Improved Motion Planning Speed and Safety using Regions of Inevitable Collision

Nicholas Chan and James Kuffner and Matthew Zucker
The Robotics Institute
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA, 15213, USA
{ncchan, kuffner, mzucker}@cs.cmu.edu

Abstract Providing safety guarantees for autonomous robots operating in environments with moving obstacles is a difficult problem, particularly for underactuated systems or systems with drift due to momentum. The conventional approach to replanning in dynamic environments typically computes partial plans within the allotted CPU time and validates explored states through robot-obstacle collision checks. However, this approach cannot provide any safety guarantees for the robot beyond the finite planning horizon. This paper explores the approximate computation of *regions of inevitable collision* for state validation in a replanning framework for dynamic systems. We present experimental results that demonstrate the effectiveness of this technique in providing dramatically improved safety for partial plans in the domain of an underactuated dynamic vehicle.

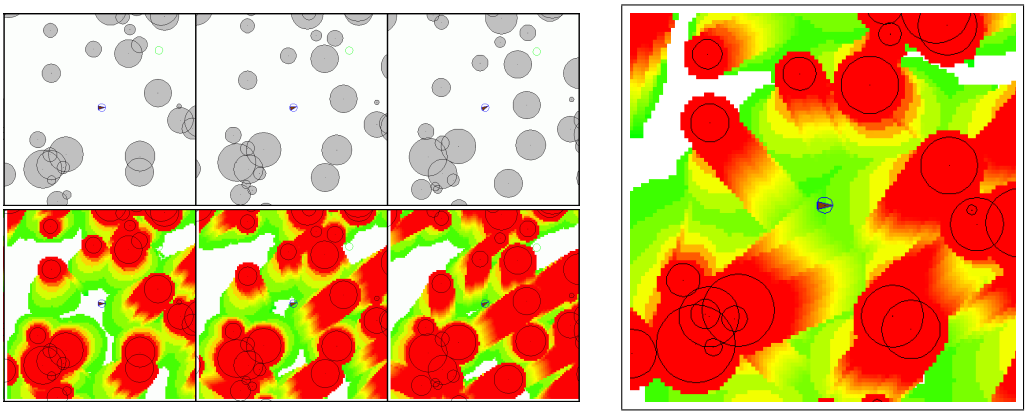

Figure 1. *Top Left:* Planning among moving obstacles for an underactuated spacecraft; *Bottom Left:* Cost maps over time showing the Regions of Inevitable Collision (RIC); *Right:* Detail of RIC combined cost map visualization.

1 Planning for Dynamic Systems

The goal of motion planning is to compute a continuous sequence of controls that enables a robotic system to accomplish a given task while obeying various kinematic and dynamic constraints. Constraints may arise from obstacles that must be avoided in the environment, or from the system dynamics due to momentum conservation, velocity and acceleration bounds, or underactuation. Path planning algorithms simplify the problem by considering only the kinematics of the problem and ignoring the system dynamics. This often allows the problem to be expressed as a search in a lower dimensional space than the full state space of the system. But ultimately, any valid solution must obey all of the constraints.

A planning algorithm examines and evaluates possible future control actions until either some goal state is reached, all possible actions have been evaluated, or the planner exceeds some time or memory limitation. For a robot operating in unknown or dynamic environments, the planner is typically iterated and interleaved with motion execution and perception processing. This replanning process typically imposes time constraints on the planner. If the planner is unable to reach the goal state in the allotted time, a *partial plan* is returned. The robot then executes some segment of the partial plan, while the planning process is iterated. The time limit imposed on the planner induces a *planning horizon*, or a future time beyond which the planner was unable to examine the state of the system. The planning horizon generally corresponds to the length of the partial plan returned during a particular iteration.

The avoidance of stationary obstacles in a motion planning algorithm is often accomplished through the use of a geometric interference detection routine (i.e. collision checking routine). Future states that result in a collision of the robot geometry with obstacle geometry are typically discarded during the search. This ensures that any partial plan made up of states in the search tree will never involve a collision. However, for many dynamic systems, states that are not immediately in collision with an obstacle *can still present just as much danger* in the context of partial plans. For example, a car-like vehicle with momentum moving at high speed towards a wide brick wall obstacle might not have enough braking distance or maneuverability to avoid impact. Thus, even though the car's current state is free of collision, *the car will inevitably collide* regardless of the future control actions applied.

2 Regions of Inevitable Collision

States that lead to collision regardless of control action make up a *region of inevitable collision* or RIC (LaValle and Kuffner (1999, 2001)), alternately referred to as *inevitable collision states* or ICS (Fraichard (2007)). During planning these states should be avoided along with states which intersect obstacles in order to guarantee safety. A limited horizon planner without inevitable collision checks could possibly select a trajectory with an unsafe terminal state that ultimately leads to a future collision outside of the planning horizon. Consider again the example of a car moving at a constant velocity towards a wall. If a future potential impact with the wall is five seconds away and the planning horizon is four seconds, it is likely that executing the computed partial plan endanger the car. If

the car requires four seconds of stopping distance, the next time the planner replans it may already be too late. Clearly in this case the problem could be resolved by increasing the planning horizon to a length safely above the vehicle's stopping distance. However, the computational resources to extend the planning horizon so may not be available. More importantly, this issue is not so easily resolved in cases with moving obstacles, or systems with more complex "stopping distances", such as the underactuated system with drift considered in this paper. In some cases, an arbitrarily long planning horizon may be required to avoid entering a region of inevitable collision, which may impractical or impossible given a time-limited planning architecture.

A limited-horizon planner that tests for RIC membership can improve overall system safety by discarding any generated state that would lead to an inevitable collision. Theoretically, if we can guarantee that the system will never enter an RIC then by definition there will always be a safe control action to take (Fraichard (2007)). If the planner can always identify safe actions, then we can guarantee that no collision will occur, regardless of the planning horizon length (assuming perfect information and control). This is the primary significant benefit to using RIC checks during planning. The second important benefit is improved planning efficiency. A planner without regions of inevitable collision checks could potentially generate many branches in the search tree that are ultimately useless because all leaf states lead to collisions. Depending on the size of the regions, the size of the time step, and the discretization of the control inputs, the planner may waste significant resources examining useless paths through regions of inevitable collision. By eliminating these states in advance, a planner with RIC checks can save a great deal of time and memory, depending on how efficiently it can identify those regions.

In this paper we present a limited horizon motion planner that uses approximate representations of the regions of inevitable collision in order to improve overall planning safety and speed. The experimental domain is a simulated underactuated spaceship vehicle with momentum that must navigate through a space of moving obstacles, similar to the popular game *"Asteroids"*. An A* search is performed over possible control actions in order to guide the ship towards a target goal location while avoiding collisions with obstacles. In addition to computing binary regions of inevitable collision (RIC) we also introduce two new concepts: the *region of potential collision* (RPC), and the *region of near-collision* (RNC). States in the RPC are those for which some set of control actions exist that lead to a collision, and states in the RNC will result in a collision unless the vehicle acts within a certain limited window of time. The RNC and the RPC represent potentially dangerous states that are scored according to risk during planning. Our experiments have shown that utilizing RIC, RNC, and RPC checks during planning resulted in moderate to significant improvements in both speed and overall safety performance.

3 Related Work

The problem of efficient planning for systems with vehicle or obstacle dynamics has been explored by a number of previous research efforts. There is a long history in the controls literature for system verification through computing implicitly or explicity the set of "reachable states" of a dynamic system. Most of this work does not consider robots

moving in environments with dynamic obstacles, so we will consider primarily the developments in the path planning community. Representative early work that considered motion planning in the presence of moving obstacles was (Reif and Sharir (1985)). Consideration of "velocity obstacles" for planning in dynamic environments was presented in (Fiorini and Shiller (1998)). The concept of the dynamic window and investigation of its effect in the domain of partial planning with feedback for localization was given in (Fox et al. (1995); Fox (1998)). Planning with limited perception within a probabilistic framework was considered in (Simmons and Koenig (1995)). The term *"regions of inevitable collision"* was introduced in (LaValle and Kuffner (1999, 2001)). Applications and analysis of inevitable collision states was explored in (Fraichard and Asama (2004)), and applied in (Petti and Fraichard (2005)). An approach that decomposes path and velocity considerations was developed in (Fraichard and Laugier (1993)) and (Laugier et al. (2006)), for the purposes of safe navigation. Motion planning for an underactuated system in the context of a game with moving obstacles was explored in (Ladd and Kavraki (2005)). (Zucker (2006)) examined the analytic RIC computation for convex polygonal obstacles and implemented a planner which exploits RICs for speed enhancements. In this paper, we present results for a planner that explicitly utilizes of regions of inevitable collision to control a system with complex dynamics, underactuation and drift.

4 Planning with Regions of Inevitable Collision

4.1 The Planning Model

Our example planner is a limited-time horizon A* planner (Hart et al. (1968)) that explores the possible control actions in order of ascending expected path cost. Expected costs are handled using precomputed cost maps, and a fixed length plan is generated and replanned at every step. The planner has a fixed relative goal, but does not terminate when it is reached, hence the robot can remain safe once the goal is reached through continually replanning. The dynamics of the robot resemble the spaceship in the classic video game *"Asteroids"*. The environment is a boundless two dimensional plane, empty except for the ship and various moving asteroid obstacles. The asteroids are modeled as circles of varying radius that move in a fixed direction at a constant speed. The asteroids do not collide with one another nor alter their velocities or heading, which simplifies the computation of approximate RICs. The ship is able to thrust forward only and rotate left or right, and has linear and angular momentum. The ship is only able to change its speed by thrusting in the direction of its current heading. Due to angular momentum, establishing a new heading can require applying a rather complex sequence of control actions.

4.2 Defining the RIC

The region of inevitable collision, denoted $\mathcal{X}_{RIC}$, is the set of all robot states for which, regardless of control action, the robot will collide with an obstacle at some point in the future. Given an exact computation of the theorectical RIC, it is possible to guarantee a planner will never hit an obstacle, regardless of limited planning horizons (Fraichard (2007)). By definition, as long as the planner avoids RICs, there will always be a control

action available to avoid obstacles. However, it is often that case that the exact computation of $\mathcal{X}_{RIC}$ is impractical or infeasible for real-world problems. Fortunately, computing approximate representations of RICs provides a number of safety and performance benefits, similar to what one would expect from an exact RIC computation (albeit without the strict safety guarantees).

4.3 Computation of exact RICs

The most obvious method to determine whether a state is a member of $\mathcal{X}_{RIC}$ is to simulate all possible control actions and determine if all possible paths lead to collision. However, there is no inherent upper bound on the length of such a computation. For states outside the RIC, this algorithm will not terminate. Alternatively, we define $\mathcal{X}_{RIC}$ iteratively with:

$$\begin{aligned} x \in \mathcal{X}_{RIC} \quad \Leftrightarrow \quad & \forall y : y \in succ(x), y \in \mathcal{X}_{RIC} \\ & x \cap O \neq \emptyset \rightarrow x \in \mathcal{X}_{RIC} \end{aligned}$$

where O represents the set of obstacles. Thus any state whose successors are all in $\mathcal{X}_{RIC}$ is also in $\mathcal{X}_{RIC}$ and any state intersecting an obstacle is also in $\mathcal{X}_{RIC}$. From this definition we can also define the complement of $\mathcal{X}_{RIC}$, $\mathcal{X}_{free}$ with:

$$x \in \mathcal{X}_{free} \Leftrightarrow x \cap O = \emptyset \wedge \exists y : y \in succ(x) \wedge y \notin \mathcal{X}_{RIC}$$

If we can determine whether a given state x is not contained in $\mathcal{X}_{RIC}$, then it follows that any predecessor states of x are also not in $\mathcal{X}_{RIC}$. Similarly, if we can show that all successors of a given state x are members of $\mathcal{X}_{RIC}$, then by definition x is also a member of $\mathcal{X}_{RIC}$.

For problems involving non-moving obstacles, there are typically many states that lie outside of $\mathcal{X}_{RIC}$. For example, consider the case of a car traveling in an environment filled with stationary obstacles. It is simple to show that any stationary state not already intersecting an obstacle can immediately be labeled as outside the RIC. The obstacles will remain stationary and the vehicle can simply perform the control action that maintains zero velocity. In this case, no collision will ever occur, and we conclude that all collision-free stationary states are members of $\mathcal{X}_{free}$.

However, in the case of moving obstacles, we can no longer assert that stationary states will remain free of collision. In fact, for the dynamic system considered in this paper, being "stationary" has little meaning. The reason is that there is no practical difference between a stationary ship with an asteroid drifting towards it, and a ship drifting towards a stationary asteroid. As a result, we are unable to prove that a particular "base case" state is a member of $\mathcal{X}_{free}$. This difficulty, combined with a limited sensing horizon and the periodic introduction of random asteroids into the environment, precludes directly reasoning about $\mathcal{X}_{RIC}$ membership. However, it is both practical and beneficial to compute approximate representations of $\mathcal{X}_{RIC}$. Rather than being forced to extend the planning horizon, our experimental results show that computing approximate RICs yields significant planner speed and safety gains at the cost of strict safety guarantees.

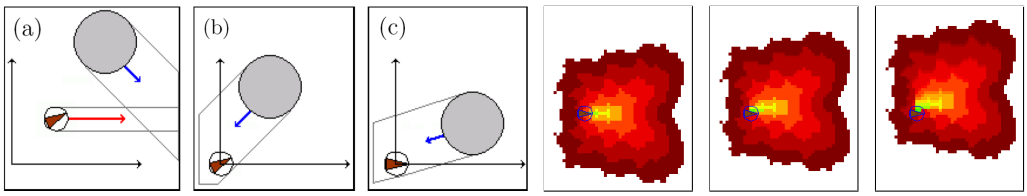

Figure 2. *Left:* RIC coordinate transformations: (a) unsimplified; (b) with ship position and velocity folded into asteroid position and velocity; (c) with ship heading zeroed. *Right:* Precomputed reachability sets with the ship angular velocity increasing from left to right.

4.4 Approximations made when computing RICs

The most significant approximation we make when testing for RIC membership is representing the full RIC as the union of the individual RICs calculated for each asteroid independently. Testing the RIC for each obstacle separately and combining the results may neglect to detect dangerous states in which the motion of several obstacles conspire against the robot. Situations in which the RIC of two obstacles is larger than the union of their individual RICs is illustrated and discussed further in (Zucker (2006)).

Another approximation we make when computing RICs is limiting the time horizon used to determine if it is possible to escape a collision. During RIC computation, if a state is found to be in danger of collision its successor states are examined to determine whether a possible escape path exists that avoids collision within the time limit. Two possibilities exist: 1) an escape route is found, or 2) all successor states examined were found to terminate in collision. Because our planner only examines successor states *up to a fixed depth*, some states will be mislabeled as belonging to the RIC when in fact an undiscovered escape route exists outside the depth limit. Note that this is a conservative approximation, and does not affect planner safety in the same way that considering obstacle RICs independently does.

4.5 Computing RICs

For the purposes of planning, we require a function that tests whether a single given state is within the RIC. Note that several aspects of a robot state will influence the size and shape of the RIC. For the example system in this paper, a ship pointing towards an oncoming asteroid will project a much larger RIC than a ship whose heading direction points away from the oncoming asteroid. In the former case, the ship must first rotate to alter its heading and then proceed to thrust away from the asteriod. Similarly, angular momentum, position, velocity and heading all influence the size and shape of the RIC. The RIC occupies a volume in n dimensions, where n is the number of state variables necessary to describe the robot state. In our example, n is six (x and y position, x and y velocity, heading, and angular velocity). The high-dimensionality of the state makes any attempt at RIC computation difficult. However, for our example, we can utilize coordinate system changes in order to effectively reduce the complexity of the problem significantly (Figure 2).

The first simplification is based on the fact that only the relative position and velocities

of the ship and asteroid matters. There is nothing to differentiate one absolute position from another, therefore we can represent the asteroid relative position and velocity in a coordinate frame attached to the origin of the ship and aligned with the ship heading. Figure 2 illustrates the original problem and the simplified problem after the change of coordinates. Ultimately, only the asteroid relative position, asteroid relative velocity, and ship angular velocity remain variables.

After the coordinate transformation, it is straightforward to determine whether a collision will occur by simply testing intersection with the solid line of the asteroid relative trajectory. States for which the ship bounding circle does not intersect the solid line are clearly outside the RIC for that asteroid. The ship can avoid collision by simply applying no action. States that do intersect the solid line are in danger of collision, and we must determine whether or not collision can be avoided. In general, we would have to exaustively explore the application of all possible control actions up to some time horizon in order to search for an escape trajectory up to some finite depth. However, we can make this computation step drastically more efficient by precomputing and storing tables of *reachability sets* (i.e. discretized tables of reachable states). Because angular velocity is the only remaining free variable in the ship state for our system, it is only necessary to store separate tables for various discrete values of the ship angular velocity. Examples of these these precomputed reachability sets are illustrated in Figure 2*Right*. Using these lookup tables, the planner can more efficiently test whether or not the ship can escape before the asteroid hits it by backtracing trajectories outside of the asteroid solid line and verifying that they are free of collision. Along with the assumptions in Section 4.4, use of these tables introduces another approximation in the computation of the RIC. In theory, an arbitrarily large asteroid would require an arbitrarily large reachability set in order to escape collsion. The limited horizon sets used here would be insufficient, resulting in the misclassification of that state as within the RIC. However, if we can bound the maximum size of the asteroid, we can ensure that any asteroid for which an escape route exists up to the resolution and discretization of our table, the ship will avoid colliding with.

If the planner is able to find an escape route, or the state is not in danger of collision, then the state is marked as not in the RIC. Otherwise, the state is classified as within the RIC. This procedure is repeated for all asteroids and if the state is within any of the individual RICs, then the state marked as such. Recall that in order to provide time-limited safety guarantees with multiple moving obstacles, we need to consider more than simply testing for containment in the union of all individual obstacle RICs. This could be done by actually enumerating and testing all possible escape routes identified from the reachability sets, and then verifying that at least one trajectory can escape all obstacle RICs. This more complete test is not implemented in the current version of our planner. In summary, although several discretizations and approximations are introduced with this method, we now have a reasonably fast and efficient test for RIC membership. Figure 3 illustrates several regions of inevitable collision under different conditions.

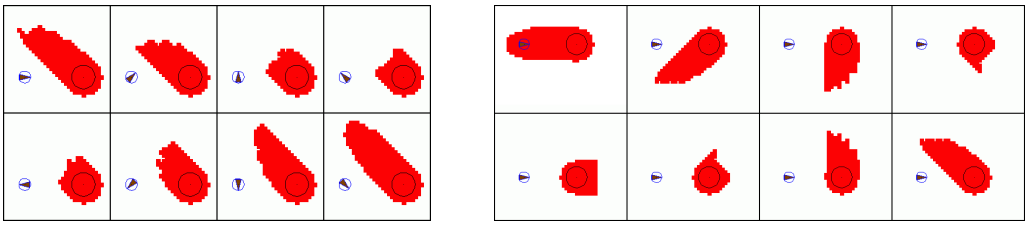

Figure 3. *Left:* RICs for an obstacle moving in a fixed northwest direction for different ship headings. *Right:* RICs for obstacles moving in different directions given a fixed ship heading.

5 Regions of Near and Potential Collision

During our experiments with developing approximate RIC membership tests, we discovered that even if a set of states is found to be outside of the RIC, not all states in the set are equally safe. Some states are more dangerous due to their proximity in both space and time to RIC states. We introduce two new classification sets of states: the *region of near collision* (RNC) and the *region of potential collision*(RPC). The purpose in defining these sets is to enhance planner safety by providing additional metrics and heuristics useful for selecting the best possible control action to apply during the next time step from among the valid set of available actions and/or RIC escape trajectories. Our experiments have found the use of the RNC and RPC heuristics useful for improving overall planner safety performance and output path quality. In addition, although we did not implement simulated noise to our planner, we believe that utilizing RPC and RNC heuristics has the potential to improve planner safety in the presence of uncertainty.

5.1 Regions of Near Collision

A state that is not inside the RIC always has at least one available trajectory that will avoid collision. However, for states near the border of the RIC, the number of available safe trajectories may be very limited. It is not always certain that the planner will be able to discover one of the few control actions that avoids entering the RIC. Additionally, even if a safe control action is selected, uncertainty and error in perception and control may inadvertently cause the robot to enter the RIC. Although being in a state outside of the RIC guarantees that at least one control action avoids collision, it does not mean that finding that control action will be possible. Thus, our planner should have some means of identifying and avoiding dangerous states very close RIC boundaries.

The *region of near collision* (RNC) is a natural extension to the concept of the RIC, and can be easily computed using our planner framework to help identify dangerous states. The RNC is defined as sets of states that are threatened by a future near-term collision, but have the possibility of escaping with a certain amount of time to spare. Like RICs, if no action is taken, a collision will eventually occur. However, RNCs are composed of non-RIC states that are in danger of eventually entering the RIC, meaning there is still some possibility of avoiding collision but that the "window of opportunity" is closing. For example, if a robot requires five seconds to avoid an oncoming obstacle, a

position four seconds away might be within the RIC, while a position six seconds might be within the RNC. In other words, if evasive action is not taken immediately in the latter case, the robot will soon find itself inside of the RIC.

Unlike binary RIC tests, a function for detecting if a state is within an RNC can return a value that represents the relative risk of being in that particular state. RIC states do not generally need danger ratings, because from the point of view of the planner, all states within the RIC are equally undesirable. However, for states within the RNC, the relative ability of the robot to avoid future collision depends on the proximity of the RIC and the skill of the planner in identifying escape routes. We implemented a simple and straightforward extension to our RIC planning framework to evaluate the relative danger of RNC states. For states in danger of collision but with several possible escape routes, one could simply count the number of available escape routes and assign a heuristic cost. States with few available escape trajectories are assigned a high cost relative to RNC states with many available escape routes. In our planner implementation, we assigned a heuristic cost not according to the number of available escape trajectories, but according to the "best" available escape route. The best escape route is defined as the trajectory that avoids the RIC with the widest margin in both space and time from among all available escape routes. Small margins indicate dangerous states, while wide margins indicate relatively safe states. States with margins below some fixed value are considered inside the RNC and their danger value is scaled linearly according to the margin width. The danger values are used by the planner to assign heuristic costs to undesirable but reachable RNC states, thus discouraging them, but allowing them if necessary to reach the goal. Figure 4 illustrates several regions of near collision.

5.2 Regions of Potential Collision

While RNC checks can improve safety in cases of impending collisions, they provide no warning to states that are not already in the path of an obstacle. It is possible for a robot that is in no danger of collision to suddenly enter an RIC after the application of a single control action. The *region of potential collision* (RPC) includes states that are in danger of entering an RIC due to the planner erroneously selecting a bad control action, or suffering from errors or uncertainty in control. As with RNCs, states within RPCs they are rated heuristically according to how dangerous they are. Computation of RPC set membership is again a simple extension of the RIC test. After the state is prepared for RIC testing, the reachability sets are used to determine the lowest cost trajectory that still leads to collision or enters the RIC. If this cost is below a certain threshold, then the state is considered inside the RPC. The state danger is associated with this cost, and the planner uses RPC danger values to heuristically increase the costs of trajectories that pass through RPC states. Hence, the planner tends to avoid both RPC and RNC states, which discourages the robot from approaching these dangerous states.

6 Results and Analysis

We have found that including approximate RIC, RNC, and RPC checks has a positive effect on planner safety and reduces the number of states examined during planning. This

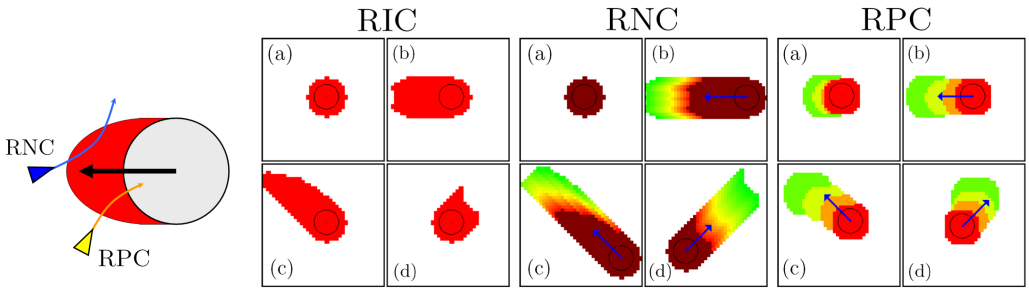

Figure 4. RICs, RNCs, and RPCs for a single obstacle under different conditions. In all images the ship is facing right. The RNCs and RPCs are color-coded to illustrate relative danger levels (dark red regions are RICs). The scenarios are: (a) no motion; (b) obstacle moving left; (c) obstacle moving towards the upper-left; (d) obstacle moving towards the upper-right.

Plan Steps	Total Collisions			Avg. Node Expansions		
	A* w/RIC	A*	Improvement	A* w/RIC	A*	Improvement
3	18	22	18.18%	7.27	7.40	1.76%
4	22	25	12.00%	12.22	12.84	4.83%
5	10	20	50.00%	23.67	27.81	14.88%
6	1	4	75.00%	44.45	65.41	32.04%

Table 1. Collision and node expansion reduction for various plan lengths (100 trials).

improvement varies with both planner and obstacle parameters. Testing involved running a limited-horizon A* planner with and without RIC checks on multiple trials involving random obstacle fields. Each planner version was allotted 100 milliseconds of planning time per iteration, and all experiments were conducted on a 1.6GHz single-processor CPU standard PC. The number of collisions caused by each planner was recorded to evaluate plan safety and the number of node expansions per plan step was recorded to evaluate planning efficiency. The results are summarized in Tables 1-4. The most noticeable variance in performance came from adjusting the length of the planning horizon. (Table 1) While increasing the plan length improved both the standard A* and A* with RIC checks planner performance, the benefits to the RIC planner were significantly greater, as evidenced by the greater reduction in total collisions with longer planning horizons. Additionally the efficiency benefits of pruning states using the RIC checks become more evident as plan length increases. As the search tree depth increases, the opportunity to eliminate large numbers of nodes saves additional computation by reducing the effective branching factor.

One interesting factor that affects RIC planner performance is obstacle size (Table 2). Bigger obstacles are more difficult to bypass, often making collisions more likely. Naturally, larger obstacles produce larger RICs. As expected, RIC planner performance increases initially, then degrades as obstacles become too large for accurate RIC computation. Planner efficiency remains fairly constant.

Obstacle Size Range	Total Collisions			Avg. Node Expansions		
	A* w/RIC	A*	Improvement	A* w/RIC	A*	Improvement
0.5 - 3.0	4	3	-33.33%	18.83	23.78	20.82%
0.5 - 5.0	6	14	57.14%	26.61	33.20	19.85%
0.5 - 6.0	6	16	62.50%	30.27	36.59	17.27%
0.5 - 7.0	11	15	26.66%	28.13	37.24	24.48%
0.5 - 8.0	17	17	0.00%	33.58	41.31	18.71%

Table 2. Collision and node expansion reduction for different obstacle size ranges (50 trials).

Obstacle Speed	Total Collisions			Avg. Node Expansions		
	A* w/RIC	A*	Improvement	A* w/RIC	A*	Improvement
0.25 - 0.5	2	3	33.33%	20.09	22.71	11.54%
0.50 - 1.0	6	14	57.14%	26.61	33.20	19.85%
0.75 - 1.5	9	12	25.00%	26.02	39.28	33.76%
1.0 - 2.0	17	25	32.00%	27.15	38.73	29.95%

Table 3. Collision and node expansion reduction for various obstacle speeds (50 trials).

Another factor in planner performance was the range of obstacle relative speeds (Table 3). Faster moving obstacles induce larger RICs, as they afford less time for evasive maneuvering. Unlike increasing obstacle size, increasing obstacle speed does not adversely effect RIC computation. However as speeds become very high relative to ship speed, the chances of an unavoidable collision increase, resulting in rapid performance gains initially, then more gradually as speeds increase.

7 Summary and Discussion

We have presented a limited-horizon planner that uses approximate representations of the RIC in order to improve overall planning safety and performance. An A* search is performed over possible control actions in order to guide an underactuated ship with drift towards a target goal location while avoiding collisions with obstacles. We also introduce the notion of the regions of near and potential collsions (RNCs and RPCs). States in the RNC will result in a collision unless the vehicle acts within a certain limited window of time, and states in the RPC are those for which some set of control actions exist that lead to a collision. The RNC and the RPC represent potentially dangerous states that are heuristically evaluated according to risk during planning. Experimental results for our example domain demonstrate that utilizing RIC, RNC, and RPC checks during planning resulted in moderate to significant improvements in both speed and overall safety and performance. Safety is improved by avoiding collisions obscured by limited planning horizons, and planning efficiency is increased by more quickly identifying and pruning large subtrees with terminal collision states from the search tree.

The primary drawbacks and limitation of our current implementation include the approximate computation of the RICs and the discretizations that are needed for an

efficient implementation. For high dimensional problems, it may not be feasible to explicitly store large tables of precomputed reachability sets. Theoretically, a planner able to compute exact representations of the RIC can provide safety guarantees even with a very short planning horizon. Computation of more accurate RICs and computation of RICs for other systems are potential areas of further investigation. Additionally, a more detailed examination and analysis of the benefits of RNCs and RPCs may prove useful to the design of "safe" high-performance planners for a variety of applications involving complex dynamic systems.

Bibliography

K. Bekris and L. Kavraki. Greedy but safe replanning under kinodynamic constraints. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, pages 704–710, April 2007.

J. Bruce and M. Veloso. Real-time randomized path planning for robot navigation. In *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, Oct 2002.

D. Ferguson, N. Kalra, and A.Stentz. Replanning with RRTs. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, May 2006.

P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research*, 17(7):760–772, 1998.

D. Fox. *Markov Localization: A Probabilistic Framework for Mobile Robot Localization and Navigation.* PhD thesis, University of Bonn, Germany, 1998.

D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, 4(1), mar 1995.

T. Fraichard. A short paper about motion safety. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, 2007.

T. Fraichard and H. Asama. Inevitable collision states - a step towards safer robots? *Advanced Robotics*, 18(10):1001–1024, 2004.

T. Fraichard and C. Laugier. Path-velocity decomposition revisited and applied to dynamic trajectory planning. In *IEEE Intl. Conf. Automation and Robotics*, pages 40–45, 1993.

E. Frazzoli, M. A. Dahleh, and E. Feron. Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Trans. on Robotics and Automation*, 21(6):1077–1091, Dec 2005.

P. Hart, N. Nilsson, and B. Rafael. A formal basis for the heuristic determination of minimum cost paths. *IEEE trans. Sys. Sci. and Cyb.*, 4:100–107, 1968.

A. Ladd and L. Kavraki. Fast tree-based exploration of state space for robots with dynamics. *Algorithmic Foundations of Robotics*, 17:297–312, 2005.

M. Lau and J. Kuffner. Precomputed search trees: Planning for interactive goal-driven animation. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 2006.

C. Laugier, S. Petti, G. Vasquez, A. Dizan, M. Yguel, T. Fraichard, and O. Aycard. Steps towards safe navigation in open and dynamic environments. In *Auton. Nav. Dyn. Environments*. Springer, 2006.

S. LaValle and J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, May 2001.

S. LaValle and J. Kuffner. Randomized kinodynamic planning. In *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA'99)*, 1999.

S. Petti and T. Fraichard. Safe motion planning in dynamic environments. In *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, pages 2210–2215, 2005.

J. Phillips, N. Bedrossian, and L. Kavraki. Guided expansive spaces trees: A search strategy for motion and cost-constrained state spaces. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, Apr 2004.

M. Pitvoraiko and A. Kelly. Generating near minimal spanning control sets for constrained motion planning in discrete state spaces. In *IROS*, August 2005.

J. Reif and M. Sharir. Motion planning in the presence of moving obstacles. In *IEEE Symposium on Foundations of Computer Science*, pages 144–154, 1985.

R. Simmons and S. Koenig. Probabilistic robot navigation in partially observable environments. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1080–1087, 1995.

J. van den Berg, D. Ferguson, and J. Kuffner. Anytime path planning and replanning in dynamic environments. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, pages 2366–2371, 2006.

M. Zucker. Approximating state-space obstacles for non-holonomic motion planning. Technical Report 06-27, Robotics Institute, Carnegie Mellon University, May 2006.