Collision Avoidance in Dynamic Environments: an ICS-Based Solution And Its Comparative Evaluation

Luis Martinez-Gomez, Thierry Fraichard

Collision Avoidance in Dynamic Environments: an ICS-Based Solution And Its Comparative Evaluation

Luis Martinez-Gomez[†] and Thierry Fraichard[†]

***Abstract*— This paper presents ICS-AVOID, a collision avoidance scheme based upon the concept of Inevitable Collision State (ICS), *ie* a state for which, no matter what the future trajectory of the robotic system is, a collision eventually occurs. By design, ICS-AVOID can handle dynamic environments since ICS do take into account the future behaviour of moving objects. ICS-AVOID is designed to keep the system away from ICS. By doing so, motion safety is guaranteed (by definition a robotic system in a non-ICS state has at least one collision-free trajectory that it can use). To demonstrate the efficiency of ICS-AVOID, it has been extensively compared with two state-of-the-art collision avoidance schemes: the first one is built upon the Dynamic Window approach and the second one on the Velocity Obstacle concept. The results obtained show that, when provided with the same amount of information about the future evolution of the environment, ICS-AVOID outperforms the other two schemes. The first reason for this has to do with the extent to which each collision avoidance scheme reasons about the future. The second reason has to do with the ability of each collision avoidance scheme to find a safe control if one exists. ICS-AVOID is the only one which is complete in this respect thanks to the concept of Safe Control Kernel.**

***Index Terms*— Motion safety; Collision Avoidance; Dynamic Environments; Inevitable Collision States, Velocity Obstacles, Dynamic Window.**

I. INTRODUCTION

A. Background and Motivations

Autonomous mobile robots/vehicles navigation has a long history by now. Remember Shakey's pioneering efforts in the late sixties [1]. Today, the situation has dramatically changed as illustrated rather brilliantly by the 2007 DARPA Urban Challenge[1]. The challenge called for autonomous car-like vehicles to drive 96 kilometers through an urban environment amidst other vehicles (11 self-driving and 50 human-driven). Six autonomous vehicles finished the race thus proving that autonomous urban driving could become a reality. Note however that, despite their strengths, the Urban Challenge vehicles have not yet met the challenge of fully autonomous urban driving (how about handling traffic lights or pedestrians for instance?).

Another point worth mentioning is that at least one collision took place between two competitors. This unfortunate mishap raises the important issue of *motion safety*, *ie* the ability for an autonomous robotic system to avoid collision with the objects of its environment. The size and the dynamics of the Urban Challenge vehicles make them potentially dangerous for themselves and their environment (especially when driving at high-speed). Therefore, before letting such autonomous systems transport around or move among people, it is vital to assert their ability to avoid collisions.

In the last forty years, the number and variety of autonomous navigation schemes that have been proposed is huge (*cf* [2]). In general, these navigation schemes intend to fulfill two key purposes: reach a goal and avoid collision with the objects of the environment. When it comes to collision avoidance, once again, many collision avoidance schemes have been proposed. Their aim of course is to ensure the robotic systems' safety. However, the analysis carried out in [3] of the most prominent navigation schemes (*ie* the ones currently used by robotics systems operating in real environments, *eg* [4]–[7]) shows that, especially in environments featuring moving objects, *motion safety is not guaranteed* (in the sense that collisions can occur even if they have full knowledge of the environment future evolution: no uncertainty or spurious information). As shown in [3], collision avoidance in dynamic environments is complex since it requires to explicitly reason about the *future behaviour* of the moving objects with a *time horizon*, *ie* the duration over which the future is taken into account, which is determined by the nature of both the moving objects and the robotic system at hand. Failure to do so yields collision avoidance schemes with insufficient motion safety guarantees.

B. Contributions

This paper draws upon the concept of *Inevitable Collision States* developed in [8] (*aka* Obstacle Shadow [9] or Region of Inevitable Collision [10], [11]) An Inevitable Collision State (ICS) for a robotic system is a state for which a collision eventually occurs no matter what the future trajectory of the system is. By design, the ICS concept can handle dynamic environments since it takes into account the future behaviour of the moving objects.

Accordingly, the paper presents an ICS-based collision avoidance scheme henceforth called ICS-AVOID. It is designed to keep the robotic system at hand away from ICS employing the knowledge provided by a given model of the future. By remaining in non-ICS states, ICS-AVOID provides a motion safety guarantee: by definition when a system is not in an ICS it has at least one collision-free trajectory that it can use. To verify whether the sate is an ICS or not, the ICS-Checker proposed in [12] is used.

To demonstrate the efficiency of ICS-AVOID, it has been extensively compared with two state-of-the-art collision avoidance schemes, both of which have been explicitly designed to handle dynamic environments. The first one comes

[†]INRIA, CNRS-LIG & Grenoble University, France.

[1]`http://www.darpa.mil/grandchallenge`.

from [13] and is henceforth called *Time-Varying Dynamic Window* (TVDW), it is a straightforward extension of the popular Dynamic Window approach [6]. The second one builds upon the concept of *Non Linear Velocity Obstacle* (NLVO) [14] which is a generalization of the Velocity Obstacle concept [7]. Like ICS-AVOID, both collision avoidance schemes make assumptions of the way the environment unfold in the future and therefore they are also susceptible to uncertainty. But if ICS-AVOID was provided with full knowledge about the future, it would guarantee motion safety no matter what. Of course, given the elusive nature of the future, this assumption is somewhat unrealistic. In practice, knowledge about the future is limited. However, the results obtained show that, when provided with the same amount of information about the future evolution of the environment, ICS-AVOID performs significantly better than the other two schemes. The first reason for this has to do with the respective time-horizon of each collision avoidance scheme thus emphasizing the fact that, reasoning about the future is not nearly enough, it must be done with an appropriate time horizon. The second reason has to do with the decision part of each collision avoidance scheme. In all cases, their operating principle is to first characterize forbidden regions in a given control space and then select an admissible control, *ie* one which is not forbidden. Accordingly motion safety also depends on the ability of the collision avoidance scheme at hand to find such admissible control. In the absence of a formal characterization of the forbidden regions, all schemes resort to sampling (with the inherent risk of missing the admissible regions). In contrast, ICS-AVOID through the concept of *Safe Control Kernel* is the only one for which it is guaranteed that, if an admissible control exists, it will be part of the sampling set.

C. Outline of the Paper

The paper is organised as follows: Section II recalls the definition and properties fundamental to the ICS characterization. Afterwards, Section III presents ICS-AVOID and introduces the Safe Control Kernel. An overview of TVDW and NLVO, the two collision avoidance schemes used for the comparative evaluation is done in Section IV. The case-study and the results obtained are presented in Section V. Discussion and concluding remarks are made in Section VI.

II. INEVITABLE COLLISION STATES

This section merely recalls the key ICS properties established in [8].

A. Notations

Let $\mathcal{A}$ denote a robotic system operating in a workspace $\mathcal{W}$=($\mathbb{R}^2$or $\mathbb{R}^3$). 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. Furthermore, 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, $i = 1, \cdots, n_b$. $\mathcal{B}$ denotes the union of the workspace objects: $\mathcal{B} = \bigcup_{i=1}^{n_b} \mathcal{B}_i$. Since an object $\mathcal{B}_i$ maybe moving, the notation $\mathcal{B}_i(t)$ is used to denote the subset of $\mathcal{W}$ occupied by $\mathcal{B}_i$ at a particular time t.

B. ICS Definition

An Inevitable Collision State 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**):*

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

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

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

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

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

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

C. ICS Properties

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

*Property 1 (**ICS Characterisation** [8]):*

$$\mathrm{ICS}(\mathcal{B}) = \bigcap_{\tilde{u} \in \tilde{\mathcal{U}}} \bigcup_{i=1}^{n_b} \mathrm{ICS}(\mathcal{B}_i, \tilde{u}) \tag{6}$$

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

*Property 2 (**ICS Approximation** [8]):*

$$\mathrm{ICS}(\mathcal{B}) \subseteq \mathrm{ICS}(\mathcal{B}, \mathcal{I})$$

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

III. ICS COLLISION AVOIDANCE SCHEME

For obvious safety reasons, a robotic system should never ever move to an ICS. Failing to do so will inevitably result in collision. If the system navigates through non-ICS states it has at least one collision-free control trajectory to apply (with the extent given by the model of the future which is used). However, employing this safety guarantee in practice is difficult due to the intrinsic complexity of ICS characterization. Nonetheless, a first step was taken in [12] where an efficient and generic ICS-Checker was presented. The goal of the ICS-Checker is to label a given state as ICS or not. Now we lay out ICS-AVOID, a reactive navigation approach that provides the mechanism to determine how to go from one safe state to the other. In this section we introduce the concept of *Safe Control Kernel* and detail the ICS-AVOID algorithm.

A. ICS-AVOID *Overview*

In general, collision avoidance schemes operate under the principle of characterizing forbidden regions in a given control space. Afterward, the system selects a valid command that it can execute safely. As it is impractical to perform a complete search in the whole control space, some sort of sampling is commonly used. The procedure usually relies on a randomize search or by discretizing the space with the risk of ending in a situation where a valid control could not be found at all. ICS-AVOID differs from others schemes by providing explicity a fallback mechanism in the entity of the Safe Control Kernel: if an admissible control exists, it will be part of the sampling set. To explain the kernel construction we recall the ICS Checking Algorithm from [12].

B. 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 is an ICS or not are:

ICS-CHECK

Input: $s, \mathcal{B}$
Output: $\mathcal{I}_s$, membership flag

1) Select $\mathcal{I}_s \subset \tilde{\mathcal{U}}$.
2) Compute $\text{ICS}(\mathcal{B}_i, \tilde{u}_j)$ for every object $\mathcal{B}_i$ and every manoeuvre $\tilde{u}_j \in \mathcal{I}_s$.
3) Compute $\text{ICS}(\mathcal{B}, \tilde{u}_j) = \bigcup_i \text{ICS}(\mathcal{B}_i, \tilde{u}_j)$ for every object $\mathcal{B}_i$.
4) Compute $\text{ICS}(\mathcal{B}, \mathcal{I}_s) = \bigcap_j \text{ICS}(\mathcal{B}, \tilde{u}_j)$ for every manoeuvre $\tilde{u}_j \in \mathcal{I}_s$.
5) Determine whether $s \in \text{ICS}(\mathcal{B}, \mathcal{I}_s)$. If so return ($\mathcal{I}_s$, True) otherwise return ($\mathcal{I}_s$, False).

The first issue raised by the previous algorithm concerns the determination of $\mathcal{I}_s$: what control trajectories should be considered? There is an intuitive answer to that question: as per Def. 1, it appears that what characterise a state that is not an ICS is the existence of at least one control trajectory yielding a collision-free trajectory. In this respect, the control trajectories that are important should correspond to *Evasive Manoeuvres* (EM), *ie* trajectories seeking to avoid collisions with the objects of the workspace. It was established in [15] that *Imitating Manoeuvres* (IM) are a good choice for $\mathcal{I}_s$. An IM for a given object $\mathcal{B}_i$ is the manoeuvre where the robotic system $\mathcal{A}$ tries to achieve and maintain a zero-relative velocity *wrt* $\mathcal{B}_i$. In other words, it tries to imitate the behaviour of $\mathcal{B}_i$ (doing so permits to avoid collision with it). The subset $\mathcal{I}_s$ could therefore include for example one IM per moving object, and a fixed number of braking manoeuvres[2].

C. Safe Control Kernel

The Safe Control Kernel is a finite subset of $\mathcal{U}$ which contains at least one control u that takes the robotic system from one non-ICS state to another. As the kernel is part of the sampling set used in ICS-AVOID, the robotic system can always relies on finding an admissible control to avoid collision. The kernel construction relies in the existence of a control trajectory that provides the transition between non-ICS states:

*Property 3 (**Safe Transition between non-ICS states**):* If $s_0 \notin \text{ICS}(\mathcal{B}, \mathcal{I})$ then $\exists \tilde{u}_j \in \mathcal{I}$ such that $s_i \notin \text{ICS}(\mathcal{B})$, $s_i = \tilde{u}_j(s_0, t_i)$.
Proof: Let $R_i = \{s \in \mathcal{S} | \forall \tilde{u} \in \mathcal{I}, \tilde{u}(s_0, t_i)\}$ be the set of reachable states at t_i using all the control trajectories in $\mathcal{I}$ and suppose $\forall s_i \in R_i$ are ICS($\mathcal{B}$) then $\forall \tilde{u} \in \mathcal{I}, \exists t, \mathcal{A}(\tilde{u}(s_0, t)) \cap \mathcal{B}(t) \neq \emptyset$ which implies $s_0 \in$ ICS($\mathcal{B}, \mathcal{I}$). Contradiction with $s_0 \notin$ ICS($\mathcal{B}, \mathcal{I}$).

If the current state s_i is not in $\text{ICS}(\mathcal{B}, \mathcal{I}_{s_i})$ then the collection of controls that define the Safe Control Kernel will correspond to the elements at time t_i of the evasive manoeuvers in $\mathcal{I}_{s_i}$.

*Def. 2 (**Safe Control Kernel**):* $K_i = \bigcup_j \tilde{u}_j(t_i)$ for every manoeuvre $\tilde{u}_j \in \mathcal{I}_{s_i}$.

D. ICS-AVOID *Algorithm*

At time t_i with a state $s_i \notin \text{ICS}(\mathcal{B}, \mathcal{I}_{s_i})$, ICS-AVOID operates in the next manner:

ICS-AVOID

Input: $s_i, \mathcal{I}_{s_i}$
Output: $u_{s_{i+1}}$

1) Compute Safe Control Kernel: $K_i = \bigcup_j \tilde{u}_j(t_i)$, $\tilde{u}_j \in \mathcal{I}_{s_i}$.
2) Include K_i into Control Space sampling set: $\mathcal{J} \subset \mathcal{U}$.
3) Select control $u \in \mathcal{J}$.
4) Compute $s_{i+1} = s_i + \int_{t_i}^{t_{i+1}} f(s_i, u) dt$.
5) If ICS-Check(s_{i+1})= True. Go to 3.
6) If ICS-Check(s_{i+1})= False. SUCCESS. Return u.

[2]Braking manoeuvres are a special case of IM where $\mathcal{A}$ imitates the behaviour of fixed objects by standing still.

IV. State-of-the-Art Collision Avoidance Schemes

As exposed in the introduction, the comparative evaluation of ICS-AVOID is focused on TVDW and NLVO. Both of them are extensions to popular collision avoidance schemes used in real-world applications: Dynamic Window (DW) and Velocity Obstacles (VO). DW has been demonstrated at relatively high speeds (up to $1\ m/s$) in complex environments with Minerva [16], Rhino [17] and Robox [18], robotic tour-guides that have operated for different time periods in different places in the United States, Germany and Switzerland. VO has been successfully tested with MAid [19], an automated wheelchair that navigated in the concourse of the central station in Ulm (DE) and during the German exhibition Hanover Fair'98.

A. Time Varying Dynamic Window

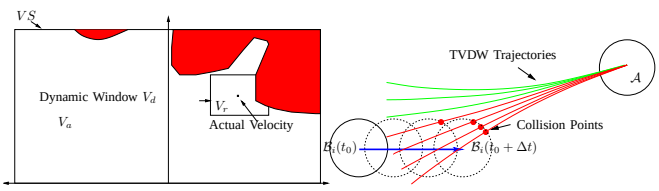
The Dynamic Window approach is a velocity space based local reactive avoidance scheme where search for admissible controls is carried out directly in the space of velocities (VS). The search space is reduced by the system kinematic and dynamic constraints to a set of reachable velocities (V_r) in a short time interval (Δt) around the current velocity vector (Fig.1a):

$$V_r = \{(v,\omega) | v \in [v_c - \dot{v_b}\Delta t, v_c + \dot{v_a}\Delta t] \wedge \omega \in [\omega_c - \dot{\omega_b}\Delta t, \omega_c + \dot{\omega_a}\Delta t]\} \quad (7)$$

where $\dot{v_a}$, $\dot{\omega_a}$, $\dot{v_b}$ and $\dot{\omega_b}$ are maximal translational/rotational accelerations and breakage decelerations. A velocity is admissible (V_a) if it allows the system to stop before hitting an object:

$$V_a = \{v,\omega \leq \sqrt{2\rho_{min}(v,\omega)\dot{v_b}} \wedge \sqrt{2\rho_{min}(v,\omega)\dot{\omega_b}}\} \quad (8)$$

An admissible velocity optimizing a given cost function is selected at each time step. This approach considers the objects in the environment as static. TVDW extends this scheme by calculating at each instant a set of immediate future obstacles trajectories in order to check for collision in the short term. In this respect TVDW is superior to DW because it reasons about the future behaviour of the obstacles. The extent of the look ahead time is set to equal the time it takes to the robotic system to stop, if no collision occurs in that time horizon the velocity is considered admissible



(a) Dynamic Window. (b) Time Varying Dynamic Window.

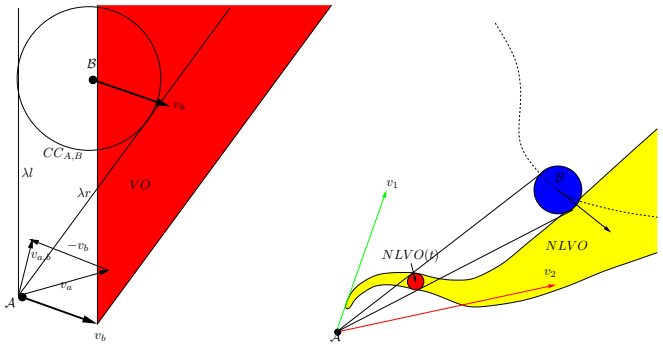
Fig. 1: Dynamic Window based approaches.

B. Non-Linear Velocity Obstacles

Velocity Obstacles is a reactive approach that also operates in the velocity space of the robotic system considered. VO takes into account the velocity of the moving objects (assumed to be moving with a constant linear velocity). Each object yields a set of forbidden velocities whose shape is that of a cone (*cf* Fig.2a depicts the linear velocity space of the robotic system, the red conical region on the right is the set of forbidden velocities that would yield a collision between the robot $\mathcal{A}$ and the moving object $\mathcal{B}$). Should the robotic system select a forbidden velocity, it would collide with the moving object at a later time (possibly infinite) in the future. In practice, velocities yielding a collision occurring after a given time horizon (TH) are considered as admissible. VO was extended by NLVO to consider known arbitrary velocity profiles for the moving objects. NLVO consist of all velocities of $\mathcal{A}$ at t_0 that would result in collision with $\mathcal{B}$ at any time $t_0 \leq t \leq TH$. Geometrically (Fig.2b), $NLVO(t)$ is a scaled $\mathcal{B}$, bounded by the cone formed between $\mathcal{A}$ and $\mathcal{B}(t)$, thus, NLVO is a warped cone with apex at $\mathcal{A}$ and formally defined as:

$$NLVO = \bigcup_{t_0 \leq t \leq TH} \frac{\mathcal{B}(t)}{t - t_0} \quad (9)$$

One issue (often overlooked) with the VO representation is that, in a closed environment, every velocity is forbidden since it eventually yield a collision. For that reason, both VO and NLVO requires a time horizon that cannot be arbitrarily large.



(a) Velocity Obstacles. (b) Non Linear Velocity Obstacles.

Fig. 2: Velocity Obstacles based approaches.

V. Case Study and Benchmarking

To assess the performance of ICS-AVOID a comparative evaluation was performed with the collision avoidance schemes just presented. A simulation environment capable of reproducing the same conditions for all the schemes was chosen to conduce the benchmarking. The robotic system, environment setup and implementation is discussed next.

A. Simulation Setup

1) Robotic System: Point Mass Model: Let $\mathcal{A}$ be modeled as a disk with point mass non-dissipative dynamics. A *state*

of $\mathcal{A}$ is defined as $s = (x, y, v_x, v_y)$ where (x, y) are the coordinates of the center of the disk and v_x, v_y are the axial components of the velocity. A control of $\mathcal{A}$ is defined by the pair (u_x, u_y) which denote the force exerted by the actuators along the x- and y-axis respectively. The motion of $\mathcal{A}$ is governed by the following differential equations:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{v_x} \\ \dot{v_y} \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} u_x + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u_y \qquad (10)$$

with a bound in the control given by the maximum acceleration: $\frac{u_x^2+u_y^2}{m^2} \leq a_{max}^2$ where m is the robot mass.

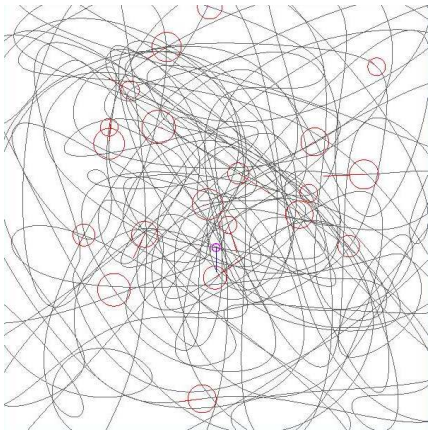


Fig. 3: Workspace example, 23 obstacles (represented by circles) along their trajectories defined by 10 random control knots B-Splines.

2) Workspace Model: $\mathcal{A}$ moves in a closed 2D workspace $\mathcal{W}$ (100 by 100 meters), cluttered up with disk-shaped moving objects (grown by the radius of $\mathcal{A}$). A total of twenty three objects move in a complex cyclic trajectories (closed B-splines with 10 random control knots). Knowledge about the future behaviour of the moving objects is provided until a fixed time horizon TH. The objects move with random constant speeds (between 1 to 10 m/s) along their trajectories. Figure 3 shows the trajectories of the objects to illustrate the complexity of the environment.

3) Implementation: The simulation environment and collision schemes were programmed entirely in C++ using OpenGL as rendering engine. The ICS-Checker presented in [12] was integrated to perform efficient state checking for ICS-AVOID. To achieve an identical reproduction of simulation conditions for each of the collision avoidance schemes in the benchmark, the random number generator was seeded with a set of identical numbers. The information about the future behaviour of the objects in the environment was made available to all the schemes with a limit of 1, 3 and 5 seconds into the future.

B. Collision Avoidance Scheme Benchmarking

The collision avoidance schemes were tested with a set of five runs with a duration of two minutes each. For each time horizon the number of collisions between $\mathcal{A}$ and the objects $\mathcal{B}_i$ are recorded in Table I.

TABLE I: Benchmarking of collision avoidance schemes.

Scheme	Run	Collisions TH=1(s)	Collisions TH=3(s)	Collisions TH=5(s)
TVDW	1	5	6	3
	2	12	4	4
	3	5	7	3
	4	12	2	4
	5	12	2	4
Average:		9.2	4.2	3.6
NLVO	1	10	2	0
	2	8	2	0
	3	12	2	0
	4	3	3	2
	5	7	2	2
Average:		8.0	2.2	0.8
ICS	1	7	0	0
	2	0	0	0
	3	1	0	0
	4	1	0	0
	5	1	0	0
Average:		2.0	0.0	0.0

TVDW (Fig. 4) performs poorly in comparison with the other two schemes. One of the main causes of failure is the limited extent in which the scheme use the information available about the future trajectories of the objects: it limits itself to a small fraction of the time at hand. NLVO (Fig. 5) exploits better the given information. It average less of one collision per run in the 5 second setup, nonetheless, it fails to guarantee the safety of the system when provided with less information. There is no clear guideline of how to set the value of the time horizon for NLVO: it depends largely on the environment and system at hand. Our experience signals that an arbitrary large value tends to reduce the admissible velocities to the point where is difficult to find a valid control. ICS-AVOID (Fig. 6) has the best performance in all the time horizon setups. ICS-AVOID is designed to reason in terms of infinite duration but even when dealing with minimal information about the future (1 second) it outperfomed the other two schemes. When given enough information (3 and 5 seconds) not a single collision occured, showing clearly that the safe control kernel allows the system to move from one safe state to the other.

VI. CONCLUSION

We have presented an ICS-based collision avoidance scheme designed to handle complex dynamic environments. ICS-AVOID takes into account the behaviour of the moving objects to reason about the future and thanks to the *Safe Control Kernel* provides a mechanism to find an admissible control to move safely from one non-ICS state to the other. The results obtained in a comparative evaluation with two state-of-the-art collision avoidance schemes show that, when provided with the same amount of information about the future evolution of the environment, ICS-AVOID outperforms them. The first reason for this has to do with the extent to which each collision avoidance scheme reasons about the

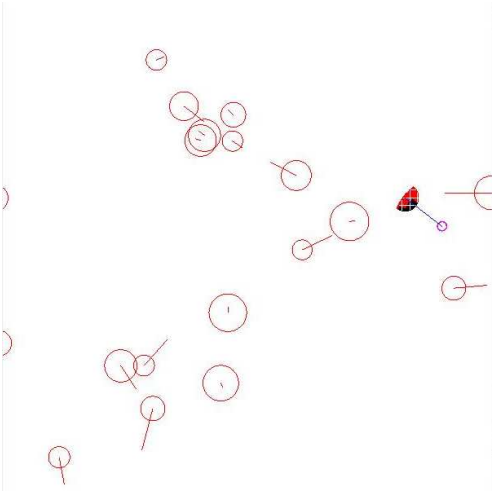


Fig. 4: TVDW. Admissible velocities (V_a) are represented in black, velocities in red are forbidden.

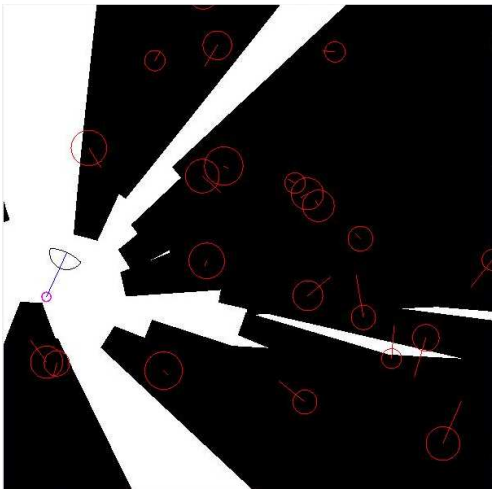


Fig. 5: NLVO. Black warped cones are forbidden velocities for the robotic system.

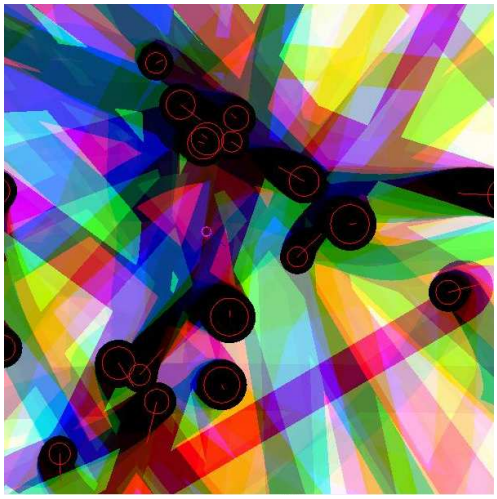


Fig. 6: ICS-AVOID. Black regions are forbidden states (ICS).

future. The second reason has to do with the ability of ICS-AVOID to find a safe control if one exists.

ACKNOWLEDGEMENTS

This work has been supported by the European Commission contract "Have-It FP7-IST-2007-212154".

REFERENCES

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

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

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

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

[5] 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.

[6] 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.

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

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

[9] 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.

[10] S. LaValle and J. Kuffner, "Randomized kinodynamic planning," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Detroit, MI (US), May 1999.

[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] 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.

[13] M. Seder and I. Petrovic, "Dynamic window based approach to mobile robot motion control in the presence of moving obstacles," *Proc. of the Int. Conf. on Robotics and Automation*, Apr. 2007.

[14] 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, September 2005.

[15] 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 (IT), Apr. 2007.

[16] S. Thrun, M. Bennewitz, W. Burgard, A. Cremers, F. Dellaert, D. Fox, D. Haehnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz, "Minerva: A second generation mobile tour-guide robot," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Detroit, MI (US), May 1999.

[17] W. Burgard, A. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, "Experiences with an interactive museum tour-guide robot," *Artificial Intelligence*, vol. 114, no. 1-2, 2000.

[18] R. Philippsen and R. Siegwart, "Smooth and efficient obstacle avoidance for a tour-guide robot," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Taiwan (TW), Sept. 2003.

[19] E. Prassler, J. Scholz, and P. Fiorini, "A robotic wheelchair for crowded public environments," *IEEE Robotics and Automation Magazine*, vol. 8, no. 1, pp. 38–45, Mar. 2001.