Safety Assessment of Trajectories for Navigation in Uncertain and Dynamic Environments

Daniel Althoff, Dirk Wollherr and Martin Buss

By: José Baca
Motivation

- Robots are finding their way into daily life of humans.
  - Navigate autonomously to known places.
  - Guide people at the Expos.
  - Assist elderly and disabled people.

Problem!

- Due to the presence of humans, motion safety of these robotic systems has to be taken into account.

- In static environments, the robot is collision-free for an infinite time-horizon.

- In dynamic environments, motion safety is more challenging, since the future behavior of other moving objects has to be considered.
This paper presents:

- Probabilistic framework for reasoning about the safety of robot trajectories in dynamic and uncertain environments with imperfect information about the future motion of surrounding objects.
- For safety assessment, the overall collision probability is used to rank candidate trajectories by considering the probability of colliding with known objects as well as the estimated collision probability beyond the planning horizon.

- They introduce a safety assessment cost metric, the probabilistic collision cost, which considers the relative speeds and masses of multiple moving objects in which the robot may possibly collide with.
- The results are integrated into a navigation framework that generates and selects trajectories that strive to maximize safety while minimizing the time to reach a goal location.
- Simulation scenarios are used to validate the overall crash probability.
Notations and preliminary definitions

The state $s(t)$ and input $u(t)$ of the considered robot system $A$ (for a point in time $t$) can take values from the state space $S$ and the control space $U$. The state $s$ of an object is represented by its position $p = \{p_x, p_y\}$ and its velocity $v = \{v_x, v_y\}$. For a given initial state $s(0)$ and a control input $u(t)$, the future states are computed by the robot model $m(\cdot)$.

In the remainder of the paper the model $\dot{s} = m(s, u)$ is used

\[
\begin{bmatrix}
\dot{x}_x \\
\dot{x}_y \\
\dot{v}_x \\
\dot{v}_y
\end{bmatrix}
\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}
\begin{bmatrix} x_x \\ x_y \\ v_x \\ v_y \end{bmatrix}
+ \begin{bmatrix} 0 \\ 0 \\ u_1 \\ u_2 \end{bmatrix} a_{\text{max}}
\]

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

• Assume a set of $n$ objects sharing an environment together with one robot.

• Each robot has an initial state $s_{\text{init}}^i$ and a goal state $s_{\text{goal}}^i$.

• No information about their navigation algorithms or the future trajectories are given.

• 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.
Problem

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 uncertainties.

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.
Fig. 1 Without considering the avoidance possibilities of the objects, no path can be found to prevent a collision. By incorporating this ability, it is possible to find a safe path.

Fig. 2 Possible kinds of collision in a scene with two objects and a robot.

Fig. 3 At time $t_0$ both possible robot trajectories are safe during the planning horizon, but at time $t_1$ one trajectory will inevitably lead to a collision.
Overall Collision Probability

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

\[ P^\infty(C|\tilde{u}) = 1 - (1 - P(C|\tilde{u}))(1 - PCS(\tilde{u})) \]

\[ \text{In where:} \]
- Collision probability of the trajectory \( P(C|\tilde{u}) \)
- Trajectory \( \tilde{u} \)
- Time interval \( I_t = [0, T] \quad I_t^+ = (T, \infty) \)
- Probabilistic Collision State \( PCS \)
Probabilistic Collision State (PCS) - 1

- It is a probabilistic generalization of an Inevitable Collision State (ICS) described in


And it is defined as

\[
\forall \tilde{u} \in \tilde{U}, \exists t, \exists B_i, A(\tilde{u}(t)) \cap B_i(t) \neq \emptyset.
\]

In where:

- Unified occupancy of all objects (written in short notation) \( \mathcal{B} = \bigcup_{i=1,\ldots,N_b} B_i \)
- Number of workspace objects \( N_b \)
- Set of input trajectories \( \tilde{U} \)
- Workspace occupancy generated from the input trajectory \( A(\tilde{u}(t)) \)

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

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


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

\[
\text{PCS}(s) = \min_{\tilde{u} \in \tilde{U}} P(C, \tilde{u})
\]
However...

This definition is not implementable!

1. There is an infinite number of input trajectories
2. There is an unlimited time horizon.

- The infinite number of input trajectories $\ddot{u}$ 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.
PCS checker based on Monte Carlo simulation (i)

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

1. Estimation of Collision Probabilities

\[ P_i(C|\tilde{u}) \approx \hat{P}_i(C|\tilde{u}) = \frac{1}{N_s} \sum_{n=1}^{N_s} \text{Ind}(C|\tilde{u}, u_{i,n}), \]

In where:
- The \( n \)th sampled trajectory of the object \( i \)
- Number of evaluated samples \( N_s \)

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.
PCS checker based on Monte Carlo simulation (ii)

2. Generation of Braking Trajectories

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.

3. 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.
Navigation Approach

Previous PCS checker allows to evaluate the safety of a final state of trajectory candidates.

In the navigation approach, the algorithm generates trajectories by using Monte Carlo simulation.

**Goal:**
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.
Navigation Approach

A. Cost Function

Two events are defined:

• \( C_A \), Collision between any objects and the robot.

• \( C_B \), Collision between any objects, excluding the robot.

The aim is to calculate the two probabilities \( P(C_A | \tilde{u}) \) and \( P(C_B | \tilde{u}) \).

The cost function \( C(.) \) is introduced, evaluating each trajectory candidate \( \tilde{u} \) of the robot.
Navigation Approach

A. Cost Function

The cost function \( C(\cdot) \) is introduced, evaluating each trajectory candidate of the robot.

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

In where:

- Overall collision probability including the robot
- Maximum collision probability excluding the robot, considering all workspace objects.
- Goal function (to rank each trajectory according to goal directness and smoothness)
Navigation Approach

A. Cost Function

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

$$P_i(C_A|\tilde{u}) = \int_{U_{CA}} \text{Ind}(C_A|\tilde{u}, u_i) f(u_i) \, du_i,$$

where $U_{CA}$ is the subset of object trajectories which do not collide with another object excluding the robot. The

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 $U_{CB}$ are considered, which are not colliding with the robot. The probability of collision excluding the robot

$$P_i(C_B|\tilde{u}) = \int_{U_{CB}} \text{Ind}(C_B|u_i) f(u_i) \, du_i$$

Where

$\text{Ind}(C_B|u)$ is an indicator function.

• 1 if a collision between any objects occurs
• 0 if no collision occurs.
Simulation

**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 $s = [0\text{ m} \ 0\text{ m} \ 1.5\frac{\text{m}}{\text{s}} \ 0\frac{\text{m}}{\text{s}}]^T$. 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.
Simulation

Overall Collision Probability

<table>
<thead>
<tr>
<th>TABLE 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIMULATION PARAMETERS</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Object Properties</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Thirty regions for $x_a [m]$</td>
<td></td>
</tr>
<tr>
<td>One region for $x_a [m]$</td>
<td></td>
</tr>
<tr>
<td>Initial velocity direction $\alpha [\text{rad}]$</td>
<td>$[\frac{3}{4} \pi, \frac{5}{4} \pi]$</td>
</tr>
<tr>
<td>Initial absolute velocity $</td>
<td></td>
</tr>
<tr>
<td>Maximum velocity $v_{\text{max}} [\text{m/s}]$</td>
<td>2.0</td>
</tr>
<tr>
<td>Maximum acceleration $a_{\text{max}} [\text{m/s}^2]$</td>
<td>2.0</td>
</tr>
<tr>
<td>Radius [m]</td>
<td>0.2</td>
</tr>
<tr>
<td>Initial covariance $\Sigma$</td>
<td>$\begin{bmatrix} 0.01 &amp; 0 &amp; 0 &amp; 0 \ 0 &amp; 0.01 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 0.01 &amp; 0 \ 0 &amp; 0 &amp; 0 &amp; 0.01 \end{bmatrix}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Trajectory Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sampling time $T_s [s]$</td>
</tr>
<tr>
<td>Sampling time of input $T_c [s]$</td>
</tr>
<tr>
<td>Time horizon $T_h [s]$</td>
</tr>
<tr>
<td>Number of object samples</td>
</tr>
<tr>
<td>Number of robot samples</td>
</tr>
</tbody>
</table>

For PCS Checker

<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of object braking trajectories</td>
</tr>
<tr>
<td>Number of robot braking trajectories</td>
</tr>
<tr>
<td>Minimum acceleration $a_{\text{min}} [\text{m/s}^2]$</td>
</tr>
</tbody>
</table>
Simulation

Overall Collision Probability

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.
Simulation

**Overall Collision Probability**

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.

![Graph showing the difference between trajectory and overall collision probability](image)

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

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: $\alpha = 0.5, \beta = 0.5, \gamma = 0.4$. 
Simulation

Navigation Approach

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.
Conclusion

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.

The presented navigation approach is based on three criteria for motion safety:
• Consider its own dynamics
• Consider the environment object’s future behavior
• Reason over an infinite time-horizon

• Additionally, it considers the avoidance possibilities of the objects and incorporates their motion safety.

• The presented algorithm is specially useful for crowded environments where it is crucial to take into account the perception and collision avoidance capabilities of the other workspace objects.