Probabilistic Collision State Checker for Crowded Environments

Daniel Althoff, Matthias Althoff, Dirk Wollherr and Martin Buss

***Abstract*— For path planning algorithms of robots it is important that the robot does not reach a state of inevitable collision. In crowded environments with many humans or robots, the set of possible inevitable collision states (ICS) is often unacceptably high, such that the robot has to stop and wait in too many situations. For this reason, the concept of ICS is extended to probabilistic collision states (PCS), which estimates the collision probability for a given state. This allows to efficiently run planning algorithms through crowded environments when accepting a certain collision probability. A further novelty is that the obstacles possibly react to the robot in order to mitigate the risk of a collision. The results show a significant difference in interaction behavior. Thus, this approach is especially suited for active and non-deterministic moving obstacles in the robot workspace.**

I. Introduction

In a static environment, the safety of a planned path can be verified by checking if the planned state trajectory does not enter the set of inevitable collision states (ICS), which depends on the geometry of the obstacles and the dynamics of the robot. The concept of inevitable collision states can be extended to scenarios in which the future behavior of dynamic obstacles is exactly known, see e.g. [1]. However, in a scenario where the future behavior of other obstacles is unknown, one cannot decide if the current robot state is an ICS since it depends on the future actions of the dynamic obstacles. For this reason, the concept of ICS is extended to probabilistic scenarios in this work. An example for such a scenario is an outdoor robot finding its way through a populated pedestrian zone, which has been investigated in the Autonomous City Explorer project, short ACE [2] (The ACE project is part of the CoTeSys Excellence Cluster [3] in which this work has been partly carried out). Another example of a robot designed for crowded and uncertain scenarios is e.g. RoboX which guided visitors at the Expo 2002 [4].

A. Related Work

In [5], ICS has been compared to many other common navigation schemes, such as *nearness diagram* [6], *dynamic window* [7] and *velocity obstacle* [8]. These alternative concepts have been evaluated with respect to the three criteria:

- A robotic system should consider its own dynamics;
- Consider the environment objects' future behavior;
- Reason over an infinite time-horizon.

The authors are with the Institute of Automatic Control Engineering (LSR) of the Technische Universität München, D-80290 München, Germany {da, althoff, dw, mb}@tum.de

The result of this evaluation was that all common approaches cannot cope with one or more of these criteria, except the ICS concept from [9]. Sets of inevitable collisions states, which are also called regions of inevitable collision (RIC) in other works, have been investigated in, e.g. [1], [10]–[12]. Under- and over-approximations of sets of ICS or RIC have been developed in [12]–[14]. Besides ICS related approaches, safety of robots in uncertain environments has been assessed by predicting the movement of dynamic obstacles with Markov chains for robotic scenarios in [15] and for autonomous cars in [16]. Other approaches use Monte Carlo simulation for the thread detection of vehicles in traffic scene in [17], [18] to verify the safety of all objects.

B. Organization of the Paper

In Sec. II, the concept of inevitable collision states is recalled from literature. This concept is extended to the case when the future behavior of other workspace objects can only be modeled probabilistically. Instead of a yes/no answer for an ICS, the extended definition of a *probabilistic collision state* (PCS) returns a probability for a collision. Since the definition of PCS is not directly implementable - which is also the case for the ICS concept as explained later, the implementation details are presented in Sec. III. This is done by first showing how the prediction of other workspace objects is conducted. Then, a finite set of behavior alternatives for the robot and the workspace objects is introduced in order to compute the robot trajectory causing the smallest collision probability. Finally, in Sec. IV, numerical results are presented which compute the probabilities for ICS in randomly generated scenarios. It is also investigated how big the impact on the probability computations is when including interaction between the workspace objects and the robot.

II. Inevitable and Probabilistic Collision States

In this section, the notion and definition of inevitable collision states is recapitulated first. This definition is then extended to a probabilistic setting which is believed to be more appropriate in highly populated and uncertain environments. Finally, it is shown that the newly introduced probabilistic collision states are a generalization of inevitable collision states.

A. Inevitable Collision States

The definition of inevitable collision states used in this work is recalled from [11]. In order to precisely define inevitable collision states some notations have to be introduced. The state $s(t)$ and input $u(t)$ of the considered robot system

(for a point in time t) can take values from the state space $\mathcal{S}$ and the control space $\mathcal{U}$. For a given initial state $s(0)$ and an input trajectory $u(t)$, the dynamics of the robot is determined by the nonlinear differential equation $\dot{s} = f(s,u)$. 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 $u(t)$, an input trajectory is denoted by $\tilde{u}$ 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 $\mathcal{A}(\tilde{u}(t))$. These notations finally allow to define an inevitable collision state.

Definition 1: **Inevitable Collision State**
The state 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 crash with another workspace object. Next, this definition is extended to a probabilistic setting.

B. Probabilistic Collision States

In crowded environments, motion planning with ICS is not reasonable. Consider the earlier mentioned scenario where a robot finds its way through many people in a pedestrian zone. Since in this scenario the workspace objects are humans, their future occupancy $\mathcal{B}_i(t)$ is unknown and can only be predicted. For this reason, their motion and their future occupancy is specified by probability density functions in this work.
The probability density function of the occupancy in the workspace caused by the object $\mathcal{B}_i$ is denoted by f_i. Since one can only formulate a probability distribution for a random vector and not an occupancy set, f_i represents the probability distribution of a point c of $\mathcal{B}_i$ 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 $(-\mathcal{B}_i + c_i)$ to $\mathcal{A}$, so that the new occupancy of the robot is $\mathcal{A}^{\mathrm{b}} = \mathcal{A} \oplus (-\mathcal{B}_i + c_i)$, which is explained in more detail in [19]. The Minkowski addition of the occupancy sets is visualized in Fig. 1 for a two-dimensional workspace with position coordinates x_x and x_y. In this work, two different kinds of objects are considered:

- Passive objects: they *ignore* the robot's trajectory. The associated probability density function of the occupancy is denoted by $f_i(x,t)$, where x is the position in the workspace and t is the time.
- Active objects: they *react* to the robot's trajectory $\tilde{u}$ in order to reduce the collision risk. The associated probability density function of the occupancy is denoted by $f_i(x,t,\tilde{u})$.

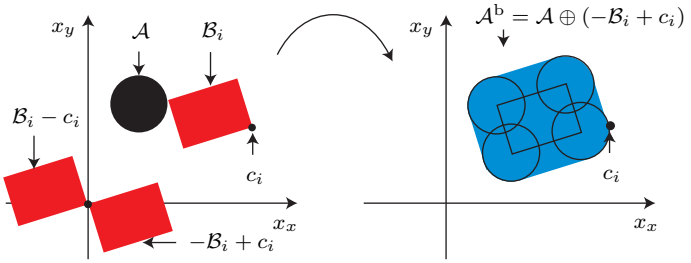


Fig. 1. Minkowski addition of the workspace occupancy of the robot and another object.

In the following, the collision probability for *active* objects is described, leading to the definition of Probabilistic Collision States for active and later also for passive objects.
The probability that a certain region R of the workspace is occupied by an object $\mathcal{B}_i$ is obtained by integration:

$$P(R \cap \mathcal{B}_i \neq \emptyset) = \int_{R \oplus (-\mathcal{B}_i + c_i)} f_i(x,t,\tilde{u})\, dx.$$

Thus, the probability that the robot system $\mathcal{A}$, applying the input trajectory $\tilde{u}$, has a collision with another probabilistic object $\mathcal{B}_i$ is computed as

$$p_i^C(t,\tilde{u}) := P(\mathcal{A}(\tilde{u}(t)) \cap \mathcal{B}_i \neq \emptyset) = \int_{\mathcal{A}^{\mathrm{b}}(\tilde{u}(t))} f_i(x,t,\tilde{u})\, dx, \tag{1}$$

where the index i of the crash probability p^C refers to the i^{th} workspace object. Additionally, the probability that a crash occurs within a time interval $[t_k, t_{k+1}[$ is considered, where $t_k = k\,T$, $k \in \mathbb{N}^+$ is a time step and $T \in \mathbb{R}^+$ is the time step size. The set occupied in the workspace for a time interval is denoted by $\mathcal{A}(\tilde{u}([t_k, t_{k+1}[))$ such that the collision for a time interval is obtained by

$$p_i^C([t_k, t_{k+1}[, \tilde{u}) = \int_{\mathcal{A}^{\mathrm{b}}(\tilde{u}([t_k, t_{k+1}[))} f_i(x, [t_k, t_{k+1}[, \tilde{u})\, dx.$$

The probability of surviving is defined as the probability that no crash occurs such that the surviving probability for a certain time interval is $p_i^S([t_k, t_{k+1}[, \tilde{u}) = 1 - p_i^C([t_k, t_{k+1}[, \tilde{u})$. An upper bound for surviving a time interval when considering all objects $\mathcal{B}_i$ is computed by

$$\overline{p}^S([t_k, t_{k+1}[, \tilde{u}) = \min_{i=1,\dots,n_b} p_i^S([t_k, t_{k+1}[, \tilde{u}).$$

Initially, it is confusing that the upper bound is computed by the min operator. The reason is that only the object with the highest collision risk (lowest survival probability) is considered and the remaining objects are neglected.
Clearly, the upper bound for the probability of survival for the infinite time horizon is computed by

$$\overline{p}^S([0,\infty[, \tilde{u}) = \prod_{k=0}^{\infty} \overline{p}^S([t_k, t_{k+1}[, \tilde{u}).$$

Since the robot system can choose any input trajectory from the set of possible input trajectories $\tilde{U}$, the input trajectory causing the maximum survival probability is chosen:

$$\overline{p}^S_{\max}([0,\infty[) = \max_{\tilde{u} \in \tilde{U}} \overline{p}^S([0,\infty[, \tilde{u}). \tag{2}$$

The upper bound of the maximum survival probability allows to define the probability of an inevitable collision state for active objects.

Definition 2: **Probabilistic Collision State for active objects**
The probability of a state leading to a collision concerning active objects is defined as the lower bound of crashing with an active obstacle under the best possible input trajectory $\tilde{u}(t)$:

$$PCS_{\mathrm{a}}(s) = 1 - \overline{p}^{S}_{\max}([0,\infty[)$$

where $\overline{p}^{S}_{\max}([0,\infty[)$ is the upper bound for the probability of survival for active objects.

The definition of $PCS_{\mathrm{a}}(s)$ includes the special case for passive objects, which is denoted by $PCS_{\mathrm{p}}(s)$. For passive objects it is sufficient to consider the probability density function $f_i(x,t)$ instead of $f_i(x,t,\tilde{u})$ since they move independently of the robot. When no index is given (PCS) it is always referred to active objects (PCS_a).
In this work, active objects do not act hostile, they always try to reduce the collision risk, so the condition

$$PCS_{\mathrm{p}}(s) \geq PCS_{\mathrm{a}}(s) \tag{3}$$

is always true for active objects. This means that if active objects are treated as passive ones, $PCS_{\mathrm{p}}(s)$ is the upper bound of $PCS_{\mathrm{a}}(s)$. The definition of PCS_{a} also allows general active objects, but as a consequence, condition (3) would not hold anymore.

In the appendix, it is shown that $PCS(s) = 1 \leftrightarrow s = ICS$ which means that the presented Def. 2 is a generalization of Def. 1. If instead of a lower bound, an upper bound of the probability of a crash would be computed, the equivalence $PCS(s) = 1 \leftrightarrow s = ICS$ would not hold anymore. The implementation issues of the given definition are discussed next.

C. Implementation Issues

The definition of ICS (see Def. 1) and the one for PCS (see Def. 2) is not implementable. The reason for this is twofold:

1) Infinite number of input trajectories: One problem is that an infinite number of input trajectories $\tilde{u} \in \tilde{\mathcal{U}}$ has to be checked. This is solved in literature, e.g. [11], by computing with a finite subset of input trajectories $\tilde{u}$. This leads to a conservative computation of an ICS, i.e. a state may not be an ICS although the computation concluded that the state is one. However, it can always be guaranteed that a state is not an ICS with a finite number of input trajectories.

2) Limited time horizon: 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. When all workspace objects including the robot are not moving anymore, the computation can be stopped. The same holds when the robot imitates the movement of workspace objects which has also been discussed in [11]. Before the imitation can be applied, the robot is in a catch-up phase. It can be shown that it is sufficient to compute the ICS for the catch-up phase. However, the imitation approach can only be applied if the workspace objects behave deterministically which is not the case in this work. Thus, only the approach of computing maneuvers that come to a standstill can be applied. Since the uncertainty in the prediction of other workspace objects increases with time, the main focus lies on braking maneuvers which come to a standstill in a reasonable time horizon. Additionally, only a finite number of braking maneuvers are considered as for the ICS implementations in literature.

III. Probabilistic Collision State Checker

Although the definition of PCS allows arbitrary workspaces and has no restriction to the object's shape, kinematics or dynamics, a PCS checker is presented for disk-shaped objects in a two-dimensional workspace with position coordinates x_x and x_y. First, the model for the motion prediction of passive and active objects is described. Second, the probability density function for passive objects is computed. Third, the optimal input trajectory of the robot is obtained which is then used to determine the probability density function for active objects. Finally, the collision probabilities are computed for passive and active objects.

A. Method for Motion Prediction

The distributions of passive objects can be computed with motion prediction techniques as described in e.g. [20]–[23]. In order to obtain a more efficient implementation of $f_i(x,t)$ and $f_i(x,t,\tilde{u})$ than in the referred literature, a constant acceleration model is used for the prediction of the workspace objects. The position, velocity and acceleration are denoted by x, v and a, respectively. The indices x and y refer to the x- and y-coordinate of the two-dimensional workspace $\mathcal{W}$. The dynamic system of the constant acceleration model is

$$\begin{bmatrix} \dot{x}_x \\ \dot{x}_y \\ \dot{v}_x \\ \dot{v}_y \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_x \\ x_y \\ v_x \\ v_y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ a_x \\ a_y \end{bmatrix},$$

where the absolute value of the acceleration $\sqrt{a_x^2 + a_y^2} \leq a^{\max}$ is limited. The velocity is indirectly limited by the initial velocity since only braking trajectories are considered as discussed in Sec. II-C. After a time discretization with $t_k = k\,T$, where $k \in \mathbb{N}^+$ has been introduced as the time step and $T \in \mathbb{R}^+$ is the time step size, the dynamic model can be exactly transformed to the discrete time form:

$$\underbrace{\begin{bmatrix} x_x \\ x_y \\ v_x \\ v_y \end{bmatrix}(t_{k+1})}_{s(t_{k+1})} = \underbrace{\begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_x \\ x_y \\ v_x \\ v_y \end{bmatrix}(t_k)}_{s(t_k)} + \underbrace{\begin{bmatrix} a_x \frac{T^2}{2} \\ a_y \frac{T^2}{2} \\ a_x T \\ a_y T \end{bmatrix}}_{u}. \tag{4}$$

It is further assumed that the initial state of the objects has a multivariate Gaussian distribution $s(0) \backsim \mathcal{N}(\mu, \Sigma)$ with mean value μ and covariance matrix Σ. From the

multiplication rule and the addition rule of independent random variables with Gaussian distributions, it follows that the mean value and the covariance of the state s in (4) are updated as

$$\begin{aligned}\mu(t_{k+1}) &= A\mu(t_k) + u \\ \Sigma(t_{k+1}) &= A\Sigma(t_k)A^T,\end{aligned} \tag{5}$$

where A^T is the transpose of the system matrix A. Note that the input u has no influence on the covariance matrix because this input is deterministic.

B. Probabilistic Density Function of Passive Objects

The Gaussian distribution of the i^{th} workspace object $\mathcal{B}_i$ can finally be formulated as

$$f_i(x,t_k) = \frac{1}{(2\pi)^2|\Sigma(t_k)|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu(t_k))^T\Sigma(t_k)^{-1}(x-\mu(t_k))\right).$$

For passive objects, the acceleration inputs are set to zero ($a_x = 0$, $a_y = 0$). This is changed when active objects are considered later.

C. Input Trajectory of the Robot

Under the assumption that the workspace objects move independently of the robot, i.e. $\forall i : f_i(x,t,\tilde{u}) = f_i(x,t)$, the input trajectory $\tilde{u}^*$ that minimizes $PCS_\mathrm{p}(s)$ is computed. The finite set of possible input trajectories are braking trajectories, i.e. trajectories for which the velocity is constantly decreasing. The state trajectories are computed from the input trajectories with the constant acceleration model of (4). The finite set of braking trajectories is chosen as follows: Use the maximum possible absolute acceleration such that $\sqrt{a_x^2 + a_y^2} = a^{\max}$, where the maximum absolute acceleration is limited through the contact friction of the robot. The parameter that is varied is the direction ϕ^R of the acceleration, where the raised R emphasizes that the direction is given in the relative coordinate system of the robot and not in the global workspace coordinates. The scalar product of the velocity vector and the direction vector ϕ^R (both in relative coordinates) is always negative to ensure braking trajectories, see Fig. 2. Next, the computed input trajectory $\tilde{u}^*$ minimizing $PCS_\mathrm{p}(s)$ is used to adapt the probability distribution $f_i(x,t)$ such that it depends on $\tilde{u}^*$: $f_i(x,t) \rightarrow f_i(x,t,\tilde{u}^*)$.

D. Probabilistic density function for active objects

One of the difficulties in the implementation is that the probability distribution $f_i(x,t,\tilde{u})$ of active workspace objects depends on the input trajectory $\tilde{u}$, while the choice of the input trajectory in (2) depends on $\overline{p}^S([0,\infty[,\tilde{u})$ and thus on the probability distribution $f_i(x,t,\tilde{u})$. This mutual dependence is broken up by first assuming that the probability distribution of the workspace objects is independent of the input trajectory $\tilde{u}$ of the robot. So the $\tilde{u} \in \tilde{\mathcal{U}}$ is determined by minimizing the crash probability $PCS_\mathrm{p}(s)$ as described above.

In order to distinguish the input trajectories of the workspace objects $\mathcal{B}_i$ from the ones of the robot, they are denoted by $\tilde{u}_i^{\mathcal{B}}$, which are element of $\tilde{\mathcal{U}}_i^{\mathcal{B}}$. The finite number of input trajectories $\tilde{u}_{i,k}^{\mathcal{B}} \in \tilde{\mathcal{U}}_i^{\mathcal{B}}$ is generated as previously presented in Sec. III-C for which the direction of the acceleration ϕ is varied while the maximum absolute acceleration $a^{\max}$ is applied. The additional index k in $\tilde{u}_{i,k}^{\mathcal{B}}$ indicates the kth input trajectory. The input trajectory causing the smallest probability $PCS_\mathrm{a}(s)$ is denoted by $\tilde{u}^{\mathcal{B}_i *}$.

The optimal input trajectories $\tilde{u}^{\mathcal{B}_i *}$ model the case when workspace objects try to avoid a collision with maximum effort or willingness. However, workspace objects may not react to the trajectory of the robot at all. For this reason, a probability distribution $f(e)$ is introduced, where e is the effort varying in the interval $[0,1]$[1]. The applied absolute acceleration for the optimal acceleration direction of $\tilde{u}^{\mathcal{B}_i *}$ is obtained as $\sqrt{a_x^2 + a_y^2} = e\,a^{\max}$. If $e = 0$, the acceleration of the object is $a_x = a_y = 0$ and if $e = 1$, the full acceleration for avoiding the robot is applied as for $\tilde{u}^{\mathcal{B}_i *}$. Since only a finite number of values of e is used, the kth values is denoted by e_k. The probability distribution of e and the acceleration direction are depicted in Fig. 3. The final probability distribution is computed as

$$f_i(x,t,\tilde{u}^*) = \sum_{k=1}^{n_e} e_k f_{i,k}(x,t,\tilde{u}^*),$$

where n_e is the number of considered values of e.

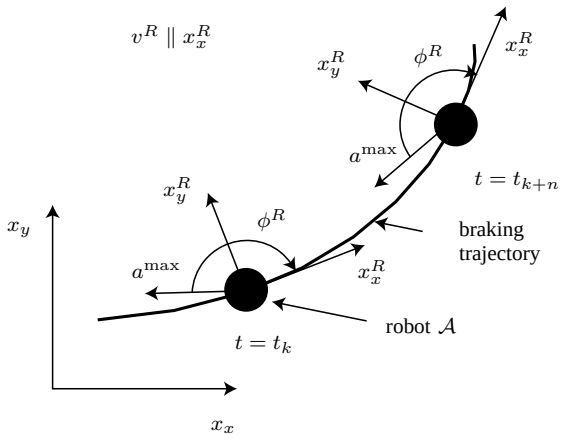


Fig. 2. Braking trajectory of the robot. The direction of the acceleration is constant in the relative coordinate system of the robot and the magnitude is constant over time.

E. Numerical Computation of the Crash Probability

Another important implementation detail is the efficient computation of the integral of the probability distribution of workspace objects according to (1). This is done by computing with an occupancy grid with equidistant segmentation as shown in Fig. 4. The occupancy of the robot $\mathcal{A}^\mathrm{b}(\tilde{u}(t))$ can be computed offline for all relevant initial conditions and input trajectories, and then stored in a database. In terms

[1]The interval $[-1,1]$ would also consider hostile objects, which are not part of the work.

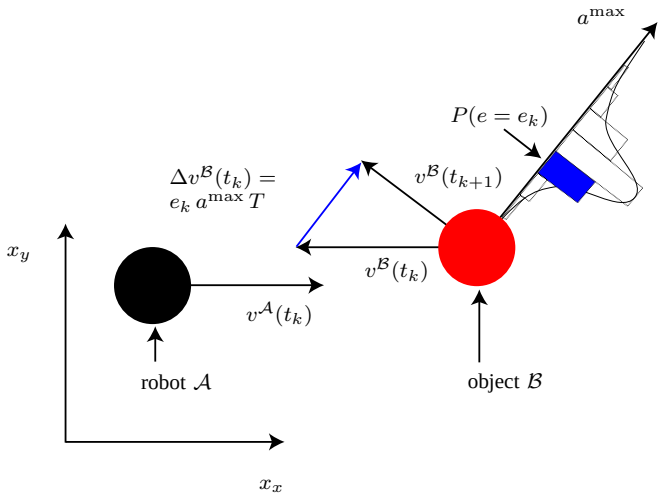


Fig. 3. Acceleration applied to an object in order to avoid the robot.

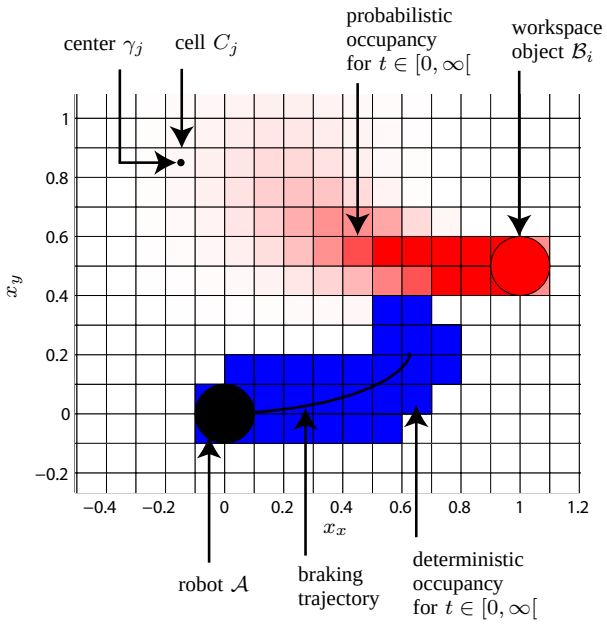


Fig. 4. Numerical integration of the crash probability.

of braking trajectories $\tilde{u}$, one has to compute only with the finite number of relative braking directions ϕ^R because the absolute value is always $a^{\max}$, see Sec. III-C. In terms of the system state, one has to compute with different initial velocities. Initial positions and directions do not have to be varied since the workspace objects are stored within the robot coordinate frame. Note that the occupancy of the robot is deterministic, i.e. a cell is either occupied or not.

The occupancy of the other objects is probabilistic and computed online. The Gaussian distributions are approximately mapped into the occupancy grid by assuming a uniform distribution within the cells C_j. The probability density value at the center γ_j of cell j is $f_i(\gamma_j, t, \tilde{u})$. From the uniform distribution follows that the probability of the occupancy is:

$$P(x(t) \in C_j | u = \tilde{u}(t)) = A f_i(\gamma_j, t, \tilde{u}), \quad A \in \mathbb{R}^+$$

where A is the area of a cell. The above formula is analogously computed for passive objects. The probability of a crash is finally obtained by summing up the probabilistic occupancies $P(x \in C_j)$ for cells j which are occupied by the robot. An example of the deterministic occupancy of the robot and the probabilistic occupancy of another workspace object is visualized in Fig. 4.

IV. Simulation Results

In this section, simulations of the presented approach for computing PCS are performed. No comparison to ICS is done since it cannot be directly applied to scenarios with non-deterministic objects.

The following simulations show the influence of the probabilistic effort $f(e)$ (for avoiding the robot) on the result of $PCS(s)$ for passive and active objects. It is assumed that the workspace objects are active objects as previous described, which means they do not behave hostile and (3) holds.

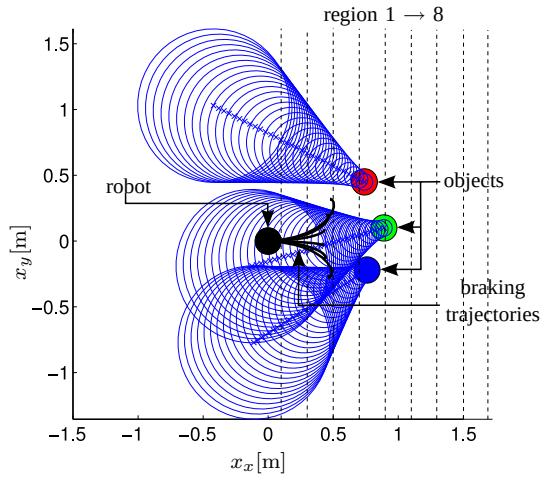
In order to show the usefulness of computing with a distribution of the effort for avoiding the robot, random scenarios are generated and evaluated. Despite the workspace objects, the initial state of the robot is fixed and has the initial state $s = \begin{bmatrix} 0\text{m} & 0\text{m} & 0.5\frac{\text{m}}{\text{s}} & 0\frac{\text{m}}{\text{s}} \end{bmatrix}^T$. The obstacles are placed randomly in front of the robot facing towards it. Each scenario consists of one robot and three workspace objects. The workspace objects are placed randomly in one of eight predefined adjacent regions which are partitioned in x-direction. The regions and other parameters for the obstacles are listed in Tab. I.

TABLE I
Simulation parameters for the obstacles

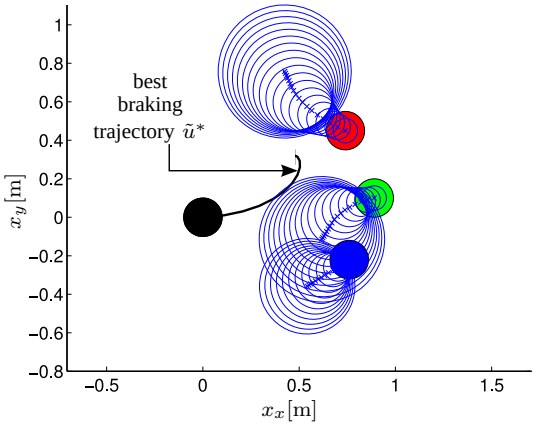
Eight regions for x_x[m]	$[0.1, 0.3]$ $\cdots$ $[1.5, 1.7]$
One region for x_y[m]	$[-0.5, 0.5]$
Initial velocity direction $\sphericalangle\vec{v}$[rad]	$[\frac{3}{4}\pi, \frac{5}{4}\pi]$
Initial absolute velocity $\|\vec{v}\|$ $[\frac{\text{m}}{\text{s}}]$	$[0.0, 0.5]$
Acceleration direction $\sphericalangle\vec{a}$[rad]	$[\frac{3}{4}\pi, \frac{5}{4}\pi]$
Absolute acceleration $[\frac{\text{m}}{\text{s}^2}]$:	$\{0.1, 0.3, 0.5, 0.7, 0.9\}$
Initial covariance $\Sigma(0)$ (see (5))	$\begin{bmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix}$

An exemplary scenario using the listed parameters is shown in Fig. 5. In this scenario, the collision probability is reduced by 45% for considering active objects. For each of the eight regions for the initial object positions, 100 randomly generated situations are computed based on two different workspace object models:

- Passive objects, i.e. objects that are not trying to avoid the robot. Thus, the probability distribution for the effort is $f(e) = \begin{cases} 0, \text{ if } e \in]0, 1] \\ \delta, \text{ if } e = 0 \end{cases}$, where δ is the Dirac impulse.
- Active objects, i.e. objects that are trying to avoid the



(a) Motion prediction assuming passive objects, the associated collision probability is $PCS_{\mathrm{p}}(s) = 40\%$



(b) Motion prediction assuming active objects, the associated collision probability is $PCS_{\mathrm{a}}(s) = 18\%$

Fig. 5. Random scenario for x_x[m] region 4: Gaussian distribution is illustrated by 2σ-ellipsoids.

robot. For the simulations, a Gaussian distribution for $f(e)$ is used with mean value $\mu = 0.5$ and standard deviation $\sigma = 0.2$. In order to obtain a finite number of effort values e_k, the Gaussian distribution is discretized.

In order to obtain significant results, randomly generated situations with a collision probability of less than 0.01 are discarded. To verify the usefulness of modeling the avoidance capabilities of the objects, the mean relative difference

$$\overline{D} = \frac{1}{n} \sum^{n} 1 - \frac{PCS_{\mathrm{a}}(s)}{PCS_{\mathrm{p}}(s)}$$

obtained from all scenarios is shown in Fig. 6.

It can be seen that there is a significant difference between the active and passive workspace objects. The maximum achieved difference is 48%. It can also be seen that the improvement depends on the distance to the obstacle when assuming the velocity range and direction as listed in Tab. I for the robot and the obstacles.

There is no difference for greater distances than 1.3m

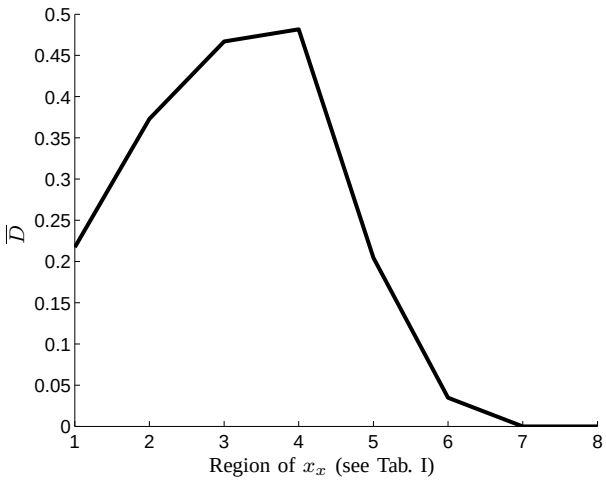


Fig. 6. Relative difference between ICS probability between active and passive obstacles.

since the crash probabilities are zero for active and passive workspace objects. During the evaluation, PCS was calculated 600 times for three workspace obstacles. The algorithm is implemented in Matlab and was executed on a AMD Phenom with 2.5 Ghz. The mean computational time of $PCS(s)$ for one robot state is 0.1s.

V. Conclusions and Future Works

This paper consists of two major contributions, a novel definition for the probabilistic computation of inevitable collision states and an exemplary implementation of this definition. The proposed definition allows to reason about the safety of planned paths in uncertain dynamic environments. Further, it is shown that this definition is a generalization of the inevitable collision state approach. The presented method is especially useful in crowded environments where the future behavior of other objects in the workspace has high uncertainty.

The exemplary implementation has shown that the willingness of objects to avoid the robot has a big impact on the collision risk. It is no noteworthy that the presented computation of PCS preserves the three criteria from [5] mentioned in the introduction: A robotic system should consider its own dynamics, consider the environment objects' future behavior and reason over an infinite time-horizon.

The simulations of the PCS checker showed that it is efficient and thus applicable to real world scenarios. It is planned to implement the PCS checker on a robot to verify the results in a crowded scenario.

VI. 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`), the Transregional Collaborative Research Centre 28 *Cognitive Automobiles*

(www.kognimobil.org) and the EU-STREP project Interactive Urban Robot (*IURO*, www.iuro-project.eu).

APPENDIX

The purpose of the appendix is to show that $PCS(s) = 1 \leftrightarrow s = ICS$ when computing $PCS(s)$ according to Def. 2. In a deterministic scenario, the position of other objects is known such that the probability distribution of all objects is a Dirac impulse δ:

$$f_i(x,t) = \begin{cases} \delta, \text{ if } x(t) = c_i(t) \\ 0, \text{ otherwise} \end{cases}$$

where $c_i(t)$ is a point of object $\mathcal{B}_i$ as introduced in Sec. II-B. From this follows directly that

$$\mathcal{A}(\tilde{u}(t)) \cap \mathcal{B}_i(t) \neq \emptyset \leftrightarrow p_i^C(t,\tilde{u}) = 1.$$

Thus, using Def. 1, the statement $s = ICS$ can be reformulated to

$$\forall \tilde{u} \in \tilde{\mathcal{U}}, \exists t, \exists i, p_i^C(t,\tilde{u}) = 1.$$

Using the computations introduced in Sec. II-B, it is shown that this statement is equivalent to $PCS(s) = 1$:

$$\begin{aligned}
&\leftrightarrow \forall \tilde{u} \in \tilde{\mathcal{U}}, \exists t, \exists i : p_i^C(t,\tilde{u}) = 1 \\
&\leftrightarrow \forall \tilde{u} \in \tilde{\mathcal{U}}, \exists k, \exists i : p_i^C([t_k, t_{k+1}[,\tilde{u}) = 1 \\
&\leftrightarrow \forall \tilde{u} \in \tilde{\mathcal{U}}, \exists k, \exists i : p_i^S([t_k, t_{k+1}[,\tilde{u}) = 0 \\
&\leftrightarrow \forall \tilde{u} \in \tilde{\mathcal{U}}, \exists k : \overline{p}^S([t_k, t_{k+1}[,\tilde{u}) = \\
&\qquad\qquad \min_{i=1,\ldots,n_b} p_i^S([t_k, t_{k+1}[,\tilde{u}) = 0 \\
&\leftrightarrow \forall \tilde{u} \in \tilde{\mathcal{U}} : \overline{p}^S([0,\infty[,\tilde{u}) = \prod_{k=0}^{\infty} \overline{p}^S([t_k, t_{k+1}[,\tilde{u}) = 0 \\
&\leftrightarrow \overline{p}^S_{\max}([0,\infty[) = \max_{\tilde{u}\in\tilde{U}} \overline{p}^S([0,\infty[,\tilde{u}) = 0 \\
&\leftrightarrow PCS(s) = 1 - \overline{p}^S_{\max}([0,\infty[) = 1.
\end{aligned}$$

REFERENCES

[1] 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.

[2] 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.

[3] M. Buss, M. Beetz, and D. Wollherr, "Cotesys – cognition for technical systems," *International Journal of Assistive Robotics and Mechatronics*, vol. 8, no. 4, pp. 25–36, 2007.

[4] 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.

[5] T. Fraichard, "A short paper about motion safety," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2007, pp. 1140–1145.

[6] J. Minguez and L. Montano, "Nearness diagram (nd) navigation: Collision avoidance in troublesome scenarios," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 1, pp. 45–59, 2004.

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

[8] P. Fiorini and Z. Shillert, "Motion planning in dynamic environments using velocity obstacles," *International Journal of Robotics Research*, vol. 17, pp. 760–772, 1998.

[9] T. Fraichard and H. Asama, "Inevitable collision states. A step towards safer robots?" *Advanced Robotics*, vol. 18, pp. 1001–1024, 2004.

[10] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006.

[11] 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.

[12] L. Martinez-Gomez and T. Fraichard, "Collision avoidance in dynamic environments: an ics-based solution and its comparative evaluation," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2009, pp. 100–105.

[13] M. Zucker, "Approximating state-space obstacles for non-holonomic motion planning," Carnegie Mellon University, Robotics Institute, Tech. Rep., 2006.

[14] N. Chan, 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, pp. 103–114.

[15] F. Rohrmüller, M. Althoff, D. Wollherr, and M. Buss, "Probabilistic mapping of dynamic obstacles using markov chains for replanning in dynamic environments," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 2504–2510.

[16] M. Althoff, O. Stursberg, and M. Buss, "Model-based probabilistic collision detection in autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, pp. 299 – 310, 2009.

[17] S. B. A. Broadhurst and T. Kanade, "Monte carlo road safety reasoning," in *IEEE Intelligent Vehicle Symposium (IV2005)*,, June 2005, pp. 319–324.

[18] A. Eidehall and L. Petersson, "Threat assessment for general road scenes using monte carlo sampling," in *2006 IEEE Intelligent Transportation Systems Conference*, September 2006, pp. 1173–1178.

[19] J.-M. Lien, "Hybrid motion planning using minkowski sums," in *Proceedings of Robotics: Science and Systems IV*, 2008.

[20] H. C. Yen, H. P. Huang, and S. Y. Chung, "Goal-directed pedestrian model for long-term motion prediction with application to robot motion planning," in *Proc. of the International Conference on Advanced Robotics and its Social Impacts*, 2008, pp. 1–6.

[21] D. Vasquez, T. Fraichard, and C. Laugier, "Incremental learning of statistical motion patterns with growing hidden markov models," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 3, pp. 403–416, 2009.

[22] S. Thompson, T. Horiuchi, and S. Kagami, "An environment driven model of human navigation intention for mobile robots," in *Proc. of The 13th IASTED International Conference on Robotics and Applications*, 2007, pp. 119–125.

[23] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical Review E*, vol. 51, pp. 4282–4286, 1995.