

Safety Assessment of Trajectories for Navigation in Uncertain and Dynamic Environments

Daniel Althoff, Dirk Wollherr and Martin Buss

Abstract—This paper presents a probabilistic threat assessment method for reasoning about the safety of robot trajectories in uncertain and dynamic environments. For safety evaluation, the overall collision probability is used to rank candidate trajectories by considering the collision probability of known objects as well as the collision probability beyond the planning horizon. Monte Carlo sampling is used to estimate the collision probabilities. This concept is applied to a navigation framework that generates and selects trajectories in order to reach the goal location while minimizing the collision probability. Simulation scenarios are used to validate the overall crash probability and show its necessity in the proposed navigation approach.

I. INTRODUCTION

One of the key challenges in the field of service robotics is safe and efficient robot navigation. Therefore, the robot should be able to navigate among static and moving obstacles. Several approaches make use of the freezing world assumption, meaning that planning is done based on the snapshot of the environment, thus dynamic objects are treated as static ones. These algorithms were successfully used in dynamically changing environments like the Expo 2002 [1]. More recent approaches use motion prediction to cope with dynamic objects as shown in [2]–[4]. This is necessary to guarantee safety in the presence of dynamic objects, under the assumption of a reliable motion prediction. In the case of a complete environment model, i.e. that the information about the environment and its future contains no uncertainty, it is possible to guarantee safety, which cannot be done in the case of an incomplete model. In [5] three criteria have been introduced to evaluate the safety of a robotic system with the result, that all common approaches cannot cope with one or more of these criteria, except for the Inevitable Collision State (ICS) concept [6]. Since environment perception is never noise free and the motion of objects with its own will, such as humans, cannot be accurately predicted, the *overall collision probability* is used to rank the safety of each trajectory. It is based on the probabilistic generalization of the ICS concept, the *Probabilistic Inevitable Collision States* [7] also called *Probabilistic Collision State* [8]. This extension requires probabilistic motion prediction for the workspace objects. Based on this prediction, the probability of the robot being in a state leading to a collision is calculated. The focus of this paper is on motion safety especially for uncertain and dynamic environments.

The authors are with the Institute of Automatic Control Engineering (LSR). Dirk Wollherr is also with Institute for Advanced Study (IAS). Both institutes are with the Technische Universität München, D-80290 München, Germany. {da, dw, mb}@tum.de

II. PROBLEM FORMULATION

The problem discussed in this paper is described as follows. For simplicity we assume that the robot and the objects are disc-shaped and move in the \mathbb{R}^2 plane. The state \mathbf{s} of an object is represented by its position \mathbf{x} and its velocity \mathbf{v} . Furthermore, a simple constant-acceleration model is used with kinematic and dynamic constraints for all objects. The presented methods can easily be extended to any polygon shape, to any motion model, and to higher dimensions. In the remainder of the paper the model $\dot{\mathbf{s}} = m(\mathbf{s}, \mathbf{u})$ is used

$$\underbrace{\begin{bmatrix} \dot{x}_x \\ \dot{x}_y \\ \dot{v}_x \\ \dot{v}_y \end{bmatrix}}_{\dot{\mathbf{s}}} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \underbrace{\begin{bmatrix} x_x \\ x_y \\ v_x \\ v_y \end{bmatrix}}_{\mathbf{s}} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ u_1 \\ u_2 \end{bmatrix}}_{\mathbf{u}} a_{\max}$$

with respect to the constraints $\sqrt{v_x^2 + v_y^2} \leq v_{\max}$ and $\sqrt{a_x^2 + a_y^2} \leq a_{\max}$. The control inputs u_1, u_2 are normalized and vary from $[-1, 1]$.

Assume a set of n objects sharing an environment together with one robot. Each object has an initial state $\mathbf{s}_i^{\text{init}}$ and a goal state $\mathbf{s}_i^{\text{goal}}$. No information about their navigation algorithms or their future trajectories are given. Only a cost function is given, which models the fact that the objects have a goal they want to reach and that they prevent trajectories with high acceleration for instance. Furthermore, it is assumed that the objects try to prevent collisions. If the optimal trajectories of the objects, according to their cost function, are not collision free, this problem cannot be solved by decoupling the problem into motion prediction of each individual object and trajectory planning of the robot, as done in classical dynamic motion planning. The reason for this is that the motion of the objects may have an influence on each other and on the robot. The goal of the proposed navigation approach is to find a trajectory which leads to the robot goal location while minimizing the collision probability. Additionally, the robot should reason about the collision probability of the other objects to minimize the risk of an indirect crash. A collision-free trajectory of the robot can lead to a collision between two other objects as shown in Fig. 1.

III. OVERALL COLLISION PROBABILITY

For navigation in dynamic environments it is often not reasonable to plan the robot motion all the way to its goal location, since the prediction of a highly dynamic environment is not reliable over a long time horizon due to

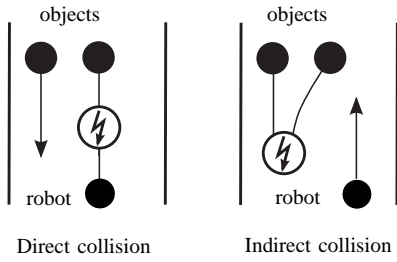


Fig. 1. Collision between two objects caused by the robot.

uncertainties. Thus the robot plan needs many adaptations, which are often unnecessary, since they take place in the distant future.

One possible solution for this problem is to make use of *partial motion planning* (PMP) proposed by [9]. The PMP approach calculates the best partial motion to the goal for a certain time horizon. The partial trajectories need to be verified for future safety, which is done by checking if the final state is an ICS or not. This idea is generalized to a probabilistic framework which is applied to environments with uncertain information. The safety of a trajectory \tilde{u} is determined by the collision probability of the trajectory $P(C|\tilde{u})$ itself and the PCS value of its last state $\text{PCS}(s(T))$. The collision probability of the trajectory considers the time interval $I_t = [0, T]$ and the PCS calculation indicates the future collision probability for the time interval $I_t^+ = (T, \infty)$. So the overall collision probability of a trajectory for an infinite time horizon is defined as follows:

Definition 1: Overall Collision Probability

The probability of a trajectory leading to a collision during or after the trajectory itself, is defined as:

$$P^\infty(C|\tilde{u}) = 1 - \underbrace{(1 - P(C|\tilde{u}))}_{t \in I_t} \underbrace{(1 - \text{PCS}(\tilde{u}))}_{t \in I_t^+}$$

In the next section, the principle of PCS is recalled as presented in [8] and a novel PCS checker is introduced.

IV. INEVITABLE AND PROBABILISTIC COLLISION STATES

A Probabilistic Collision State (PCS) is a probabilistic generalization of an Inevitable Collision State (ICS) [6] (*aka* Region of Inevitable Collisions (RIC) [10] or Obstacle Shadow [11]). In the following, the definition of ICS and PCS are recalled.

In order to precisely define inevitable collision states some notations have to be introduced. The state $s(t)$ and input $\mathbf{u}(t)$ 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 $\mathbf{u}(t)$, the dynamics of the robot is determined by the nonlinear differential equation $\dot{s} = m(s, \mathbf{u})$. 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{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 2: 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 collision with another workspace object.

This definition is extended to a probabilistic setting as presented in [8] and [7]. It is necessary to calculate the probability $P_i(C|\tilde{u})$ that the robot system, applying the input trajectory \tilde{u} , has a collision with the i th object. In [8] only the maximum collision probability regarding all objects was used. This is replaced by the product of their probabilities, that no collision occurs

$$P(C|\tilde{u}) = 1 - \prod_{i=1}^{N_b} (1 - P_i(C|\tilde{u})).$$

Since the robot system can choose any input trajectory from the set of possible input trajectories $\tilde{\mathcal{U}}$, the input trajectory causing the minimum collision probability allows to define the probability of an inevitable collision state.

Definition 3: Probabilistic Collision State

The probability of a state leading to a collision is defined as the minimum collision probability under the best possible input trajectory:

$$\text{PCS}(s) = \min_{\tilde{u} \in \tilde{\mathcal{U}}} P(C|\tilde{u})$$

In the following section a possible implementation of the PCS concept is shown based on Monte Carlo simulation of the workspace objects.

V. PCS CHECKER BASED ON MONTE CARLO SIMULATION

Definition 3 of PCS 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. This leads to a conservative computation of an PCS.

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. In [8] a model-based PCS checker was presented, which has some drawbacks since only one trajectory of each object is considered. Therefore, a novel PCS checker

is introduced which is based on Monte Carlo simulation allowing to investigate multiple braking trajectories of each object.

A. Estimation of Collision Probabilities

As described in Sec. IV, it is necessary to determine the collision probability between the robot and an object until all objects come to a standstill for determining the PCS of a robot state. The time interval during all objects come to a standstill is denoted by $I_t = [0, T_b]$ and T_b is the brake time. For calculating the collision probability, the stochastic variable \mathbf{u}_i is introduced, which contains the discretized control inputs for the time interval I_t for object i :

$$\mathbf{u}_i = (u_{i,1}, \dots, u_{i,N_c}), \quad u \in \mathcal{U}$$

where N_c is the number of control inputs. For the discretization, the time T_c is used and during the time interval $I_c = [kT_c, (k+1)T_c]$ the control input is constant. Using the motion model $\dot{\mathbf{s}} = m(\mathbf{s}, \mathbf{u})$, the object is simulated and the object states are obtained. A sampling time T_s ($T_s < T_c < T_b$) is defined for discretizing the object states

$$\mathbf{s}_i(\mathbf{u}_i) = (s_{i,1}, \dots, s_{i,N_d}).$$

State \mathbf{s}_i contains all sampled states of the object control input \mathbf{u}_i . This leads to the collision probability that the i th object collides with the robot trajectory \tilde{u} during the time interval I_t

$$P_i(C|\tilde{u}) = \int_{\mathcal{U}^B} \text{Ind}(C|\tilde{u}, \mathbf{u}_i) f(\mathbf{u}_i) d\mathbf{u}_i,$$

where $f(\mathbf{u}_i)$ is the density function of the control inputs, \mathcal{U}^B is the set of all possible braking trajectories and $\text{Ind}(C|\tilde{u}, \mathbf{u}_i)$ is an indicator function

$$\text{Ind}(C|\tilde{u}, \mathbf{u}_i) = \begin{cases} 1, & \text{collision occurs} \\ 0, & \text{collision free.} \end{cases}$$

This integral is approximated by Monte Carlo simulation

$$P_i(C|\tilde{u}) \approx \hat{P}_i(C|\tilde{u}) = \frac{1}{N_s} \sum_{n=1}^{N_s} \text{Ind}(C|\tilde{u}, \mathbf{u}_{i,n}), \quad (1)$$

where $\mathbf{u}_{i,n}$ is the n th sampled trajectory of the object i and N_s is the number of evaluated samples.

By the strong law of large numbers $\hat{P}(C|\tilde{u})$ will almost surely (a.s.) converge to $P(C|\tilde{u})$ [12] with $N_s \rightarrow \infty$. In other words, the probability that the robot trajectory will collide with an object is the ratio between the number of object trajectories leading to a collision and collision-free trajectories. As described in Sec. IV, the collision probability $P(C|\tilde{u})$ considering all objects is used to calculate the PCS of a robot state \mathbf{s} .

B. Generation of Braking Trajectories

Trajectories for the objects need to be generated in order to get the discretized object states $\mathbf{s}_i(\mathbf{u}_i)$. Therefore, the initial state is sampled according to the according density function representing the uncertainty. In addition, the input

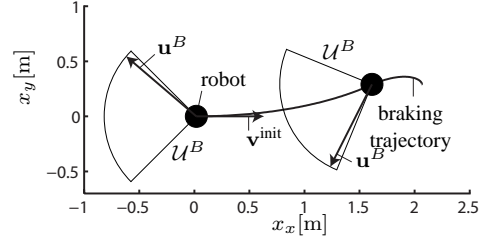


Fig. 2. Exemplary generation of a braking trajectory of the robot with two different acceleration directions. The direction of the acceleration is in the relative coordinate system of the object.

space of the objects is uniformly sampled and applied to the motion model of the object. The same algorithm is used for generating the brake trajectories for the robot. Since only braking trajectories are used, the reduced control input set $\mathcal{U}^B \subset \mathcal{U}$ is applied. It is easier to give all control inputs in the relative coordinate system of the object and not in the global workspace coordinates. The control input $\mathbf{u}^B \in \mathcal{U}^B$ is given in polar coordinates $[u_r^B, u_\theta^B]$ with azimuth $u_\theta^B \in (\frac{3}{4}\pi, -\frac{3}{4}\pi)$ and the radius $u_r^B \in [a_{\min}, a_{\max}]$. Under the assumption of $a_{\min} > 0$, the object comes to a standstill within a finite time horizon. Fig. 2 illustrates one example braking trajectory. The resulting collision probability $P_i(C|\tilde{u})$ for one robot trajectory \tilde{u} is according to Eq. (1) the ratio of the number of crashed trajectories and the number of all evaluated trajectories of the i th object.

C. Collision Detection

A collision between two objects is determined by checking the distance between two objects. If the distance is smaller than the sum of both radii, a collision occurs. An efficient method for objects with polygon shape is the OBB-Tree algorithm [13].

D. Remarks

It is also possible to generate not only braking trajectories in order to get a less conservative approximation of PCS. The only requirement is, that all workspace objects come to a standstill within a finite time horizon. But simulating longer trajectories increases the computational effort which could be used for the safety assessment of more trajectory candidates.

VI. NAVIGATION APPROACH

In the previous section a novel PCS checker was introduced which allows to evaluate the safety of a final state of trajectory candidates. In this section, an algorithm is presented, which generates these trajectories by using Monte Carlo simulation. The goal is to generate promising trajectories for all workspace objects. The result of the trajectory generation and the evaluation of the collision probabilities will lead to a novel framework for navigation in uncertain and dynamic environments. It is inspired by the work of [14]–[16] which uses Monte Carlo simulation for threat analysis of road scenes.

As described in Sec. II it is necessary to consider the safety

of the robot and all other workspace objects. That means the robot has to check if its trajectory will not collide with an object and if its trajectory will not lead to a collision between other objects. Since the trajectories may influence each other, there exists a mutual dependency of each trajectory. To break up this dependency, independent trajectories for each workspace object are generated by Monte Carlo simulation for a fixed time horizon. This allows to calculate an individual collision probability of each object. Furthermore, the robot chooses the most promising trajectory according to a cost function considering the overall collision probability.

A. Cost Function

Similar to [16], two events are defined: C_A , collision between any object and the robot and C_B , collision between any objects, excluding the robot. The aim is to calculate the two probabilities $P(C_A|\tilde{u})$ and $P(C_B|\tilde{u})$ for a robot trajectory \tilde{u} , which are the two essential elements for the following cost function. The cost function $C(\cdot)$ is introduced, evaluating each trajectory candidate \tilde{u} of the robot

$$C(\tilde{u}) = \alpha P^\infty(C_A|\tilde{u}) + \beta P_{\max}(C_B|\tilde{u}) + \gamma(1 - g(\tilde{u})),$$

where $P^\infty(C_A|\tilde{u})$ is the overall collision probability including the robot and $P_{\max}(C_B|\tilde{u})$ is the maximum collision probability excluding the robot, considering all workspace objects. Additionally, the goal function $g(\cdot)$ is used to rank each trajectory according to goal directness and smoothness as explained in Sec. VI-B.

1) *Collision Including the Robot:* For the calculation of $P(C_A|\tilde{u})$ it is required to calculate the collision probability between the robot trajectory \tilde{u} and the i th object

$$P_i(C_A|\tilde{u}) = \int_{\mathcal{U}^{C_A}} \text{Ind}(C_A|\tilde{u}, \mathbf{u}_i) f(\mathbf{u}_i) d\mathbf{u}_i,$$

where \mathcal{U}^{C_A} is the subset of object trajectories which do not collide with another object excluding the robot. The difference between the calculation of the PCS analysis is, that *not* only braking trajectories are considered and that the time interval $I_t = [0, T_h]$ is used. T_h is the considered prediction horizon for the trajectories and replaces the brake time T_b . The set of all braking trajectories \mathcal{U}^B is replaced by the set of all control inputs \mathcal{U} . The collision probability considering all objects is defined as

$$P(C_A|\tilde{u}) = 1 - \prod_{i=1}^{N_b} (1 - P_i(C_A|\tilde{u})).$$

2) *Collision Excluding the Robot:* The calculation of $P(C_B|\tilde{u})$ is similar to the calculation of $P(C_A|\tilde{u})$. Only the subset of object trajectories \mathcal{U}^{C_B} are considered, which are not colliding with the robot. The probability of collision excluding the robot

$$P_i(C_B|\tilde{u}) = \int_{\mathcal{U}^{C_B}} \text{Ind}(C_B|\mathbf{u}_i) f(\mathbf{u}_i) d\mathbf{u}_i$$

is determined by considering all future control inputs \mathcal{U}^{C_B} which do not collide with the robot. Where $\text{Ind}(C_B|\mathbf{u})$ is

an indicator function, which is 1 if a collision between any objects occurs and 0 if no collision occurs. The collision probability considering all objects

$$P_{\max}(C_B|\tilde{u}) = \max_{i=1 \dots N_b} P_i(C_B|\tilde{u})$$

is defined by the maximum collision probability of all objects in the workspace.

B. Estimation of Collision Probabilities

As presented in [16] the trajectories are evaluated to model the fact, that objects have a goal they want to reach. Furthermore, they have a preferred velocity and want to prevent high accelerations and fast changes in the motion direction. Hence, a goal function $g(\cdot)$ is introduced, which evaluates each sample according to this behavior. Since the goal function of [16] was designed for traffic scenarios, the path deviation is replaced by a goal directness and the lateral and longitudinal accelerations are replaced by the absolute acceleration.

$$g(\mathbf{u}_i) = \lambda_1 \underbrace{d(s_i^{\text{goal}}, s_i^{\text{final}})}_{\text{goal directness}} + \int_{I_t} [\lambda_2 \underbrace{(v(t) - v_d)^2}_{\text{favors desired velocity } v_d} + \lambda_3 \underbrace{a(t)^2}_{\text{favors smooth trajectories}}] dt$$

Where $d(\cdot)$ is a distance function between the final state of the object s_i^{final} and its goal state s_i^{goal} . This goal function is used to determine the density function

$$f(\mathbf{u}_i) = a e^{-g(\mathbf{u}_i)},$$

where a is a normalizing constant. Since it is not possible to directly generate samples according to $f(\cdot)$ importance sampling [17] is used to overcome this problem.

For importance sampling, the samples are generated from a different distribution rather using the distribution of interest. The samples are now generated with respect to the density $p(\cdot)$, which is called the *importance sampling density*. Furthermore, weighted samples are used leading to the weighted sample estimator [17]

$$\hat{P}_i(C|\tilde{u}) = \frac{\sum_{n=1}^{N_s} \text{Ind}(C|\tilde{u}, \mathbf{u}_{i,n}) w_n}{\sum_{n=1}^{N_s} w_n}.$$

Where $w_n = \frac{f(\mathbf{u}_{i,n})}{p(\mathbf{u}_{i,n})}$ is the weight of the random sample $\mathbf{u}_{i,n}$. In this work, a uniform importance sampling density $p(\mathbf{u}_{i,n}) = c, \forall \mathbf{u}_{i,n} \in \mathcal{U}$ is used, that simplifies the weights to $w_n = c f(\mathbf{u}_{i,n})$ and the weighted sample estimator to

$$\hat{P}_i(C|\tilde{u}) = \sum_{n=1}^{N_s} \text{Ind}(C|\tilde{u}, \mathbf{u}_{i,n}) w_n,$$

where $\mathbf{u}_{i,n}$ is the n th sampled trajectory of the i th object. Loosely speaking, the weighted samples $\{\mathbf{u}_{i,n}, w_n\}$ can be seen as an approximation of $f(\cdot)$ which is used for generating the samples for the following PCS analysis. The

TABLE I
SIMULATION PARAMETERS

Object Properties	
Thirty regions for x_x [m]	$[0.4, 0.6] \cdots [6.2, 6.4]$
One region for x_y [m]	$[-1.0, 1.0]$
Initial velocity direction $\angle \mathbf{v}$ [rad]	$[\frac{3}{4}\pi, \frac{5}{4}\pi]$
Initial absolute velocity $\ \mathbf{v}\ $ [$\frac{m}{s}$]	$[1.0, 2.0]$
Maximum velocity v_{\max} [$\frac{m}{s}$]	2.0
Maximum acceleration a_{\max} [$\frac{m}{s^2}$]	2.0
Radius [m]	0.2
Initial covariance Σ	$\begin{bmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix}$
Trajectory Generation	
Sampling time T_s [s]	0.025
Sampling time of input T_c [s]	0.25
Time horizon T_h [s]	1.0
Number of object samples	20
Number of robot samples	10
For PCS Checker	
Number of object braking trajectories	5
Number of robot braking trajectories	5
Minimum acceleration a_{\min} [$\frac{m}{s^2}$]	1.0

object trajectories for calculating the collision probabilities $P(C_A|\tilde{u})$ and $P(C_B|\tilde{u})$ are generated in the same way as for the PCS calculation, except that the complete control input set \mathcal{U} is used.

VII. SIMULATION

In this section, simulations of the presented navigation approach are performed. The following simulations show the usefulness of the overall collision probability $P^\infty(C|\tilde{u})$ and compare the results to the collision probability of the trajectory without considering PCS. Furthermore, simulations are used to show the necessity of considering the collision probability between workspace objects which depends on the robot trajectory. Finally, the proposed navigation algorithm from Sec. VI is applied to a multi-robot problem.

A. Overall Collision Probability

In order to show the usefulness of the overall collision probability, random scenarios are generated and the difference between the overall and the trajectory collision probability is determined. Despite the workspace objects, the initial state of the robot is fixed and has the initial state $\mathbf{s} = [0\text{m} \ 0\text{m} \ 1.5\frac{m}{s} \ 0\frac{m}{s}]^T$. The obstacles are placed randomly in front of the robot facing towards it. Each of the scenarios consists of one robot and three workspace objects. The workspace objects are placed randomly in one of thirty predefined adjacent regions which are partitioned in x -direction. Each region is evaluated by 50 trials and the applied parameters for the objects are listed in Tab. I.

An example scenario using the listed parameters is shown in Fig. 3. To verify the necessity of the *overall collision probability*, the mean \bar{D} and the maximum difference D_{\max} of $P^\infty(C_A|\tilde{u}) - P(C_A|\tilde{u})$ are obtained from all scenarios

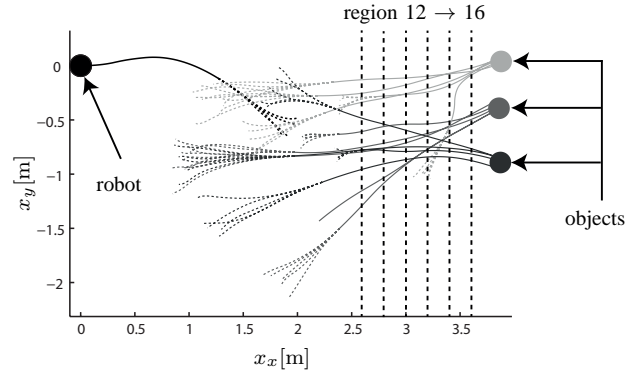


Fig. 3. Random scenario for region 18 in x -direction.: The solid lines depict the trajectories within the planning phase and the dashed lines depict the braking trajectories for the PCS calculation.

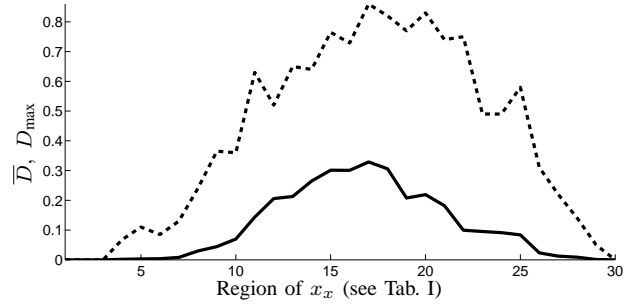


Fig. 4. The solid line depicts the mean difference and the dashed line depicts the maximum difference of one x -region between $P(C_A|\tilde{u})$ and $P^\infty(\tilde{u})$.

and the result is shown in Fig. 4.

It can be seen that there is a significant difference between the trajectory collision probability and the overall collision probability. The maximum achieved difference is 86%. It can also be seen that the difference 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.

B. Collision Excluding the Robot

In order to show the necessity of considering the collision probability between workspace objects excluding the robot $P(C_B|\tilde{u})$, random scenarios are generated. Therefore, the same setup as in Sec. VII-A is used. However, the workspace objects are only generated for the 10th x -region and 100 scenarios including 10 different robot trajectories are investigated. Hence, the relative difference of $P(C_B|\tilde{u}^C)$ and $P(C_B)$ is determined for 1000 robot trajectories \tilde{u} . Since we want to prevent collisions, we are only interested in scenarios, in which the robot trajectory \tilde{u} leads to a higher collision probability between the workspace objects. These trajectories are denoted as \tilde{u}^C . To obtain the difference of both probabilities, the relative mean difference of $P(C_B|\tilde{u}^C)$ and $P(C_B)$ and the maximum relative difference for all scenarios and trajectories \tilde{u}^C are computed. The simulation results are shown in Tab. II. It can be seen that there is a significant difference in 17% of all cases with a mean relative difference of 10.5 and the maximum observed relative difference was

TABLE II
SIMULATION RESULTS

Number of \tilde{u}	1000
Number of \tilde{u}^C	171 (17.1%)
Mean value of $\left \frac{P(C_B \tilde{u}^C) - P(C_B)}{P(C_B)} \right $	10.5
Max value of $\left \frac{P(C_B \tilde{u}^C) - P(C_B)}{P(C_B)} \right $	39.3

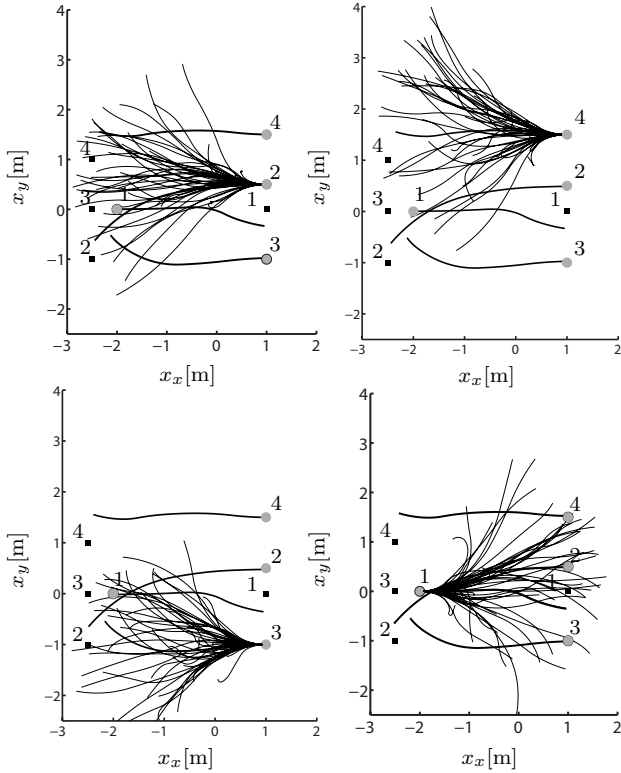


Fig. 5. The results for each workspace object with the proposed navigation algorithm is illustrated. The blue lines depict all trajectory candidates and the green line is the most promising one. The squares illustrate the goal position for each object.

39.3.

C. Navigation Approach

The proposed navigation algorithm from Sec. VI is also suited for multi-robot navigation without communication. Therefore, one scenario is simulated and each of the workspace objects is applied with the same navigation algorithm. The results are illustrated in Fig. 5. Four trajectories are found, which are not colliding during the time horizon $T_h = 2s$. The parameters presented in [14] are used for the goal function and the weights for the cost function are: $\alpha = 0.5$, $\beta = 0.5$, $\gamma = 0.4$.

VIII. CONCLUSION AND FUTURE WORK

This paper consists of two major contributions, the definition of the overall collision probability which assesses the safety of partial trajectories and a novel navigation algorithm is presented which considers the collision probability between workspace objects. The proposed definition

allows to reason about the safety of a planned trajectory and its future. The simulations showed that it is necessary to consider not only the collision probability of the robot but rather considering additionally the collision probability of all workspace objects. Otherwise, a safe trajectory of the robot can lead to a collision between the workspace objects. It is planned to implement the navigation algorithm on a robot to verify the results.

IX. 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). Special thanks to Matthias Althoff for the inspiring discussions.

REFERENCES

- [1] R. Philippsen, “Motion planning and obstacle avoidance for mobile robots in highly cluttered dynamic environments,” Ph.D. dissertation, ETH Zürich, Institute of Robotics and Intelligent Systems, 2004.
- [2] C. Fulgenzi, “Autonomous navigation in dynamic uncertain environment using probabilistic models of perception and collision risk prediction,” Ph.D. dissertation, INRIA Rhne-Alpes, 2009.
- [3] M. Bennewitz, “Mobile robot navigation in dynamic environments,” Ph.D. dissertation, University of Freiburg, Department of Computer Science, 2004.
- [4] J. P. van den Be, “Path planning in dynamic environments,” Ph.D. dissertation, Utrecht University, Utrecht, The Netherlands, 2007.
- [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] T. Fraichard and H. Asama, “Inevitable collision states. A step towards safer robots?” *Advanced Robotics*, vol. 18, pp. 1001–1024, 2004.
- [7] A. Bautin, L. Martinez-Gomez, and T. Fraichard, “Inevitable collision states: a probabilistic perspective,” in *Proc. of the IEEE Int. Conference on Robotics and Automation*, 2010.
- [8] D. Althoff, M. Althoff, D. Wollherr, and M. Buss, “Probabilistic collision state checker for crowded environments,” in *Proc. of the IEEE Int. Conference on Robotics and Automation*, 2010.
- [9] S. Petti and T. Fraichard, “Safe motion planning in dynamic environments,” in *Proc. of the IEEE Int. Conference on Intelligent Robots and Systems*, 2005.
- [10] 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.
- [11] J. Reif and M. Sharir, “Motion planning in the presence of moving obstacles,” *J. ACM*, vol. 41, pp. 764–790, 1994.
- [12] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan, “An introduction to mcmc for machine learning,” *Machine Learning*, vol. 50, pp. 5–43, 2003.
- [13] S. Gottschalk, M. C. Lin, and D. Manocha, “Obb-tree: A hierarchical structure for rapid interference detection,” in *Proc. on ACM Siggraph* 96, 1996.
- [14] A. Broadhurst, S. Baker, and T. Kanade, “Monte carlo road safety reasoning,” in *Proc. of the IEEE Intelligent Vehicle Symposium*, 2005.
- [15] S. Danielsson, L. Petersson, and A. Eidehall, “Monte carlo based threat assessment: Analysis and improvements,” in *Proc. of the IEEE Conference on Intelligent Vehicles Symposium*, 2007.
- [16] A. Eidehall and L. Petersson, “Statistical threat assessment for general road scenes using monte carlo sampling,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, pp. 137–147, 2008.
- [17] J. Handschin and D. Mayne, “Monte carlo techniques to estimate the conditional expectation in multi-stage non-linear filtering,” *Int. J. Control*, vol. 9, pp. 547–559, 1969.