

Computing Unions of Inevitable Collision States and Increasing Safety to Unexpected Obstacles

Daniel Althoff¹, Christoph N. Brand², Dirk Wollherr³ and Martin Buss⁴
Institute of Automatic Control Engineering^{1,2,3,4}, Institute for Advanced Study³
Technische Universität München
D-80290 München, Germany
{da¹, dw³, mb⁴}@tum.de, brand_christoph@mytum.de²

Abstract—For reasoning about the safety of a robot system, it is sufficient to pretend the robot to reach an Inevitable Collision State (ICS). Otherwise, there exists no future trajectory which can avoid a collision. The usage of ICS is limited due to its computational complexity. One reason for this is, that the ICS computation cannot be done separately for each obstacle. Hence, ICS needs to be recomputed from scratch if another object appears in the scene. The main contribution of this paper is a modified ICS calculation which allows to compute the union of ICS sets in a sequential manner, thus reducing the computational requirements in case of new obstacles. Therefore, two novel ICS-Checker algorithms are presented reducing the computational effort. Furthermore, this novel calculation is used to reduce the probability of being in an ICS regarding an unforeseen obstacle.

I. INTRODUCTION

Recently, numerous projects for autonomous navigation have been carried out. Most are in the field of autonomous driving [1]–[3] or in the field of service robotics [4]–[7]. The general aim is to improve their navigation capabilities in order to operate in the daily life of humans. Due to their possible kinematic energy these systems are potentially harmful to humans and thus motion safety is a major issue. Over the years collision avoidance approaches have been widely studied, an overview is given in [8]. But most approaches are never evaluated according to their safety. They were used for navigation in dynamic environments like the Expo.02 [9], the Munich pedestrian zone [10] or the central station of Ulm [11]. However, no global motion safety guarantees are made for these approaches. They are only proven to be collision-free regarding static obstacles or for a finite time horizon. In [12] three criteria are pointed out for evaluating collision avoidance schemes with the result, that all common approaches do not satisfy all. Only the Inevitable Collision State (ICS) approach [13] satisfies all three criteria: A robotic system should consider *its own dynamics, the future behavior of workspace objects* and reason over an *infinite-time horizon*. Recent approaches use different implementations of the ICS concept [14]–[17]. All these approaches have one drawback, that the union of ICS sets is not equal to the ICS set of the union of obstacles. As a result, all ICS calculations are invalid if an obstacle appears or disappears in the workspace. The purpose of this paper is to address this problem.

A modification for the ICS calculation is presented allow-

ing to perform a valid union of ICS calculations. Based on this, two novel ICS-Checker algorithms are presented which are more computationally efficient than former approaches. Furthermore, based on the ICS concept it is shown, that a higher maneuverability of the robot decreases the probability of collision regarding unforeseen obstacles. For validation simulation results of the novel ICS-Checkers are presented.

The paper is organized as follows: first the definition of ICS and its properties are recalled from literature in Sec. II. Then the problem of computing the union of ICS sets is addressed and a solution is presented in Sec. III. Based on this, two novel ICS-Checker algorithms are presented in Sec. IV. In Sec. V the problem of motion safety regarding unexpected obstacles is presented and a possibility to decrease the probability of ending in an ICS is shown. The simulation results for validating the new approaches are presented in Sec. VI. Finally, a discussion and a conclusion of the results and the approaches are drawn in Sec. VII.

II. INEVITABLE COLLISION STATES

For defining an Inevitable Collision State, some notations have to be introduced. The state of the robot at time t is represented by $\mathbf{s}(t)$. The initial state is denoted as $\mathbf{s}(0)$. The state $\mathbf{s}(t)$ contains the position $\mathbf{x}(t)$ and the velocity $\mathbf{v}(t)$. The input $\mathbf{u}(t)$ and the state $\mathbf{s}(t)$ of the robot system can take values from the control space \mathcal{U} and the state space \mathcal{S} . For a given initial state $\mathbf{s}(0)$ and an input trajectory $\mathbf{u}(t)$, the dynamics of the robot is determined by the nonlinear differential equation $\dot{\mathbf{s}} = \mathbf{m}(\mathbf{s}, \mathbf{u})$. The workspace of the robot is denoted by \mathcal{W} and the subset of the workspace occupied by the robot is expressed as $\mathcal{A} \subset \mathcal{W}$. The occupancy of other objects in the workspace is denoted by \mathcal{B}_i and by $\mathcal{B}_i(t)$ if they are moving. The unified occupancy of all objects is written in short notation as $\mathcal{B} = \bigcup_{i=1, \dots, n_b} \mathcal{B}_i$ where n_b is the number of workspace objects. In order to distinguish complete input trajectories from values of inputs $\mathbf{u}(t)$, an input trajectory is denoted by $\tilde{\mathbf{u}}$ which maps the time t to the input space: $[0, \infty) \rightarrow \mathcal{U}$. The set of input trajectories is denoted by $\tilde{\mathcal{U}}$ and the workspace occupancy generated from the input trajectory is denoted by $\mathcal{A}(\tilde{\mathbf{u}}(t))$. In the following, the definition of Inevitable Collision State (ICS) [13] (*aka* Region of Inevitable Collisions (RIC) [18] or Obstacle Shadow [19]) is recalled.

Definition 1 (Inevitable Collision State):

The state \mathbf{s} is an ICS iff

$$\forall \tilde{u} \in \tilde{\mathcal{U}}, \exists t, \exists \mathcal{B}_i, \mathcal{A}(\tilde{u}(t)) \cap \mathcal{B}_i(t) \neq \emptyset.$$

Loosely speaking, the robot is in an inevitable collision state if there exists no input trajectory \tilde{u} which can avoid a collision with another workspace object.

In the following the two ICS properties presented in [13] are summarized. A conservative approximation of ICS can be calculated by using only a subset of all possible future trajectories.

Property 1 (ICS Approximation [13]):

$$\text{ICS}(\mathcal{B}, \tilde{\mathcal{U}}) \subseteq \text{ICS}(\mathcal{B}, \mathcal{I}), \text{ with } \mathcal{I} \subset \tilde{\mathcal{U}} \quad (1)$$

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

Property 2 (ICS Characterization [13]):

$$\text{ICS}(\mathcal{B}) = \bigcap_{\tilde{u} \in \tilde{\mathcal{U}}} \bigcup_{i=1}^{n_b} \text{ICS}(\mathcal{B}_i, \tilde{u}) \quad (2)$$

It is pointed out, that the union of ICS computations is *not* equal to the ICS computation of the union of obstacles.

Property 3 (ICS Union):

$$\text{ICS}(\mathcal{B}) \neq \bigcup_{i=1}^{n_b} \text{ICS}(\mathcal{B}_i) \quad (3)$$

This is a major drawback of ICS computation, since the computation must start from scratch, if a new obstacle appears. In the following section a modified computation of ICS is introduced, which allows to compute the union of ICS sets. Furthermore, this method is computationally more efficient.

III. UNION OF ICS SETS

As mentioned in (3), the union of ICS sets is not equal to the ICS set of the union of obstacles. The computation is modified, thus that the union of ICS sets can be efficiently computed in a sequential manner. We show, that the union of ICS can be computed by using only the reduced set of trajectories which are still collision-free. Therefore, we consider the case of two obstacles $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$. The union can be calculated as

$$\text{ICS}(\mathcal{B}_1 \cup \mathcal{B}_2, \tilde{\mathcal{U}}) = \text{ICS}(\mathcal{B}_1, \tilde{\mathcal{U}}) \cup \text{ICS}(\mathcal{B}_2, \tilde{\mathcal{U}}^a(\mathcal{B}_1)), \quad (4)$$

where $\tilde{\mathcal{U}}^a(\mathcal{B}_1) \subseteq \tilde{\mathcal{U}}$ are all admissible trajectories, which are not leading to a collision with \mathcal{B}_1

$$\tilde{\mathcal{U}}^a(s, \mathcal{B}_1) = \{\tilde{u} \in \tilde{\mathcal{U}} | \forall t, \mathcal{A}(\tilde{u}(s, t)) \cap \mathcal{B}_1 = \emptyset\}.$$

For the ICS calculation of \mathcal{B}_2 , only $\mathcal{U}^a(\mathcal{B}_1)$ needs to be considered. The proof is done for a single state \mathbf{s} .

According to Def. 1, each trajectory \tilde{u} needs to be checked for collision with all obstacles \mathcal{B} . If one trajectory \tilde{u} exists which is not colliding with one of the obstacles \mathcal{B} , the state is

not an ICS. Hence, it is sufficient to check if all trajectories collide with at least one obstacle. We want to show that

$$\text{ICS}(\mathbf{s}, \mathcal{B}, \tilde{\mathcal{U}}) = \text{ICS}(\mathbf{s}, \mathcal{B}_1, \tilde{\mathcal{U}}) \vee \text{ICS}(\mathbf{s}, \mathcal{B}_2, \tilde{\mathcal{U}}^a(\mathbf{s}, \mathcal{B}_1))$$

is true, which is the equivalent of (4) for just one state. We show that all possible trajectories $\tilde{\mathcal{U}}$ are checked for collision with all obstacles \mathcal{B} . The first term $\text{ICS}(\mathbf{s}, \mathcal{B}_1, \tilde{\mathcal{U}})$ determines which trajectories collide with the obstacles \mathcal{B}_1 and which are collision free, denoted as $\tilde{\mathcal{U}}^a(s, \mathcal{B}_1)$. Since it is sufficient that a trajectory collides with \mathcal{B}_1 or \mathcal{B}_2 only the set $\tilde{\mathcal{U}}^a(s, \mathcal{B}_1)$ needs to be considered for obstacle \mathcal{B}_2 . If there exists one trajectory of $\tilde{\mathcal{U}}^a(s, \mathcal{B}_1)$ which does not collide with \mathcal{B}_2 , the state \mathbf{s} is not an ICS according to Def. 1, because one trajectory \tilde{u} exists which does not collide with \mathcal{B}_1 or \mathcal{B}_2 , so

$$\mathcal{A}(\mathbf{s}, \tilde{u}) \cap \mathcal{B} = \emptyset.$$

Since $\tilde{\mathcal{U}}^a(s, \mathcal{B})$ is the set of trajectories which does not collide with \mathcal{B} for the state \mathbf{s} the same can be done for each possible state

$$\text{ICS}(\mathcal{B}, \tilde{\mathcal{U}}) = \bigcup_{\mathbf{s} \in \mathcal{S}} \text{ICS}(\mathbf{s}, \mathcal{B}, \tilde{\mathcal{U}}).$$

Hence, it is shown that (4) is valid. For the general case, the union is computed as

$$\text{ICS}(\mathcal{B}) = \bigcup_{i=1}^{n_b} \text{ICS}(\mathcal{B}_i, \tilde{\mathcal{U}}_i^a),$$

where

$$\tilde{\mathcal{U}}_i^a = \begin{cases} \tilde{\mathcal{U}}, & i = 1 \\ \tilde{\mathcal{U}}^a(\mathcal{B}_1, \dots, \mathcal{B}_{i-1}), & i \neq 1. \end{cases}$$

Additionally, we can show with ICS Prop. 1 that

$$\text{ICS}(\mathcal{B}_2) \subseteq \text{ICS}(\mathcal{B}_2, \tilde{\mathcal{U}}^a(\mathcal{B}_1))$$

since $\tilde{\mathcal{U}}^a(\mathcal{B}_1) \subset \tilde{\mathcal{U}}$. This property allows to calculate ICS in a sequential manner. Based on this property, two novel ICS-Checkers are presented in the next section.

IV. ICS-CHECKERS

The ICS Def. 1 is not implementable, for two reasons: There is an infinite number of input trajectories and an unlimited time horizon. The infinite number of input trajectories \tilde{u} of the robot, is approximated by computing a finite subset of input trajectories. According to ICS Prop. 1, this leads to a conservative computation of the ICS set.

The problem of computing with an infinite time horizon can be solved by applying only maneuvers that come to a standstill after a finite time horizon. Since the computational effort increases with time, the main focus lies on braking maneuvers which come to a standstill within a reasonable time horizon. So only a subset $\mathcal{I} \subset \tilde{\mathcal{U}}$ is used for the computation. In the following two ICS-Checker algorithms are presented. The first one considers only the trajectory set \mathcal{I}^a , which contains all admissible trajectories \tilde{u} , which are not colliding with the regarded obstacles \mathcal{B} . An overview of the sequential ICS-Checker is given in Alg. 1.

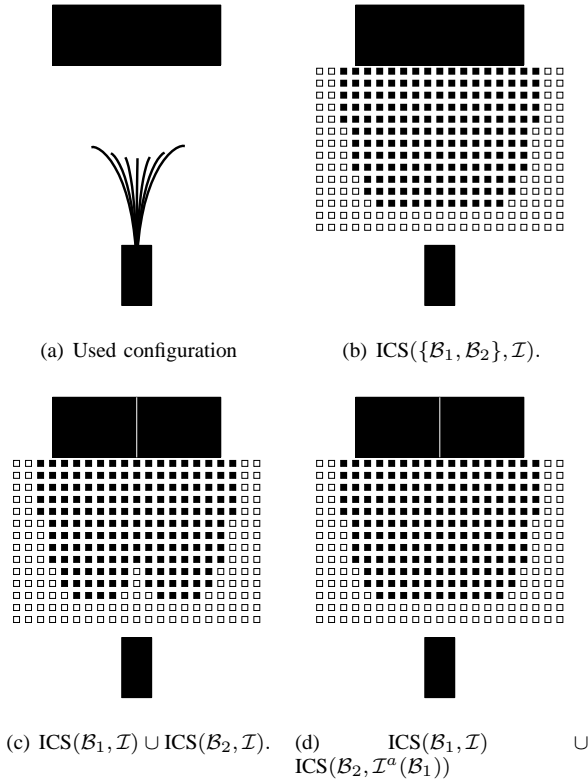


Fig. 1. Different ICS calculations for the same setup.

Additional to the ICS property of a state s , the algorithm also determines all trajectories of \mathcal{I} which are collision-free regarding all obstacles \mathcal{B} . The aim of the second ICS-

Algorithm 1: ICS Checker

Input : $s, \{\mathcal{B}_1, \dots, \mathcal{B}_{n_b}\}$
Output : ICS flag, \mathcal{I}^a

Initialize: Select $\mathcal{I} \subset \tilde{\mathcal{U}}, \mathcal{I}^a \leftarrow \mathcal{I}, \mathcal{B}^a \leftarrow \emptyset$

for $i \leftarrow 1$ **to** n_b **do**

$\mathcal{I}^a(s, \{\mathcal{B}_i, \mathcal{B}^a\}) \leftarrow \text{ICS}(s, \mathcal{B}_i, \mathcal{I}^a(s, \mathcal{B}^a))$

$\mathcal{B}^a = \{\mathcal{B}_i, \mathcal{B}^a\}$

if $\mathcal{I}^a(s, \mathcal{B}^a) = \emptyset$ **then**

return true, \emptyset

return false, \mathcal{I}^a

Checker is to determine as fast as possible the ICS status of the state s . Hence, it checks sequentially or parallel all trajectories \tilde{u} if one exists which is collision-free for all obstacles \mathcal{B} . If such a trajectory is found, the state s is not an ICS and the algorithm returns. Compared to the previous algorithm, only one trajectory is known to be collision-free and not the set \mathcal{I}^a . An overview of the ICS-Checker is given in Alg. 2.

Compared to all other known ICS implementations [14]–[17], these two are more efficient, since only collision-free trajectories are considered. In the following section, the sequential computation of ICS sets according to Sec.III

Algorithm 2: ICS Checker 2

Input : $s, \{\mathcal{B}_1, \dots, \mathcal{B}_{n_b}\}$

Output : ICS flag, \tilde{u}^a

Initialize: Select $\mathcal{I} \subset \tilde{\mathcal{U}}$

foreach $\tilde{u} \in \mathcal{I}$ **do**

 collfree = true

for $i \leftarrow 1$ **to** n_b **do**

if $\text{ICS}(s, \mathcal{B}_i, \tilde{u})$ **then**

 collfree = false

break

if collfree equal true **then**

return false, \tilde{u}

return true, \emptyset

is used to decrease the collision probability of unexpected obstacles.

V. INCREASING SAFETY TO UNEXPECTED OBSTACLES

When robots traverse through partially known environments, unforeseen obstacles can appear caused by imperfect perception capabilities. In [20] the robot velocity profile for a given path is adopted such that the robot can come to standstill before colliding with any mobile object possibly intercepting its future path. Therefore, the maximum possible velocity of the objects and the limited field of view of the robot is taken into account. In the following we show how to increase the safety of the robot assuming that an unforeseen obstacle appears.

The workspace contains the known obstacles \mathcal{B} and the current state of the robot system is $s_R \notin \text{ICS}(\mathcal{B})$. Additionally, one unexpected static obstacle \mathcal{B}_u exists, which is uniformly distributed in \mathcal{W} . Since one can only formulate a probability distribution for a random vector and not an occupancy set, $f(\cdot)$ represents the probability distribution of a point c of \mathcal{B}_u and the size of the object is considered by enlarging the occupancy \mathcal{A} of the robot system. The enlargement is performed by Minkowski addition of the area $(-\mathcal{B}_u + c)$ to \mathcal{A} , so that the new occupancy of the robot is $\mathcal{A}^b = \mathcal{A} \oplus (-\mathcal{B}_u + c)$ which is explained in more detail in [21]. The probability of collision at the time step t

$$P(C|\tilde{u}) := P(\mathcal{A}(s_R, \tilde{u}(t)) \cap \mathcal{B}_i \neq \emptyset) = \int_{\mathcal{A}^b(\tilde{u}(t))} f(x, t) dx$$

is equal for every time step, since $f(\cdot)$ is a uniform distribution. Thus, the collision probability for every trajectory is equal. We want to show that the probability of being in an ICS regarding the known obstacles and the unforeseen one can be reduced by increasing the maneuverability of the robot system. The robot has a finite set of trajectories \mathcal{I} and its maneuverability is proportional to the number of collision-free trajectories regarding the known obstacles.

Definition 2 (Maneuverability):

The maneuverability $\mathcal{M}(s, \mathcal{B})$ of a robot state s is defined

as

$$\mathcal{M}(\mathbf{s}, \mathcal{B}) = \frac{N_{\mathcal{B}}^a(\mathbf{s})}{N^a(\mathbf{s})}, \mathcal{M}(\mathbf{s}, \mathcal{B}) \in [0, 1]$$

where $N_{\mathcal{B}}^a(\mathbf{s})$ is the number of admissible trajectories regarding obstacles \mathcal{B} and $N^a(\mathbf{s})$ are the number of all trajectories in \mathcal{I} .

The robot system is not in an ICS, if there exists one trajectory \tilde{u} which does not collide either with \mathcal{B} or \mathcal{B}_u . According to (4) it is sufficient to calculate $\text{ICS}(\mathcal{B}_u, \mathcal{I}^a(\mathcal{B}))$ for determining the ICS status, because $\mathbf{s}_R \notin \text{ICS}(\mathcal{B})$. Since every trajectory $\tilde{u} \in \mathcal{I}^a(\mathcal{B})$ has the same probability to collide with the unexpected obstacle $P(C_{\mathcal{B}_u}|\tilde{u})$, the probability that the robot system is in an ICS is

$$P(\mathbf{s}_R \in \text{ICS}(\mathcal{B}_u, \mathcal{I}^a(\mathcal{B}))) = P(C_{\mathcal{B}_u}|\tilde{u})^{N_{\mathcal{B}}^a},$$

where $N_{\mathcal{B}}^a$ is the number of admissible trajectories of $\mathcal{I}^a(\mathcal{B})$. This is equivalent to the probability that all trajectories, which are not colliding with \mathcal{B} , collide with \mathcal{B}_u meaning that there exists no trajectory with is collision-free for \mathcal{B} and \mathcal{B}_u . The probability can be decreased by increasing the number of admissible trajectories $N_{\mathcal{B}}^a$ resulting in a higher maneuverability. Loosely speaking, if an unexpected obstacle appears in the workspace, the probability of being in an ICS can be reduced by increasing the maneuverability regarding the known obstacles. In Sec. VI simulation results confirm this theorem.

VI. SIMULATION

For evaluating the performance of the novel ICS-Checker algorithms and for validating the influence of the maneuverability to the collision probability of unexpected obstacles, a simulation setup as in [17] is used.

A. Simulation Setup

a) Robot Model: The state \mathbf{s} of the robot is represented by its position \mathbf{p} and its velocity \mathbf{v} . The dynamics of the robot is determined by the nonlinear differential equation $\dot{\mathbf{s}} = m(\mathbf{s}, \mathbf{u})$

$$\underbrace{\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{v}_x \\ \dot{v}_y \end{bmatrix}}_{\dot{\mathbf{s}}} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \underbrace{\begin{bmatrix} p_x \\ p_y \\ v_x \\ v_y \end{bmatrix}}_{\mathbf{s}} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ u_1 \\ u_2 \end{bmatrix}}_{\mathbf{u}} a_{\max} \quad (5)$$

with respect to the velocity constraint $\sqrt{v_x^2 + v_y^2} \leq v_{\max}$ and the acceleration constraint $\sqrt{a_x^2 + a_y^2} \leq a_{\max}$. For the simulation, $v_{\max} = 3 \frac{\text{m}}{\text{s}}$ and $a_{\max} = 2 \frac{\text{m}}{\text{s}^2}$ was used. The disc-shaped robot has a radius of 2 m.

b) Workspace Model: The workspace \mathcal{W} has a size of $100 \text{ m} \times 100 \text{ m}$ and is populated with 15 moving obstacles. Identical to [17] the obstacles trajectories are modeled as closed B-splines with 10 random control knots. The disc-shaped obstacles stir with a random constant velocity between $1 \rightarrow 2 \frac{\text{m}}{\text{s}}$ and their radius is 2 m. A snapshot of the simulation environment is depicted in Fig. 2.

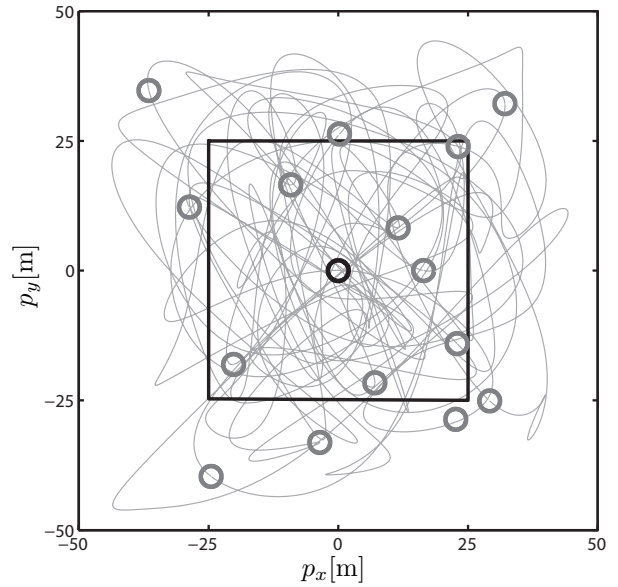


Fig. 2. Snapshot of workspace, gray circles depict the 15 objects and the gray lines depict the associated trajectories. The robot is shown as a black circle and the inner black square illustrates the workspace of the robot.

To prevent the robot to drive and stay at corner points, where usually no obstacle trajectories disturb the robot, the workspace of the robot \mathcal{W}_R is limited to $50 \text{ m} \times 50 \text{ m}$ and centrally positioned in \mathcal{W} .

Additional to the known workspace obstacles, 5 random static obstacles with radius 2 m are placed every 5 s at a random position in \mathcal{W}_R which has a minimum distance of 6 m to the robot to prevent instantaneous collisions.

c) Navigation Algorithm: The robot is applied with the ICS-Avoid algorithm from [17]. An overview is given in Alg. 3. The robot system has a fixed number of control inputs \mathcal{J} and the resulting states \mathbf{s} are checked one after another for $\mathbf{s} \in \text{ICS}(\mathcal{B})$. The trajectories are generated with the sampling time $T_s = 0.1 \text{ s}$ and the control inputs are constant for $T_c = 1.0 \text{ s}$. The set \mathcal{J} contains five different control inputs $[u_1, u_2] = \{[0, 0], [1, 0], [-1, 0], [0, 1], [0, -1]\}$. The set of collision-free trajectories \mathcal{K} of the ICS-Checker is defined as the *Safe Control Kernel*. For the next time step, the Safe Control Kernel is added to the set of control inputs \mathcal{J} . Thus, ICS-Avoid can guarantee a fallback mechanism due to the Safe Control Kernel.

Algorithm 3: ICS-Avoid [17]

Input : $\mathbf{s}(t), \{\mathcal{B}_1, \dots, \mathcal{B}_{n_b}\}, \mathcal{I}_{\mathbf{s}(t)}$

Output : $\mathbf{u}_{\mathbf{s}(t+T_c)}$

Initialize: Compute Safe Control Kernel:

$$K(t) = \bigcup_j \tilde{u}_j(t), \tilde{u}_j \in \mathcal{I}_{\mathbf{s}(t)}$$

foreach $\mathbf{u} \in \mathcal{J}$ **do**

$$\mathbf{s}(t+T_c) = \mathbf{s}(t) + \int_t^{t+T_c} m(\mathbf{s}(t), \mathbf{u}) dt$$

if *ICS-Check*($\mathbf{s}(t+T_c)$) = *false* **then**

return \mathbf{u}

As shown in Sec. V it is possible to decrease the probability of being in an ICS regarding an unexpected obstacle by increasing the robot maneuverability. Therefore, the ICS-Avoid algorithm is extended so that the maneuverability of the robot is taken into account. The algorithm is shown in Alg. 4. The robot chooses the control input $\mathbf{u} \in \mathcal{J}$ which maximizes its maneuverability, thus increasing its motion safety.

Algorithm 4: ICS-Avoid including maneuverability

Input : $\mathbf{s}(t), \{\mathcal{B}_1, \dots, \mathcal{B}_{n_b}\}, \mathcal{I}_{\mathbf{s}(t)}$

Output : $\mathbf{u}_{\mathbf{s}(t+T_c)}$

Initialize: Compute Safe Control Kernel:

$$K(t) = \bigcup_j \tilde{u}_j(t), \tilde{u}_j \in \mathcal{I}_{\mathbf{s}(t)}$$

foreach $\mathbf{u} \in \mathcal{J}$ **do**

$\mathbf{s}(t + T_c) = \mathbf{s}(t) + \int_t^{t+T_c} m(\mathbf{s}(t), u) dt$
if *ICS-Check*($\mathbf{s}(t + T_c)$) = *false* **then**
└ Compute Maneuverability $M(u)$

return $\arg \max_u M(u)$

B. ICS-Checkers

While the robot is navigating in the workspace by applying ICS-Avoid, the performance of three different ICS-Checkers are evaluated: ICS-Checker by [16], ICS-Checker from Alg. 1 and ICS-Checker from Alg. 2. All algorithms use the same set of trajectories, which contains only pure braking trajectories. The braking trajectories are generated based on a constant deceleration in the robot coordinate frame. As presented in [22], the robot comes to a standstill within a finite time horizon, if the direction of the resulting acceleration vector is chosen from the interval $(\frac{3}{4}\pi, \frac{5}{4}\pi)$ and its magnitude is greater than zero. For this evaluation, 7 directions are generated with an equidistant step size of 0.2 beginning at $\frac{3}{4}\pi$. The corresponding magnitude is chosen, such that the duration of the longest braking time lasts not longer than 5 s.

The workspace was populated by 20 known obstacles and their trajectories are known for a time horizon of 5 s. For comparison, the mean number of trajectory checks \bar{N}_C and the mean computation times are listed in Tab. I for 1654 ICS calculations. The standard ICS-Checker from [16] needs to evaluate 7 braking trajectories for all 20 workspace objects. Hence, 140 checks are needed to determine the ICS property of one state. The ICS-Checker from Alg. 1 needs 46.69% less collision checks compared to the ICS-Checker presented in [16], despite both algorithms have the same result containing the ICS-flag and the set of admissible trajectories. The ICS-Checker Alg. 2 needs even less collision checks, since only the ICS flag is computed. In the following section, random workspaces are generated to evaluate the two different ICS-Avoid algorithms according to their motion safety.

TABLE I
ICS-CHECKER PERFORMANCE

Algorithm	Checks		Mean 10^{-3} [s]
	\bar{N}_C	Δ to [16] [%]	
ICS-Checker [16]	140.00	-	17.93
ICS-Checker Alg. 1	74.64	46.69	10.37
ICS-Checker Alg. 2	58.15	58.46	7.74

C. Motion Safety

For validating the concept of Sec. V, improving safety by increasing maneuverability of the robot, the two different ICS-Avoid algorithms are evaluated in the same simulation environment. Therefore, 5 different random workspaces (runs) are generated which are identical for both algorithms. As presented in [16], the future trajectories of the objects is only provided for a fixed time horizon T_H . Three different time horizons are considered: 1, 3 and 5 s. Each run was evaluated according to the collisions regarding the known obstacles \mathcal{B} , while ignoring the unexpected obstacles \mathcal{B}_u , and according to the total number of collisions regarding all obstacles $\mathcal{B} \cup \mathcal{B}_u$. The number of collisions considering only the unexpected obstacles \mathcal{B}_u are not discussed, since the robot may collide with a known obstacle while avoiding an unknown one. Furthermore, the maneuverability is determined for each run and the simulation results are summarized in Tab. II.

The ICS-Avoid algorithm maximizing the maneuverability of the robot reduces the average number of collisions compared to the other one. Regarding the known obstacles \mathcal{B} , the average number of collision for Alg. 3 is 9.80 and for Alg. 4 is 6.80 which is a relative difference of 30.61%. The difference for the unknown obstacles \mathcal{B}_u is more significant. The average number of collision for Alg. 3 is 18.13 and for Alg. 4 is 12.33 which is a relative difference of 31.99%. The reason for the difference regarding the known obstacles is because of the limited prediction horizon T_H , thus no significant difference can be observed for longer time horizons. The time horizon T_H has less influence on the difference between both algorithms regarding all workspace obstacles $\mathcal{B} \cup \mathcal{B}_u$. The biggest relative difference of the mean values is 48.00% and occurs with the longest time horizon $T_H = 5$ s. This is mainly because most collisions occur due to the unforeseen obstacles, which confirms the influence of the maneuverability to the motion safety of the robot regarding unexpected obstacles.

VII. CONCLUSION

The sequential computation for unions of ICS sets and the concept of robot maneuverability for increasing motion safety were presented. Furthermore, two novel ICS-Checker algorithms are introduced allowing a more efficient computation as former implementations. For evaluation, a novel ICS-Avoid algorithm considering the maneuverability of the robot was used. Simulation results validate this concept and showed a significant reduction of robot collisions especially

TABLE II
EVALUATION OF ICS-AVOID

Algorithm	Run	Collisions						Maneuverability					
		$T_H = 1s$		$T_H = 3s$		$T_H = 5s$		$T_H = 1s$		$T_H = 3s$		$T_H = 5s$	
		\mathcal{B}	$\mathcal{B} \cup \mathcal{B}_u$	\mathcal{B}	$\mathcal{B} \cup \mathcal{B}_u$	\mathcal{B}	$\mathcal{B} \cup \mathcal{B}_u$	\mathcal{B}	$\mathcal{B} \cup \mathcal{B}_u$	\mathcal{B}	$\mathcal{B} \cup \mathcal{B}_u$	\mathcal{B}	$\mathcal{B} \cup \mathcal{B}_u$
ICS-Avoid Alg. 3	1	25	26	4	10	4	9	0.50	0.42	0.67	0.53	0.54	0.48
	2	27	33	3	7	0	4	0.40	0.38	0.68	0.63	0.67	0.55
	3	14	38	6	15	5	15	0.55	0.37	0.58	0.46	0.62	0.40
	4	25	35	4	15	4	10	0.47	0.43	0.66	0.46	0.55	0.42
	5	18	31	7	12	1	12	0.57	0.40	0.59	0.47	0.64	0.43
	Mean	21.8	32.6	4.8	11.8	2.8	10.0	0.50	0.50	0.63	0.51	0.60	0.46
ICS-Avoid Alg. 4	1	19	29	5	5	5	3	0.55	0.47	1.00	0.80	0.78	0.87
	2	12	15	1	9	0	6	0.70	0.65	0.95	0.70	0.90	0.73
	3	16	15	3	17	2	7	0.66	0.63	0.86	0.56	0.84	0.77
	4	19	22	5	12	1	6	0.63	0.51	0.77	0.60	0.98	0.68
	5	12	28	0	7	2	4	0.73	0.55	0.97	0.80	0.85	0.75
	Mean	15.6	21.8	2.8	10.0	2.0	5.2	0.65	0.56	0.91	0.69	0.87	0.76

for unexpected obstacles or for obstacles with a limited motion prediction.

VIII. ACKNOWLEDGMENTS

The authors gratefully acknowledge partial financial support of this work by the Deutsche Forschungsgemeinschaft (German Research Foundation) within the excellence initiative research cluster *Cognition for Technical Systems – CoTeSys* (www.cotesys.org), and the EU-STREP project Interactive Urban Robot (*IURO*, www.iuro-project.eu).

REFERENCES

- [1] M. Goebel, M. Althoff, M. Buss, G. Färber, F. Hecker, B. Heilig, S. Kraus, R. Nagel, F. P. Leon, F. Rattei, M. Russ, M. Schweitzer, M. Thuy, C. Wang, and H.-J. Wünsche, “Design and capabilities of the munich cognitive automobile,” in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2008.
- [2] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, “Winning the darpa grand challenge,” *Journal of Field Robotics*, vol. 23, pp. 661–692, 2006.
- [3] C. Urmson, C. Baker, J. M. Dolan, P. Rybski, B. Salesky, W. R. L. Whittaker, D. Ferguson, and M. Darms, “Autonomous driving in traffic: Boss and the urban challenge,” *AI Magazine*, vol. 30, pp. 17–29, 2009.
- [4] E. Prassler, J. Scholz, and M. Strobel, “Maid: mobility assistance for elderly and disabled people,” in *Proc. of the 24th Annual Conference of Industrial Electronics Society*, 1998.
- [5] K. O. Arras, N. Tomatis, and R. Siegwart, “Robox, a remarkable mobile robot for the real world,” in *Experimental Robotics VIII*, ser. Springer Advanced Robotics Series (STAR), B. Siciliano and P. Dario, Eds. Springer, 2003.
- [6] A. Bauer, K. Klasing, G. Lidoris, Q. Mühlbauer, F. Rohrmüller, S. Sosnowski, T. Xu, K. Kühnlenz, D. Wollherr, and M. Buss, “The autonomous city explorer: Towards natural human-robot interaction in urban environments,” *International Journal of Social Robotics*, vol. 1, no. 2, pp. 127–140, 2009.
- [7] H.-M. Gross, H.-J. Boehme, C. Schroeter, S. Mueller, A. Koenig, C. Martin, M. Merten, and A. Bley, “Shopbot: Progress in developing an interactive mobile shopping assistant for everyday use,” in *Proc. Int. Conf. on Systems, Man and Cybernetics (SMC)*, 2008.
- [8] J.-C. Latombe, *Robot Motion Planning*. Norwell, MA, USA: Kluwer Academic Publishers, 1991.
- [9] R. Philippsen and R. Siegwart, “Smooth and efficient obstacle avoidance for a tour guide robot,” in *Proc. of the IEEE International Conference on Robotics and Automation*, 2003, pp. 446–451.
- [10] G. Lidoris, F. Rohrmüller, D. Wollherr, and M. Buss, “The autonomous city explorer (ace) project - mobile robot navigation in highly populated urban environments,” in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2009.
- [11] E. Prassler, J. Scholz, and P. Fiorini, “Navigating a robotic wheelchair in a railway station during rush hour,” *The International Journal of Robotics Research*, vol. 18, no. 7, pp. 711–727, 1999.
- [12] T. Fraichard, “A short paper about motion safety,” in *Proc. of the IEEE International Conference on Robotics and Automation*, 2007, pp. 1140–1145.
- [13] T. Fraichard and H. Asama, “Inevitable collision states. a step towards safer robots?” *Advanced Robotics*, vol. 18, pp. 1001–1024, 2004.
- [14] R. Parthasarathi and T. Fraichard, “An inevitable collision state-checker for a car-like vehicle,” in *Proc. of the IEEE International Conference on Robotics and Automation*, 2007, pp. 3068–3073.
- [15] N. Chan, J. J. Kuffner, and M. Zucker, “Improved motion planning speed and safety using regions of inevitable collision,” in *17th CISM-IFToMM Symposium on Robot Design, Dynamics, and Control*, 2008.
- [16] L. Martinez-Gomez and T. Fraichard, “An efficient and generic 2d inevitable collision state-checker,” in *Proc. of the IEEE International Conference on Intelligent Robots and Systems*, 2008, pp. 234–241.
- [17] —, “Collision avoidance in dynamic environments: an ics-based solution and its comparative evaluation,” in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2009, pp. 100–105.
- [18] S. M. LaValle and J. J. Kuffner, “Randomized kinodynamic planning,” *The International Journal of Robotics Research*, vol. 20, pp. 378–401, 2001.
- [19] J. Reif and M. Sharir, “Motion planning in the presence of moving obstacles,” *J. ACM*, vol. 41, pp. 764–790, 1994.
- [20] K. M. Krishna, R. Alami, and T. Simeon, “Safe proactive plans and their execution,” *Robotics and Autonomous Systems*, vol. 54, pp. 244–255, 2006.
- [21] J.-M. Lien, “Hybrid motion planning using minkowski sums,” in *Proceedings of Robotics: Science and Systems IV*, 2008.
- [22] D. Althoff, D. Wollherr, and M. Buss, “Safety assessment of trajectories for navigation in uncertain and dynamic environments,” in *Proc. of the IEEE International Conference on Robotics and Automation*, 2011.