Safety, Challenges, and Performance of Motion Planners in Dynamic Environments

Hao-Tien Lewis Chiang, Baisravan HomChaudhuri, Lee Smith, Lydia Tapia

Abstract Providing safety guarantees for autonomous vehicle navigation is an ultimate goal for motion planning in dynamic environments. However, due to factors such as robot and obstacle dynamics, e.g., speed and nonlinearity, obstacle motion uncertainties, and a large number of moving obstacles, identifying complete motion planning solutions with collision-free safety guarantees is practically unachievable. Since complete motion planning solutions are intractable, it is critical to explore the factors that impact planning success. One such factor is the planning environment, e.g., obstacle speed, obstacle motion uncertainty, and number of obstacles. In this paper, we explore the impact of the environmental parameters on the performance of a set of thirteen planning algorithms for navigating in dynamic environments. We aim to answer: 1) How do these algorithms perform relative to each other under increasingly more challenging environments? 2) What factors in an environment make planning in dynamic environments challenging? We classify and compare the algorithms in two planning environments with varying types and magnitudes of environmental challenges. Results show that state of the art planning algorithms were unable to consistently identify collision-free paths even in simple geometric planning problems with moving obstacles with stochastic dynamics. Results also demonstrate that given accurate obstacle predictions, planning algorithms that work in state-time space can typically generate real-time solutions in a limited planning horizon with higher success rates than other methods. In addition, in the presence of obstacle motion uncertainty, accepting paths with non-zero collision probability may lead to higher success rates.

Hao-Tien (Lewis) Chiang, Lydia Tapia
Department of Computer Science, University of New Mexico, Albuquerque, NM 87131, USA, e-mail: lewispro@unm.edu, tapia@cs.unm.edu

Baisravan HomChaudhuri, Lee Smith
Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM 87131, USA, e-mail: bhomchaudhuri@unm.edu, lsmith22@unm.edu

1 Introduction

Motion planning algorithms for identifying collision-free paths in presence of moving obstacles are critical for robotic applications including self-driving cars [29], UAVs [16], service robots [27], mobile manipulators [32] and autonomous wheelchairs [28]. However, motion planning in dynamic environments is challenging. Even in the simplest case, where a 2D holonomic robot must avoid collision with polygonal obstacles moving at constant velocities, planning has been shown to be NP-Hard [5] and in PSPACE [4]. Thus, identifying complete solutions, in real-time and in the presence of robot dynamic constraints, obstacle motion uncertainty, and a large number of obstacles, is practically unachievable. Therefore, many real-time planning algorithms sacrifice completeness for practicality [23]. These algorithms vary drastically in methodology, obstacle information usage, and computational requirements. This diversity in algorithms has motivated prior surveys, which discussed the variety in methodology [23] and their desirable properties [17]. In addition, in depth discussion on the complexity of planning in dynamic environments can be found in [14]. A review focusing on sampling-based methods for dynamic environments is available in [25]. We extend all prior review work by contributing in this work a detailed comparison of a set of algorithms implemented in navigation that requires dynamic obstacle avoidance.

Since complete motion planning solutions are intractable, it is important to identify the critical factors that impact planning success of state of the art algorithms. More specifically, two important questions are not answered by previous surveys: 1) *How do these algorithms perform relative to each other under increasingly more challenging environments?* 2) *What factors in a planning environment make planning in dynamic environments challenging?*

In this paper, an effort has been made to answer these questions by classifying planning algorithms in terms of methodology and requirements. We have identified three environmental challenges fundamental to planning in dynamic environments: 1) speed of the robot and obstacles, 2) obstacle motion uncertainty, and 3) number of obstacles. In order to investigate the impact of these challenges, we implement thirteen different planning algorithms in two tunable environments with various environmental challenges. Algorithm performance metrics such as success rate, finish time, and computation time per time step are collected, compared, and analyzed.

2 Motion Planning Algorithms for Dynamic Environments

For many robotic applications, a planning algorithm for dynamic environments must generate planning solutions in real-time, avoid moving obstacles, and assume replanning is critical to success since obstacles are observed as the robot moves. In this paper, we selected thirteen popular planning algorithms for dynamic environments that were shown to satisfy the above conditions in some cases (Table 1). These methods are classified on a variety of features detailed below.

Name	Methodology	State vs State-time	Planning Horizon	Obstacle Prediction	Precomputation
Khatib APF [18]	APF	State	Reactive	None	None
Gaussian APF [19]	APF	State	Reactive	None	None
Ge et. al (Ge APF) [15]	APF	State	Reactive	None	None
APFSR [7][21]	APF	State	Reactive	Dynamics required†	Backward SR‡
A* [2]	Discretized state graph	State	Global	None	None
Velocity Obstacle (VO) [11]	Geometric	State	Reactive	Velocity extrapolation	None
Dynamic RRT (D-RRT)[10]	Sampling-based	State	Global	None	None
State-Time A* (ST-A*) [12]	Discretized state graph	State-time	Global	Velocity extrapolation	None
Dynamic Window (D. Window) [24]	Sampling-based	State-time	Partial	Velocity extrapolation	None
State-Time RRT (ST-RRT) [23]	Sampling-based	State-time	Global	Velocity extrapolation	None
Partial Motion Planning (PMP) [3]	Sampling-based	State-time	Partial	Velocity extrapolation	None
Stochastic Ensemble Simulation (SES) [9]	Sampling-based	State-time	Partial	Dynamics required†	Monte Carlo prediction
Dynamic Risk Tolerance (DRT) [6]	Sampling-based	State-time	Partial	Dynamics required†	Forward SR‡

Table 1: Features of thirteen selected motion planning algorithms for dynamic environments. †These methods require knowledge of obstacle stochastic dynamics for prediction. ‡SR stands for stochastic reachability analysis [1].

Methodology classifies the planners based on the underlying algorithm. First, *APF* (Artificial Potential Field) was proposed in [18]. It works by constructing an attractive potential from the goal and repulsive potentials around obstacles. The robot control is then obtained from the derivative of the linear combination of attractive and repulsive potentials. Next, *Geometric* methods, such as Velocity Obstacle (VO) [11], compute control actions in the robot's velocity space using the geometry and velocity of the robot and its nearby obstacles. The obstacles are assumed to move at a constant velocity. *Discretized state graph* methods discretize the state and action space in order to form a roadmap. A path is then identified using graph search techniques such as A* [2]. Finally, the selected *Sampling-based* methods either randomly sample the state space or the state-time space and grow a tree rooted at the robot's current state. The tree consists of robot states that are generated by incrementally applying control actions to nodes/states in the tree to obtain new collision free states. A sequence of control inputs, a path, can be extracted from this tree which the robot executes.

State vs State-Time describes if planning is done in the robot's state space (often workspace or configuration space) or if the state space is extended with time.

Authors in [12] proposed to augment the robot's state space with time in order to explicitly identify collision free paths in dynamic environments. However, planning in state-time space generally requires a form of obstacle motion prediction.

Planning Horizon can be classified as one of the three types. First, *Global* methods consider all obstacles in the environment while planning a path connecting the robot's current state to the goal. *Partial* motion planning was first proposed in [3]. These methods only consider obstacles within a finite range and plan a finite horizon path that may not reach the goal. This reduces the computation cost and is suitable for environments that are only partially observable by the robot. However, it is difficult to guarantee safety and optimality since the paths are built with partial information. Finally, *Reactive* methods also only consider obstacles within a finite range but compute an action at every time step instead of planning over a finite horizon. For reasons similar to partial motion planning, it is also difficult to guarantee safety and optimality of the planned path.

Obstacle Prediction, specifically prediction of obstacle motion, enables algorithms to select better informed paths. However, it requires more information about the obstacle that maybe difficult to obtain, such as velocity, acceleration, and intention.

Precomputation considers any task that needs to be run before planning occurs. Sometimes, time consuming tasks such as simulating an obstacle's future position distribution [9] or computing an optimal single obstacle avoidance strategy [7] can be precomputed offline. The results from these computations can be queried at run time to reduce online computation. However, this can restrict the allowed obstacle dynamics to be non-interacting.

3 Challenges of Planning in Dynamic Environments

We analyze three fundamental challenges faced by most planning algorithms for dynamic environments and investigate their impact on planning success. While these challenges have been previously studied [17, 33, 21, 6, 9], we present the first detailed comparison across thirteen planning algorithms. It should be noted that in realistic environments there are many additional challenges, and we intend this systematic study of these fundamental issues to serve as a foundation to the evaluation of additional planning challenges.

Obstacle and robot speed. Increasing the obstacle speed or lowering the max robot speed increases the difficulty of planning in dynamic environments [17, 33]. It was shown in [34] that safety guarantees can only be given if the robot's maximum speed is higher than that of the obstacle's speed. In many applications, however, the robot's maximum speed is lower than that of the obstacles, e.g., the maximum speed of assistive robots may be limited due to safety concerns of surrounding pedestrians. In such scenarios the size of Inevitable Collision States (ICS) [13], states that will result in collision regardless of control action, is larger than the geometric size of the obstacle. The size of ICS increases in relation to the max speed ratio, i.e., the ratio

of obstacle speed to max robot speed. We investigate scenarios where the robot's maximum speed is above or below the speed of the obstacles in Section 4.4.

Obstacle Motion Uncertainty. Another challenge experienced by autonomous vehicles is the fact that obstacle motion may not be predicted exactly. This can be caused by the robot's noisy sensors, also known as uncertainty in environment sensing [20], or by the stochastic nature of the obstacles, e.g., pedestrians or human drivers, also known as uncertainty in environment predictability [20]. Regardless of the source, obstacle motion uncertainty causes the possible future obstacle locations to increase over time, and therefore it reduces the solution space of the robot. As a result, there may not exist a robot state-time coordinate with zero collision probability in crowded environments, and hence no safety guarantee can be given when planning over a finite time horizon. This is known as the freezing robot problem [26] and is analyzed in [6]. We empirically evaluate how obstacle motion uncertainty, in the form of stochastic obstacle speed, impacts planning algorithms in Section 4.5.

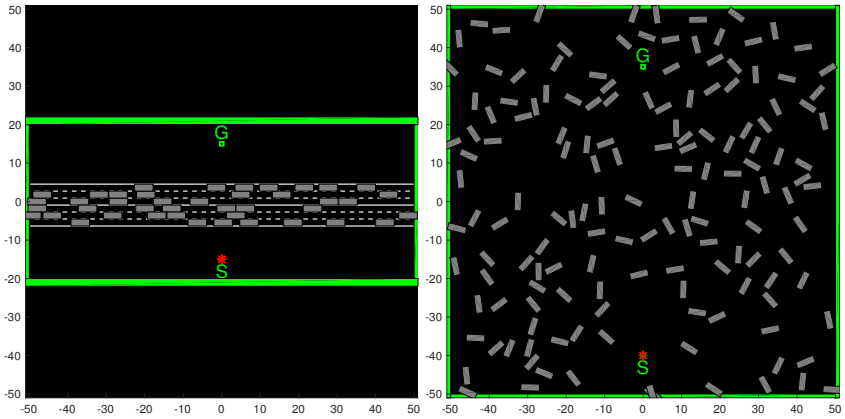
Number of Obstacles. Robots often operate in environments with multiple moving obstacles. High obstacle density reduces the size of the collision-free planning solution space. As a result, planning algorithms often perform poorly in crowded environments, i.e., sampling-based methods often struggle to find solutions in narrow corridors formed by moving obstacles [6, 9], and APF-based methods are often trapped in local minima formed by the repulsive potential of multiple obstacles [8]. On the other hand, even if the obstacle density is kept constant, increasing the number of obstacles while increasing the size of the environment at the same time forces planning algorithms to avoid more obstacles and often increases computation time. Section 4.6 investigates the impact of increasing number of obstacles with both constant obstacle density and increasing density.

4 Evaluation of Planning Algorithms

In this section, we evaluated the performance of the thirteen algorithms in two highly challenging environments. We also introduce a variant of the ST-A* planning algorithm in order to investigate the impact of obstacle motion uncertainty.

4.1 Planning Environments

In the Lanes environment (Figure 1(a)), obstacles move in horizontal lanes resembling real-world traffic. Obstacles in the lower lanes move from left to right while the upper lanes move from right to left. The height and width of the environment is 40m and 100m, respectively. For a run to be successful, the holonomic point robot must travel from the start (0m,-15m) to the goal (0m,15m) within 100s without colliding with obstacles. The obstacles are rectangles that resemble mid-sized sedans with 1.81m width and 4.23m length. In order to maintain a constant obstacle den-



(a) (b)

Fig. 1: Lanes (a) and Random (b) dynamic environments. The point robot must navigate from start (S) to goal (G) without colliding with moving obstacles (gray rectangles).

sity, an obstacle is immediately transported to the opposite side of the environment in the same lane if the center of the obstacle reaches the edge of the environment. The number of lanes, number of obstacles per lane, and obstacle speed are tunable parameters. These parameters can be varied in order to investigate the challenges of planning in dynamic environments.

In the Random environment (Figure 1(b)), obstacles are initialized with random positions and headings in a 100m by 100m environment. The robot and obstacles are identical to Lanes. To maintain constant obstacle density the obstacle is transported to the antipodal position with unchanged speed and heading if the center of an obstacle reaches the boundary. This randomized environment forces the robot to avoid collision from multiple directions, and moving obstacles may form temporary complex structures that requires the robot to execute long-term evasion plans.

All methods were implemented in C++. The world simulation time step is 0.01s. The VO algorithm was adapted from the RVO2 C++ code base [31] implementation of the Optimal Reciprocal Collision Avoidance (ORCA) algorithm. This algorithm [30] was modified to allow for single-agent collision avoidance, removing the reciprocal aspect of ORCA while maintaining many of ORCA's linear programming optimizations. All experiments are repeated 100 times on a single core of an Intel I7-6820HQ at 2.7GHz with 16GB of RAM.

4.2 Analysis Tools

In order to compare the performance of planning algorithms, we collect the following metrics for each algorithm: success rate (the ratio of successful navigation to the total number of runs), finish time (the amount of time the robot takes to navigate

from start to goal), and computation time per planning step. In order to establish a baseline for comparison, the performance metric of a Non-Reactive (NR) method is also given where the robot traverses the shortest path from start to goal without avoiding collision with obstacles.

In order to gain performance insights, we developed a tool to identify the set of ICS ($\mathscr{X}_{ICS}$) and its complement set $\mathscr{X}_{ICS}^{C}$ for any given planning scenario. These sets are identified by discretizing the state and action space of the robot to a fixed resolution. For our holonomic robot, the state space is discretized into grids in 2D space with a resolution set to the maximum distance the robot can move within Δ=0.2 seconds. The robot action space is discretized into five actions: remaining stationary and moving up, down, left, and right at maximum speed.

For all robot states in a given scenario, i.e., current position, velocity and orientation of all obstacles, we can identify if a state is in $\mathscr{X}_{ICS}$ by enumerating through all action sequence within a horizon. A state is in $\mathscr{X}_{ICS}$ if no action sequence can avoid collision within the horizon, otherwise, the state is in $\mathscr{X}_{ICS}^{C}$. (A horizon of 6 seconds was used.) As a metric to reflect the difficulty of a planning problem, we also compute the $\mathscr{X}_{ICS}^{C}$ ratio as defined by the number of states in $\mathscr{X}_{ICS}^{C}$ divided by the total number of discretized robot states in the environment.

4.3 Algorithm Performance Comparison

We first try to find out how the algorithms perform relative to each other under a baseline condition. In Lanes, there are 6 lanes (3 right-bound and 3 left-bound) with 4 obstacles per lane. The obstacles have a constant speed of 4.47m/s (10 mph) while the holonomic point robot's maximum speed is 2.68m/s (6 mph). There are 75 obstacles in Random. The robot and obstacle dynamics are identical to Lanes.

The blue bars in Figure 2(a) and 2(c) show that the success rate of a planning method depends largely on methodology and whether the algorithm utilizes obstacle prediction. In general, methods that utilize obstacle predictions outperform the ones that utilize only information of the current obstacle position, which is intuitive. Among the methods that utilize obstacle predictions, state-time, sampling-based, and discretized state-time graph methods have the best success rates since the future position of obstacles can be exactly predicted over a large time horizon, thus enabling algorithms to identify whether a state-time coordinate is in collision. Note that D. Window finds paths with the longest finish time among all methods since it does not explicitly optimize finish time.

Algorithms with the next highest success rates, VO and APFSR, also utilize obstacle predictions. However, these methods either approximate the obstacle motion, as in through a velocity obstacle computation, or approximate the $\mathscr{X}_{ICS}$ associated with multiple moving obstacles, as in APFSR. As a result, these methods have lower success rates than discretized state-time graph and state-time sampling methods. Note that VO has a low success rate in Lanes which is likely due to the circular approximation of obstacles that increases the area occupied by obstacles. For example,

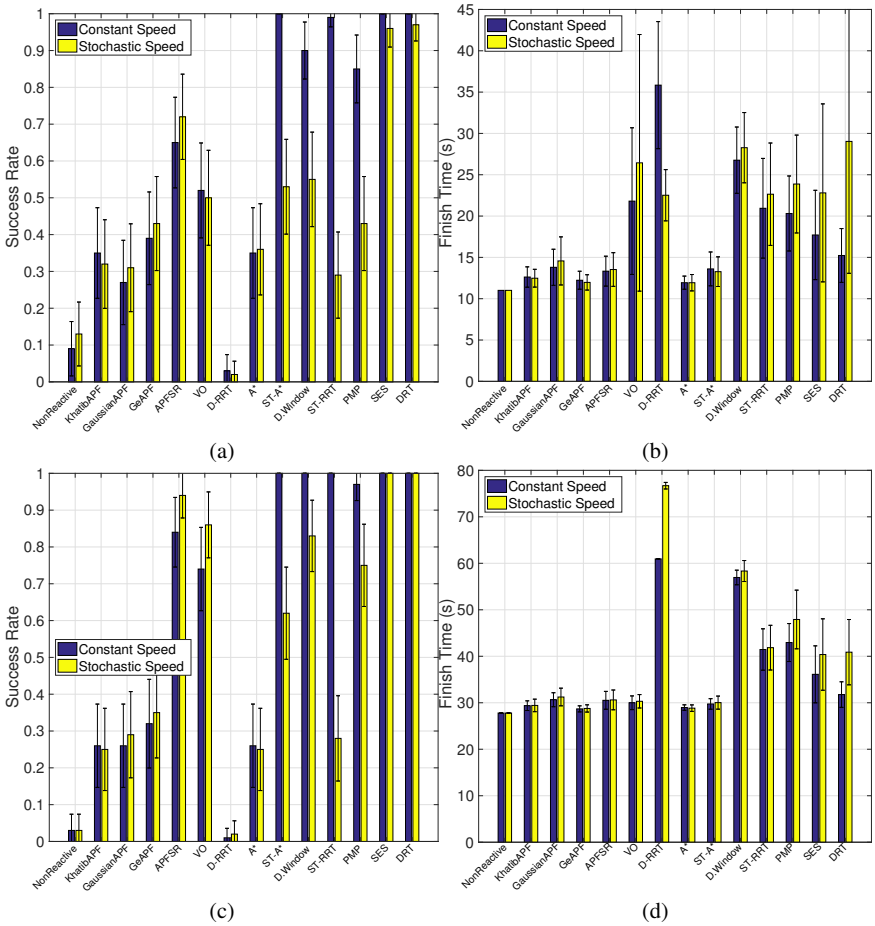


Fig. 2: Success rates and finish time of planning algorithms in both Lanes (a,b) and Random (c,d). Obstacles are moving at a constant speed (blue bars) or a speed stochastically sampled every 0.05s (yellow bars).

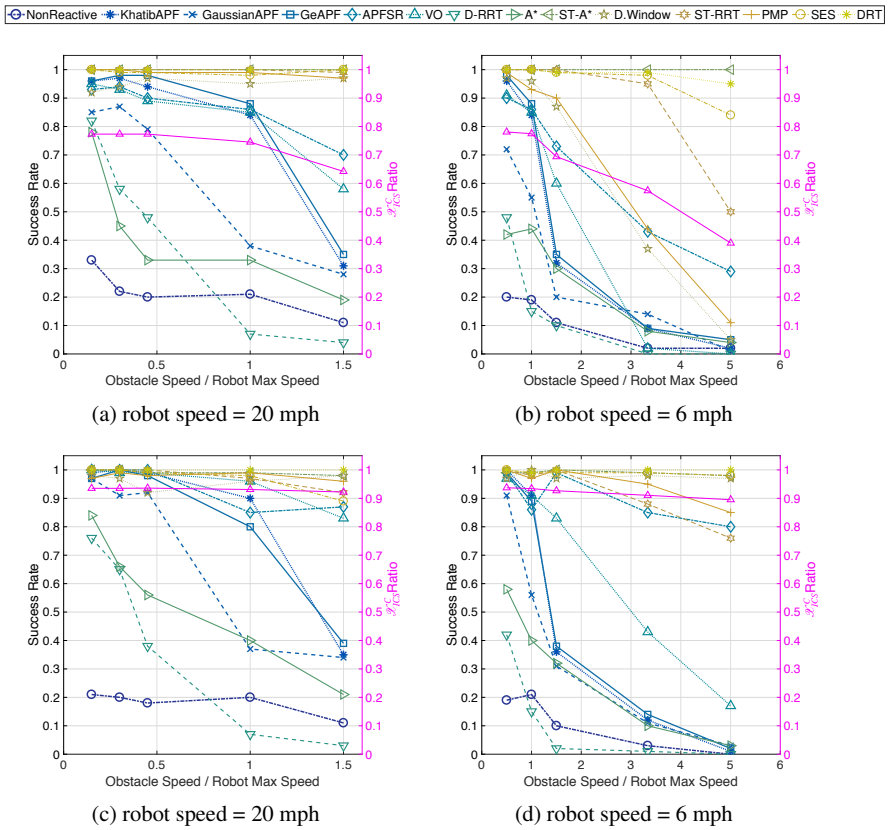
VO approximates obstacles as circles which have the diameter of 4.61m. This spans more than two lanes in Lanes.

Methods that do not utilize obstacle predictions are limited in success rate, even for algorithms such as A*, as these methods do not have a reliable way to identify and avoid $\mathscr{X}_{ICS}$. Among these methods, D-RRT has a particularly low success rate and a long finish time. This is due to constant pruning and regrowth of a large portion of the tree in these highly dynamic environments. In addition, the sampling nature of RRT returns different paths every time the tree is pruned and re-grown, which greatly increases finish time and reduces success rate. Khatib, Gaussian, and Ge APF methods have low success rates since the repulsive potentials are constructed

heuristically around obstacles. These potentials do not approximate $\mathscr{X}_{ICS}$ well and may guide the robot into $\mathscr{X}_{ICS}$.

4.4 Impact of Robot And Obstacle Speed

After analyzing the general performance of the planning algorithms, we investigate how planning algorithms perform under various robot and obstacle speeds. The setup of Lanes and Random are identical to Section 4.3. Two cases are considered where the max speed of the robot is 6 mph and 20 mph, respectively, while the speed of all obstacles belongs to the set $\{3,6,9,20,30\}$ mph ($\{1.34,2.68,4.02,8.94,13.68\}$ m/s).



(a) robot speed = 20 mph

(b) robot speed = 6 mph

(c) robot speed = 20 mph

(d) robot speed = 6 mph

Fig. 3: Success rates of planning algorithms in both Lanes (a), (b) and Random (c), (d) environments as a function of max speed ratio (Obstacle speed/robot max speed). The magenta solid line is the $\mathscr{X}_{ICS}^{C}$ ratio.

Figure 3 shows that the success rate of most algorithms decreases with the increase in obstacle speed, but success rate increases with an increase in the robot's max speed. Additionally, comparing the high robot max speed (Figures 3(a) and (c)) with low robot max speed (Figures 3(b) and (d)), we can see that the success rate of a method with a given max speed ratio is roughly the same regardless of max robot speed. This result shows that the max speed ratio is a good indicator of environmental difficulty for most methods except for those that discretize or sample the state space (A* and D-RRT). This indicator applies well for methods that do not predict obstacle motion (Khatib, Gaussian and Ge APF), as the success rates drop drastically (50-60%) if the max speed ratio exceeds one. This is intuitive as these methods can usually avoid an obstacle if the robot can move faster than the obstacle.

To gain some insight why the max speed ratio is a good indicator, we plotted the $\mathscr{X}_{ICS}^{C}$ ratio (magenta solid line) at the beginning of the planning scenario in Figure 3. It is clear that the $\mathscr{X}_{ICS}^{C}$ ratio decreases as max speed ratio increases. Notice that when the obstacles are five times faster than the robot in Lanes, more than 60% of the states are in $\mathscr{X}_{ICS}$.

APFSR and PMP perform well in high $\mathscr{X}_{ICS}^{C}$ ratio environments. However, their success rate reduces as $\mathscr{X}_{ICS}^{C}$ ratio decreases since they work by computing reachable sets for every obstacle robot pair and then joining all individually computed sets. These conjoined reachable sets approximate $\mathscr{X}_{ICS}^{C}$, and the quality of the approximation reduces when the max speed ratio is high.

Since a large portion of the robot states are in $\mathscr{X}_{ICS}$ when the max speed ratio is high, it may be difficult for ST-RRT to identify a global collision-free path due to its use of sampling. Therefore, it often finds partial paths that eventually end up in $\mathscr{X}_{ICS}$. Similarly, D. Window performs poorly since it typically has a short fixed planning horizon (2s), and thus cannot identify inevitable collisions beyond the planning horizon. On the other hand, state of the art state-time sampling-based methods SES and DRT have success rates higher than 85% even when a large portion of states are in $\mathscr{X}_{ICS}$. This is likely due to the τ-safety criterion, the algorithm chooses a path if it is at least τ seconds long, and tree replanning, the algorithm replans a tree when the remaining path is less than τ seconds long, features of these methods.

4.5 Impact of Obstacle Motion Uncertainty

In this section, we investigate the impact of stochastic obstacle speed in planning algorithms. The setup of the two environments are identical to Section 4.3. However, instead of a constant speed of 4.47m/s, the obstacle samples speed every 0.05s uniformly from the set of possible speeds $\{2.25, 3.375, 4.47, 5.625, 6.75\}$ m/s ($\{5, 7.5, 10, 12.5, 15\}$ mph), thus has an average speed of 4.47m/s. Note that this kind of obstacle speed uncertainty is common in robotics applications as velocity estimation from sensor data is typically inaccurate [22].

The success rates and finish time of planning algorithms that do not predict obstacle motion are not impacted by the obstacle motion uncertainty (Figure 2, yellow

(a) $V_o = 6$ mph (b) Stochastic

Fig. 4: (a) Inevitable Collision States (ICS) for a single moving obstacle (moving towards right) and a holonomic point robot with max speed of 6 mph. (b) Probability of collision from a backward SR set calculation (also used as the repulsive potential of APFSR). The obstacle changes speed stochastically as described in Section 4.5.

bars). This is expected as the area of ICS posed by a single obstacle traveling at the average speed (4.47m/s) (Figure 4(a)) is identical to the backward SR set (Figure 4(b)). However, none of these methods have a success rate higher than 45% in both environments.

Motion uncertainty reduces the success rate and increases the finish time of all methods that predict an obstacle's future position by extrapolating the current velocity (ST-A*, D. Window, ST-RRT, PMP). This is also expected since these methods use an approximate obstacle prediction method to plan a partial or global path which can lead to $\mathscr{X}_{ICS}$. The only exception to this reduction in success rate and increase in finish time for methods with prediction is VO that observes continuously and plans reactively. Thus, it is capable of averaging out the imprecise obstacle predictions.

Methods that are designed to handle obstacle motion uncertainty (APFSR, SES and DRT) maintain a high success rate despite the presence of uncertainty. However, the finish time mean and variance increase significantly for SES and DRT. These methods predict the probability of obstacle occupancy in the future (via Monte Carlo or forward stochastic reachability analysis) and identify paths with low probability of collision by sampling the state-time space. This allows the planner to make much more informed decisions in presence uncertainty. However, since the possible area occupied by obstacles increases in the presence of uncertainty, the robot takes a longer path in order to avoid potential future collisions.

Note that ST-A* is no longer a complete algorithm in this environment since velocity extrapolation cannot exactly predict the future position of stochastically moving obstacles. Therefore, we also tested the performance of a variant of ST-A*, ST-A*$_{FSR}$, where the velocity extrapolation is replaced by the exact obstacle occupancy distribution prediction obtained from forward reachability analysis [6]. Since the obstacle occupancy is probabilistic, the collision probability for a given state-time coordinate is also probabilistic, and this variant treats any node with collision probability below some threshold $P_{threshold}$ as collision-free.

Table 2 shows that the success rate of $P_{threshold} = 0$ is much lower than other values in both environments. This is due to the freezing robot problem. Since the possible future position of obstacles increases over time, based on the prediction, there may not exist a robot state-time coordinate with zero collision probability in the future. The paths identified by setting $P_{threshold} = 0$ are guaranteed to be collision-free, but many feasible paths cannot be identified. However, $P_{threshold} > 0$ produces

a higher success rate. But, collisions occur since the collision probability of feasible paths can be non-zero. Note that $P_{threshold}$ is the threshold of collision probability of nodes, not the entire trajectory.

$P_{threshold}$		0%	1%	5%	10%
Lanes	Success Rate	49 ± 13%	96 ± 4%	99 ± 1%	98 ± 2%
	Finish Time	30.3 ± 12.6m	21.6 ± 7.3m	18.5 ± 4.9m	17.0 ± 4.3m
Random	Success Rate	62 ± 12%	95 ± 5%	94 ± 6%	96 ± 4%
	Finish Time	45.3 ± 7.3m	35.6 ± 4.3m	33.4 ± 2.9m	32.9 ± 2.9m

Table 2: Performance of ST-A*$_{FSR}$ with various collision probability thresholds $P_{threshold}$

No tested method achieves a 100% success rate in presence of obstacle motion uncertainty. This includes state of the art methods (DRT and SES) designed to work in such environments and ST-A*$_{FSR}$ that employs exact obstacle prediction. Therefore, obstacle motion uncertainty remains a difficult challenge and an open problem for motion planning.

4.6 Impact of Number of Moving Obstacles

This section investigates the impact of increasing number of obstacles, with and without increasing obstacle density. Since the number of obstacles directly impacts computation load, we also compare the computation time per planning step of planning algorithms. To vary the obstacle density, we vary the number of obstacles from 50 to 150 in Random and the number of obstacles per lane from 2 to 7 in Lanes. A snapshot of 7 obstacles per lane and 150 obstacles in Lanes and Random are shown in Figures 1(a) and 1(b), respectively. We also investigated the effect of varying the number of obstacles, without increasing obstacle density, in Lanes. This is done by varying the number of lanes from 2 to 10 without increasing the number of obstacles per lane. All other settings are identical to Section 4.3.

Figures 5(a) and 5(c) show that increasing obstacle density decreases success rate for all methods except for ST-A* and DRT. This is expected as obstacle density greatly decreases the $\mathscr{X}_{ICS}^{C}$ ratio. For example, in the environment in which 34% of the area is occupied by obstacles, roughly 20% of the states are in $\mathscr{X}_{ICS}^{C}$. As discussed in Section 4.4, this poses a significant challenge for planning algorithms.

Increasing the number of obstacles while keeping the density constant decreases the success rate of all methods except for ST-A* and DRT, as shown in Figure 6(a). However, this is not due to decrease of $\mathscr{X}_{ICS}^{C}$ ratio, since the $\mathscr{X}_{ICS}^{C}$ ratio (magenta solid line in Figure 6(a)) remains roughly the same. This is likely due to the robot needing to traverse through more regions of possible collision (more lanes), thus increasing the probability of collision along the trajectory. The decrease in success rates also indicates that the difficulty of planning in dynamic environments is very hard to quantify. We utilized $\mathscr{X}_{ICS}^{C}$ ratio to explain the performance of many

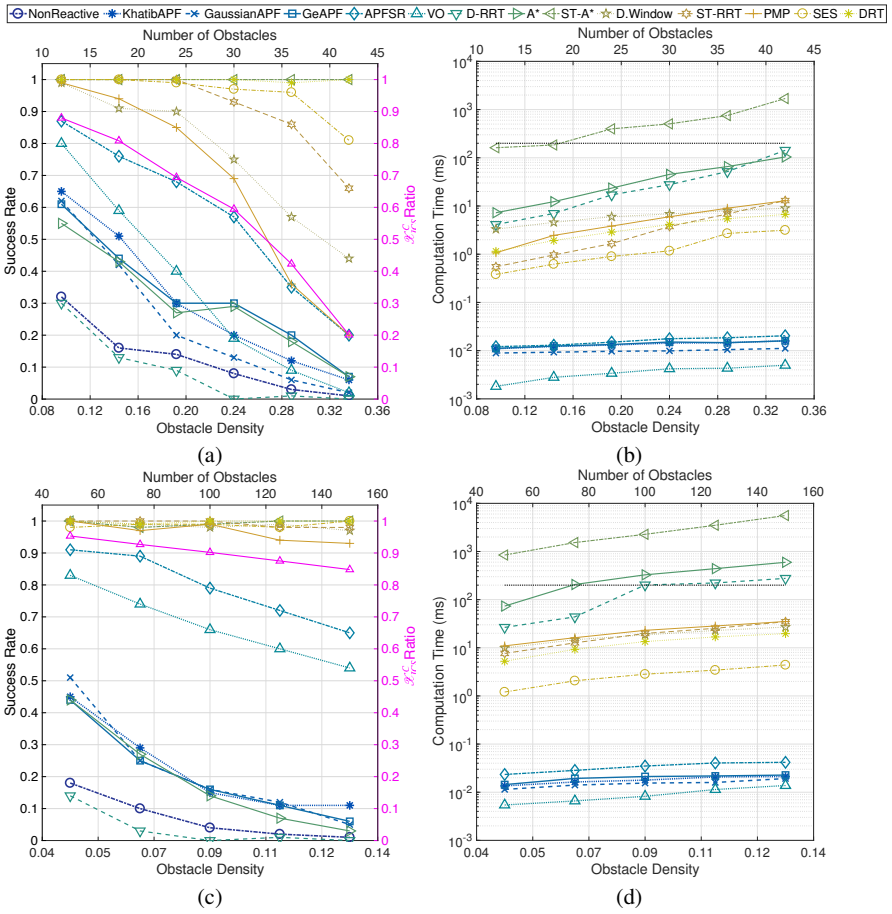
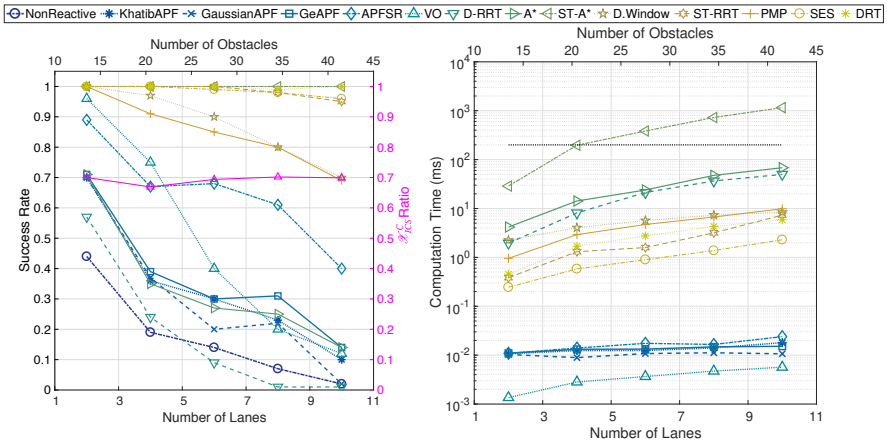


Fig. 5: Success rate and computation time per planning step with varying obstacle density in Lanes (a,b) and Random (c,d). The black dotted line is the real-time limit (200ms).

planning algorithms. However, $\mathscr{X}_{ICS}^{C}$ ratio does not fully quantify the difficulty of planning in dynamic environments as it failed to explain the success rate decrease as the number of lanes increase.

Comparing Figures 5(a) and (c), we can see the density of Random is lower than that of the Lanes. However, even though the obstacles have the same (relative) speed, it can be seen that success rate of most methods in the highest density in Random (13% occupied) is lower than the a similar density in Lanes (13% occupied). This indicates that the structure of the environment may impact the success rate of planning algorithms. For example, the robot can safely stay outside the lanes to wait until an opening occurs. Such safe zones do not exist in Random.

Computation time per planning step is typically sensitive to the number of obstacles. Figures 5 (b), (d) and Figure 6(b) show the computation time per online



(a) (b)

Fig. 6: Success rate and computation time per planning step as a function of number of lanes in Lanes. Note that the obstacle density remains constant but the number of obstacles increases with number of lanes. The black dotted line is the real-time limit (200ms).

planning step has three groupings. First, reactive methods (APF methods, VO, and APFSR) are extremely fast, in the order of 10μs in the 150 obstacle environment. Since these methods return an action every time step (10ms), they are real-time capable. Next, the computation time of state-time sampling-based methods (D. Window, ST-RRT, PMP, SES and DRT) ranges from 0.1ms to 35.2ms, while returning actions every 200ms. Since 200ms is generally considered as the requirement for real-time planning algorithms [23], these methods are also real-time capable. Lastly, Dynamic RRT, A* and ST-A* are no longer real-time capable in these environments with more than 100 obstacles. In particular, ST-A* is not real-time capable except for the environment with less than 12 obstacles. This is due to ST-A* explicitly discretizing state and time, resulting in a graph with $O(1/\Delta^3)$ number of nodes (in contrast, A* has $O(1/\Delta^2)$ number of nodes), where Δ is the temporal discretization size.

5 Conclusion

In this paper, we classify, implement and compare thirteen planning algorithms for dynamic environments in two tunable environments. Using success rates, finish times, and computation times per planning step, we identified and analyzed the impact of environmental challenges on these algorithms. These challenges include max speed ratio (obstacle speed/robot max speed), obstacle motion uncertainty, and number of obstacles. We show that planning in dynamic environments remains unsolved as state of the art planning algorithms failed to consistently identify collision-free paths even in simple geometric planning problems. Our detailed analysis demon-

strates an initial attempt at obtaining insights about the specific challenges of motion planning algorithms, and we hope may motivate new solutions to this challenging problem.

6 Acknowledgments

This material is based upon work supported by the National Science Foundation (NSF) under Grant Numbers IIS-1528047 and IIS-1553266 (Tapia, CAREER). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

1. A. Abate, M. Prandini, J. Lygeros, and S. Sastry. Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. *Automatica*, 44(11):2724–2734, 2008.
2. A. Bacha, C. Bauman, R. Faruque, M. Fleming, C. Terwelp, C. Reinholtz, D. Hong, A. Wicks, T. Alberi, D. Anderson, et al. Odin: Team victortango's entry in the DARPA urban challenge. *Journal of Field Robotics*, 25(8):467–492, 2008.
3. R. Benenson, S. Petti, T. Fraichard, and M. Parent. Integrating perception and planning for autonomous navigation of urban vehicles. In *Proc. IEEE Int. Conf. on Intel. Robot. Sys. (IROS)*, pages 98–104, 2006.
4. J. Canny. Some algebraic and geometric computations in pspace. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 460–467, 1988.
5. J. Canny and J. Reif. New lower bound techniques for robot motion planning problems. In *Foundations of Computer Science, 28th Annual Symposium on*, pages 49–60, 1987.
6. H.-T. Chiang, B. HomChaudhuri, A. P. Vinod, M. Oishi, and L. Tapia. Dynamic risk tolerance: Motion planning by balancing short-term and long-term stochastic dynamic predictions. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 3762–3769, 2017.
7. H.-T. Chiang, N. Malone, K. Lesser, M. Oishi, and L. Tapia. Aggressive moving obstacle avoidance using a stochastic reachable set based potential field. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 73–89, 2014.
8. H.-T. Chiang, N. Malone, K. Lesser, M. Oishi, and L. Tapia. Path-guided artificial potential fields with stochastic reachable sets for motion planning in highly dynamic environments. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2347–2354, 2015.
9. H.-T. Chiang, N. Rackley, and L. Tapia. Stochastic ensemble simulation motion planning in stochastic dynamic environments. In *Proc. IEEE Int. Conf. on Intel. Robot. Sys. (IROS)*, pages 3836–3843, 2015.
10. D. Ferguson, N. Kalra, and A. Stentz. Replanning with RRTs. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1243–1248, 2006.
11. P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *Int. J. Robot. Res.*, 17(7):760–772, 1998.
12. T. Fraichard. Trajectory planning in a dynamic workspace: a state-time space approach. *Advanced Robotics*, 13(1):75–94, 1998.
13. T. Fraichard and H. Asama. Inevitable collision states–a step towards safer robots? *Advanced Robotics*, 18(10):1001–1024, 2004.
14. K. Fujimura. *Motion planning in dynamic environments*. Springer, 1991.
15. S. S. Ge and Y. J. Cui. Dynamic motion planning for mobile robots using potential field method. *Auto. Robot.*, 13(3):207–222, 2002.

16. C. Goerzen, Z. Kong, and B. Mettler. A survey of motion planning algorithms from the perspective of autonomous UAV guidance. *Journal of Intelligent & Robotic Systems*, 57(1):65–100, 2010.
17. F. Kamil, S. Tang, W. Khaksar, N. Zulkifli, and S. Ahmad. A review on motion planning and obstacle avoidance approaches in dynamic environments. *Advances in Robotics & Automation*, 4(2):134–142, 2015.
18. O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 500–505, 1985.
19. C.-P. Lam, C.-T. Chou, K.-H. Chiang, and L.-C. Fu. Human-centered robot navigation towards a harmoniously human–robot coexisting environment. *IEEE Trans. Robot.*, 27(1):99–112, 2011.
20. S. M. LaValle and R. Sharma. A framework for motion planning in stochastic environments: modeling and analysis. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 3057–3062, 1995.
21. N. Malone, H.-T. Chiang, K. Lesser, M. Oishi, and L. Tapia. Hybrid dynamic moving obstacle avoidance using a stochastic reachable set-based potential field. *IEEE Trans. Robot.*, pages 1124–1138, 2017.
22. J. Miura and Y. Shirai. Modeling motion uncertainty of moving obstacles for robot motion planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2258–2263, 2000.
23. B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Trans. on Intell. Veh.*, 1(1):33–55, 2016.
24. M. Seder and I. Petrovic. Dynamic window based approach to mobile robot motion control in the presence of moving obstacles. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1986–1991, 2007.
25. A. Short, Z. Pan, N. Larkin, and S. van Duin. Recent progress on sampling based dynamic motion planning algorithms. In *Proc. IEEE/ASME Int. Conf. Adv. Mech. (AIM)*, pages 1305–1311, 2016.
26. P. Trautman and A. Krause. Unfreezing the robot: Navigation in dense, interacting crowds. In *Proc. IEEE Int. Conf. on Intel. Robot. Sys. (IROS)*, pages 797–803, 2010.
27. R. Triebel, K. Arras, R. Alami, L. Beyer, S. Breuers, R. Chatila, M. Chetouani, D. Cremers, V. Evers, M. Fiore, et al. Spencer: A socially aware service robot for passenger guidance and help in busy airports. In *Field and Service Robotics*, pages 607–622, 2016.
28. S. Tzafestas. Research on autonomous robotic wheelchairs in europe. *Robot. Automat. Mag.*, 8(1):4–6, 2001.
29. C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.
30. J. van den Berg, S. J. Guy, M. C. Lin, and D. Manocha. Reciprocal n-body collision avoidance. In *Int. J. Robot. Res.*, pages 3–19, 2009.
31. J. van den Berg, S. J. Guy, J. Snape, M. C. Lin, and D. Manocha. RVO2 library: Collision avoidance for real-time multi-agent simulation. `http://gamma.cs.unc.edu/RVO2/`.
32. J. Vannoy and J. Xiao. Real-time adaptive motion planning (RAMP) of mobile manipulators in dynamic environments with unforeseen changes. *IEEE Trans. Robot.*, 24(5):1199–1212, 2008.
33. R. Vatcha and J. Xiao. Detection of robustly collision-free trajectories in unpredictable environments in real-time. *Auto. Robot.*, 37(1):81–96, 2014.
34. A. Wu and J. P. How. Guaranteed infinite horizon avoidance of unpredictable, dynamically constrained obstacles. *Auto. Robot.*, 32(3):227–242, 2012.