2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

An Inevitable Collision State-Checker for a Car-Like Vehicle

Rishikesh Parthasarathi & Thierry Fraichard
Inria Rhône-Alpes & LIG-CNRS Lab., Grenoble (FR)

***Abstract*— An Inevitable Collision State (ICS) for a robotic system is a state for which, no matter what the future trajectory followed by the system is, a collision with an obstacle eventually occurs [1]. The ICS concept takes into account both the dynamics of the robotic system and the future motion of the moving objects of the environment. For obvious safety reasons, a robotic system should never ever end up in an ICS hence the interest of the ICS concept when it comes to safely drive robotic systems in dynamic environments. In theory, determining whether a given state is an ICS requires to check for collision *all possible future trajectories of infinite duration* that the robotic system can follow from this particular state! In practise, it is fortunately possible to build a conservative approximation of the ICS set by considering only a finite subset of the whole set of possible future trajectories. The primary contribution of the paper is a general principle to select the subset of trajectories based upon the concept of *imitating manoeuvres*, *ie* trajectories leading the robotic system to duplicate the behaviour of the environment objects (fixed or moving), it is shown how a good approximation of the ICS set can be obtained. The second contribution of the paper is an ICS-Checker for a car-like vehicle moving in a dynamic environment. This ICS-Checker integrates the above-mentioned selection principle. It is efficient and could be used in practise to compute truly safe motions for a car-like vehicle amidst moving objects.**

***Keywords*— Collision avoidance, dynamic environment**

I. Introduction

A. Background and Motivation

Today, mobile robotic systems are leaving the research laboratories. They are trying to operate in the real world and are also increasingly interacting with human beings. The characteristic feature of most real-world environments, irrespective of whether they are natural or man-made, structured or unstructured, hostile or friendly, is that they are *dynamic*, they feature moving objects (human beings, animals, vehicles, other robotic systems, *etc*). This raises an important question concerning the safety of the robotic systems and the environments in which these interact: how safe are these systems in dynamic environments? what guarantee is there that collisions will not happen? As soon as the size and dynamics of a robotic system makes it potentially harmful for itself or its environment, motion safety is critical (especially with human beings around).

Motion autonomy is a long standing issue in mobile robotics. Since Shakey's pioneering attempts at navigating around autonomously in the late sixties [2], the number and variety of autonomous navigation schemes that have been proposed is huge (*cf* [3]). From the motion determination perspective, these navigation architectures can be broadly classified into *deliberative* (*aka motion planning-based*) versus *reactive* approaches: deliberative approaches aim at computing a complete motion all the way to the goal using motion planning techniques, whereas reactive approaches determine the motion to execute during the next time-step only[1]. Deliberative approaches have to solve a motion planning problem [8]: they require a model of the environment as complete as possible and their intrinsic complexity is such that it may preclude their application in dynamic environments[2]. Reactive approaches on the other hand can operate on-line using local sensor information: they can be used in any kind of environment whether unknown, changing or dynamic. This accounts for the large number of reactive approaches that have been developed over the years, *eg* [9], [10], [11], [12], [13], [14], [15], *etc*. Most of today's reactive approaches however face a major challenge: as shown in [16], *motion safety in dynamic environments is not guaranteed* (in the sense that these robotic systems may end up in a situation where a collision inevitably occurs at some point in the future). Ref. [16] reaches this conclusion after introducing three motion safety criteria and establishing that all the autonomous navigation approaches currently used in real-world applications fail to satisfy them all (hence the collision risk). Ref. [16] also establishes that the concept of Inevitable Collision States (ICS) introduced in [1] does satisfy all three criteria.

An ICS for a robotic system is a state for which, no matter what the future trajectory followed by the system is, a collision with an object eventually occurs. For obvious safety reasons, a robotic system should never ever end up in an ICS. The ICS concept has already been used in a number of applications. The first one concerns the safe motion of a mobile robot subject to sensing constraints, *ie* a limited field of view, and moving in a partially known static environment [1]. The second one concerns a car-like vehicle moving in a roadway-like environment [7]. In both cases, the future motion of the robotic system at hand is guaranteed never to take the system in an ICS. To that end, an ICS-Checker is used: as the name suggests, it determines whether a given state is an ICS or not.

B. Contribution and Paper Outline

Although the ICS concept offers a theoretical answer to the motion safety issue, using it in practise raises a major

[1] In a few approaches, the motion is computed for a number, fixed or arbitrary, of time-steps [4], [5], [6], [7].

[2] Arguments about this issue can be found in [5] and [7].

problem: that of the characterisation of the set of ICS for a given robotic system. In theory, determining whether a given state is an ICS or not requires to check for collision *all possible future trajectories of infinite duration* that the robotic system can follow from this particular state!

In practise, it is fortunately possible to use the approximation property established in [1]. This property says that a conservative approximation of the ICS set of a given robotic system can be obtained by considering only a finite subset of the whole set of possible future trajectories (this property was used in [1] and [7]). What the approximation property does not say is how to select this subset. This is unfortunate because the quality of the approximation largely depends on the subset considered. If the approximation is too coarse, one might end up with most states being labelled as ICS (when in fact they are not).

The primary contribution of the paper is a general principle to select the subset of trajectories that can be used to determine whether a state is an ICS or not. By introducing the concept of *imitating manoeuvres*, *ie* trajectories leading the robotic system to duplicate the behaviour of the environment objects (fixed or moving), it is shown how a good approximation of the ICS set can be obtained.

The second contribution of the paper is an ICS-Checker for a car-like vehicle moving in a dynamic environment. It is an extension of the algorithm used in [1] that considered static environments only. This ICS-Checker integrates the above-mentioned selection principle. It is efficient (it has a polynomial complexity) and could be used in practise to compute truly safe motions for a car-like vehicle amidst moving objects.

The paper is organised as follows: first, section II recalls the definition and the properties fundamental to the ICS characterisation. Then, section III introduces the concept of imitating manoeuvres. Afterwards, section IV presents ICS-Checker in its general form. The instantiation of ICS-Checker to the case of a car-like vehicle moving in a dynamic environment is finally presented in section V.

II. Inevitable Collision States

The concept of *Inevitable Collision State* (ICS) was laid down and explored in [1]. This section merely recalls the definition of an ICS and its main characterising properties. The reader is referred to [1] for more details.

A. ICS Definition

Let $\mathcal{A}$ denote a robotic system. It is assumed that its dynamics can be described by a differential equation such as: $\dot{s} = f(s,u)$ 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 $\phi : [0,\infty[\longrightarrow \mathcal{U}$ denote a *control input*, *ie* a time-sequence of controls. Starting from an initial state s_0 (at time 0) and under the action of a control input ϕ, the state of $\mathcal{A}$ at time t is denoted by $\phi(s,t)$. ϕ equivalently represents a trajectory for $\mathcal{A}$. The set of possible control inputs is denoted by Φ, it represents the set of future trajectories that $\mathcal{A}$ can follow. Similarly, $\phi^{-1}(s_0,\ \mathrm{t})$ denotes the state s such that $\phi(s,t) = s_0$.

Let $\mathcal{ST}$ denote the state-time space of $\mathcal{A}$, *ie* its state space augmented of the time dimension [17]. A fixed or moving object in the workspace $\mathcal{W}$ of $\mathcal{A}$ yield a set of collision state-times denoted by $\mathcal{B}$. If $(s,t) \in \mathcal{B}$, it means that at time t a collision takes place between $\mathcal{A}$ and the corresponding object. Henceforth $\mathcal{B}(t)$ denotes the t-slice of $\mathcal{B}$, *ie* the state-times whose time coordinate is t.

Let us now recall the definition of an ICS and of its companion concept, the Inevitable Collision Obstacle (ICO):

*Def. 1 (**Inevitable Collision State**):* s is an ICS iff $\forall \phi \in \Phi, \exists t, \phi(s,t)$ is a collision state at time t.

*Def. 2 (**Inevitable Collision Obstacle**):* Given a set of forbidden state-times $\mathcal{B}$,

$$\mathrm{ICO}(\mathcal{B}) = \{s \in \mathcal{S} | \forall \phi, \exists t, \phi(s,t) \in \mathcal{B}\}$$

B. ICS Properties

Three properties established in [1] are now recalled:

*Property 1 (**Control Input Intersection**):*

$$\mathrm{ICO}(\mathcal{B}) = \bigcap_{\Phi} \mathrm{ICO}(\mathcal{B}, \phi)$$

Assuming now that $\mathcal{B} = \bigcup_i \mathcal{B}_i$,

*Property 2 (**Obstacles Union**):*

$$\mathrm{ICO}(\bigcup_i \mathcal{B}_i, \phi) = \bigcup_i \mathrm{ICO}(\mathcal{B}_i, \phi)$$

Finally,

*Property 3 (**ICO Characterisation**):*

$$\mathrm{ICO}(\mathcal{B}) = \bigcap_{\Phi} \bigcup_i \mathrm{ICO}(\mathcal{B}_i, \phi)$$

$\mathrm{ICO}(\mathcal{B})$ can obviously be derived from $\mathrm{ICO}(\mathcal{B}, \phi)$ for every possible control input ϕ. In general, complex systems have an infinite number of control inputs and hence the approximation property was introduced. This property is of practical interest since it permits to compute a conservative approximation of $\mathrm{ICO}(\mathcal{B})$ by using a subset only of the whole set of possible control inputs.

*Property 4 (**ICO Approximation**):* Let $\mathcal{I}$ denote a subset of the set of possible control inputs Φ,

$$\mathrm{ICO}(\mathcal{B}) \subset \bigcap_{\mathcal{I}} \mathrm{ICO}(\mathcal{B}, \phi)$$

C. Control Input Selection

The approximation property raises an important issue: what type of control inputs should be considered for the subset $\mathcal{I}$? This issue is important because the quality of the approximation largely depends on the subset considered. If the approximation is too coarse, one might end up with most states being labelled as ICS (when in fact they are not).

There is an intuitive answer to that problem: as per Def. 1, it appears that what characterise a state that is not an ICS is the existence of at least one control input yielding a collision-free trajectory. In this respect, the control inputs that are important should correspond to *evasive manoeuvres*, *ie* trajectories seeking to avoid collisions with the objects of the workspace. It is this principle that guided the determination of $\mathcal{I}$ in [1]. It considered a car-like vehicle subject to sensing constraints, *ie* a limited field of view, and moving in a partially known static environment. In this case, a straightforward evasive manoeuvre is to brake down and stop (while possibly steering to the left or to the right). Accordingly, $\mathcal{I}$ featured braking manoeuvres.

Now, in the presence of moving objects, what constitutes a good evasive manoeuvre? There is no straightforward answer to that question. However, the next section proposes a solution to this problem that turns out to be a logical extension to braking manoeuvres.

III. Evasive Manoeuvres and Moving Objects

A. Zero-Relative Velocity Paradigm

In a static environment, if $\mathcal{A}$ can perform a braking manoeuvre without any collision, its safety is guaranteed *forever*. It is argued that a braking manoeuvre is nothing but a control input that tries to achieve and maintain a zero-relative velocity *wrt* the static obstacles. It is obvious that two objects with zero-relative velocity will never collide in the future unless they are already in collision.

In a dynamic environment, braking manoeuvres are not so good from the safety point of view. Even if $\mathcal{A}$ can stop safely, it still can be hit by a moving object. This leads to the following question: what is the easiest way to escape a dynamic object? The solution to this question proposed herein lies in the extension of the idea of braking manoeuvres. A robotic system can escape a dynamic object if it can achieve and maintain a zero relative velocity *wrt* the object. This type of manoeuvre are called *imitating manoeuvres* as the system tries to imitate the moving object's behaviour. They are presented in the next section.

B. Imitating Manoeuvres

Given a moving object, the corresponding imitating manoeuvre (IM) is the control input leading $\mathcal{A}$ to imitate the moving object's future motion (in the workspace) so as to maintain a zero-relative velocity between them. IM exists provided that the dynamic properties of $\mathcal{A}$ and the moving object are similar. Henceforth it is assumed that they are equal.

Let us consider a moving object whose future motion is determined by the control input $\phi_{mo} : [0, \infty[\longrightarrow \mathcal{U}$. Henceforth $\mathcal{B}$ is used both to denote the moving object and the corresponding set of collision state-times.

Let us assume first that $\mathcal{A}$ is in a state with zero-relative velocity *wrt* $\mathcal{B}$. In this case, it can start imitating (in the workspace) the future motion of $\mathcal{B}$ right away (Fig.1-left). IM is exactly ϕ_{mo} and the following property can be established:

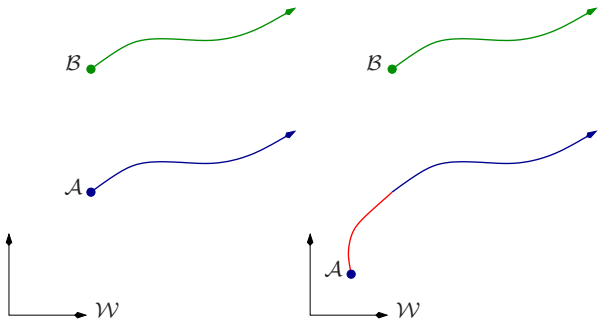

Fig. 1. $\mathcal{A}$ imitates $\mathcal{B}$'s behaviour: $\mathcal{A}$ can imitate right away (left); $\mathcal{A}$ must first "catch-up" (red part of the motion) before imitating (right).

Property 5: $\mathrm{ICO}(\mathcal{B}, \phi_{mo}) = \mathcal{B}(0)$

where $\mathrm{ICO}(\mathcal{B}, \phi_{mo})$ denote the set of ICS obtained when considering the control input ϕ_{mo} alone. This property establishes the fact that, unless $\mathcal{A}$ and $\mathcal{B}$ are already in collision at time 0, they will never collide provided that $\mathcal{A}$ executes the trajectory corresponding to ϕ_{mo}.

Proof:

$$\begin{aligned}
\mathrm{ICO}(\mathcal{B}, \phi_{mo}) &\overset{2}{=} \bigcup_t \mathrm{ICO}(\mathcal{B}(t), \phi_{mo}) \\
&\overset{2}{=} \bigcup_t \bigcup_{\mathcal{B}} \mathrm{ICO}(b(t), \phi_{mo}) \\
&= \bigcup_t \bigcup_{\mathcal{B}} \phi_{mo}^{-1}(b(t), t) \\
&= \bigcup_t \bigcup_{\mathcal{B}} b(0) = \mathcal{B}(0)
\end{aligned}$$

■

In general, $\mathcal{A}$ will not be in a state with zero-relative velocity *wrt* $\mathcal{B}$. Accordingly, $\mathcal{A}$ cannot start imitating $\mathcal{B}$ right away (for instance, it does not have the proper orientation or the proper velocity). In such a situation, IM comprises two parts (Fig.1-right):

- The "catch-up" part at the end of which $\mathcal{A}$ achieves a zero-relative velocity with $\mathcal{B}$.
- The "follow" part during which $\mathcal{A}$ duplicates $\mathcal{B}$'s control input.

As per property 5, if $\mathcal{A}$ can perform the catch-up trajectory without any collision, its safety *wrt* $\mathcal{B}$ is guaranteed forever. In this respect, IM are good candidates for the subset $\mathcal{I}$. Since an imitating manoeuvre is defined *wrt* to a given moving objects, there should be one imitating manoeuvre per moving objects. Finally, it can be noticed that a braking manoeuvre is just a special imitating manoeuvre: by braking down and stopping, $\mathcal{A}$ is simply imitating the behaviour of a fixed object (which is standing still). Unlike imitating manoeuvres, braking manoeuvres are not object-dependent.

IV. ICS Checking Algorithm

Using the ICO characterisation and approximation properties along with the principle guiding the choice of the

evasive manoeuvres, it is possible to design *ICS-Checker*, *ie* an generic algorithm whose purpose is to determine whether a given state is an ICS or not. ICS-Checker is a boolean function taking as input s, the state that is to be checked, and the current model of the environment given as a list of objects with corresponding future trajectories (null for fixed objects). The environment model is used to determine, for each object, the corresponding set of collision state-times, $\mathcal{B}_i$. The steps involved in determining if s is an ICS are:

1) To begin with, ICS-Checker determines the evasive manoeuvres that will compose $\mathcal{I}$, the set of evasive manoeuvres, namely:
   - A fixed number of braking manoeuvres.
   - One imitating manoeuvre per moving object.
2) Compute ICO($\mathcal{B}_i$, ϕ_j) for every object $\mathcal{B}_i$ and every evasive manoeuvre $\phi_j \in \mathcal{I}$.
3) Compute $\text{ICO}(\mathcal{B}, \phi_j) = \bigcup_i \text{ICO}(\mathcal{B}_i, \phi_j)$ for every object $\mathcal{B}_i$ (property 2).
4) Compute $\text{ICO}(\mathcal{B}) = \bigcap_j \text{ICO}(\mathcal{B}, \phi_j)$ for every evasive manoeuvre $\phi_j \in \mathcal{I}$ (property 1).
5) Determine whether $s \in \text{ICO}(\mathcal{B})$. If so return True otherwise return False.

The next section describes the instantiation of this algorithm to the particular case of a car-like vehicle moving in a dynamic environment.

V. Case Study: Car-Like Vehicle

Ref. [1] gave a characterisation of the ICS for a car-like vehicle moving among fixed objects only. The moving object case is addressed here.

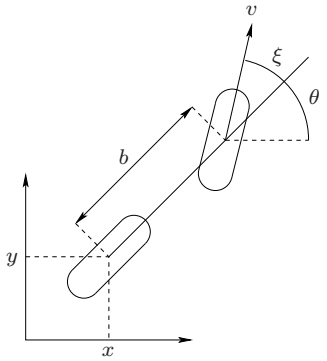

Fig. 2. The car-like vehicle $\mathcal{A}$ (bicycle model).

A. Model of the Car-Like Vehicle

Let us consider a point robot $\mathcal{A}$ that moves like a car-like vehicle and whose dynamics follows the bicycle model (Fig. 2). A state of $\mathcal{A}$ is defined by the 4-tuple $s = (x, y, \theta, v)$ where (x, y) are the coordinates of the rear wheel, θ is the main orientation of $\mathcal{A}$, and v is the linear velocity of the front wheel. A control of $\mathcal{A}$ is defined by the couple (u^ξ, u^v) where u^ξ is the steering angle and u^v the linear acceleration. The motion of $\mathcal{A}$ is governed by the differential equations:

$$\begin{cases} \dot{x} &= v \cos\theta \cos u^\xi \\ \dot{y} &= v \sin\theta \cos u^\xi \\ \dot{\theta} &= v \sin u^\xi / b \\ \dot{v} &= u^v \end{cases} \quad (1)$$

with $|u^\xi| \leq \xi_{\max}$ and $|u^v| \leq u^v_{\max}$. b is the wheelbase of $\mathcal{A}$.

$\mathcal{A}$ moves on a planar workspace $\mathcal{W}$ cluttered up with fixed and moving convex polygonal objects. It is assumed that the moving objects move with a constant linear velocity.

B. ICS-Checker Particulars

The efficiency of the ICS checking algorithm presented in [1] was obtained by computing the ICS corresponding to two-dimensional slices of the four-dimensional state space $\mathcal{S}$ of $\mathcal{A}$ (instead of attempting to perform computation in the full four-dimensional space). The slices considered were slices with fixed orientation and velocity. Should $s = (x, y, \theta, v)$ be the state to be checked, ICS-Checker would compute the ICS set of the θv-slice and then test if s belongs to the ICS set obtained.

As shown in [1], this approach yield an efficient ICS-Checker given that computing the ICS set of a θv-slice only requires to perform operations on two-dimensional generalised polygons (*eg* Minkowsky Sums, intersection and unions), operations that can be performed efficiently.

The same principle applies here and the following sections illustrates how to compute ICO($\mathcal{B}$, ϕ_j) for the different combinations of object and evasive manoeuvre types (so as to carry out step 3 of the ICS checking algorithm presented in §IV). Due to lack of space, the following sections merely outline how ICO($\mathcal{B}$, ϕ_j) is computed, the reader is referred to [18] for more details.

C. ICO (Fixed Object, Braking Manoeuvre)

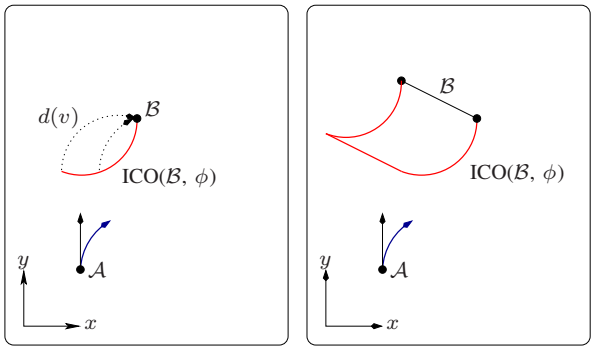

Fig. 3. Computing ICO($\mathcal{B}$, ϕ) for a fixed object and a braking manoeuvre.

The combination considered here is that of a fixed object and a braking manoeuvre. The set of braking manoeuvres considered here and selected for $\mathcal{I}$ comprises a discrete number of manoeuvres ϕ_b where $\mathcal{A}$ steers with a constant steering angle u^ξ while braking down until it stops.

Let us consider a point object $\mathcal{B}$ first. In this case, $\mathcal{A}$ eventually crashes into $\mathcal{B}$ iff it is on a collision course and its distance to $\mathcal{B}$ is less than its braking distance denoted by $d(v)$. Accordingly, $ICO(\mathcal{B}, \phi_b)$ is the circular arc of radius $b/\tan u^{\xi}$ and arc length $d(v)$ starting from $\mathcal{B}$ in the $-\theta$ direction (Fig. 3-left).

When $\mathcal{B}$ is a solid obstacle, property 2 is used. $ICO(\mathcal{B}, \phi_b)$ is the union of the $ICO(\mathcal{B}_i, \phi_b)$ for every point $\mathcal{B}_i$ of $\mathcal{B}$, *ie* the Minkowsky Sum between $\mathcal{B}$ and the circular arc computed earlier (Fig. 3-right).

D. ICO (Moving Object, Braking Manoeuvre)

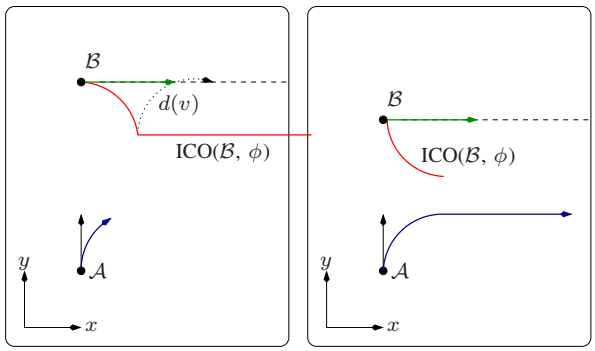

Fig. 4. Computing ICO($\mathcal{B}$, ϕ) for a moving object and a) a braking manoeuvre (left); b) an imitating manoeuvre (right).

The combination considered here is that of a moving object $\mathcal{B}$ and a braking manoeuvre ϕ_b. It is assumed that $\mathcal{B}$ is moving with a constant linear velocity. In this case, should $\mathcal{A}$ stops in the heading direction of $\mathcal{B}$, it will eventually be hit by it. Accordingly, ICO($\mathcal{B}$, ϕ_b) comprises two parts: a) a preliminary part containing the states from which $\mathcal{A}$ reaches a state occupied by $\mathcal{B}$ *at the same time instant*, and b) a half-line running parallel to $\mathcal{B}$'s path (Fig. 4-left).

E. ICO (Moving Object, Imitating Manoeuvre)

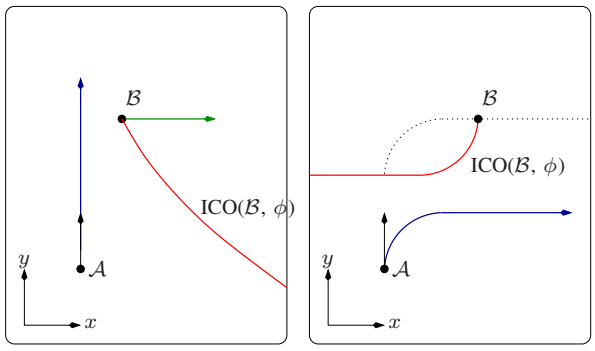

Fig. 5. Computing ICO($\mathcal{B}$, ϕ) for a) a moving object and an arbitrary imitating manoeuvre (left); and b) a fixed object and an arbitrary imitating manoeuvre (right).

The combination considered here is that of a moving object $\mathcal{B}$ and the corresponding imitating manoeuvre ϕ_i. As mentioned in §III, ϕ comprises two parts: a catch-up and a follow part. The purpose of the catch-up part is to change the orientation and the velocity of $\mathcal{A}$ so that they match that of $\mathcal{B}$. Once done, $\mathcal{A}$ move at constant linear velocity (that of $\mathcal{B}$). The catch-up part consists in turning to the left or to the right (depending on the respective orientations of $\mathcal{A}$ and $\mathcal{B}$) with maximum steering angle and with maximum acceleration/deceleration until $\mathcal{A}$ achieves zero-relative velocity *wrt* $\mathcal{B}$. Let us assume that $\mathcal{A}$ achieves zero-relative velocity *wrt* $\mathcal{B}$ at time t_c. As per property 5, ICO($\mathcal{B}$, ϕ_i) reduces to (Fig. 4-right):

$$ICO(\mathcal{B}, \phi_i) = \bigcup_{t \leq t_c} \phi_i^{-1}(\mathcal{B}(t), t).$$

That was for the case where the imitating manoeuvre was the one corresponding to the moving object considered. Since ICS-Checker has to compute the ICS set for every possible pair of object and evasive manoeuvre, it is necessary to consider the combination between a moving object $\mathcal{B}$ and an arbitrary imitating manoeuvre ϕ_e. In this case, $\mathcal{A}$ never achieves zero-relative velocity *wrt* $\mathcal{B}$ and ICO($\mathcal{B}$, ϕ_e) is an infinite curve originating at $\mathcal{B}(0)$(Fig. 5-left). It is defined as:

$$ICO(\mathcal{B}, \phi_e) = \bigcup_{t} \phi_e^{-1}(\mathcal{B}(t), t).$$

F. ICO (Fixed Object, Imitating Manoeuvre)

The last combination to be considered is that of a fixed object $\mathcal{B}$ and an arbitrary imitating manoeuvre ϕ_i. Computing ICO($\mathcal{B}$, ϕ_i) is straightforward in this case (Fig. 5-right).

G. Software Implementation

A prototype version of the algorithm proposed in §IV has been implemented in C++. Step 3 of the algorithm, *ie* computing ICO($\mathcal{B}$, ϕ_j) for the different combinations of object and evasive manoeuvre types, is carried out using the techniques presented above. Step 4 reduces to computing generalised polygon intersection. Fig. 6 depicts results of the ICS checking software: each snapshot shows the ICS set of a given (θ, v)-slice in two different environments: the dark regions are the objects, either fixed or moving (with a constant linear velocity).

VI. Conclusion

The concept of Inevitable Collision States (ICS) was introduced in [1] to answer the problem of safe motions (in dynamic environments in particular). In theory, determining whether a given state is an ICS requires to check for collision *all possible future trajectories of infinite duration* that the robotic system can follow from this particular state! In practise, it is fortunately possible to build a conservative approximation of the ICS set by considering only a finite subset of the whole set of possible future trajectories.

The paper has presented a general principle to select this subset of trajectories based upon the concept of *imitating manoeuvres*, *ie* trajectories leading the robotic system to duplicate the behaviour of the environment objects (fixed or moving), it has shown how a good approximation of the ICS set could be obtained. Then the paper has presented an

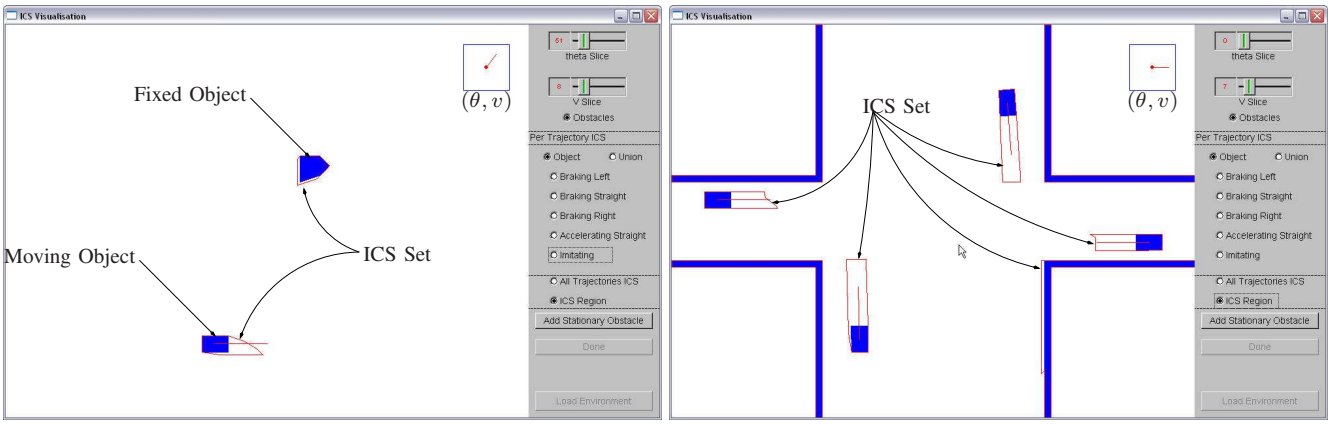

Fig. 6. Snapshots of the ICS checking software.

ICS-Checker for a car-like vehicle moving in a dynamic environment. This ICS-Checker integrates the above-mentioned selection principle. It is efficient and could be used in practise to compute truly safe motions for a car-like vehicle amidst moving objects.

Future works include integrating the ICS-Checker within the *Partial Motion Planner* navigation scheme of [7] and testing it on a real vehicle (such as the Cycab[3]). Another goal is to design an ICS-Checker for a more realistic car-like vehicle model, *ie* a model wherein the controls are the linear acceleration and the steering velocity (instead of the steering angle). Designing an ICS-Checker both generic, *ie* applicable to arbitrary robotic systems, and efficient remains a challenge worth being pursued.

REFERENCES

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

[2] N. J. Nilsson, "Shakey the robot," AI Center, SRI International, Menlo Park, CA (US), Technical note 323, Apr. 1984.

[3] I. R. Nourbaskhsh and R. Siegwart, *Introduction to Autonomous Mobile Robots*. MIT Press, 2004.

[4] I. Ulrich and J. Borenstein, "VFH*: Local obstacle avoidancd with look-ahead verification," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, San Francisco, CA (US), Apr. 2000, pp. 2505–2511.

[5] E. Frazzoli, M. A. 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.

[6] 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, Sept. 2005.

[7] 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.

[8] S. M. Lavalle, *Planning Algorithms*. Cambridge University Press, 2006.

[9] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. Journal of Robotics Research*, vol. 5, no. 1, 1986.

[10] J. Borenstein and Y. Korem, "The vector field histogram — fast obstacle avoidance for mobile robts," *IEEE Trans. Robotics and Automation*, vol. 7, no. 3, pp. 278–288, June 1991.

[11] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, pp. 23–33, Mar. 1997.

[12] N. Y. Ko and R. Simmons, "The lane-curvature method for local obstacle avoidance," in *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, Victoria, BC (CA), Oct. 1998, pp. 1615–1621.

[13] O. 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 (US), May 1999, pp. 341–346.

[14] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *Int. Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, July 1998.

[15] J. Minguez and L. Montano, "Nearness diagram (ND) navigation: collision avoidance in troublesome scenarios," *IEEE Trans. on Robotics and Automation*, vol. 20, no. 1, pp. 45–59, Feb. 2004.

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

[17] ——, "Trajectory planning in a dynamic workspace: a 'state-time space' approach," *Advanced Robotics*, vol. 13, no. 1, pp. 75–94, 1999.

[18] R. Parthasarathi, "Characterization of the inevitable collision states for a car-like vehicle," Master's thesis, Inst. Nat. Polytechnique de Grenoble, Grenoble (FR), June 2006.

[3] http://www-lara.inria.fr/cycaba