Avoiding Inevitable Collision States: Safety and Computational Efficiency in Replanning with Sampling-based Algorithms

Kostas E. Bekris

***Abstract*— Safety concerns arise when planning for systems with dynamics among moving obstacles, where a collision-free trajectory leads to an Inevitable Collision State (ICS). Identifying whether a state is ICS, however, is computationally challenging. This has led to approximations, varying from conservative schemes, which never characterize an ICS as a safe state, to schemes with weaker guarantees but fast online resolution of an ICS query. The computational cost of the approach is critical in problems that require replanning. This report presents various alternatives for identifying whether a state is ICS from the related literature. It also discusses different ways for integrating such schemes with sampling-based planners in safe replanning frameworks so as to reduce the computational overhead of avoiding ICS.**

I. INTRODUCTION

It becomes increasingly important to reason about the dynamics of robots during planning so as to achieve a higher degree of autonomy by produing plans that can be safely and directly followed. Examples of motion constraints that have to be respected are bounds in velocity and acceleration that cause drift. At the same time, in many realistic tasks, robots have only partial information about their environment. Examples incude planning among unknown static obstacles, exploration and planning in dynamic environments. Solving such problems requires an interleaving of sensing, planning and execution. This means that a planner is called frequently and has finite time to replan a trajectory.

This report focuses on the safety concerns that arise when replanning for systems that exhibit non-trivial kinodynamic constraints. Safety becomes an issue because a collision-free trajectory may still bring a dynamical system close to an obstacle region with high-velocity and no possible maneuver to avoid collisions into the future as Figure 1 illustrates. Similar problems also arise when planning in dynamic environments, even for systems without challenging dynamics, because the obstacles may block escaping maneuvers. Such challenges were identified early on in the motion planning literature. Collision-free states that inevitably lead to collisions have been referred to in the past as Obstacle Shadows [1], or Regions of Inevitable Collision [2] or Inevitable Collision States (ICS) [3]. This work follows the ICS terminology.

Over the last decade, sampling-based planners that construct a tree data structure of feasible trajectories, such as RRT [2] and its variants [4], [5], have become popular for planning with dynamic constraints. Thus, it is important to show how ICS avoidance can be achieved effectively in the

Kostas E. Bekris is with the Computer Science and Engineering department, University of Nevada-Reno, Reno, NV, 89557 email: bekris@cse.unr.edu

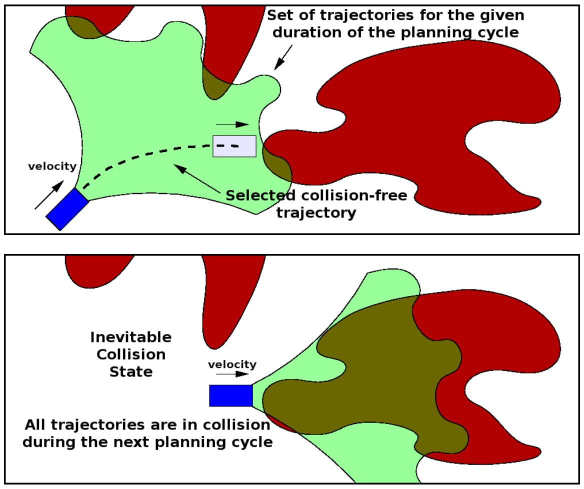
Set of trajectories for the given duration of the planning cycle
velocity
Selected collision-free trajectory
Inevitable Collision State
velocity
All trajectories are in collision during the next planning cycle

Fig. 1. A collision-free trajectory may still lead to an ICS.

context of tasks that require replanning using sampling-based algorithms. Nevertheless, identifying whether a state is ICS or not is computationally challenging for most realistic problems. For moving obstacles, it also requires knowledge of the obstacles' paths. This report reviews various alternatives from the related literature for avoiding ICS and how these schemes can be integrated with sampling-based planners.

Some methods for ICS avoidance employ machine learning offline or approximation methods in order to construct an ICS identifier that can be used online to prune those states identified as ICS. These methods have the advantage of being fast in their ICS determination during the online step but they can misclassify states. Thus, these methods do not provide safety guarantees even for simpler replanning tasks, such as planning among unknown static obstacles. Conservative alternatives identify a state as safe by explicitly computing the ICS set and reasoning over an infinite time horizon. These methods provide stronger safety guarantees, which is highly desirable, but they also require more online computations, which can also effect the practical safety of the approach. A slow planner delays taking into account new sensing information, while a faster planner provides a more diverse set of plans in the same amount of time.

The speed of a replanning scheme that aims to avoid ICS does not only depend on the computational cost of the method for identifying ICS. It also depends on how many calls to an ICS-identification method are made by the high-level replanning approach. This report surveys alternative schemes for safe replanning and discusses their computational efficiency.

II. BACKGROUND

There are two high-level approaches for problems that require online recomputation of a robot's path: (i) reactive approaches and (ii) replanning using a global planner. Reactive methods determine the motion the robot should execute so as avoid collisions and are designed for unknown or dynamic environments. An early, successful reactive method was the Vector Field Histogram approach [6], which, however, did not reason about robot dynamics. More recent alternatives do reason about dynamics and include the Nearness Diagram Navigation [7], the Dynamic Window Approach [8]–[10] and Velocity Obstacles [11], [12]. The first two assume static obstacles, while the later typically assumes obstacles with constant linear velocity. Path deformation methods compute a flexible path, which is adapted on the fly so as to avoid moving obstacles [13].

The focus of this report is on replanning by iteratively calling a global planner. For problems where the state space can be effectively discretized, the D* family of algorithms are very effective [14], [15]. Sampling-based planners, popularized by the Probabilistic Roadmap Method [16] and the Rapidly-Exploring Random Tree (RRT) approach [2], are effective in exploring state spaces that cannot be effectively discretized otherwise. Methods that construct trees of feasible trajectories, such as RRT, are especially effective in planning with dynamics. Various methods have been proposed in the literature for adapting sampling-based planners to replanning tasks [17]–[23]. Some of these methodologies focus on systems with dynamics and consider the safety issues that arise [3], [4], [24]–[32]. The following discussion details the contributions of these later approaches.

Wikman et al. [24] worked on controllers that maintained systems with dynamics in the collision-free part of the space for static scenes. The approach employed braking maneuvers to evaluate whether states led to inevitable collisions or not. Frazzoli et al. [25] proposed a real-time kinodynamic planner for dynamic environments. They formulated the concept of τ-safety as a guarantee of no collision during τ seconds for each node of a tree created with a sampling-based algorithm. Hsu et al. [4] proposed a kinodynamic sampling-based planner that calculated an escape maneuver in case it failed to find a complete trajectory to the goal during the allotted time. The escape maneuver brought the system to a collision-free state given the dynamic obstacles. The approach was tested on real air-cushioned robots. Bruce and Veloso extended previous work on replanning for kinematic systems [18] by employing a controller for generating motion commands that respected constraints together with a procedure for searching safe paths by incorporating braking maneuvers [27].

Alami at al. [26] limited the planning within the robot's visibility region and employed worst-case assumptions that its boundary was defined by moving obstacles. The planner computed a path and a maximum velocity profile such that the robot can decelerate to a halt before colliding with moving obstacles. A similar approach is followed by the recent work by Vatcha and Xiao [32] for higher-DOF robots.

Fraichard and collaborators [3], [33]–[38] have provided a comprehensive study of Inevitable Collision States (ICS) over the last few years. In particular, Fraichard and Asama [3] presented a formalization and discussed how to acquire conservative approximations of the ICS set. They applied their approach on navigation among unexpected obstacles, where braking maneuvers are sufficient. Petti et al. [33] integrated this ICS avoidance approach with a Partial Motion Planning framework, which employed sampling-based algorithms, and applied it to planning in dynamic environments. Parthasarathi et al. [34] proposed imitating maneuvers for problems involving car-like obstacles and vehicles. This work has been extended to general planar robots [36].

Fraichard [35] also emphasized 3 criteria for motion safety. A robot should consider (i) its own dynamics, (ii) the environment objects' future behavior and (iii) reason over an infinite-time horizon, to achieve safety. A technique that avoids ICS will satisfy these criteria and is guaranteed safety. Nevertheless, traditional approaches for reactive navigation fail to provide motion safety in dynamic environments. The Nearness Diagram Navigation and the Dynamic Window approaches do not reason about the motion of obstacles. Velocity Obstacles do not reason over an infinite-time horizon. Similarly, τ-safety does not provide safety. Experimental comparisons [37], [38] have also shown the practical advantages of ICS-avoidance against other reactive navigation schemes [10], [12]. A recent variation of Velocity Obstacles does reason about the time horizon [39].

Bekris and Kavraki [28] proposed an integration of a kinodynamic sampling-based planner with ICS avoidance schemes. They applied the approach on an exploration problem in a static environment using breaking maneuvers. The focus was on minimizing the computational overhead of providing safety guarantees by reducing the number of states along the sampling-based tree that have to be shown safe. This is an issue discussed in detail in section V.

Kalisiak and Van den Panne [29] have proposed avoiding ICS during one-shot planning for systems with dynamics to speed up the process. They've employed learning-based methods to identify the set of ICS. They have also utilized ICS in order to maintain safety when a dynamical system is controlled by a human user [40].

Tsianos and Kavraki [30] have used replanning to construct an efficient kinodynamic planner by incrementally computing safe trajectories. They tested their approach on a Segway-like system moving among dynamic obstacles.

Kuffner and collaborators [31] have proposed ICS approximations to improve overall planning safety and speed. The domain was a simulated underactuated spaceship vehicle with momentum that must navigate through moving obstacles, similar to the game of "Asteroids". The work by Zucker [41] describes similar approximation for computing state$\times$time space obstacles.

The following section will present a formulation of ICS, while section IV details alternative methods for detecting whether a state is ICS or not. Section V covers different ways to integrate ICS-avoidance with sampling-based planning.

III. INEVITABLE COLLISION STATES

Consider a **workspace** $\mathcal{W}(t)$, which is either dynamic or partially known to a robot and discovered through sensing. These assumptions result in a time dependent representation. An individual **obstacle** in the workspace at time t will be denoted as $\mathcal{O}_i(t)$, while the entire subset of $\mathcal{W}(t)$ that corresponds to obstacles will be $\mathcal{O}(t)$. For dynamic environments, the assumption is that the future path of each $\mathcal{O}_i$ is known.

The focus is on robots with interesting dynamics that have to deal with inertia and potentially additional constraints, e.g., underactuated, non-holonomic systems. In particular, assume a **robot** $\mathcal{A}$ whose motion is governed by $\dot{x} = f(x,u)$ and $g(x,\dot{x}) \leq 0$, where $x \in \mathcal{X}$ is a **state**, $\dot{x}$ is its time derivative, $u \in \mathcal{U}$ is a **control** and f, g are smooth. $\mathcal{X}$ and $\mathcal{U}$ respectively denote the **state space** and the **control space** of $\mathcal{A}$. $\mathcal{X}_f(t)$ denotes the collision-free part of the state space at time t given $\mathcal{O}(t)$.

A **plan** is a time-sequence of controls

$$p(dt) = \{(u_1, dt_1), \ldots, (u_n, dt_n)\}$$

where $dt = \sum_i dt_i$. Then the set of all possible plans of duration dt is denoted as $\mathcal{P}(dt)$. When a plan $p(dt)$ is executed at state $x(t)$, it defines a **trajectory**:

$$\pi(x(t), p(dt)).$$

A trajectory is the sequence of states propagated according to $\dot{x} = f(x,u)$, and which also respects $g(x,\dot{x}) \leq 0$. A state along a trajectory π at time $t' \in [t : t+dt]$ is denoted as $x^{\pi}(t')$. A trajectory $\pi(x(t), p(dt))$ is **collision-free** if

$$\forall\ t' \in [t : t+dt] :\ x^{\pi(x(t),p(dt))}(t') \in \mathcal{X}_f(t').$$

Given the above notation, it is now possible to define what constitutes an **Inevitable Collision State** (ICS). A state $x(t)$ is ICS given $\mathcal{P}(\infty)$ and $\mathcal{O}(t)$ if and only if:

$$\begin{aligned}\forall\ p(\infty) \in \mathcal{P}(\infty), \forall\ \mathcal{O}_i(t) : \exists\ dt \in [t, \infty) \text{ so that}\\ x^{\pi(x(t),p(\infty))}(dt) \notin \mathcal{X}_f(dt).\end{aligned} \qquad (1)$$

Then it is possible to define the set $\texttt{ICS}(\mathcal{O}(t), \mathcal{P}(\infty))$, which contains all the states $x(t)$ for which Eq. 1 is true. Conversely, a **safe state** $x(t)$ is not in $\texttt{ICS}(\mathcal{O}(t), \mathcal{P}(\infty))$. This means that $x(t)$ is safe if and only if:

$$\begin{aligned}\exists\ p(\infty) \in \mathcal{P}(\infty),\ \text{so that}\\ \forall\ \mathcal{O}_i(t), \forall\ dt \in [t, \infty) :\ x^{\pi(x(t),p(\infty))}(dt) \in \mathcal{X}_f(dt).\end{aligned}$$

IV. IDENTIFYING AND AVOIDING ICS

Computing whether a state $x(t)$ is in $\texttt{ICS}(\mathcal{O}(t), \mathcal{P}(\infty))$ is infeasible, since it requires to consider all plans in $\mathcal{P}(\infty)$, the infinite set of all possible plans of infinite duration out of $x(t)$! This section describes the methods that have been developed in the literature for evaluating state safety. Such modules can be used as black-boxes within a motion planner to detect and avoid ICS, the same way a collision-checker is traditionally employed to detect states that are in collision.

A. *Conservative Approximation of the ICS set*

While the computation of $\texttt{ICS}(\mathcal{O}(t), \mathcal{P}(\infty))$ is not feasible, it is possible to define a conservative approximation of this set that in many cases can be computed effectively. Consider a set of maneuvers $\Gamma(\infty)$, referred to as *contingencies* here, which is a subset of the entire space of infinite duration plans $\mathcal{P}(\infty)$. Then it is true [3] that:

$$\texttt{ICS}(\mathcal{O}(t), \mathcal{P}(\infty)) \subset \texttt{ICS}(\mathcal{O}(t), \Gamma(\infty)).$$

So all the states at time t that are not in $\texttt{ICS}(\mathcal{O}(t), \Gamma(\infty))$ are safe. If the set $\Gamma(\infty)$ is a discrete set, then it is possible to enumerate all of the plans $\gamma(\infty) \in \Gamma(\infty)$. Given this idea, Fraichard and collaborators [34], [36] have proposed the following scheme for checking whether a state $x(t)$ is safe or not:

1) Select $\Gamma(\infty)$.
2) Compute $\texttt{ICS}(\mathcal{O}_i(t), \gamma(\infty))$ for every obstacle $\mathcal{O}_i(t)$ and contingency $\gamma(\infty) \in \Gamma(\infty)$.
3) Compute for every contingency:
$$\texttt{ICS}(\mathcal{O}(t), \gamma(\infty)) = \cup_{\mathcal{O}_i(t) \in \mathcal{O}(t)} \texttt{ICS}(\mathcal{O}_i(t), \gamma(\infty)).$$
4) Compute the overall ICS set:
$$\texttt{ICS}(\mathcal{O}(t), \Gamma(\infty)) = \cap_{\gamma(\infty) \in \Gamma(\infty)} \texttt{ICS}(\mathcal{O}(t), \gamma(\infty)).$$
5) Determine whether $x(t) \in \texttt{ICS}(\mathcal{O}(t), \Gamma(\infty))$. If so return "not safe", otherwise return "safe".

For replanning problems in static environments (e.g., planning among unexpected obstacles, exploration of an unknown static environment), braking maneuvers, where the robot decelerates as much as possible, are sufficient for the set $\Gamma(\infty)$ [3], [24], [28]. For a moving obstacle $\mathcal{O}_i$ with a known future plan, imitating maneuvers have been proposed where the robot tries to achieve and maintain a zero-relative velocity with regard to $\mathcal{O}_i$ [34]. If there are multiple obstacles, then the set $\Gamma(\infty)$ can contain one imitating maneuver per obstacle together with a fixed number of braking maneuvers [34].

The computation of $\texttt{ICS}(\mathcal{O}_i(t), \gamma(\infty))$, requires to compute all the states $x(t)$ that lead to a collision with $\mathcal{O}_i(t)$ if the plan $\gamma(\infty)$ is executed from $x(t)$. This is still an expensive computation. For planar robots, however, a methodology has been proposed that brings the promise of computational efficiency [34], [36]. The idea is to focus on a 2D projection of the state space that corresponds to the Cartesian coordinates of the robot. The obstacles are grown so as to represent the robot as a single point. For example, for a state $x =< c_x, c_y, \theta, v, \phi >$ of a second-order car-like system, the approach defines the 2D projection of X on Cartesian coordinates for the values $< \theta, v, \phi >$ of x. Then for every point o_i on the surface of an obstacle $\mathcal{O}_i$, either static or moving, it is possible to compute for a maneuver $\gamma(\infty)$ the subset of the 2D projection that corresponds to ICS. In order to define the ICS for a polygonal obstacle, it is then possible to take the union of the ICS sets for all the points on the obstacle's boundary. This procedure has an infinite time horizon and provides safety guarantees.

Furthermore, the approach is amenable to offline computation, by discretizing the non-Cartesian coordinates (e.g., $<\theta, v, \phi>$) and precomputing the ICS set for different values of these parameters, contingencies and obstacle configurations. This discretization, however, results in an approximation that is not conservative anymore so it might be desirable to compute the ICS sets online. Furthermore, computing the unions and intersections of ICS sets has to be executed online. Note that the overall approach becomes increasingly more difficult to execute as the robotic system becomes more complex (e.g., 3D problems).

B. Simulating Contingencies

The previous approach first constructs the set $\texttt{ICS}(\mathcal{O}(t), \Gamma(\infty))$ (at least in a 2D projection of $\mathcal{X}$) and then computes whether a specific state $x(t)$ is part of this set. This requires the computation of the preimage of a state for a specific maneuver, which for incresingly more complex system it becomes increasingly more challenging to compute. An alternative is to compute the trajectories $\pi(x(t), \gamma(\infty))$ that result by executing the contingencies $\gamma(\infty)$ at $x(t)$ and then whether these trajectories collide with the future paths of the obstacles. Then the following procedure can be defined:

1) Select $\Gamma(\infty)$.
2) Compute the trajectories $\pi(x(t), \gamma(\infty))$ for all contingencies $\gamma(\infty) \in \Gamma(\infty)$.
3) Then compute the intersection of each $\pi(x(t), \gamma(\infty))$ with the path of each obstacle $\mathcal{O}_i$.
4) If there is a path $\pi(x(t), \gamma(\infty))$ that does not collide with any path of any obstacle, then return "safe", otherwise return "not safe".

At a high level, this is the approach that has been used in many of the existing efforts towards the computation of safe paths [4], [25], [28], [30], where the intersection of paths is executed in an incremental fashion by sampling along the trajectories, the same way that traditional collision checking is executed in sampling-based algorithms. None, however, of these implementations considers contingency trajectories of infinite duration, thus not providing safety by violating the third criterion of motion safety [35].

Consequently, the challenge is how to execute conservative collision checking between two trajectories of infinite duration for two general systems. One direction is to bound at each state $x^{\pi(x(t), \gamma(\infty))}(dt)$ along the robot's contingency trajectory, where $dt \in [t, \infty)$, the subset of the space that $\mathcal{A}$ can occupy after time dt. For example, consider a contingency maneuver that forces an airplane-like robot to execute circular paths. Then the subset of $\mathcal{X}$ that can be occupied by the robot is bounded by a disk (for a planar problem - the reasoning extends to higher dimensions as well). Similar bounds can be computed for an obstacle assumed to move along a path with constant linear velocity, the typical assumption in reactive navigation (e.g., Velocity Obstacles). At each state the obstacle will never again reach the half-plane behind it defined by the line that goes through its position and perpendicular to its velocity vector. Given such bounds, it is possible to guarantee collision-avoidance between the systems without having to execute an infinite number of collision-checking calls.

Alternatively it is possible to conservatively approximate the swept volumes defined by the robot's contingency and the obstacle's path. If the swept volumes do not intersect, collision avoidance is guaranteed. If they do intersect, then it is possible to check whether the corresponding systems arrive at the intersection point at the same point in time. If appropriate bounds cannot be found easily for two trajectories, then they can be considered as colliding, which results in a conservative computation of $\texttt{ICS}(\mathcal{O}(t), \Gamma(\infty))$.

This implies that the selection of the contingency maneuvers is again critical. It is important to consider maneuvers which quickly bound the space that the robot can occupy into the future or for which it is easy to compute a conservative approximation of the swept volume. Thus, braking or circling maneuvers are appropriate, since they limit the space a robot can visit to a local region of $\mathcal{W}$. Similarly, maneuvers that move the robot along a straight line path are also appropriate as they quickly exclude a large part of the workspace and lead to easy definitions of the swept volume. The effectiveness of this approach depends on the complexity of the obstacles' paths. A complex obstacle path that does not easily provide any meaningful bound or an easy swept volume computation will be harder to address. Furthermore, although the bounds for prespecified maneuvers can be computed offline, this approach again requires a lot of the computations to occur online. This approach has the prospect of generalizing to complex systems as it depends only on forward integration of controls.

C. Relaxing Guarantees and Simplifying Computation

The above approaches aim to provide a conservative approximation of the set $\texttt{ICS}(\mathcal{O}(t), \mathcal{P}(\infty))$. This means that no state in this set would be returned as safe. This guarantee comes at the expense of computational cost. Thus, it might be appropriate in certain cases to relax the guarantees provided and simplify the computation of the ICS set. This is the methodology followed by Kuffner and collaborators [31], [41].

This approach does not make use of a set of contingency maneuvers but instead tries to compute directly $\texttt{ICS}(\mathcal{O}(t), \mathcal{P}(\infty))$ for all possible plans of infinite duration. It employs the following tools to approximate the set:

a) In the general case, the set $\texttt{ICS}(\mathcal{O}_1(t) \cup \mathcal{O}_2(t), \mathcal{P}(\infty))$ is not equal to the union of $\texttt{ICS}(\mathcal{O}_1(t), \mathcal{P}(\infty))$ and $\texttt{ICS}(\mathcal{O}_2(t), \mathcal{P}(\infty))$. The approximation employed is to ignore this issue, and represent the full ICS set as the union of the individual ICS sets calculated for each individual obstacle. This means that ICS states might be returned as safe by the approach [31].

b) If a state $x(t+dt)$ is shown to be safe, then any predecessor state $x(t)$, for which there exists a plan $p(dt)$ so that $x(t+dt) = x^{\pi(x(t), p(dt))}(t+dt)$ is also safe. The approach searches for safe successor states of state $x(dt)$ only within a limited time horizon. Although limiting the time horizon when checking for collisions to prove that a state is ICS is problematic (violates the third criterion of motion safety), the opposite, to limit the time horizon when checking for an escape route, is a conservative approximation and does not affect planner safety [31].

The approach also precomputes tables of ICS sets through a discretized representation for different configurations of individual obstacles relative to the robot. These reachability sets for all the obstacles are then transformed online given the parameters of a specific state. If a state belongs to at least one ICS set for an individual obstacle, it is marked as an ICS overall. The precomputation assists in reducing the online overhead of the approach but introduces new approximations.

Finally, this method introduces two additional concepts: the Region of Potential Collision (RPC) and the Region of Near-Collision (RNC). States in the RPC are those for which there exist plans that lead to a collision. States in the RNC will result in a collision unless the vehicle acts within a certain limited window of time. The approach proposes to heuristically avoid such states during the planning process. This additional level of granularity in the definition of safe states has experimentally shown good results.

D. Learning Approximations

If weak safety guarantees are acceptable, then it is also possible to utilize machine learning algorithms for characterizing ICS sets. This is the approach followed in the work by Kalisiak and Van de Panne [29], [40].

The idea is to train an ICS model using examples of safe states collected through experiments offline and utilize it during the online identification of ICS states. The input for the training procedure are descriptors of the form: (x, s), where x is the state of the robot and s is descriptor of the robot's local surroundings. In the work by Kalisiak et al. [29], the focus was on static environments and s corresponded to the distance from the agent to the nearest obstacle along the robot's velocity vector. The environment descriptor can potentially be extended for the case of moving obstacles to include the velocity of obstacles in the vicinity of the robot.

In order to collect examples of safe states it is possible to make use of very long random-walk trajectories, created by applying random control actions at each time step, and backtracking upon failure. Every state along the random walk that is followed by a collision-free trajectory of sufficient duration is considered to be an example of a safe state. Thus, this training approach does not yield examples of ICS but only of safe states. Furthermore, it only provides approximate examples of safe states given a limited time horizon. In the existing work [29] a learning procedure using SVMs was employed because it has the capability to learn even when the examples belong to only one class.

The advantage of this procedure is that once the learned model has been constructed, then it is really fast during the online planning to get an answer to the question of whether a state is safe or not. It is interesting to investigate, however, the level of accuracy in ICS identification that such learning-based approaches can potentially achieve, especially for problems involving moving obstacles.

E. Trade-offs

The above alternatives have divergent objectives. The first two aim to provide a conservative approximation of the ICS, so that every state that is deemed safe by the technique is truly safe and an available contingency maneuver of infinite duration can be provided that avoids collisions, at least for problems where the motion of dynamic obstacles is known. The last two methods relax their guarantees, so they may be wrong in the identification of a safe state, but they aim to provide a toolbox that quickly answers ICS queries.

Providing stronger guarantees is clearly preferable and desirable. Nevertheless, the scalability of all the proposed schemes in complex high-dimensional systems and challenging dynamics scenes remains to be shown. Moreover, the speed with which a technique can identify ICS online is of critical importance for the practical safety of a robotic system. For example, consider a robot that operates in the safe subset of the state space. Then a new moving obstacle appears in the workspace. The robot has to be able to quickly recompute the new $\texttt{ICS}(\mathcal{O}(t), \mathcal{P}(\infty))$, otherwise it may quickly enter this set and will not be able to avoid collisions. Benchmarks and experimental comparisons between the discussed alternatives are needed to evaluate the relative effectiveness of the various techniques, including setups where the assumption that the motion of the obstacles is known is violated.

One direction could be a combination of the current approaches. During the online operation of a planner, a fast technique could be first employed, either learning-based or approximation-based, to quickly prune potentially unsafe states. Then for states deemed to be safe, the conservative but slower methods can be used to guarantee safety.

V. Safe Replanning

Checking whether a state is ICS or not has a computational overhead, which is typically higher than the overhead of computing whether a state is collision-free or not. Thus it is important to minimize the number of calls to the ICS-identification module during replanning. What is, however, the minimum number of states which a replanning process has to check for ICS to guarantee safety?

This section describes various alternatives from the related literature for replanning with a global planner so as to compute safe trajectories. Since sampling-based algorithms are popular for such problems, the discussion starts by an outline of these methods, which will assist the description of their integration into safe replanning frameworks.

A. Sampling-based Planning for Systems with Dynamics

Sampling-based algorithms incrementally construct a tree data structure that stores trajectories in $\mathcal{X}$ with the objective to quickly cover the state space as quickly as possible. The description here follows the popular RRT method [2]. The tree is rooted at the initial state $x_0(t_0)$ of the robot. Then at each iteration, the algorithm randomly samples a state $x_r \in \mathcal{X}$. Given a distance metric in $\mathcal{X}$, the closest state $x_c(t_c)$ along the currently expanded tree of trajectories can be determined. A set of n constant control plans $\{p_1(\epsilon), \ldots, p_n(\epsilon)\}$ are then applied from $x_c(t_c)$ for a duration ϵ each, yielding n candidate local trajectories $\pi_i(x_c(t_c), p_i(\epsilon))$. Out of these trajectories, typically only one is kept, the one with the resulting state $x^{\pi_i(x_c(t_c), p_i(\epsilon))}(t_c + \epsilon)$ which is closer to x_r according to the distance metric in $\mathcal{X}$. For single-shot planning, this process is repeated until one of the states $x^{\pi_i(x_c(t_c), p_i(\epsilon))}(t_c + \epsilon)$ falls within the goal region.

B. Partial Motion Planning Framework

For the dynamic problems considered in this report, a robot has to plan a motion given a limited amount of time and execute it in order to remain safe. Let's denote as δ_c the duration of a planning cycle during which the robot much calculate a new motion. During each cycle, it is possible to employ a sampling-based planner to compute a partial plan given the current model of the world. Unfortunately, sampling-based planners are only probabilistically (or resolution) complete. This means that they do not provide any running-time upper bound: there is no guarantee that a complete solution can be found within time δ_c.

Fraichard and collaborators [33] have proposed a Partial Motion Planning (PMP) framework that avoids ICS given the existence of a module for the identification of such states. The scheme assumes that the initial state of $\mathcal{A}$ is ICS-free. Then during the cycle (t_i, t_{i+1}), where $t_{i+1} = t_i + \delta_c$, the following steps are executed:

1. An updated model of the workspace is acquired.
2. A tree data structure of feasible trajectories is created by an algorithm similar to the one described in V-A. The tree is rooted at state $x(t_{i+1})$, the initial state of $\mathcal{A}$ in the beginning of the next cycle.
3. During the expansion of the tree, every resulting state $x^{\pi_i(x_c(t_c), p_i(\epsilon))}(t_c+\epsilon)$ of a local propagation step is checked whether it is ICS. If it is, then the corresponding trajectory $\pi_i(x_c(t_c), p_i(\epsilon))$ is pruned and not inserted in the tree. In this way, the entire tree stores only safe trajectories.
4. As time approaches t_{i+1}, the current cycle is over and the best safe partial trajectory $\pi(x(t_{i+1}), p(\delta_c))$ is selected so as to best complete the current task.
5. If the planning process failed to produce any safe trajectory and the state $x(t_{i+1})$ was safe, the robot can follow the first δ_c part of a contingency maneuver $\gamma(\infty) \in \Gamma(\infty)$ defined for state $x(t_{i+1})$.

The state of the robot in the beginning of each planning cycle is guaranteed to be safe, because the algorithm selects either safe paths or contingencies, which are also safe. Thus, a maneuver $\gamma(\infty) \in \Gamma(\infty)$ will always exist at t_{i+1}. Thus, the PMP framework guarantees safety as long as the ICS-identification module can truly prune all the ICS state (i.e., satisfy the three criteria for motion safety, including to reason over an infinite time horizon). This framework, however, checks all the nodes of the tree for safety. This is also the approach employed in the work by Frazzoli et al. [25], where all the nodes of the tree are checked for safety, alas without reasoning over an infinite time horizon.

C. Minimizing the Number of ICS Checks

Bekris and Kavraki [28] follow a similar high-level approach for replanning like the Partial Motion Planning framework. Instead of step 3 in the previous subsection, however, their approach checks a different, smaller set of states for ICS in order to provide safety guarantees. The idea is that it is not necessary to guarantee the safety of the entire tree but only those states along the tree that are potential initial states during the next planning cycle. After all, the safety of the PMP framework depends only the existence of a contingency at the beginning of the next planning cycle.

Thus, the step 3 from the previous procedure can be replaced by the following operations.

- For every candidate trajectory $\pi_i(x_c(t_c), p_i(\epsilon))$ during the expansion of the tree, check whether this trajectory intersects the beginning of the next planning cycle. That is check whether $t_{i+2} \in [t_c, t_c + \epsilon]$. If this is true, then check $x^{\pi_i(x_c(t_c), p_i(\epsilon))}(t_{i+2})$ for ICS. If this state is deemed to be ICS then prune the corresponding trajectory.

Note that the state $x^{\pi_i(x_c(t_c), p_i(\epsilon))}(t_{i+2})$ is not necessarily a node of the tree, most of the time it actually occurs along an edge. Furthermore, the number of states along the tree which intersect a particular point in time are guaranteed to be less than the number of nodes along the tree. Thus, this approach calls the ICS-identification procedure fewer times than the original PMP framework. Moreover, this is the minimum number of states that have to be checked in order to provide safety guarantees in the context of replanning with a sampling-based algorithm.

Care, however, has to be taken during the selection of the path given this change. In the original PMP framework, it was also possible to select a trajectory of duration δ shorter than the duration of the planning cycle ($\delta < \delta_c$). This is because the resulting state of such a short trajectory (i.e., a node of the tree data structure) is checked for safety. The robot can execute the corresponding plan up to time δ and then switch to the corresponding contingency for the resulting state, which was used to prove its safety. This is not possible in the scheme by Bekris and Kavraki [28], since trajectories stored along the tree of duration $\delta < \delta_c$ have not been checked for safety. So step 4 of the PMP framework has to be slightly adapted. In step 4, a solution trajectory must be at least of duration δ_c before selected. Such trajectories have been checked for safety.

Moreover, the above procedure results in a tree where all the trajectories within the interval $[t_{i+1}, t_{i+2}]$ are safe but future trajectories are not guaranteed to be safe. This does not cause any issues as long as during the next cycle the tree construction starts from scratch. It is possible, however, to consider an optimization step, where in the beginning of a new planning cycle $[t_{i+1}, t_{i+2}]$, the algorithm retains the part of the tree computed during the previous cycle $[t_i, t_{i+1}]$ that is still valid, which is the part of the tree past time t_{i+2} reachable from $x(t_{i+2}$. Since this part of the tree was not guaranteed to be safe during the previous cycle, it has to be checked during the tree retainment step. Again the algorithm will check only those states along the retained tree which occur in the beginning of the consecutive cycle, that is t_{i+2}.

An alternative scheme has been employed by Tsianos and Kavraki [30], which also utilizes the idea that only states that occur in the beginning of a planning cycle have to be checked for safety. The idea is to split the planning cycle of a robot into two steps: the tree expansion step and the safety check step. During the tree expansion procedure, no safety check is executed. During the safety check procedure, the algorithm starts from the last state along the tree that intersects the beginning of a planning cycle. So if the tree is expanded during the planning cycle $[t_i, t_{i+1}]$, then the algorithm will consider states that occur on $\{t_{i+2}, t_{i+3}, t_{i+4}, \ldots\}$ but in reverse order. Checking first future states aims to utilize the property discussed in section IV-C: if a state $x(t + dt)$ is determined to be safe, then any predecessor state $x(t)$ along a feasible trajectory is also safe. Consequently, if a state $x(t_{i+2})$ along the tree is a predecessor of a state $x(t_{i+4})$ which is shown to be safe, then $x(t_{i+2})$ does not have to be checked for safety.

The last approach does not have to execute additional safety checking during tree retainment. An argument can be made, however, that it actually checks a larger number of states for safety than the first alternative in this subsection. Furthermore, the further into the future, the less reliable the model of the world, especially in dynamic environments, so the less reliable the ICS-identification procedure.

VI. DISCUSSION

This report reviewed in detail methods from the literature for (i) identifying Inevitable Collision States (ICS) and (ii) properly integrating ICS-identification modules with replanning frameworks that employ sampling-based algorithms. The overall objective is to assist in the development of safe methods for sampling-based replanning that are also computationally efficient by (a) reducing the cost of identifying whether a state of a system is ICS and (b) minimizing the number of states that have to be checked for ICS during a replanning process. A step towards this direction will be the experimental comparison of many of the described techniques on a common set of benchmarks, in a similar fashion to recent work that compared the effectiveness of ICS-based tools versus reactive navigation schemes [38].

The tools described in this report can be used in many interesting applications where robots with non-trivial dynamics have to recompute their path on the fly, such as planning among unknown static obstacles, exploration of completely unknown environments and especially for planning in dynamic environments. While the discussion focused mostly on such single-agent challenges, some of the techniques discussed here have also been successfully applied in motion coordination problems for multi-robot teams [42]. It is also interesting to investigate the importance of ICS-based work in the context of pursuit-evasion games for systems with dynamics, where competing agents that can effectively reason about Inevitably Evading States will have an advantage over their opponent.

ACKNOWLEDGMENTS

Work on this report has been supported by NSF CNS 0932423.

REFERENCES

[1] J. Reif and M. Sharir, "Motion planning in the presence of moving obstacles," in *Proc. of the IEEE Int. Symp. on Foundations of Computer Science*, Portland, OR, 1985.
[2] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *IJRR*, vol. 20, no. 5, pp. 378–400, May 2001.
[3] T. Fraichard and H. Asama, "Inevitable collision states: A step towards safer robots?" *Advanced Robotics*, pp. 1001–1024, 2004.
[4] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *Int. Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, March 2002.
[5] A. M. Ladd and L. E. Kavraki, "Fast tree-based exploration of state space for robots with dynamics," in *Workshop on the Algorithmic Foundations of Robotcs (WAFR)*, 2005, pp. 297–312.
[6] J. Borenstein and Y. Korem, "The vector field histogram - fast obstacle avoidance for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, 1991.
[7] J. Minguez and L. Montano, "Nearness diagram (nd) navigation: Collision avoidance in troublesome scenarios," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 1, pp. 45–59, 2004.
[8] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, 1997.
[9] P. Brock and O. Khatib, "High-speed navigation using the global dynamic window approach," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Detroit, MI, USA, May 1999.
[10] M. Seder and I. Petrovic, "Dynamic window based approach to mobile robot motion control in the presence of moving obstacles," in *Proc. of the Int. Conf. on Robotics and Automation*, April 2007.
[11] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *Int. Journal of Robotics Research*, vol. 17, no. 7, 1998.
[12] F. Large, C. Laugier, and Z. Shiller, "Navigation among moving obstacles using the nlvo: Principles and applications to intelligent vehicles," *Autonomous Robots Journal*, vol. 19, no. 2, pp. 159–171, 2005.
[13] Y. Yang and O. Brock, "Elastic roadmaps: Globally task-consistent motion for autonomous mobile manipulation in dynamic environments," in *Proc. of Robotics: Science and Systems II*, 2006.
[14] A. Stentz, "The focussed d* algorithm for real-time replanning," in *International Joint Conference on Artificial Intelligence*, Montreal, Quebec, 1995, pp. 1652–1659.
[15] S. Koenig and M. Likhachev, "D* lite," in *Proceedings of the AAAI Conference of Artificial Intelligence (AAAI)*, 2002, pp. 476–483.
[16] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," vol. 12, no. 4, pp. 566–580, August 1996.
[17] P. Level and S. Hutchinson, "A framework for real-time path planning in changing environments," *Int. Journal of Robotics Research*, pp. 999–1030, 2002.
[18] J. Bruce and M. Veloso, "Real-time randomized path planning for robot navigation," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Lausanne, CH, October 2002.
[19] M. Kallman and M. Mataric, "Motion planning using dynamic roadmaps," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2004.
[20] J. Van den Berg, D. Ferguson, and J. J. Kuffner, "Anytime path planning and replanning in dynamic environments," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2006.
[21] D. Ferguson, N. Kalra, and A. Stentz, ""replanning with rrts,"," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, May 2006, pp. 1243–1248.
[22] M. Zucker, J. J. Kuffner, and M. Branicky, "Multi-partite rrts for rapid replanning in dynamic environments," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2007, pp. 1603–1609.
[23] R. Gayle, K. R. Klinger, and P. G. Xavier, "Lazy reconfiguration forest: An approach for planning with multiple tasks in dynamic environments," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Rome, Italy, April 10-14 2007, pp. 1316–1323.

[24] M. S. Wikman, M. Branicky, and W. S. Newman, “Reflexive collision avoidance: A generalized approach,” in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 1993.

[25] E. Frazzoli, M. Dahleh, and E. Feron, “Real-time motion planning for agile autonomous vehicles,” *AIAA Journal of Guidance, Control and Dynamics*, vol. 25, no. 1, pp. 116–129, 2002.

[26] R. Alami, T. Simeon, and K. Madhava Krishna, “On the influence of sensor capacities and environment dynamics onto collision-free motion plans,” in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Lausanne, CH, September-October 2002.

[27] J. Bruce and M. Veloso, “Real-time multi-robot motion planning with safe dynamics,” in *Proc. of the International Workshop on Multi-Robot Systems*, A. C. Schultz, L. E. Parker, and F. E. Schneider, Eds., 2003.

[28] K. E. Bekris and L. E. Kavraki, “Greedy but safe replanning under kinodynamic constraints,” in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Rome, Italy, April 2007.

[29] M. Kalisiak and M. Van de Panne, “Faster motion planning using learned local viability models,” in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Roma, Italy, May 2007.

[30] K. Tsianos and L. E. Kavraki, “Replanning: A powerful planning strategy for hard kinodynamic problems,” in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, September 2008.

[31] N. Chan, J. J. Kuffner, and M. Zucker, “Improved motion planning speed and safety using regions of inevitable collision,” in *17th CISM-IFToMM Symposium on Robot Design, Dynamics, and Control (RoManSy’08)*, July 2008.

[32] R. Vatcha and J. Xiao, “Perceived ct-space for motion planning in unknown and unpredictable environments,” in *Workshop on the Algorithmic Foundations of Robotcs (WAFR)*, Mexico, 2008.

[33] S. Petti and T. Fraichard, “Safe motion planning in dynamic environments,” in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Edmonton, AB, Canada, August 2005.

[34] R. Parthasarathi and T. Fraichard, “An inevitable collision state-checker for a car-like vehicle,” in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Roma, Italy, April 2007.

[35] T. Fraichard, “A short paper about motion safety,” in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Rome, Italy, 2007.

[36] L. Martinez-Gomez and T. Fraichard, “An efficient and generic 2d inevitable collision state-checker,” in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Nice, France, September 2008.

[37] ——, “Collision avoidance in dynamic environments: An ics-based solution and its comparative evaluation,” in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Kobe, Japan, May 2009.

[38] ——, “Benchmarking collision avoidance schemes for dynamic environments,” in *ICRA Workshop on Safe Navigation in Open and Dynamic Environments*, 2009.

[39] O. Gal and Z. Shiller, “Mapping obstacles to collision states for online motion planning in dynamic environments,” in *ICRA Workshop on Safe Navigation in Open and Dynamic Environment*, 2009.

[40] M. Kalisiak and M. Van de Panne, “Approximate safety enforcement using computed viability envelopes,” in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, New Orleans, LA, USA, April 2004.

[41] M. Zucker, “Approximating state-space obstacles for non-holonomic motion planning,” Robotics Institute, Carnegie Mellon University, Tech. Rep. CMU-RI-TR-06-27, May 2006.

[42] K. E. Bekris, K. Tsianos, and L. E. Kavraki, “Safe and distributed kinodynamic replanning for vehicular networks,” *Mobile Networks and Applications (MONET)*, vol. 14, no. 3, pp. 292–308, February 2009.