2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Inevitable Collision States: a Probabilistic Perspective

Antoine Bautin[†‡], Luis Martinez-Gomez[†] and Thierry Fraichard[†]

Abstract— **For its own safety, a robot system should never find itself in a state where there is no feasible trajectory to avoid collision with an obstacle. Such a state is an Inevitable Collision State (ICS). The ICS concept is particularly useful for navigation in dynamic environments because it takes into account the future behaviour of the moving objects. Accordingly it requires a model of the future evolution of the environment. In the real-world, the future trajectories of the obstacles are generally unknown and only estimates are available. This paper introduces a probabilistic formulation of the ICS concept which incorporates uncertainty in the model of the future trajectories of the obstacles. It also presents two novel probabilistic ICS-checking algorithms that are compared with their deterministic counterpart.**

Index Terms— **Motion safety; Collision avoidance; Uncertainty; Dynamic Environments; Inevitable Collision States.**

I. Introduction

A. Background and Motivations

Since the early days of mobile robotics the ability to move while avoiding collisions has been a major concern. Navigation (motion planning and obstacle avoidance) has evolved from simple models of the environment (static and known) to complex and realistic ones (dynamic and uncertain). Numerous approaches spanning the full range can be found in the literature. Motion planning techniques [1] are well suited for controlled and unchanging environments. Reactive methods [2]–[4] are popular due to their responsiveness to unforeseen objects in static and dynamic environments. Other methods such as [5] go a step further and explicitly reason about the future behaviour of the moving objects.

However, as analyzed in [6], safe navigation in dynamic environments is especially hard to achieve since it requires explicit reasoning about the *future behaviour* of the moving objects with an appropriate *lookahead*, *i.e.* the duration over which the future is considered, which is determined by the nature of both the moving objects and the robotic system at hand. Failure to do so yields navigation schemes with insufficient motion safety guarantees.

Assuming a model of the future is available (estimated for instance using motion prediction techniques such as [7] or [8]), the remaining (and not straightforward) issue is how to use this model in order to produce safe navigation strategies. To address this issue, the concept of *Inevitable Collision States* (ICS) developed in [9] (*aka* Obstacle Shadow [10] or Region of Inevitable Collision [11]) was proposed. An ICS for a robotic system is a state for which a collision eventually occurs no matter what the future trajectory of the system is. The formal characterization of the ICS is based upon the future behaviour of the moving objects with an infinite lookahead. Accordingly ICS can be used to ensure motion safety: for obvious safety reasons, a robotic system should never ever decide on a move that would take it to an ICS state. ICS have already been used in a number of applications: (i) mobile robot subject to sensing constraints, *i.e.* a limited field of view, and moving in a partially known static environment [9], (ii) car-like vehicle moving in a roadway-like environment [12], [13], (iii) spaceship moving in an asteroid field [11]. A generic ICS-checker, *i.e.* an algorithm that determines whether a given state is an ICS or not, was developed in [14] and later integrated within a provably safe collision avoidance scheme [15].

B. Contributions and Paper Outline

So far the characterization of the ICS has been based upon deterministic models of the future. In other words, each moving object was assumed to follow a given nominal trajectory (known *a priori* or predicted). Such deterministic models provide a clear-cut answer to the motion safety issue: a given state is an ICS or not (simple binary answer). However, such models are not well suited to capture the uncertainty that prevails in real world situations, in particular the uncertainty concerning the future behaviour of the moving objects.

The purpose of this paper is precisely to address this issue and to study to what extent the ICS concept can be extended to handle the uncertainty inherent to the future. Its primary contribution is a *probabilistic formulation of the ICS concept*. Probabilistic ICS permit the characterization of the motion safety likelihood of a given state, a likelihood that can later be used to design safe navigation strategies in real world situations. This is the first step towards the applicability of the ICS framework to real robots operating in uncertain dynamics environments. The paper also presents two *Probabilistic ICS-Checkers*, *i.e.* algorithms that determine the motion safety likelihood of a given state. The first is the probabilistic version of the ICS-Checker presented in [14]. The second is novel and more efficient.

The paper is organised as follows: first, §II recalls the key ICS properties and outlines a generic ICS-checking algorithm (in its deterministic form). Then, §III introduces a model of the future that captures the uncertainty about its future evolution. Afterwards, two novel probabilistic ICS-checking algorithms are presented in §IV. The results obtained from comparing the proposed probabilistic algorithms with their deterministic counterpart are presented in §V. Discussion and concluding remarks are made in §VI.

[†]INRIA, CNRS-LIG & Grenoble University, France.
[‡]Now at INRIA-LORIA - Henri Poincaré University, France.
This work has partially been supported by the European Commission contract "Have-It FP7-IST-2007-212154".

II. Inevitable Collision States

This section briefly summarizes the key ICS properties established in [9].

A. Notations

Let $\mathcal{A}$ denote a robotic system operating in a workspace $\mathcal{W}=\mathbb{R}^2$with a fixed Cartesian frame $\mathcal{F}_\mathcal{W}$. The dynamics of $\mathcal{A}$ is described by a state transition equation of the form:

$$\dot{s} = f(s, u) \tag{1}$$

where $s \in \mathcal{S}$ is the state of $\mathcal{A}$, $\dot{s}$ its time derivative and $u \in \mathcal{U}$ a control. $\mathcal{S}$ and $\mathcal{U}$ respectively denote the state space and the control space of $\mathcal{A}$. Let $s(t)$ denote $\mathcal{A}$'s state at time t and $\mathcal{A}(s)$ denote the closed subset of $\mathcal{W}$ occupied by $\mathcal{A}$ when in the state s. Let $\tilde{u} : [t_0, \infty(\longrightarrow \mathcal{U}$ denote a *control trajectory* (a time-sequence of controls), $\tilde{u}(t)$ denote the element of $\tilde{u}$ at time t. The set of all possible control trajectories over $[t_0, \infty($ is denoted $\tilde{\mathcal{U}}$. Abusing notation, let $\tilde{u}(s_0, t)$ denote the state reached by $\mathcal{A}$ at time t starting from an initial state $s_0 = s(t_0)$ while applying a control trajectory $\tilde{u}$ by integrating (1). The time-sequence of states is a *state trajectory*, a curve in $\mathcal{S} \times \mathcal{T}$ where $\mathcal{T}$denotes the time dimension.

The workspace $\mathcal{W}$ contains a set of n_b fixed and moving objects defined as closed subsets. Let $\mathcal{B}_i$ denote such an object, $i = 1, \ldots, n_b$. A Cartesian frame $\mathcal{F}_{\mathcal{B}_i}$ is attached to $\mathcal{B}_i$ so that each point in the object has fixed coordinates in $\mathcal{F}_{\mathcal{B}_i}$. $\mathcal{O}_{\mathcal{B}_i}$ denote the origin of $\mathcal{F}_{\mathcal{B}_i}$ and is called the reference point of $\mathcal{B}_i$. A configuration $\mathbf{q}^{\mathcal{B}_i} = (x, y, \theta)$ of $\mathcal{B}_i$ specifies the position and orientation of $\mathcal{F}_{\mathcal{B}_i}$ with respect to $\mathcal{F}_\mathcal{W}$. An arbitrarily selected configuration is called the reference configuration and denoted $\mathbf{q}_0^{\mathcal{B}_i}$. The subset of $\mathcal{W}$ occupied by $\mathcal{B}_i$ at configuration $\mathbf{q}^{\mathcal{B}_i}$ is denoted by $\mathcal{B}_i(\mathbf{q}^{\mathcal{B}_i})$. Similarly, $\mathcal{B}_i(t)$ is used to denote the subset of $\mathcal{W}$ occupied by $\mathcal{B}_i$ at a particular time t. Finally, $\mathcal{B}$ denotes the union of the workspace objects: $\mathcal{B} = \bigcup_{i=1}^{n_b} \mathcal{B}_i$.

B. ICS Definition

An ICS is informally defined as a state for which, no matter what the future trajectory followed by $\mathcal{A}$ is, a collision eventually occurs. Hence the following formal definition:

*Def. 1 (**Inevitable Collision State**):*

$$\text{ICS}(\mathcal{B}) = \{s \in \mathcal{S} | \forall \tilde{u} \in \tilde{\mathcal{U}}, \exists t, \mathcal{A}(\tilde{u}(s,t)) \cap \mathcal{B}(t) \neq \emptyset\} \tag{2}$$

Consequently, it is possible to define the set of ICS yielding a collision with a particular object $\mathcal{B}_i$:

$$\text{ICS}(\mathcal{B}_i) = \{s \in \mathcal{S} | \forall \tilde{u} \in \tilde{\mathcal{U}}, \exists t, \mathcal{A}(\tilde{u}(s,t)) \cap \mathcal{B}_i(t) \neq \emptyset\} \tag{3}$$

Likewise, the ICS set yielding a collision with $\mathcal{B}_i$ for a given trajectory $\tilde{u}$ (or a given set of trajectories $\mathcal{E} \subset \tilde{\mathcal{U}}$) is:

$$\text{ICS}(\mathcal{B}_i, \tilde{u}) = \{s \in \mathcal{S} | \exists t, \mathcal{A}(\tilde{u}(s,t)) \cap \mathcal{B}_i(t) \neq \emptyset\} \tag{4}$$

$$\text{ICS}(\mathcal{B}_i, \mathcal{E}) = \{s \in \mathcal{S} | \forall \tilde{u} \in \mathcal{E}, \exists t, \mathcal{A}(\tilde{u}(s,t)) \cap \mathcal{B}_i(t) \neq \emptyset\} \tag{5}$$

Finally, the ICS set yielding a collision with $\mathcal{B}_i$ for a given trajectory $\tilde{u}$ and time t is:

$$\text{ICS}(\mathcal{B}_i, \tilde{u}, t) = \{s \in \mathcal{S} | \mathcal{A}(\tilde{u}(s,t)) \cap \mathcal{B}_i(t) \neq \emptyset\} \tag{6}$$

C. ICS Properties

The first property shows that $\text{ICS}(\mathcal{B})$ can be derived from $\text{ICS}(\mathcal{B}_i, \tilde{u})$ for every object $\mathcal{B}_i$ and every possible future trajectory $\tilde{u}$.

*Property 1 (**ICS Characterisation** [9]):*

$$\text{ICS}(\mathcal{B}) = \bigcap_{\tilde{u} \in \tilde{\mathcal{U}}} \bigcup_{i=1}^{n_b} \text{ICS}(\mathcal{B}_i, \tilde{u}) \tag{7}$$

Complex systems having an infinite number of control inputs, the following property allows computation of a conservative approximation of $\text{ICS}(\mathcal{B})$ by using a subset only of the whole set of possible future trajectories.

*Property 2 (**ICS Approximation** [9]):*

$$\text{ICS}(\mathcal{B}) \subseteq \text{ICS}(\mathcal{B}, \mathcal{E})$$

with $\mathcal{E} \subset \tilde{\mathcal{U}}$, a subset of the whole set of possible future trajectories.

D. General ICS Checking Algorithm

Properties 1 and 2 provide the basis for a general ICS checking scheme. The steps involved in checking whether a given state s_c is an ICS or not are given in Algorithm 1. Besides the state to be checked, the algorithm takes as input the model of the environment, *i.e.* the list of the objects (fixed and moving) and their future behaviour.

Algorithm 1: General ICS Checking Algorithm.

```
Input: $s_c$, the state to be checked and $\mathcal{B}_i, i = 1 \cdots n_b$
Output: Boolean value

1 Select $\mathcal{E}$;
2 Compute $\text{ICS}(\mathcal{B}_i, \tilde{u}_j, t)$ for all t, every $\mathcal{B}_i$ and every
  $\tilde{u}_j \in \mathcal{E}$;
3 Compute $\text{ICS}(\mathcal{B}_i, \tilde{u}_j) = \bigcup_{t=0\ldots\infty} \text{ICS}(\mathcal{B}_i, \tilde{u}_j, t)$ for
  every $\mathcal{B}_i$ and every $\tilde{u}_j \in \mathcal{E}$;
4 Compute $\text{ICS}(\mathcal{B}, \tilde{u}_j) = \bigcup_{i=1\ldots n_b} \text{ICS}(\mathcal{B}_i, \tilde{u}_j)$ for every
  $\tilde{u}_j \in \mathcal{E}$;
5 Compute $\text{ICS}(\mathcal{B}, \mathcal{E}) = \bigcap_{\tilde{u}_j \in \mathcal{E}} \text{ICS}(\mathcal{B}, \tilde{u}_j)$;
6 Check whether $s_c \in \text{ICS}(\mathcal{B}, \mathcal{E})$, return TRUE or FALSE
  accordingly;
```

III. Modeling the future

ICS characterization has been defined so far with a deterministic model of the future in which the outcome is precisely known. In this paper we focus on a probabilistic model that assigns a probability measure to the obstacle's future trajectories to express the degree of belief that they will occur. Uncertainty is explicitly represented and can be properly handled with probability theory. A probabilistic model can be built using methods proposed in the literature. For example in [16] the model is obtained through an Extended Kalman Filter. Obstacle trajectories are forward simulated using the prediction step resulting in a distribution which is an estimate of the obstacle future position and uncertainty as a function of time. Other methods [17], [18] learn motion patterns from a set of observations which are later used to perform prediction of future motion.

For our purposes we assume that one such method was used and a probabilistic model of the future is readily available. In our general representation we define independent stochastic processes for each of $\mathcal{B}_i$'s configuration parameters. Thus we have time indexed independent random variables for the reference point coordinates and orientation of $\mathcal{B}_i$. The random variables have associated probability density functions $fx_t^{\mathcal{B}_i}(x)$, $fy_t^{\mathcal{B}_i}(y)$ and $f\theta_t^{\mathcal{B}_i}(\theta)$ which are used to obtain the probability that a point (x_w, y_w) in $\mathcal{W}$ is occupied by $\mathcal{B}_i$ at time t. We define the set of reference point coordinates that makes the shape of $\mathcal{B}_i$ overlap with (x_w, y_w) given an orientation θ as:

$$\mathcal{B}_i{}'(\theta) = (x_w, y_w) \ominus \mathcal{B}_i(\mathbf{q}_\theta^{\mathcal{B}_i}) \tag{8}$$

where $\ominus$ denotes the Minkowski difference and $\mathbf{q}_\theta^{\mathcal{B}_i}$ is $\mathcal{B}_i$'s reference configuration rotated by θ. Then, the probability of occupation for the point (x_w, y_w) is computed by integrating the coordinates probability density functions over $\mathcal{B}_i{}'(\theta)$ for all possible orientations and weighting the result of the integral by the probability of the orientation:

$$P_{occ[\mathcal{B}_i,t]}(x_w,y_w)=\int_0^{2\pi}\int_{\mathcal{B}_i{}'(\theta)} fx_t^{\mathcal{B}_i}(x) fy_t^{\mathcal{B}_i}(y) f\theta_t^{\mathcal{B}_i}(\theta)\,dx\,dy\,d\theta \tag{9}$$

IV. Probabilistic ICS

In this section we introduce the probabilistic computation of ICS which employs some of the relevant principles defined for its deterministic counterpart. Its fundamental difference is the provided model of the future. Having a probabilistic model implies that it is not possible to unerringly determine if a collision will occur or not in the future. Consequently, determining if a given state is an ICS can not have a binary answer. Instead, a probabilistic ICS can be defined as the probability of being in an ICS or equivalently as the minimum collision probability of all possible collision trajectories (if $\mathcal{A}$ has a trajectory with an almost zero collision probability then the probability that the state is an ICS is equally low).

*Def. 2 (**Probabilistic Inevitable Collision State**):*

$$P_{ICS}(s) = P(s \in \text{ICS}(\mathcal{B})) = \min_{\forall \tilde{u} \in \tilde{\mathcal{U}}} (P_{ICS[\tilde{u},\mathcal{B}]}(s)) \tag{10}$$

where $P_{ICS[\tilde{u},\mathcal{B}]}(s)$ is $\mathcal{A}$'s collision probability with $\mathcal{B}$ while applying a control trajectory $\tilde{u}$ from state s.

In the next sections we present two ICS-Checking algorithms that differ from each other in the way of computing $P_{ICS[\tilde{u},\mathcal{B}]}(s)$. The first one starts in $\mathcal{W}$ with the probability of occupation of obstacles and back-computes the probability in the space of interest $\mathcal{S}$. The second algorithm starts directly in $\mathcal{S}$ and forward-computes the probability using the required probabilities of occupation in $\mathcal{W}$. Both algorithms take as input the state to be checked s_c and the probabilistic model of the future as defined in §III.

A. Backward Probabilistic ICS-Check *Algorithm*

The Backward PICS-Check Algorithm is the probabilistic version of Algorithm 1. It can be summarized as first performing for each time t a probabilistic mapping between the workspace $\mathcal{W}$ and the state space $\mathcal{S}$. The probability of occupation $P_{occ[\mathcal{B}_i,t]}(x,y)$ is used to compute $P_{ICS[\mathcal{B}_i,\tilde{u}_j,t]}(s)$ which denotes the collision probability with $\mathcal{B}_i$ at time t for the state s given the control trajectory $\tilde{u}_j$. Next it computes a probabilistic union concerning time to obtain $P_{ICS[\mathcal{B}_i,\tilde{u}_j]}(s)$, *i.e.* the collision probability along all the duration of $\tilde{u}_j$. Afterwards a second probabilistic union is done to get a collision probability $P_{ICS[\tilde{u}_j]}(s)$ that considers all obstacles in the environment. Finally it repeats the previous steps for all control trajectories $\tilde{u}_j \in \mathcal{E}$ and assigns the probability of a state being an ICS as the minimum collision probability computed for all of them. The complete method is given in Algorithm 2 and detailed below.

Algorithm 2: Backward PICS-Check Algorithm

```
Input: $s_c$, the state to be checked, the probabilistic
       model of the environment $P_{occ[\mathcal{B}_i,t]}(x,y)$
Output: $P_{ICS}(s_c)$

1 Select $\mathcal{E}$;
2 Compute $P_{ICS[\mathcal{B}_i,\tilde{u}_j,t]}(s)$ for all $t$,every $\mathcal{B}_i$ and every
  $\tilde{u}_j \in \mathcal{E}$, $s \in \hat{\mathbf{z}}_\mathbf{c}$ (see text);
3 Compute $P_{ICS[\mathcal{B}_i,\tilde{u}_j]}(s) = \bigcup_t P_{ICS[\mathcal{B}_i,\tilde{u}_j,t]}(s)$ for
  every $\mathcal{B}_i$ and every $\tilde{u}_j \in \mathcal{E}$;
4 Compute $P_{ICS[\tilde{u}_j]}(s) = \bigcup_{i=1\cdots n_b} P_{ICS[\mathcal{B}_i,\tilde{u}_j]}(s)$ for
  every $\tilde{u}_j \in \mathcal{E}$;
5 Compute $P_{ICS}(s_c) = \min(P_{ICS[\tilde{u}_j]}(s_c))$;
6 return $P_{ICS}(s_c)$;
```

1) Computing $P_{ICS[\mathcal{B}_i,\tilde{u}_j,t]}(s)$: This step performs for a time t the mapping between the probability of occupation for an obstacle in $\mathcal{W}$ to the collision probability of a state in $\mathcal{S}$ given a control trajectory $\tilde{u}_j$. We start by defining the subset of points in $\mathcal{W}$ which have a non-negligible probability of occupation (a threshold p_{min} is introduced) for $\mathcal{B}_i$ at time t:

$$X_{\mathcal{B}_i} = \{\mathbf{x} \in \mathcal{W} | P_{occ[\mathcal{B}_i,t]}(\mathbf{x}) > p_{min}\} \tag{11}$$

We want to compute the image of $X_{\mathcal{B}_i}$ in a 2D slice of the state space $\mathcal{S}$ of $\mathcal{A}$ (see Fig. 1). For a planar robot, such slices can be obtained for an arbitrary state s by rewriting it as $s = (x, y, \hat{\mathbf{z}})$ with (x, y) the workspace coordinates of $\mathcal{A}$'s reference point, and $\hat{\mathbf{z}}$ the rest of the state parameters. The set of states that share the same values for the $\hat{\mathbf{z}}$ parameters is called the $\hat{\mathbf{z}}$-slice. We are interested first in the slice corresponding to $s_t = \tilde{u}_j(s_c, t)$, that is the state reached by $\mathcal{A}$ at time t starting from the initial state s_c (the state to be checked) while applying a control trajectory $\tilde{u}_j$. The image of $X_{\mathcal{B}_i}$ in the $\hat{\mathbf{z}}_\mathbf{t}$-slice is then defined as:

$$X_{\mathcal{B}_i}^{\hat{\mathbf{z}}_\mathbf{t}} = \{s \in \hat{\mathbf{z}}_\mathbf{t}\text{-slice} | \mathcal{A}(s) \cap X_{\mathcal{B}_i} \neq \emptyset\} \tag{12}$$

The state to be checked s_c does not belong to the $\hat{\mathbf{z}}_\mathbf{t}$-slice where our image $X_{\mathcal{B}_i}^{\hat{\mathbf{z}}_\mathbf{t}}$ is placed. Thus is necessary to back compute the set of states in $\hat{\mathbf{z}}_{\mathbf{c}}$-slice that reaches $X_{\mathcal{B}_i}^{\hat{\mathbf{z}}_{\mathbf{t}}}$ when following $\tilde{u}_j$. This can be done by finding the unique geometric transformation featuring both a translation and rotation that describe the motion of $\mathcal{A}$ in $\mathcal{W}$ when passing from s_c to s_t. Let $T_{\tilde{u}_j}(t)$ denote this transformation (it is a function of both $\tilde{u}_j$ and t): $\mathcal{A}(s_t) = T_{\tilde{u}_j}(t)\mathcal{A}(s_c)$ and conversely $\mathcal{A}(s_c) = T_{\tilde{u}_j}^{-1}(t)\mathcal{A}(s_t)$. Accordingly, we can define the image of $X_{\mathcal{B}_i}^{\hat{\mathbf{z}}_{\mathbf{t}}}$ in the $\hat{\mathbf{z}}_{\mathbf{c}}$-slice as:

$$X_{\mathcal{B}_i}^{\hat{\mathbf{z}}_{\mathbf{c}}} = \{s \in \hat{\mathbf{z}}_{\mathbf{c}}\text{-slice} | \mathcal{A}(s) = T_{\tilde{u}_j}^{-1}(t)\mathcal{A}(s_k), \forall s_k \in X_{\mathcal{B}_i}^{\hat{\mathbf{z}}_{\mathbf{t}}}\} \tag{13}$$

Finally we obtain the collision probability with $\mathcal{B}_i$ at time t for any state $s \in X_{\mathcal{B}_i}^{\hat{\mathbf{z}}_{\mathbf{c}}}$ given the control trajectory $\tilde{u}_j$ by:

$$P_{ICS[\mathcal{B}_i,\tilde{u}_j,t]}(s) = \max_{(x,y)\in\mathcal{A}(\tilde{u}_j(s,t))} \left(P_{occ[\mathcal{B}_i,t]}(x,y)\right) \tag{14}$$

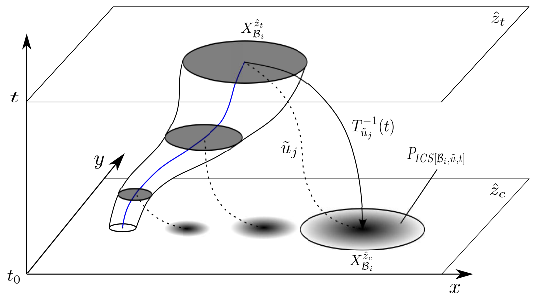

$X_{\mathcal{B}_i}^{\hat{z}_t}$
$\hat{z}_t$
t
$T_{\tilde{u}_j}^{-1}(t)$
y
$\tilde{u}_j$
$P_{ICS[\mathcal{B}_i,\tilde{u},t]}$
$\hat{z}_c$
t_0
$X_{\mathcal{B}_i}^{\hat{z}_c}$
x

Fig. 1: Backward PICS-CHECK:Computing $P_{ICS[\mathcal{B}_i,\tilde{u}_j,t]}(s)$.

2) Union of time - Computing $P_{ICS[\mathcal{B}_i,\tilde{u}_j]}(s)$: Having computed the collision probability for all time t, it is now possible to aggregate the probabilities all along the control trajectory:

$$P_{ICS[\mathcal{B}_i,\tilde{u}_j]}(s) = \bigcup_t P_{ICS[\mathcal{B}_i,\tilde{u}_j,t]}(s) \tag{15}$$

The probability at each time step is assumed to be independent of the other time steps, so the probability of having a collision free trajectory is the product of the probabilities of being collision free at each time step.

$$P_{ICS[\mathcal{B}_i,\tilde{u}_j]}(s) = 1 - \prod_{i=0}^{\infty}\left(1 - P_{ICS[\mathcal{B}_i,\tilde{u}_j,t]}(s)\right) \tag{16}$$

3) Union of obstacle - Computing $P_{ICS[\tilde{u}_j]}(s)$: To ensure safety, consideration of all the obstacles present in the environment is required, then:

$$P_{ICS[\tilde{u}_j]}(s) = \bigcup_{i=1..n_b} P_{ICS[\mathcal{B}_i,\tilde{u}_j]}(s) \tag{17}$$

Considering that obstacle trajectories are independent:

$$P_{ICS[\tilde{u}_j]}(s) = 1 - \prod_{i=1..n_b}\left(1 - P_{ICS[\mathcal{B}_i,\tilde{u}_j]}(s)\right) \tag{18}$$

4) Intersection of controls - Computing $P_{ICS}(s_c)$: Now is time to assign the probability of a state being an ICS given the collision probabilities associated with each control $\tilde{u}_j \in \mathcal{E}$. For deterministic ICS, a state is not an ICS if at least one control trajectory is collision free. Here we follow the same reasoning. We assign the probability as the minimum of the collision probabilities of the chosen control trajectories:

$$P_{ICS}(s) = \min_{\tilde{u}_j\in\mathcal{E}}\left(P_{ICS[\tilde{u}_j]}(s)\right) \tag{19}$$

An almost certainly collision free trajectory implies that the state is almost certainly not an ICS. The state is very likely an ICS when only high collision probabilities are present.

B. Forward Probabilistic ICS-CHECK *Algorithm*

This second PICS-CHECK Algorithm is a more efficient alternative to the backward version. It starts from the state of interest s_c in $\mathcal{S}$ and forward computes $P_{ICS[\tilde{u}_j,t]}(s)$ which denotes the collision probability of the system with all obstacles in the environment at time t. This is done only in the workspace area occupied by the system at the state $s_t = \tilde{u}_j(s_c,t)$ which reduces considerably the computation load. An arbitrary lookahead time t_H is set to perform a union over time which give as result $P_{ICS[\tilde{u}_j]}(s)$, the collision probability along the trajectory $\tilde{u}_j$. Finally, as in Algorithm 2, the minimum value among the different trajectories is set as the probability of the state being an ICS. The complete method is given in Algorithm 3 and the relevant steps are detailed below.

Algorithm 3: Forward PICS-CHECK Algorithm

```
Input: $s_c$, the state to be checked, the stochastic
       processes of each obstacle $P_{occ[\mathcal{B}_i,t]}(x,y)$
Output: $P_{ICS}(s_c)$

1 Select $\mathcal{E}$;
2 Compute $P_{ICS[\tilde{u}_j,t]}(s)$ for all $t$ and every $\tilde{u}_j \in \mathcal{E}$;
3 Compute $P_{ICS[\tilde{u}_j]}(s) = \bigcup_{t_0..t_H} P_{ICS[\tilde{u}_j,t]}(s)$ for every
  $\tilde{u}_j \in \mathcal{E}$;
4 Compute $P_{ICS}(s_c) = \min(P_{ICS[\tilde{u}_j]}(s_c))$;
5 return $P_{ICS}(s_c)$;
```

Computing $P_{ICS[\tilde{u}_j,t]}(s)$ The goal of this step is to compute the collision probability of a state at a given time with any obstacle in the environment. To consider all obstacles, we first compute the collision probability of a state with each obstacle at each time step. For a time t the collision probability in $\mathcal{S}$ given a control trajectory $\tilde{u}_j$ is equal to the maximum probability of occupation in the subset of $\mathcal{W}$ occupied by the system at the state $s_t = \tilde{u}_j(s_c,t)$:

$$P_{ICS[\mathcal{B}_i,\tilde{u}_j,t]}(s) = max\Big(P_{occ[\mathcal{B}_i,t]}(\mathcal{A}(s_t))\Big) \tag{20}$$

The probability of collision of the state s_t with all obstacles is obtained with the probabilistic union of the collision probabilities with each obstacle:

$$P_{ICS[\tilde{u}_j,t]}(s) = \bigcup_{i=1...n_b} P_{ICS[\mathcal{B}_i,\tilde{u}_j,t]}(s) \tag{21}$$

The next step merges the probability values for each time into a single one to obtain the collision probability value along the trajectory up to the lookahead time t_H. Then by repeating the procedure for all $\tilde{u}_j \in \mathcal{E}$ as shown in Figure 2 and selecting the minimum value of the collision probabilities we obtain the probability of the checked state s_c of being an ICS.

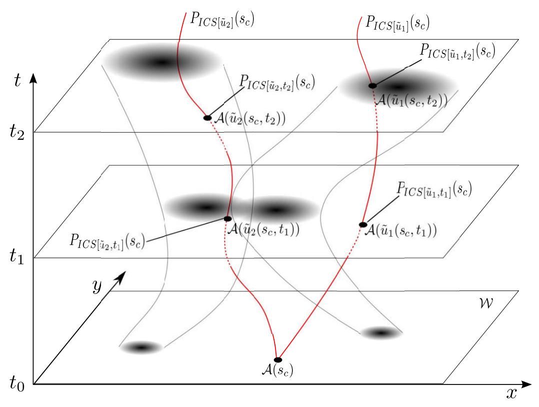

$P_{ICS[\tilde{u}_2]}(s_c)$
$P_{ICS[\tilde{u}_1]}(s_c)$
$P_{ICS[\tilde{u}_1,t_2]}(s_c)$
$P_{ICS[\tilde{u}_2,t_2]}(s_c)$
$\mathcal{A}(\tilde{u}_1(s_c,t_2))$
$\mathcal{A}(\tilde{u}_2(s_c,t_2))$
t
t_2
$P_{ICS[\tilde{u}_1,t_1]}(s_c)$
$P_{ICS[\tilde{u}_2,t_1]}(s_c)$
$\mathcal{A}(\tilde{u}_2(s_c,t_1))$
$\mathcal{A}(\tilde{u}_1(s_c,t_1))$
t_1
y
$\mathcal{W}$
$\mathcal{A}(s_c)$
t_0
x

Fig. 2: Forward PICS-CHECK Algorithm.

V. RESULTS

Algorithms 2 and 3 have been implemented in C++ and tested in a simulation environment. An illustrative example is presented here to show the results of both algorithms and compare them to the deterministic version of ICS-CHECK.

A. *Robotic System and Environment*

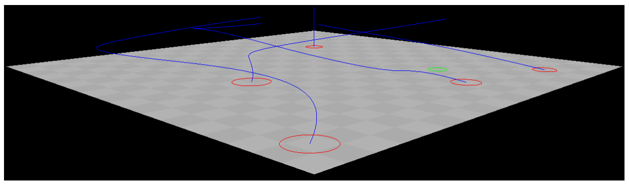
$\mathcal{A}$ is modeled as a disk with point mass non-dissipative dynamics moving in a closed 2D workspace $\mathcal{W}$ cluttered with disk-shaped fixed and moving objects. A total of five objects populate the environment (1 static and 4 moving).

B. *Model of the future*

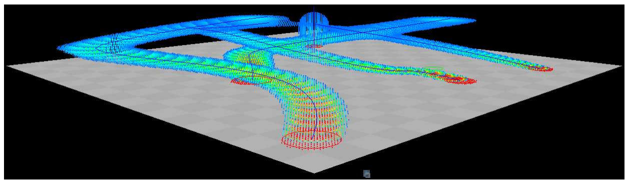
To validate the PICS-CHECK Algorithms we compare them with the result of a deterministic ICS-CHECK. The probabilistic algorithms model the future trajectories of the obstacles with stochastic processes. They are characterized by a mean and covariance functions. The mean function is set equal to the deterministic model of the future for comparison purposes. Uncertainty is added as Gaussian noise with an increasing variance value across time. Thus the random variables have normally distributed probability density functions at each sampling time. For example, the x parameter of the obstacle configuration has:

$$fx_t^{\mathcal{B}_i}(x) = \mathcal{N}(\mu x_t^{\mathcal{B}_i}, \sigma^2 x_t^{\mathcal{B}_i})$$

where $\mu x_t^{\mathcal{B}_i}$ is the mean value of the pdf of x at time t, set equal to $bx_t^{\mathcal{B}_i}$ and $\sigma^2 x_t^{\mathcal{B}_i}$ is its associated variance. Figure 3 illustrates the two models of the future.



(a) Deterministic: future obstacle's trajectories are known

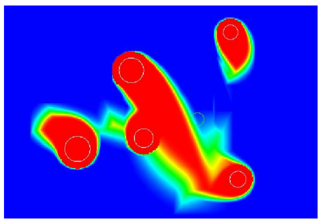


(b) Probabilistic: uncertainty grows with time

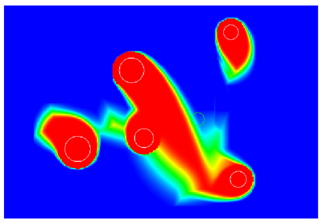
Fig. 3: Models of the future (time dimension pointing upwards).

C. *Validation*

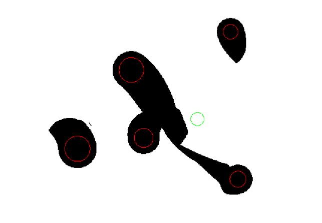
At the upper row of Figure 4 the result of the Backward and Forward PICS-CHECK Algorithms is presented. In the figures, the 2D $\hat{\mathbf{z}}_\mathbf{c}$-slice for the current state of the system is shown. The highest probability value is indicated by red and the lowest value with blue. For the same state slice the result of the deterministic ICS-CHECK is shown in the lower left figure. Black regions indicate the set of states that belongs to the ICS set. Both probabilistic algorithms have similar results and clearly encompass the result of the deterministic ICS as shown in the overlay of the bottom right figure.



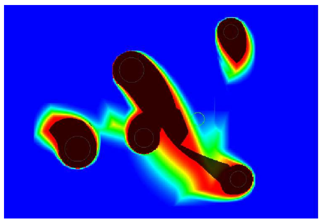
(a) Backward PICS-CHECK



(b) Forward PICS-CHECK



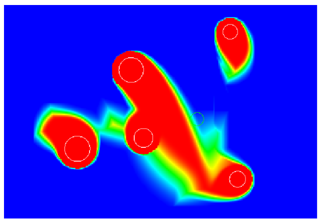
(c) Deterministic ICS-CHECK [14]

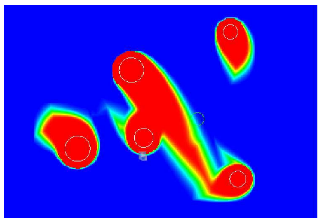


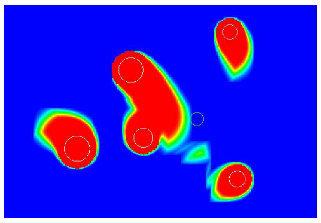
(d) Overlay

Fig. 4: Comparing PICS-CHECK with ICS-CHECK.

The backward and forward algorithms differ from each other mainly in the extent of the lookahead time used to reason about the future behavior of the obstacles. The forward algorithm explicitly set a limit in the lookahead value which impacts the obtained result as illustrated in Figure 5. Reducing the value has the effect of "shrinking" the states that have a high probability of being an ICS. A valid lookahead should consider the point in time where the probability of occupation in all the workspace reach a negligible value as if it has disappeared or exited the workspace. From this point no more relevant information can be used to decide about the safety of the system. The backward algorithm considers this with a minimum probability threshold at step 1.


(a) 20 s


(b) 10 s


(c) 5 s

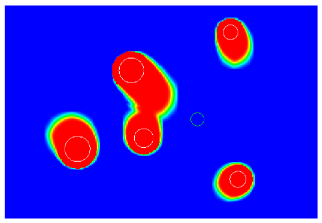

(d) 2.5 s

Fig. 5: Forward PICS-CHECK with decreasing lookahead time.

D. Performance

The algorithms performance depend on several parameters: number of obstacles, discretization (cell size), number of evasive maneuvers (i.e. the control trajectory subset used) and lookahead time. Figure 6 illustrates the effect on the computation time when changing two of those parameters. The main difference between the algorithms can be observed when changing the number of obstacles. This is explained by the fact that for the Backward version we do not know beforehand which obstacle will influence $P_{ICS}(s_c)$. Therefore it is needed to compute P_{ICS} for all states in $\hat{\mathbf{z}}_{\mathbf{c}}$ that lead to collision with an obstacle in the future before focusing on the state that we are checking. In contrast, the Forward version only computes the probability of collision in the states touched along the trajectories of the evasive maneuvers, greatly reducing the computation time.

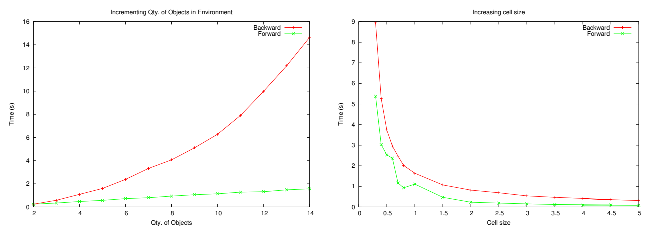


Fig. 6: Running time performances *wrt* the number of objects (left) and the cell size (right).

VI. CONCLUSION

We have presented a probabilistic approach for ICS which accounts for the uncertainty affecting the prediction of the future evolution of the moving objects. Two novel probabilistic ICS checking algorithms were introduced and compared with their deterministic counterpart. The results obtained with PICS-CHECK encompass those of ICS-CHECK demonstrating the ability of the probabilistic algorithms to incorporate the uncertainty of the model of the future. Examining both algorithms we found that the backward version offers a good theoretical answer with no limit in the lookahead time but is computationally expensive. On the other hand, the forward version is more efficient but requires the setting of a lookahead time horizon which directly impacts in the results. The next step of this work is the development of a navigation scheme which incorporates the results of probabilistic ICS. Safety will then be conditioned by the quality of the model.

REFERENCES

[1] J.-C. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[2] J. Borenstein and Y. Korem, "The vector field histogram — fast obstacle avoidance for mobile robts," *IEEE Trans. Robotics and Automation*, vol. 7, no. 3, June 1991.

[3] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, Mar. 1997.

[4] J. Minguez and L. Montano, "Nearness diagram (ND) navigation: collision avoidance in troublesome scenarios," *IEEE Trans. on Robotics and Automation*, vol. 20, no. 1, Feb. 2004.

[5] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *Int. Journal of Robotics Research*, vol. 17, no. 7, July 1998.

[6] T. Fraichard, "A short paper about motion safety," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Roma (IT), Apr. 2007.

[7] W. Hu, X. Xiao, Z. Fu, D. Xie, T. Tan, and S. Maybank, "A system for learning statistical motion patterns," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 28, no. 9, Sept. 2006.

[8] D. Vasquez, T. Fraichard, and C. Laugier, "Growing Hidden Markov Models: a Tool for Incremental Learning and Prediction of Motion," *Int. Journal of Robotics Research*, In Press.

[9] T. Fraichard and H. Asama, "Inevitable collision states. a step towards safer robots?" *Advanced Robotics*, vol. 18, no. 10, pp. 1001–1024, 2004.

[10] 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 (US), Oct. 1985.

[11] N. Chan, M. Zucker, and J. Kuffner, "Towards safe motion planning for dynamic systems using regions of inevitable collision," in *Proc. of the workshop on Collision-free Motion Planning for Dynamic Systems*, Rome (IT), Apr. 2007, workshop held in association with the IEEE Int. Conf. on Robotics and Automation.

[12] S. Petti and T. Fraichard, "Safe motion planning in dynamic environments," in *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, Edmonton, AB (CA), Aug. 2005.

[13] R. Benenson, S. Petti, T. Fraichard, and M. Parent, "Toward urban driverless vehicles," *Int. Journal of Vehicle Autonomous Systems*, vol. 6, no. 1/2, 2008.

[14] L. Martinez-Gomez and T. Fraichard, "An efficient and generic 2d inevitable collision state-checker," in *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, Nice (FR), Sept. 2008.

[15] ——, "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 (JP), May 2009.

[16] A. Kushleyev and M. Likhachev, "Time-bounded lattice for efficient planning in dynamic environments," *Robotics and Automation, 2009 IEEE International Conference on*, vol. 1, 2009.

[17] D. Vasquez, T. Fraichard, O. Aycard, and C. Laugier, "Intentional motion on-line learning and prediction," *Machine Vision and Applications*, vol. 19, pp. 411–425, 2008.

[18] C. Tay and C. Laugier, "Modelling smooth paths using gaussian processes," *Proc. of the Int. Conf. on Field and Service Robotics*, vol. 42, pp. 381–390, 2007.