Safe Temporal Planning for Urban Driving

Bence Cserna and **William J. Doyle** and **Tianyi Gu** and **Wheeler Ruml**
Department of Computer Science
University of New Hampshire
Durham, NH 03824 USA
bence, doyle, gu, ruml at cs.unh.edu

Abstract

A self-driving car must always have a plan for safely coming to a halt. Often, finding these safe plans is treated as an afterthought. In this paper, we demonstrate that techniques explicitly designed for safety can yield higher quality plans and lower latency than conventional planners in an urban driving setting. We adopt ideas from a previously-proposed safe online real-time heuristic search method to the spatiotemporal state lattices used when planning for autonomous driving. We experimentally compare our proof-of-concept implementation to conventional methods and find significantly improved performance while still maintaining passenger comfort and safety.

Introduction

A central goal of artificial intelligence is the construction of autonomous systems. It is becoming more common that these systems interact closely with humans. Perhaps the most intimate way that humans can interact with an autonomous system is to climb inside it and put their lives in the hands of its control system. Interestingly, it is exactly such systems, in the form of self-driving automotive mobility systems, that are predicted to become widespread in the coming decades. It is crucial that the AI systems that decide on the actions of autonomous vehicles be designed with safety as a fundamental aspect.

Self-driving vehicle technology has many potential benefits for individuals, such as reduced collisions, improved mobility, and reclaiming time spent driving. They also have the potential for broad benefits to society and the environment, such as reduced car ownership, space devoted to parking, and the number of vehicles required. However, there are still significant problems to be surmounted to achieve this vision. For example, in addition to avoiding actual collisions, for this technology to be successfully adopted, individuals need to feel subjectively safe and comfortable while using an autonomous vehicle. In this paper, we address the problem of planning trajectories for autonomous vehicles, taking into account both objective safety, given the predicted trajectories of nearby vehicles and pedestrians, and subjective passenger comfort.

Background

Trajectory planning for urban driving has proven to be a complex problem because of the inherently dynamic environment. A typical driver can encounter hundreds of other vehicles and many more pedestrians. Not only does the planner need to consider the spatial location of each part of this environment but their temporal evolution needs to be built into the planning techniques that are employed. These two aspects need to be intelligently combined to produce high quality and computationally feasible algorithms to address the urban driving problem.

Hierarchical methods are state-of-the-art for addressing the difficulty of autonomous urban driving (Paden et al., 2016). At the highest level, a vehicle must be able to select a route from a road network based upon its current position in that network and the desired destination of the passenger. This network can be represented as a graph with millions of weighted edges. Traditional offline algorithms such as A* are computationally infeasible, as planners are expected to run at a rate of ten times per second. Once a route has been identified, the vehicle must select a sequence of behaviors to use along that route. These behaviors correspond to situations such as ‘highway cruising’ or ‘stopping at a stop sign’ and identify the pertinent rules of the road with respect to other aspects of the environment, such as pedestrians and other drivers. Next, a motion planning algorithm will decide the vehicle trajectory used to achieve the next behavior. This motion plan then becomes a reference trajectory for a low-level controller to achieve using throttle and steering inputs while correcting for errors and inaccuracies from the vehicle model. These four components of route planning, behavior selection, motion planning, and control comprise the hierarchical approach to autonomous driving.

Our work addresses urban driving at the level of motion planning: we want to send trajectories to the vehicle controller such that the motions are comfortable for passengers while simultaneously ensuring that they are safe. We define a safe trajectory as one that reaches a safe state. In urban driving, a safe state is one in which the vehicle is stopped (Shalev-Shwartz, Shammah, and Shashua, 2017).[1] We assume that the vehicle replans frequently, thus only the first part of a safe trajectory will be executed in most cases before a new trajectory is computed to replace the current plan.

The planner also needs to take into account the comfort of

[1]For highway driving, a safe state might be one where the car is pulled over at the side of the road. We do not address this in this paper.

its passengers. Specific ranges of acceleration, when applied to the autonomous vehicle, create an uncomfortable riding experience. Allowing a high amount of acceleration to be chosen by the planner and executed on the vehicle leads to passenger discomfort due to the physical stress the human body undergoes when the vehicle quickly accelerates.

McNaughton et al. (2011) devise acceleration profiles for a spatiotemporal lattice. These profiles can be used to construct a constraint on the planner to maintain the acceleration of the autonomous vehicle within comfortable ranges. Our work addresses these two crucial aspects of the urban driving problem. We construct a planner that can remain safe while limiting the control of the vehicle for the passengers to be comfortable.

In this paper, we study a general online method for guaranteeing that the planner will find a series of safe actions for the agent to execute. We dynamically allow the agent to plan for its current goals and devise a way to balance the safety of the trajectories the agent is executing while being as close to the edge of non-safe action execution as possible. We define a general problem setting for the use of this technique and study the use of it in the domain of simple urban driving. We empirically test this method on a simulated vehicle with inertia and find that the new techniques dramatically outperform the conventional ones.

Previous Work

A state lattice (Pivtoraiko, Knepper, and Kelly, 2009) is a discretization of a continuous state space. A Spatiotemporal state lattice (Ziegler and Stiller, 2009) is the result of combining a traditional state lattice with time and velocity dimensions. The urban driving domain requires the planner to be able to consider both time and space while planning. The state lattice gives us a method for searching through a static environment; however, adding in time and velocity to the state space lattice can lead to an exponential blowup of the size of the search space. Even assuming a modest number of possible accelerations applied to a vehicle, the state space can contain nearly 12 million trajectory edges that would need to be evaluated during each planning iteration (Pivtoraiko, Knepper, and Kelly, 2009). On the other hand, it is a difficult task to compose search heuristics for the complex cost function of urban driving. For example, McNaughton et al. (2011) applied exhaustive search on a spatiotemporal state lattice.

CL-RRT (Kuwata et al., 2009) is a real-time motion planning algorithm that can guarantee safety. To deal with dynamic obstacles, in each planning iteration, the algorithm cleans the motion tree by removing the invalid tree nodes and saving unconnected subtrees into a stand-by forest. A random state is sampled by biasing the nearest tree in the forest or the goal. If the new sampled state is in one of the subtrees, then it will try to connect the nearest state on the current motion tree to the sampled state by solving a boundary value problem. Otherwise, it will perform the conventional RRT extend routine. They guarantee safety by ensuring that the vehicle is stopped and safe at the end of the trajectory. We will refer to these approaches as plan-to-stop. However, the random feasible solution that is constructed by the RRT could be a highly sub-optimal plan (Karaman and Frazzoli, 2011).

Algorithm 1: SafeTLP

```
   Input: $s_{root}$
1  perform BEST-FIRST SEARCH on lattice to find a
     potentially unsafe trajectory $T$ from $s_{root}$ to a partial
     goal
2  $s_{current} \leftarrow T.last$
3  while $s_{current}$ exists do
4      perform BEST-FIRST SEARCH on $d_{safe}$
           from the node $s_{current}$ to $s_{goal}$
5      if $s_{goal}$ is found then
6          cache the safe partial path from $s_{current}$ to $s_{goal}$
7          return $\langle s_{root} \dots s_{current} \rangle \langle s_{current} \dots s_{goal} \rangle$
8      else
9          $s_{current} \leftarrow s_{current}.predecessor$ in $P$
10 BEST-FIRST SEARCH to find a safe trajectory $T$ from
     $s_{root}$ to $s_{goal}$
11 return $T$
```

SafeRTS (Cserna et al., 2017) is a real-time search algorithm that can guarantee safety. It explicitly tries to prove nodes are safe and finds a plan to a safe node. SafeRTS searches for a partial real-time plan that has a frontier node with the most promising f value and guarantees that plan could be lead to a safe node in the lattice. The authors introduce a safety heuristic, $d_{safe}(n)$, that estimates the distance through the state space from a given node n forward to the nearest safe state. As an online search algorithm, SafeRTS distributes the expansion allowance between exploring the best f state and attempting to prove its safety. The safety proof performs a best-first search on d_{safe}.

Safe Temporal Lattice Planning

We now turn to applying Cserna et al. (2017)'s notion of safe planning to spatiotemporal state lattice planning. The planner's goal state is a location a predefined distance along the current route with a zero velocity. Our technique, Safe Temporal Lattice Planning (SafeTLP), first performs a best-first-search from the root state s_{root} until a partial goal state is expanded (line 1 of Alg. 1). A partial goal state is a state that matches the location of the goal state but may not have the correct speed. The priority function of the best-first search prioritizes states with lower distance-to-goal, earlier goal achievement time, and higher speed.

This naïve solution is inherently unsafe as it does not consider upcoming obstacles and dead-ends states beyond the explored spaces or partial goal. To ensure safety SafeTLP proves that a prefix of the naïve trajectory is safe by constructing a safe trajectory starting with this prefix. The algorithm first attempts to prove that the last state of the trajectory is safe (line 4). If the safety proof is not successful, SafeTLP falls back to the preceding state on the trajectory(line 9). The last few states in the naïve trajectory may be skipped if a maximum allowed deceleration would not allow the agent to come to a full stop from these states.

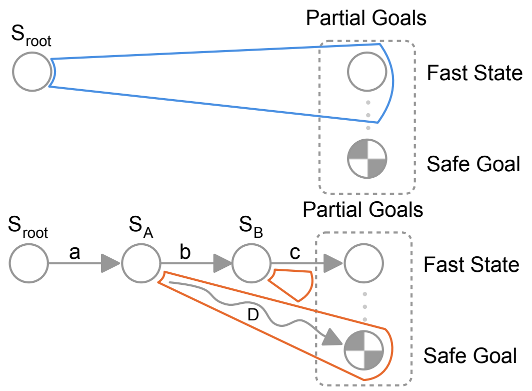


Figure 1: SafeTLP search graph example.

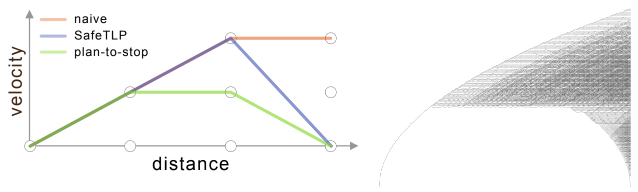
As we attempt to prove safety, we store every node encountered in a special safety closed list. If reencountered during a subsequent attempt, such nodes are not expanded as we have already attempted to prove safety from them.

Similar to SafeRTS's d_{safe} heuristic , SafeTLP prioritizes states that are closer to the goal state. The safety proof expands the search tree under the state $s_{current}$ (Cserna et al., 2017) with expansion ordered based on the distance to goal and speed (the lower, the better). If the safety open list becomes empty, the proof is unsuccessful and $s_{current}$ is labeled as unsafe. Upon the expansion of the goal node, the safety search terminates and returns the trajectory leading to the goal from $s_{current}$. $s_{current}$ is now proven safe and its naïve partial plan is augmented by the discovered safe trajectory to the goal to form a complete safe plan from the agent's current state to a goal state (line 7).

We use Fig. 1 to demonstrate the behavior of SafeTLP. SafeTLP first expands a search tree (blue on Fig. 1) from s_{root} that optimizes for velocity until it reaches a state in the partial goal set. In our example the sequence of a, b, and c trajectory segments represents a trajectory that leads to a partial goal state $s_{partial\_goal}$. Then SafeTLP attempts to find a safe trajectory to s_{goal} from each intermediate state identified by the $\langle a, b, c\rangle$ trajectory starting from the state closest to the partial goal. First, it attempts to prove that $s_{partial\_goal}$ is safe, then it falls back to S_b, S_a, and lastly to s_{root}. These proofs are independent search trees marked with orange on Fig. 1. The proof searches are optimized on d_{safe}. In our example the safety proof from $s_{partial\_goal}$ and S_b fails and only succeeds from S_a. SafeTLP constructs a safe trajectory to s_{goal} by appending the prefix $\langle a\rangle$ trajectory that leads to S_a with the trajectory segment sequence D.

Theorem 1 *SafeTLP is guaranteed to find a safe plan if one exists in the state space.*

Proof: If all safety proofs initiated from the states on the naïve path (line 5 of Alg. 1) fail, SafeTLP will perform a best-first search from the agent's current state (line 10). Because best-first search does not prune states from the search tree and the state space is finite, this search will eventually find paths from s_{root} to every other reachable state. Best-first search is exhaustive and complete in finite state spaces, thus eventually it will identify a path to a safe state if one exists. □



(a) (b)

Figure 2: (a): An cartoon sketch of algorithm behavior for spatiotemporal planning. (b): Distance-velocity projection of the plan-to-stop search tree. The horizontal axis represents space and the vertical axis velocity. Note that the temporal aspect of the states is not captured by the projection.

Theorem 2 *In the worst case, SafeTLP expands no more than twice as many states as the naïve plan-to-stop method expands in the worst case.*

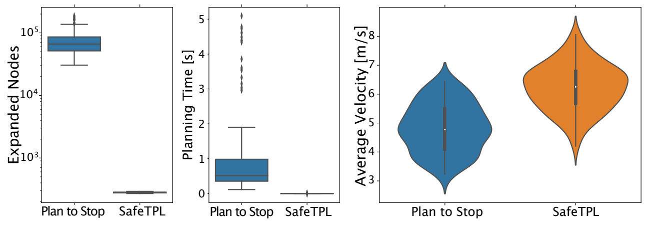
Proof: Recall that, due to the safety closed list, any node will be expanded at most once by a safety proof attempt. If all the safety proofs fail, then SafeTLP will perform best-first search initiated from the agent's current state. This exhaustive search will expand every state at most once, as the standard closed list of best-first search will prevent duplicate expansions. Thus any state will be expanded at most twice (once by a safety proof attempt and once by the exhaustive search). The naïve method expands each node at most once. □

In motivating our approach, we present three different approaches in Figure 2a. The abstract diagram shows the distance towards the goal along the horizontal axis and the velocity of the vehicle along the vertical axis. Ideally, we want an algorithm which allows the vehicle to travel as fast as possible towards the goal while guaranteeing that there is an alternate course of actions to travel to a safe state in the case of an emergency or unexpected situation.

The basic naïve approach, shown in orange, performs a best-first search towards the goal. By optimizing distance-to-goal, this search tends to accelerate the vehicle until it reaches the maximum velocity and then maintain that speed. This technique is clearly not safe as it does not guarantee a way for the vehicle to reach a safe state in case of an emergency. However, it does fulfill the task of accelerating the vehicle to a very high speed. On the other hand, the current state-of-the-art is a more exhaustive approach, shown in green and labeled plan-to-stop. This performs an exhaustive breadth-first search expanding all the possible trajectories for the vehicle to undertake with the goal of completely stopping the vehicle at the end as shown in Fig. 2b. This approach is able to find a safe way of reaching the goal at the price of many node expansions and a slow velocity for the vehicle. To be able to plan to slow down at the current planning horizon, it needs to plan to have a lower average velocity than the naïve approach would find. In essence, we have devised a hybrid approach that performs a best-first search to find a trajectory leading to a very high speed, and then performs a search on safety to find a series of safe actions leading us towards the goal safely represented by the blue line. In this fashion, we guarantee that even though we are

	1	2	3	4
comfortable acceleration	0.8	1.0	1.2	1.5
comfortable deceleration	0.8	1.0	1.2	1.5
aggressive deceleration	1.6	1.8	2.0	2.2

Table 1: Acceleration limit sets (m/s^2). Inferred from Powell and Palacín (2015), Martin and Litwhiler (2008), and Hoberock (1977)



(a) (b) (c)

Figure 3: (a): Distribution of the number of expanded nodes during search. (b):Planning time distribution. (c): Average vehicle velocity distribution.

accelerating the vehicle very quickly towards the goal, there will be a series of actions the vehicle can take to reach a safe state in the event of an emergency or unexpected action. This purpose-designed hybrid algorithm combines the low number of expansions of the best-first search's high-velocity actions with the guarantee of safety of the exhaustive search. Ultimately, this approach yields a high-quality solution that is both fast and guaranteed to be safe.

Empirical Evaluation

We demonstrate planning in simplified spatiotemporal lattice. The goal is 100 meters away from the start position of the vehicle. The road is discretized by half meters, so 200 states in total on the distance dimension of the lattice. From each state, the vehicle can apply three actions: accelerate, maintain the velocity, and decelerate. The following studies Powell and Palacín (2015), Martin and Litwhiler (2008), and Hoberock (1977) show that 0.8 to 1.5 m/s^2 are comfortable acceleration values for urban driving, and people can tolerate up to 2.2 m/s^2 without injury. Following this, we design four sets of comfortable acceleration and deceleration pairs, along with the aggressive deceleration that would be allowed to apply in our approach. Table 1 shows the acceleration and deceleration sets we used in our experiment. We fixed the maximum velocity limit at 15 m/s. We vary the starting velocity from 0 to 5 m/s with 0.2 increment.

Figure 3a shows the number of expanded node for the algorithms to find a safe trajectory. As we can see, SafeTLP expands many (about three magnitudes) fewer nodes than plan-to-stop. This is because plan-to-stop performs an exhaustive search of the state space while SafeTLP explicitly reasons about the optimal safe solution. Figure 3b shows the planning time. As expected, SafeTLP consumes much less time than plan-to-stop to find a safe plan. In a real application setting, fast planning enables the planner to run in a higher frequency, thereby enabling the vehicle to react to the environment more quickly and thus be safer.

Figure 3c present violin plots showing the distribution of the average velocity for each test instances. As we can see, SafeTLP produces higher performance trajectories than plan-to-stop.

Conclusion

We have introduced a new and more effective method for safe action selection in spatiotemporal planning. Previous methods are extremely conservative in how they guarantee safety by expanding drastically more nodes than is required. Our planner can quickly generate comfortable and safe plans online. We demonstrate that planning time is drastically reduced while simultaneously being able to achieve a higher average velocity for the vehicle. In combining techniques from real-time heuristic search and spatiotemporal planning, we introduced a method for selecting safe and fast plans quickly. We hope this work encourages further development in the applicability of safe real-time and online search.

References

Cserna, B.; Doyle, W.; Ramsdell, J.; and Ruml, W. 2017. Avoiding dead ends in real-time heuristic search. In *Proceedings of AAAI-18*, 1306–1313.

Hoberock, L. L. 1977. A survey of longitudinal acceleration comfort studies in ground transportation vehicles. *J. Dyn. Sys., Meas., and Control* 99(2):76–84.

Karaman, S., and Frazzoli, E. 2011. Sampling-based algorithms for optimal motion planning. *IJRR* 30(7):846–894.

Kuwata, Y.; Teo, J.; Fiore, G.; Karaman, S.; Frazzoli, E.; and How, J. P. 2009. Real-time motion planning with applications to autonomous urban driving. *IEEE Transactions on Control Systems Technology* 17(5):1105–1118.

Martin, D., and Litwhiler, D. 2008. An investigation of acceleration and jerk profiles of public transportation vehicles. In *ASEE, Conference Proceedings*.

McNaughton, M.; Urmson, C.; Dolan, J. M.; and Lee, J.-W. 2011. Motion planning for autonomous driving with a conformal spatiotemporal lattice. In *ICRA-2011*, 4889–4895.

Paden, B.; Čáp, M.; Yong, S. Z.; Yershov, D.; and Frazzoli, E. 2016. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles* 1(1):33–55.

Pivtoraiko, M.; Knepper, R. A.; and Kelly, A. 2009. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics* 26(3):308–333.

Powell, J., and Palacín, R. 2015. Passenger stability within moving railway vehicles: limits on maximum longitudinal acceleration. *Urban Rail Transit* 1(2):95–103.

Shalev-Shwartz, S.; Shammah, S.; and Shashua, A. 2017. On a formal model of safe and scalable self-driving cars. *arXiv preprint arXiv:1708.06374*.

Ziegler, J., and Stiller, C. 2009. Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios. In *IROS-2009*, 1879–1884.