Nicholas Jones
ENGW 3302
Unit 2: Academic Review
Tom Akbari
April 10, 2015

THE ROAD TO AUTONOMOUS VEHICLES

A Safe Review of Trajectory Planning

Abstract

This paper will review recent research into the development of intelligent trajectory planning in autonomous vehicles. It will focus on safety and reliability in novel, unknown, dynamic environments, because such vehicles must protect human life and property regardless of circumstance. Various algorithms that have been employed to achieve this goal will be discussed, including Graph Search, Fuzzy Logic, and Machine Learning. It is proposed that safe reaction to novel situations requires flexible decision making based on a wealth of prior experience. Machine Learning exemplifies these characteristics the best; however, care must be taken to verify implementation correctness.

Introduction

In the past five years, self-driving vehicles have begun the move from proof-of-concept research to sellable products. These new intelligent vehicles present solutions to many of the major issues that plague our roads today. First and foremost, they will eliminate most, if not all, crashes that result from human negligence. In 2013, the number of fatalities in the United States by traffic accident was 33,808 [1]. According to a study released by the World Health Organization:

> Road traffic injuries are the eighth leading cause of death globally, and the leading cause of death for young people aged 15–29 (1, 2). More than a million people die each year on the world's roads, and the cost of dealing with the consequences of these road traffic crashes runs to billions of dollars (3). Current trends suggest that by 2030 road traffic deaths will become the fifth leading cause of death unless urgent action is taken (2) [1:8].

Autonomous vehicle technology will, at the very least, enhance our driving ability by double checking our mistakes and taking corrective action. It will also serve as a way to take diving out of the hands of those that are too tired, drunk, or distracted to pay attention to the road, and free others who would rather work, text, or sleep than drive.

This paper will cover the, as of yet, pubescent field of trajectory planning for autonomous vehicles, focusing on safety and reliability in new, unknown, changing environments. While an autonomous car may be safe most of the time, computer systems do not naturally have a human-like caution of the unknown. As a result, unknown environments may cause an otherwise functional vehicle to perform strange, life-threatening actions such as to driving off a cliff or into a wall. If were the case, the technology would not only be dangerous, but it would lose the sense of trust that is critical for it to prosper. We will present several classes of algorithms for trajectory planning and discuss the flexibility and safety of each under varying environments.

Graph Search

Review

Graph search algorithms attempt to find paths by iteratively exploring a domain of states interconnected by actions. In the classical formulation, this collection of environment states is usually known before-hand, so an optimal path can be calculated once and performed offline (without further thought.) Exploring many states is computationally intense and generally cannot be performed in real-time at large scales. Furthermore, graph search algorithms traditionally do not take into account probability such as imperfect actions or environments that change during action [2].

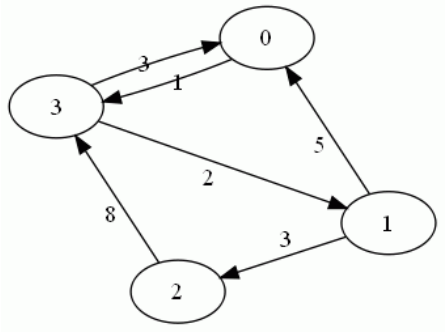

FIGURE 1 A DIRECTED, WEIGHTED GRAPH [3].

In the field of autonomous robotics, A* (A-Star) is the most commonly used graph search algorithm. In A*,

computation begins at the source node. From here the algorithm chooses the neighbor node that has the lowest cost to reach, and adds it to a collection of frontier nodes. It repeats this process, choosing from unvisited neighbors of frontier nodes, until it either reaches the destination node or it explores all possible nodes that were reachable from the source (meaning there is no path.) If it reaches the destination, it then back-tracks its path to the source to produce the optimal path [2]. This is a common formula for graph search algorithms, however, the important part of A* is the cost values, which are calculated via equation 1.

$$Cost(s) = g(s) + h(s)$$

EQUATION 1 COST CALCULATION OF A STATE USING A* [2]

In this equation, s is the state we are moving to, $g(s)$ is the cost to reach that state from the source, and $h(s)$ is a heuristic of the expected remaining cost to reach the destination from s. If we set $h(s)$ to 0, then we choose the optimal action available to us at each step, disregarding future actions. This is called uniform-cost search, and it results in an optimal path from the source to the goal; however, it also results in sub-optimal performance in the average case, since it doesn't necessarily expand in the direction of the goal. For example, if there were two paths, one being a 100-step, one-cost path, and the other being a one-step, five-cost path, we would expand six times along the ten-step path before taking the five-cost action. Whereas, in A* we would only expand in that direction three times, because each step away from the goal increases $h(s)$. In trajectory planning, we're typically dealing with location states, so a good heuristic for the remaining cost is the Euclidean distance to the goal. This is an optimistic guess of the remaining cost, which is a required property of $h(s)$. As a result of this optimization, we expand via the sequence of actions that have the lowest total expected cost for the entire path. If $h(s)$ is always lower than the actual remaining cost and it is consistent—meaning that all values of h obey the triangle inequality rule—the resulting path will still be optimal [2].

$$h(s') \leq h(s) + distance(s, s')$$

EQUATION 2 TRIANGLE INEQUALITY RULE [2]

Overall, graph search computes the entire path from a source to a destination via a series of actions. In implementation, it is typically done beforehand and performed offline, because this computation is costly. As a result it doesn't deal with dynamic changes very well, since it must re-compute the entire path each time. Despite this, A* is still a reasonable solution to trajectory planning if the environment is sufficiently stable and known, because it always produces the mathematically optimal path from the provided information.

Research

A* is an incredibly powerful algorithm, however, it still faces limitations which limit its applicability to real time robotics such as autonomous vehicles. As a result, most research into graph search algorithms has focused on overcoming these limitations.

One of the primary limitations that A* faces is that it needs a complete map of the environment to function. In robotics, this is rarely the case. In 1994, Anthony Stentz at CMU released a paper on D* to account for this flaw. In the scenario he considered, a robot enters the environment with imperfect information, but has a sensor that it can use to read its environment. Previous research also made this assumption, however, the results were suboptimal paths, because they decided to locally avoid obstacles or wander randomly rather than re-compute the optimal path. This was a reasonable decision under the assumption that global re-optimization is costly. However, Stentz proposed that it is possible to re-compute only the section of the path which had changed [4].

To achieve this, the robot would first calculate an optimal path, and attempt to follow it. If it sees cost information that conflicts with its model of the world, it updates that state information and adds the neighbors to an OPEN list with either a RAISE or LOWER flag indicating direction of the cost change. The algorithm then expands these OPEN states until the minimum value of the heuristic over all states in the OPEN list is greater than or equal to the heuristic at the robot's state. In this way, the algorithm fixes the costs and re-achieves the optimal path, while only touching a small part of the state space [4]. In later work, Stentz improved D* further, resulting in Focused D*, which decreased the total time required for the initial path calculation and re-planning operations, and achieved a complete generalization of A* for dynamic environments [5].

Other research by De et al. looked into applying vehicle kinematic constraints to A*. "Kinematic A* (KA*) includes a simple kinematic model of the vehicle to evaluate the moving cost between the waypoints of the path in a tridimensional environment [6]." It constrained movements by turning radius and maximum rate of climb and added a required distance from obstacles to create a safe buffer-zone. Furthermore, simulations by the authors using aircraft models integrated wind patterns into the world, so they could find the optimal

sequence of actions to reach their goal regardless of condition. Using this algorithm, they noticed that the paths that they achieved were significantly smoother and safer than paths generated using A* [6].

Analysis

Graph search algorithms are an incredibly powerful tool for computing autonomous vehicle trajectories. Unfortunately, classical implementations were not flexible enough to run in real-time, dynamic environments where the agent has imperfect knowledge. D*, Focused D*, and KA* present interesting improvements on A* to resolve these issues.

Computationally, D* is superior to A* given that it doesn't need to re-compute the entire environment when it encounters inconsistencies in its model. In autonomous vehicle trajectory planning this is invaluable, since it means that the vehicle can react to changes and still achieve efficient operation. KA* is also valuable, since the paths that it calculates take into account physical limitations. While this is important for feasibility concerns, it's also valuable, because as vehicles approach their limitations, it is expected that issues such as equipment malfunction or imperfect path performance are more probable. If a safe path from A* relies on a vehicle turning 360 degrees on the spot, who knows what will happen if it can't?

Despite these concerns, there are still several issues with graph search algorithms for trajectory planning. First, in their current state, they are incapable of thinking ahead. While a path may be optimal and safe now, it may not be in 10 seconds when a child runs across the road. A vehicle that does not account for such events may find itself unable to respond in time. Additionally, the current algorithms consider all environments to be approximately the same, and actions will occur in the same optimal manner. While KA* took into account vehicle limitations, we should perform more research on representing environment state such as weather, traffic, and other environmental features in graph planning algorithms. Despite these flaws, graph-search algorithms are a powerful way to compute optimal vehicle trajectory. Although they may not be the best choice for autonomous vehicles in dynamic, complex environments they are a staple of the autonomous robotics industry.

Fuzzy Logic

Review

Fuzzy logic is class of algorithm used in autonomous trajectory planning and other control systems. Overall, the concept can be seen as an approximation of how humans solve problems by fitting information into categories and reacting to category combinations based on rules.

For example, in table 2 we present several rules that map input values to actions. If the room is COLD, we should HEAT the room, if the room is HOT, we should COOL the room, and if the room is WARM, we should do nothing. Note that rules could require multiple features to activate. For example, maybe we should only HEAT if we have enough money to pay the heating bill. Overall, the simplicity of these rules and the ability to easily write more is a powerful feature of fuzzy logic, since it allows for the modification of instance behavior without re-implementing the entire algorithm.

TABLE 2 RULES THAT MAP VARIABLES ACTIONS

Value	Action
COLD	HEAT
WARM	OFF
HOT	COOL

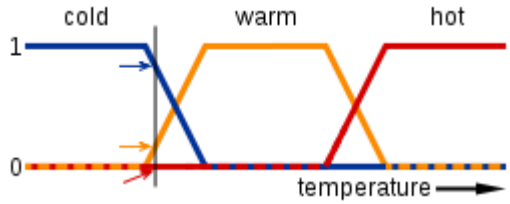

FIGURE 3 A MEMBERSHIP FUNCTION THAT MAPS TEMPERATURE TO 3 VARIABLES [7]

Fig. 3 shows a membership function that maps input temperature to the variables COLD, WARM, and HOT. Again, this separation of knowledge from algorithm presents a simple way to change the behavior of an implementation with re-implementing the entire algorithm. We could easily change the limits and gradients or add new variables such as KINDA_WARM or ICE_AGE_COLD. While these functions are simple, they clearly mimic the way a human might think about temperature. Rather than consider precisely the value 71.2 degrees Fahrenheit, we are more likely to think, "It's warm", and act accordingly.

Research

In autonomous vehicle research, fuzzy logic as has been applied to just about everything from trajectory planning to heater control.

Farhi et al. researched fuzzy control algorithms to optimally control the trajectory of a car within a road. They took sensor data regarding the distance to the

curb on either side of the car and the distance to the curb in front of the car, and parsed them using various membership functions. Braking reserve became either ZERO, MIDDLE, or HIGH, and delta distance front became CLOSER, CONSTANT, or FURTHER. Their algorithm utilized two weighted rule sets to find the optimal curve while staying within the lane of the road. The power of this algorithm is that the rules for matching the environment were weighted higher than the goal rules. This meant that the car could push towards the optimal goal curve as much as possible but never drive off the road. [8]

Milanés et al. used fuzzy logic to create an intelligent overtaking system that relies on vision for vehicle detection. They used stereoscopic cameras measure the width, length, and time-to-collision for the vehicle that they wanted to overtake. From here they calculated the angle at which to turn using fuzzy logic that maintained both comfort and safety by taking into account the velocity of the overtaking car. Overtake speed was calculated as a function of the relative velocities of the vehicles and the length of the overtaken vehicle clamped by the speed limit. As a result, the overtake maneuver gets activated automatically when the autonomous vehicle becomes aware that a collision will occur. The autonomous vehicle changes to the overtake lane, accelerates to reduce the overtake time, and finally merges back into the slow lane in front of the overtaken vehicle [9].

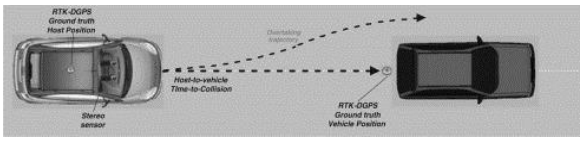

FIGURE 4 A DEPICTION OF THE SCENARIO STUDIED IN MILANÉS ET AL. [9]

Naranjo et al. used fuzzy logic to implement the guidance system of two Citroën Berlingo vans. Functionalities that this included were steering wheel and vehicle velocity control as well as adaptive cruise control (ACC), stop-and-go maneuvers, and overtaking. For steering control they used two fuzzy variables: "lateral_error" and "angular_error" that mapped errors in pose to left and right values. For example if the angle of error is counterclockwise, the angular_error is left. They combined these values to produce steering_wheel, the direction the wheel should turn to correct the errors. To model the human knowledge of this problem they needed only two rules:

1. if lateral_error or angular_error is left:
   steering_wheel = right
2. if lateral_error or angular_error is right:
   steering_wheel = left

While these rules are simple, they match closely with human behavior under most circumstances [10].

Analysis

Fuzzy logic is a powerful concept in autonomous vehicle trajectory planning. Compared to graph search algorithms it has the incredible ability to use human decision making strategies to make generalized decisions over multiple domains of data. Furthermore, the separation of algorithm and knowledge mean that rules can be optimized without touching the underlying code. This means that with enough man power and foresight, we could potentially think of every event and write rules for each situation to account for them. This is unlikely. While this knowledge works well on a general level, and will function will in most environments, it still relies on human forethought to come up with rules. If we miss a rule that is important in a particular situation, the system will fail.

On the other hand, fuzzy logic is easy to test, since we need only check that all rules respond to the appropriate input. For this reason, it's likely that autonomous vehicles will use fuzzy logic for some time to come. Whereas with graph search we would need to run many tests to confirm that our concept of an optimal path is correct, with fuzzy logic we need only confirm that our rules cover an acceptable percentage of likely situations. This is a clear metric, which industry can follow.

Machine Learning

Review

Machine learning is potentially the most powerful tool we have today, although there is still a lot of work to complete before it is ready. The most powerful feature of machine learning is its direct interaction with data. Whereas, with graph search we wrote the algorithm based on the concept of an optimal path, and with fuzzy logic humans wrote the rules, with machine learning we need only provide billions of data points over a collection of features [2]. Given sufficient training data, the agent will teach itself how to act so it can achieve optimal behavior in most situations based on the probability of events and different responses to these events [2].

This is similar to fuzzy logic, because the autonomous vehicle is essentially developing its own knowledge set, however, it is significantly more powerful, since it can achieve precisely tailored results to all situations contained in the data. If a vehicle were to have issues

with a particular situation, the most likely solution is to get more data on that situation and have the vehicle teach itself how to react [11].

Research

Wang et al. performed research into teaching autonomous vehicles to cruise control in environments with "unknown dynamics and external disturbances [12]." For their cruise controller they used a proportional-integral (PI) controller linked to a kernel-based least-squares policy iteration (KLSPI). Using data obtained from real cars, they trained models to stay within as close to a desired speed as possible. They then tested their cruise controller on autonomous vehicles in both urban and rural environments, confirming that it was possible to develop an offline planner that could control vehicle velocity without prior knowledge of vehicle kinematics or environment characteristics [12].

Bagnell et al. discuss DARPA's UGCV-Perceptor Integration (UPI) program in which they built Crusher and Spinner, autonomous robots designed to traverse novel, unstructured (rural) terrain to reach a goal 0.2-2 km away from the start in optimal time. Over the course of the project, they used machine learning to solve various problems. For example, "Because Crusher is a six-wheeled skid-steer vehicle, its motion is difficult to model; self-supervised learning was applied (see the "Vehicle Modeling" section) to learn response to control inputs and improve following of commanded paths [13]." They also used multiclass logistic regression to classify the terrain based on features from LADAR scanners and cross-calibrated color and

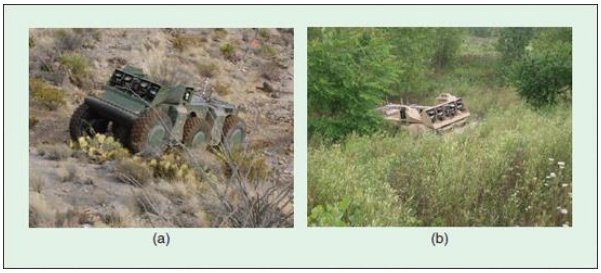

FIGURE 5 CRUSHER ROBOTS FROM THE UPI PROGRAM [13]

near infrared cameras [13]. Additionally, they implemented a novel "learning to search" (LEARCH) algorithm that used inverse optimal control to train the robots to find the goal using a collection of paths that the researchers demonstrated [13]. Overall, throughout the project "machine learning techniques were heavily used to provide robust and adaptive performance, while simultaneously reducing the required development and deployment time." The robots developed were tested in completely unknown, extreme environments over numerous tests and more than 1000 km. Over the course of the experiments, they averaged human intervention once every 20 km. Compared to previous attempts using manually engineered cost functions, researchers found that the robot would require significantly less time to configure using LEARCH. Additionally, they found that the robot chose to drive faster through safer terrain than when using a human-engineered interpretation of the environment [13].

Analysis

Machine learning is incredibly powerful, and has revolutionized the field of autonomous robotics in recent years. Whereas other algorithms require humans to interpret the environment, machine learning algorithms need only see the result of various actions and change their models to suit the environment. This eliminates much of the overhead from configuring new systems, since engineers need not spend as much time finding the correct values for cost functions such as is traditionally done with graph search algorithms.

Overall, machine learning is incredibly promising, and the future of autonomous vehicles rests on its shoulders. Over the next decade or so, we should continue to research machine learning, focusing on teaching machines using human demonstration as was done in UPI. It is important, however, that we verify that machine learning algorithm implementations are trained correctly before they are put in positions of power. This may be more difficult that with fuzzy logic, since it is harder to know what a machine learning algorithm thinks of an environment.

Conclusion

This paper has reviewed recent research into trajectory planning in autonomous vehicles, focusing on safety and reliability in novel, unknown, dynamic environments. These characteristics are important because the primary concern of such vehicles must always be to protect human life and property regardless of circumstance.

Of the algorithms discussed graph search is the most efficient, since it revolves around finding an optimal path from a source to a destination. While classical variations of graph search were limited to known, static environments, recent research has enabled autonomous robots such as cars to react to changes in real time and factor physical limitations in their planning process. Despite this, they still face the limitations of foresight

into a situation, which makes them suitable only in controlled situations.

Based on a review of the field, it would seem that fuzzy logic is currently the most commonly used algorithm of the three. This is primarily because it allows researchers to quickly map their knowledge of the field to robotic actions, and its performance relies only on the completeness of its rule-set. Furthermore, many control problems can be described simply using only a few rules and variables.

Finally, machine learning is probably the most powerful tool we have moving forward. Its direct interaction with data is similar to fuzzy logic's rules. However, we need only ensure that the system has enough data to have seen situations before and be able to predict the best way to react. Unfortunately, this may also be its downside, since the logic of the robot is located within the precise interaction of weighed variables based on potentially hundreds of billions of data points [14]. Therefore, while machine learning is more flexible and responsive than fuzzy logic, it will likely be harder to verify that implementations are free from bugs.

Overall, autonomous trajectory planning is just beginning to be viable. Companies and researchers around the world have developed feasible solutions that respond well to common problems such as overtaking, lane changing, and obstacle detection. From here, it is recommended that we focus on achieving reliable behavior in most situations rather than push for new features. We also need to consider the best way to combine research for various sub-problems into a single coherent model. In the meantime, it may be appropriate to trial features in traditional cars as human-complement systems to test responsiveness and to gather data for testing and training models. All in all, there's still a lot to do, but we should see autonomous vehicles entering public roads en-mass within the next decade as current research matures.

References

[1] "WHO | Global status report on road safety," *WHO*. [Online]. Available: http://www.who.int/violence_injury_prevention/road_safety_status/2013/report/en/. [Accessed: 10-Apr-2015].

[2] S. Russell and P. Norvig, *Artificial Intelligence - A Modern Approach*, 3rd ed. Pearson.

[3] "Boost Graph Library: Graph Theory Review - 1.57.0." [Online]. Available: http://www.boost.org/doc/libs/1_57_0/libs/graph/doc/graph_theory_review.html. [Accessed: 10-Apr-2015].

[4] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *, 1994 IEEE International Conference on Robotics and Automation, 1994. Proceedings*, 1994, pp. 3310–3317 vol.4.

[5] A. Stentz, "The Focussed D* Algorithm for Real-time Replanning," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, San Francisco, CA, USA, 1995, pp. 1652–1659.

[6] L. De and G. Guglieri, "Advanced Graph Search Algorithms for Path Planning of Flight Vehicles," in *Recent Advances in Aircraft Technology*, R. Agarwal, Ed. InTech, 2012.

[7] "261px-Fuzzy_logic_temperature_en.svg.png (PNG Image, 261 × 106 pixels)." [Online]. Available: http://upload.wikimedia.org/wikipedia/commons/thumb/6/61/Fuzzy_logic_temperature_en.svg/261px-Fuzzy_logic_temperature_en.svg.png. [Accessed: 01-Apr-2015].

[8] O. Farhi and Y. Chervenkov, "Optimal fuzzy control of autonomous robot car," in *Intelligent Systems, 2008. IS '08. 4th International IEEE Conference*, 2008, vol. 1, pp. 4–62–4–65.

[9] V. Milanés, D. F. Llorca, J. Villagrá, J. Pérez, C. Fernández, I. Parra, C. González, and M. A. Sotelo, "Intelligent automatic overtaking system using vision for vehicle detection," *Expert Syst. Appl.*, vol. 39, no. 3, pp. 3362–3373, Feb. 2012.

[10] J. E. Naranjo, M. A. Sotelo, C. Gonzalez, R. Garcia, and S. MA, "Using Fuzzy Logic in Automated Vehicle Control," *IEEE Intell. Syst.*, vol. 22, no. 1, pp. 36–45, Jan. 2007.

[11] *GPU Technology Conference 2015 - Dr. Andrew Ng.* .

[12] J. Wang, X. Xu, D. Liu, Z. Sun, and Q. Chen, "Self-Learning Cruise Control Using Kernel-Based Least Squares Policy Iteration," *IEEE Trans. Control Syst. Technol.*, vol. 22, no. 3, pp. 1078–1087, May 2014.

[13] J. A. Bagnell, D. Bradley, D. Silver, B. Sofman, and A. Stentz, "Learning for Autonomous Navigation," *IEEE Robot. Autom. Mag.*, vol. 17, no. 2, pp. 74–84, Jun. 2010.

[14] *Peter Norvig: How Computers Learn*. 2015.