $c_a = -5$, collision cost constant $c_c = -10000$, and vehicle length = 3 are fixed.

We see that this difference can be explained by the increased magnitude of the acceleration cost in the MCIM model, as longer vehicles will need to more drastically slow down to avoid collisions.

If we weight time cost higher than acceleration cost, then the MCIM model performs better in terms of average utility than the stoplight model even with vehicles of long lengths. For $c_a = -5$ and $c_t = -10$, we see that the expected utility of a vehicle in MCIM is -128.59 compared to -150.0 in the stoplight model; this shows how the cost in the stoplight model is caused more from time cost, whereas the cost in MCIM comes more from acceleration cost.

7. CONCLUSION

In scenarios with short vehicle lengths, our model achieves greater efficiency in moving vehicles across an intersection than the traditional traffic light model while also considering the passenger experience by avoiding abrupt accelerations and decelerations.

In addition, our study shows that the utility in MCIM is highly influenced by the vehicle length. The MCIM model performs worse than the traffic light model in terms of utility in scenarios with both long vehicle lengths and a low time cost constant. In scenarios with either a short vehicle length or high time cost constant, MCIM always performs better, in terms of both expected utility and trip time.

As for future work, we can consider modifications to the cost function such as squared acceleration cost to more heavily penalize larger changes in acceleration. We can also incorporate priorities of each car, such as giving higher priority to ambulances, in determining intersection flow.

8. CONTRIBUTIONS

Todd Macdonald worked on the Intersection Management class (getting next states and actions, incorporating car lengths, simulating traffic light baseline) as well as creating the tables and graphs. Ashwin Sreenivas worked on formulating and coding the MCIM algorithm (getting best actions and executing them). Jessica Zhao worked on the Intersection Management class (initializing state, reward function, calculating utilities) and researching related work. All team members worked on brainstorming the MCIM algorithmic approach and writing the paper.

References

[1] Masoud Bashiri, Hassan Jafarzadeh, and Cody Fleming. Paim: Platoon-based autonomous intersection management. *arXiv preprint arXiv:1809.06956*, 2018.

[2] Dustin Carlino, Stephen D Boyles, and Peter Stone. Auction-based autonomous intersection management. In *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*, pages 529–534. IEEE, 2013.

[3] Kurt Dresner and Peter Stone. A multiagent approach to autonomous intersection management. *Journal of artificial intelligence research*, 31:591–656, 2008.

[4] Matthew Hausknecht, Tsz-Chiu Au, and Peter Stone. Autonomous intersection management: Multi-intersection optimization. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 4581–4586. IEEE, 2011.

[5] David Schrank, Bill Eisele, Tim Lomax, and Jim Bak. *2015 Urban Mobility Scorecard*. 2015.