On Provably Safe Obstacle Avoidance for Autonomous Robotic Ground Vehicles

Stefan Mitsch  
Carnegie Mellon University

Khalil Ghorbal  
Carnegie Mellon University

Andre Platzer  
Carnegie Mellon University

Follow this and additional works at: http://repository.cmu.edu/compsci

Published In
On Provably Safe Obstacle Avoidance for Autonomous Robotic Ground Vehicles

Stefan Mitsch, Khalil Ghorbal and André Platzer
Carnegie Mellon University, Computer Science Department
5000 Forbes Avenue, Pittsburgh PA 15213, USA,
Email: {smitsch,kghorbal,aplatzer}@cs.cmu.edu,
WWW home page: http://www.symbolaris.com

Abstract—Nowadays, robots interact more frequently with a
dynamic environment outside limited manufacturing sites and in
close proximity with humans. Thus, safety of motion and obstacle
avoidance are vital safety features of such robots. We formally
study two safety properties of avoiding both stationary and mov-
ing obstacles: (i) passive safety, which ensures that no collisions
can happen while the robot moves, and (ii) the stronger passive
friendly safety in which the robot further maintains sufficient
maneuvering distance for obstacles to avoid collision as well. We
use hybrid system models and theorem proving techniques that
describe and formally verify the robot’s discrete control decisions
along with its continuous, physical motion. Moreover, we formally
prove that safety can still be guaranteed despite location and
actuator uncertainty.

I. INTRODUCTION

With the increased introduction of autonomous robotic
ground vehicles as consumer products—such as autonomous
household appliances [17] or driverless cars on regular Californi-
nian roads [14]—we face an increased need for ensuring safety not
only for the sake of the consumer, but also the manufacturer.
Since those robots are designed for environments that occupy
stationary as well as moving obstacles, motion safety and
obstacle avoidance are vital safety features for such robots
[3, 22, 25, 27].

In this paper, we prove safety for the obstacle avoidance
control algorithm of a robot. One of the conceptual difficulties is
what safety means for an autonomous robot. We would want
it to be collision-free, but that usually requires other vehicles
to be sensible, e.g., not actively try to run into the robot when
it is just stopped in a corner. One way of doing that is to
assume stringent constraints on the behavior of obstacles (see,
e.g., [3, 12]). In this paper, we want to refrain from doing
so and allow arbitrary obstacles with an arbitrary continuous
motion with a known upper bound on their velocity. Then our
robot is safe, intuitively, if no collision can ever happen where
the robot is to blame. The first notion we consider is passive
safety [14]. Passive safety can be guaranteed with minimal
assumptions about obstacles. It requires that the robot does
not actively collide, i.e., if a collision occurs at all then only
while the robot is stopped and the (moving) obstacle ran into
the robot. The difficulty with that notion is that it still allows
the robot to go kamikaze and stop in unsafe places, creating
unavoidable collision situations in which an obstacle has no
control choices left that would prevent a collision. The second
notion we consider is passive friendly safety [14], which aims
for more careful robot decisions that respect the features of
moving obstacles (e.g., their braking capabilities). Thus, a
passive friendly robot not only ensures that it is itself able to
stop before a collision occurs, but it also maintains sufficient
maneuvering room for obstacles to avoid a collision as well.

Motion safety and obstacle avoidance lead to interesting
cognitive robotics questions: how much does the robot have to
know about the goals and constraints of other vehicles so as
not to be considered to blame? In this paper, we successively
construct models and proofs that increase the level of assumed
knowledge and explicitly expressed uncertainty. We start with
(i) passive safety, which assumes a known upper bound on the
velocity of obstacles. Then we extend to (ii) passive
friendly safety for a known lower bound of an obstacle’s
braking power and consider an upper bound on the reaction
time that we allow an obstacle to start collision avoidance
attempts. Note that all our models use symbolic bounds so
they hold for all choices of the bounds. As a result, we can
account for uncertainty in several places (e.g., by instantiating
upper bounds on acceleration or time with values including
uncertainty). We additionally show how further uncertainty that
cannot be attributed to such bounds (in particular location and
actuator uncertainty) can be modeled and verified explicitly.

We use the well-known dynamic window algorithm [9]
and verify passive safety and passive friendly safety. Unlike
existing work on obstacle avoidance (e.g., [1, 17, 24–26]), we
use hybrid models and verification techniques that describe
and verify the robot’s discrete control choices along with its
continuous, physical motion. In summary, our contributions are
(i) hybrid models of navigation and obstacle avoidance control
algorithms of robots, and (ii) proofs that they guarantee passive
and passive friendly safety in the presence of stationary and
moving obstacles despite sensor and actuator uncertainty. The
models and proofs of this paper are bundled with KeYmaera.

The paper is organized as follows. In the next section, we
discuss related work on navigation and obstacle avoidance of
robots that focuses on verification. Section [11] recalls
differential dynamic logic that we use as a modeling formalism
for the hybrid dynamics of a robot and the safety constraints. In
Sect. [14] we introduce models of obstacle avoidance control
and physical motion, and prove that they guarantee passive
safety and passive friendly safety with stationary as well as
moving obstacles. We then model uncertainty explicitly and
prove that motion is still safe. Sect. [17] concludes the paper.

1 http://www.nytimes.com/2010/10/10/science/10google.html?_r=0
2 http://symbolaris.com/info/KeYmaera.html
II. RELATED WORK

Isabelle has recently been used to formally verify that a C program implements the specification of the dynamic window algorithm [25]. This is interesting, but the algorithm itself and its impact on motion of the robot was considered in an informal pen-and-paper argument only. We, instead, formally verify the correctness of the dynamic window algorithm using a hybrid verification technique. Thus, we complement the work in [25] in a twofold manner: First, we create hybrid models of the control and the motion dynamics of the robot and formally verify correctness of the dynamic window algorithm control for the combined dynamics. Second, we model stationary as well as moving obstacles and prove two versions of safety. These complementary results together present a strong safety argument from concept (this paper) to implementation [25].

PASSAVID [3] is a navigation scheme, which avoids braking inevitable collision states (i.e., states that regardless of the robot’s trajectory lead to a collision) to achieve safety in the presence of moving obstacles. Since PASSAVID is designed to operate in completely unknown environments, it ensures that the robot is at rest when a collision occurs (passive safety). The motion dynamics of the robot have only been considered in simulation. We prove the stronger passive friendly safety using a hybrid verification technique (i.e., algorithm and motion dynamics), which ensures that the robot does not create unavoidable collision situations by stopping in unsafe places.

For the purpose of guaranteeing infinite horizon safety, velocity obstacle sets [27] assume unpredictable behavior for obstacles with known forward speed and maximum turn rate (i.e., Dubin’s cars). The authors focus only on the obstacle behavior; the robot’s motion is explicitly excluded from their work. We complement their work and show that a robot, which has a known upper bound on its reaction time and considers discs as velocity obstacle sets (i.e., known forward speed and unknown turn rate, as allowed by [27]), moves safely.

Hybrid models of driver support systems in cars [12, 16] have been verified with a model of the continuous dynamics of cars. That points out interesting safety conditions for vehicles on straight lines, but not in the general two-dimensional case that we consider in this work.

Safety of aircraft collision avoidance maneuvers in the two-dimensional plane was verified for constant translational velocity and a rotational velocity that stays constant during the maneuver [13, 22]. Our models include acceleration for both translational and rotational velocity.

LTLMoP contains an approach [23] to guarantee high-level behavior for map exploration when the environment is continuously updated. The approach synthesizes and re-synthesizes plans, expressed in linear temporal logic, of a hybrid controller, when new map information is discovered. This work focuses on preserving the state and task completion history, and thus on guaranteeing that the robot will follow a high-level behavior (e.g., visit all rooms) even when the controller is re-synthesized, not on safe obstacle avoidance.

Pan et al. [17] propose a method to smooth the trajectories produced by sampling-based planners in a collision-free manner. Our paper proves that such trajectories are indeed safe when considering the control choices of a robot and its continuous dynamics.

LQG-MP [26] is a motion planning approach that takes into account the sensors, controllers, and motion dynamics of a robot while working with uncertain information about the environment. The approach assesses randomly generated paths by the approximated probability of a collision with an obstacle. One goal is to select paths with low collision probability; however, guaranteeing collision-free motion is not their focus, since a collision-free path may not even have been generated.

Althoff et al. [11] use a probabilistic approach to rank trajectories according to their collision probability. To further refine such a ranking, a collision cost metric is proposed, which derives the cost of a potential collision by considering the relative speeds and masses of the collision objects.

In summary, this paper addresses safety of robot obstacle avoidance in the following manner.

- Unlike [3, 23–25, 27], we study combined models of the hybrid dynamics in terms of discrete control and differential equations for continuous physical motion of the robot as well as the obstacles, not discrete control alone or only the behavior of obstacles.
- Unlike [12, 13, 16, 22, 25], we verify safety in the two-dimensional plane not one-dimensional space and do not assume constant translational and rotational velocity, but include accelerations for both, as needed for ground vehicles.
- Unlike [1, 17, 23, 24, 26], we produce formal, deductive proofs in a formal verification tool. Note, that in [25], the correctness of the implementation w.r.t. the algorithm’s specification was formally verified, but not the motion of the robot.
- Unlike [17, 23, 25], we verify safety even in the presence of moving obstacles.
- Unlike [3, 17, 23, 25], we verify passive friendly safety, which is important because passive (non-friendly) safe robots may cause unavoidable collisions by stopping in unsafe places so that obstacles will collide with them.
- Unlike [3, 12, 22, 23], we consider sensor and actuator uncertainty in our verification results.
- Unlike [1, 24, 26], we do not minimize or probabilistically minimize collisions, but prove that collisions can never occur (as long as the robot system fits to the model).

III. PRELIMINARIES: DIFFERENTIAL DYNAMIC LOGIC

A robot and the moving obstacles in its environment form a hybrid system: they make discrete control choices (e.g., compute the actuator set values for acceleration, braking, or steering), in which in turn influence their actual physical behavior (e.g., slow down to a stop, move along a curve). Hybrid systems have been considered as joint models for both components, since verification of either component alone does not capture the full behavior of a robot and its environment.
In order to verify safe obstacle avoidance, we use differential dynamic logic \(\text{dC}^{3,20,21}\), which has a notation for hybrid systems as \textit{hybrid programs}. We use hybrid programs for modeling a robot that follows the dynamic window algorithm as well as for modeling the behavior of moving obstacles. \text{dC} allows us to make statements that we want to be true for all runs of the program (safety) or for at least one run (liveness). Both constructs are necessary to verify safety: for all possible control choices and entailed physical motion, our robot must be able to stop, while at the same time there must be at least one possible execution in which the obstacle is able to stop without collision as well. Table I summarizes the syntax of hybrid programs together with an informal semantics.

### Table I: Hybrid program representations of hybrid systems.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\alpha; \beta)</td>
<td>sequential composition, first run (\alpha), then (\beta)</td>
</tr>
<tr>
<td>(\alpha \cup \beta)</td>
<td>nondeterministic choice, following either (\alpha) or (\beta)</td>
</tr>
<tr>
<td>(\alpha^*)</td>
<td>nondeterministic repetition, repeats (\alpha) (n) (\geq 0) times</td>
</tr>
<tr>
<td>(x := \theta)</td>
<td>assign value of term (\theta) to variable (x) (discrete jump)</td>
</tr>
<tr>
<td>(x := *)</td>
<td>assign arbitrary real number to variable (x)</td>
</tr>
<tr>
<td>(\langle x'_1 = \theta_1, \ldots, x'_n = \theta_n, \phi \rangle)</td>
<td>restricted to maximum evolution domain (F)</td>
</tr>
</tbody>
</table>

Sequential composition \(\alpha; \beta\) says that \(\beta\) starts after \(\alpha\) finishes (e.g., first let the robot choose acceleration, then steering angle). The nondeterministic choice \(\alpha \cup \beta\) follows either \(\alpha\) or \(\beta\) (e.g., let the robot decide nondeterministically between remaining stopped or accelerating). The nondeterministic repetition operator \(\alpha^*\) repeats \(\alpha\) zero or more times (e.g., let the robot repeatedly control and drive). Assignment \(x := \theta\) instantaneously assigns the value of the term \(\theta\) to the variable \(x\) (e.g., let the robot choose maximum braking), while \(x := *\) assigns an arbitrary value to \(x\) (e.g., let the robot choose any acceleration). \(x' = \theta\) and \(F\) describes a continuous evolution of \(x\) within the evolution domain \(F\) (e.g., let the velocity of the robot change according to its acceleration, but never lower than zero). The test \(?F\) checks that a particular condition \(F\) holds, and aborts if it does not (e.g., test whether or not the distance to an obstacle is large enough). A typical pattern that involves assignment and tests is to limit the assignment of arbitrary values to known bounds (e.g., limit an arbitrarily chosen acceleration to the physical limits of the robot, as in \(x := *; ?x \leq A\), which says \(x\) is any value less or equal \(A\)).

The set of \text{dC} formulas is generated by the following EBNF grammar (where \(\sim \in \{<, \leq, =, \geq, >\}\) and \(\theta_1, \theta_2\) are arithmetic expressions in \(+, -, \cdot, \div\) over the reals):

\[
\phi ::= \theta_1 \sim \theta_2 \mid \neg \phi \mid \phi \land \psi \mid \phi \lor \psi \mid \phi \rightarrow \psi \mid \forall x \phi \mid \exists x \phi \mid (\alpha) \phi \mid (\langle \alpha \rangle) \phi
\]

Further operations, such as Euclidian norm \(\|\theta\|\) and infinity norm \(\|\theta\|_{\infty}\) of a vector \(\theta\), are definable.

To specify the desired correctness properties of hybrid programs, \text{dC} formulas of the form \(F \rightarrow [\alpha]G\) mean that all executions of the hybrid program \(\alpha\), which start at a state in which formula \(F\) is true, lead to states in which formula \(G\) is true. Dually, formula \(F \rightarrow (\langle \alpha \rangle)G\) expresses that there is a state reachable by the hybrid program \(\alpha\) that satisfies formula \(G\).

Differential dynamic logic comes with a verification technique to prove those correctness properties. We did all our proofs in the verification tool KeYmaera, which implements this verification technique \cite{18,20}. KeYmaera supports hybrid systems with nonlinear discrete jumps, nonlinear differential equations, differential-algebraic equations, differential inequalities, and systems with nondeterministic discrete or continuous input. This makes KeYmaera more readily applicable to robotic verification than other hybrid system verification tools, such as SpaceEx \cite{10}, which focuses on piecewise linear systems. KeYmaera implements automatic proof strategies that decompose hybrid systems symbolically \cite{20}. This compositional verification principle helps scaling up verification, because KeYmaera verifies a big system by verifying properties of subsystems. Strong theoretical properties, including relative completeness results, have been shown about \text{dC} \cite{18,21} indicating how this composition principle can be successful.

### IV. Robotic Ground Vehicle Navigation

The robotics community has come up with a large variety of robot designs, which differ not only in their tool equipment, but also (and more importantly for the discussion in this paper) in their kinematic capabilities. We focus on wheel-based vehicles. In order to make our models applicable to a large variety of robots, we use only limited control options (e.g., do not move sideways to avoid collisions). We consider robots that drive forward (non-negative translational velocity) in sequences of arcs in two-dimensional space. Such trajectories can be realized by robots with single wheel drive, differential drive, Ackermann drive, synchro drive, or omni drive \cite{4}.

Many different navigation and obstacle avoidance algorithms have been proposed for such robots, e.g. dynamic window \cite{9}, potential fields \cite{11}, or velocity obstacles \cite{8}. For an introduction to various navigation approaches for mobile robots, see \cite{2,6}. In this paper, we focus on the dynamic window algorithm \cite{9}, which is derived from the motion dynamics of the robot and thus discusses all aspects of a hybrid system (models of discrete and continuous dynamics). Other approaches, such as velocity obstacles \cite{8}, are interesting for further work on provably safe moving obstacle avoidance.

We want to prove motion safety of a robot that avoids obstacles by dynamic window navigation. Under passive safety \cite{14}, the vehicle is in a safe state if it is able to come to a full stop before making contact with an obstacle (i.e., the vehicle does not itself collide with obstacles, so if a collision occurs at all then while the vehicle was stopped). Passive safety, however, puts the burden of avoiding collisions mainly on other objects. The safety condition that we want to prove additionally is the stronger passive friendly safety \cite{14}: we want to guarantee that our robot will come to a full stop safely under all circumstances, but will also leave sufficient maneuvering room for moving obstacles to avoid a collision \cite{11}.

We consider both models and both safety properties to show the differences between the assumed knowledge and the safety guarantees that can be made. The verification effort and complexity difference is quite instructive: passive safety can be guaranteed by proving safety of all robot choices, whereas passive friendly safety additionally requires liveness proofs for...

---

\(^3\) If the radius of such a circle is infinite, the robot drives (forward) on a straight line.  
\(^4\) The robot ensures that there is enough room for the obstacle to stop before a collision occurs. If the obstacle decides not to, the obstacle is to blame and our robot is still considered safe.
the obstacle. In the following sections, we discuss models and verification of the dynamic window algorithm in detail.

A. Passive Safety of Obstacle Avoidance

The dynamic window algorithm is an obstacle avoidance approach for mobile robots equipped with synchro drive [9] but can be used for other drives too [5]. It uses circular trajectories that are uniquely determined by pairs of translational and rotational velocity \((v, \omega)\). The algorithm is roughly organized into two steps: (i) The range of all possible pairs of translational and rotational velocities is reduced to admissible ones that result in safe trajectories (i.e., avoid collisions since those trajectories allow the robot to stop before it reaches the nearest obstacle). The admissible pairs are further restricted to those that can be realized by the robot within a short time frame (the so-called dynamic window). If the set of admissible and realizable velocities is empty, the algorithm stays on the previous safe curve (such curve exists unless the robot started in an unsafe state). (ii) Progress towards the goal is optimized by maximizing a goal function. For safety verification, we can omit the second step and verify the stronger property that all choices that are fed into the optimization are safe.

\[ a_x = \frac{2v_x\omega_y - v_y\omega_x}{r_x^2 + r_y^2}, \quad a_y = \frac{2v_y\omega_x - v_x\omega_y}{r_x^2 + r_y^2}, \quad a_z = \frac{v_x^2 + v_y^2 - v_c^2}{r_x^2 + r_y^2} \]

(a) Position \((p_x^r, p_y^r)\), translational velocity \(v_r\) and rotational velocity \(\omega_r\) for positive acceleration on a circle.

(b) Position \((p_x^r, p_y^r)\), translational velocity \(v_r\) and rotational velocity \(\omega_r\) for braking to a complete stop on a circle.

(c) Position \((p_x^r, p_y^r)\), translational velocity \(v_r\) and rotational velocity \(\omega_r\) for translational acceleration on a spiral.

(d) \((p_x^r, p_y^r)\) motion plot for acceleration [a] (e) \((p_x^r, p_y^r)\) motion plot for braking [b] (f) \((p_x^r, p_y^r)\) motion plot for [c]

Fig. 1: Trajectories of the robot.

\[ d_x = (\cos \theta, \sin \theta)^T \]

\[ d_c = (\cos \theta, \sin \theta)^T \]

\[ d_c = (\cos \theta, \sin \theta)^T \]

As stated earlier, we study unidirectional motion: the robot moves along its direction, that is the vector \(d_x\) gives the direction of the velocity vector.

5 The derivative with respect to time is denoted by superscript prime (\(^\prime\)).
curve when braking (the robot brakes along the curve and comes to a complete stop before completing the circle). If the rotational velocity is constant (\(\omega_r = 0\)), the robot drives an Archimedean spiral with the translational and rotational accelerations controlling the spiral’s separation distance (\(a_r/\omega^2\)). The corresponding plots are shown in Figures [1] and [2].

As in the dynamic window algorithm, we assume bounds for the acceleration \(a_r\) in terms of a maximum acceleration \(A \geq 0\) and braking power \(b > 0\), as well as a bound \(\Omega\) on the rotational velocity \(\omega_r\). We use \(\varepsilon\) to denote the upper bound for the control loop time interval (e.g., sensor and actuator delays, sampling rate, and computation time). That is, the robot may react as quickly as it wants, but it can take no longer than \(\varepsilon\). Notice that, without such a time bound, the robot would not be safe, because its control might never run. In our model, all these bounds will be used as symbolic parameters and not concrete numbers. Therefore, our results apply to all values of these parameters and can be enlarged to include uncertainty.

An obstacle has vector state variables describing its current position \(p_o = (p^x_o, p^y_o)\) and velocity \(v_o = (v^x_o, v^y_o)\). The obstacle model is very liberal. The only restriction about the dynamics is, that within \(\varepsilon\) time units, they move continuously with bounded velocity \(\|v_o\| \leq V\). Note, that the dynamic window algorithm considers a special case \(V = 0\) (obstacles are stationary). Depending on the relation of \(V\) to \(\varepsilon\), moving obstacles can make quite a difference, e.g., when fast obstacles meet communication-based virtual sensors as in RoboCup[3].

In order to determine admissible velocities, the dynamic window algorithm requires the distance to the nearest obstacle for every possible curve. In the presence of moving obstacles, however, all obstacles must be considered and tested for safety (e.g., in a loop). To capture this requirement, our model nondeterministically picks any obstacle \(p_o := (*,*)\) and tests its safety, which includes safety for the worst-case behavior \(V\) of the nearest obstacle (ties are included) and is thus safe for all possible obstacles. In the case of non-point obstacles, \(p_o\) denotes the obstacle perimeter point that is closest to the robot (this fits naturally to obstacle point sets delivered by radar sensors, from which the closest point will be chosen). At each active step of the robot, the position \(p_o\) is updated (again nondeterministically to summarize all obstacles). In this process, the robot may or may not find another safe trajectory. If it does, the robot can follow that safe trajectory w.r.t. any nondeterministically chosen obstacle (again, \(V\) ensures that all other obstacles will stay more distant than the worst case of the nearest). If not, the robot can still brake on the previous trajectory, which is known to be safe.

Model[1] represents the common controller-plant model: it repeatedly executes control choices followed by dynamics, cf. [1]. For the sake of clarity we restrict the study to circular trajectories with constant positive radius, that is \(|p_r - p_c| > 0\), which also covers straight line trajectories if the radius is infinite. Thus, the condition \(|p_r - p_c| |\omega_r|\) of \(\omega_r\) can be rewritten as \(\omega_r = \frac{a_r}{|p_r - p_c|}\). We have bundled adjusted dynamics for spinning behavior \(|p_r - p_c| = 0, \omega_r \neq 0\) and Archimedean spiral \((\omega_r = 0, a_r \neq 0)\) with KeYmaera. The control of the robot is executed in parallel to that of the obstacle, cf. [2]. The obstacle itself may choose any velocity in any direction up to the worst-case velocity \(V\) assumed about obstacles (\(\|v_o\| \leq V\)), cf. [3]. This uses the modeling pattern from Section [11] we assign an arbitrary value to the obstacle’s velocity \((v_o := (*,*))\), which is then restricted to any value up to the worst-case velocity using a subsequent test \(\|v_o\| \leq V\).

The robot is allowed to brake at all times since [3] has no test. If the robot is stopped, it may choose to stay in its current spot, cf. [5]. Finally, the robot may choose a new safe curve in its dynamic window: it chooses any acceleration within the bounds of its braking power and acceleration [6], and any rotational velocity in the bounds, cf. [7]. This corresponds to testing all possible acceleration and rotational velocity values at the same time. An actual implementation would use loops to enumerate all possible values and all obstacles and test each pair \((v_o, \omega_r)\) separately w.r.t. every obstacle. The admissible pairs would be stored in a data structure (as e.g., in [25]).

The curve is determined by the robot following a circular trajectory around the rotation center \(p_c\), starting in direction \(d_r\) with angular velocity \(\omega_r\), cf. [8]. The distance to the nearest obstacle on that curve is measured in [9]. The trajectory starts at \(p_r\) tangential to the circle centered at \(p_c\)—which has positive radius—with translational velocity \(v_r\) and rotational velocity \(\omega_r\), as defined by \(\omega_r |p_r - p_c| = v_r\) in [10].

A circular trajectory around the rotation center \(p_c\) ensures passive safety if it allows the robot to stop before it collides with the nearest obstacle. Consider the worst case where the

\[ d_r = \frac{(p_r - p_c)}{|p_r - p_c|} \]


\[ d_r = \frac{p^x_r - p^x_c}{|p_r - p_c|} \wedge d^y_r = \frac{p^y_r - p^y_c}{|p_r - p_c|}. \]

---

center \( p_c \) is at infinity and the robot travels on a straight line. In this case, the distance between the robot’s current position \( p_r \) and the nearest obstacle \( p_o \) must account for the following components: First, the robot needs to be able to brake from its current velocity \( v_r \) to a complete standstill, cf. \( \frac{v_r^2}{2b} = \int_0^{\psi_{ps}} (v_r - b t) \, dt \) in (11). Second, the robot may not react before \( \varepsilon \) time units. Thus, we must additionally take into account the distance that the robot may travel w.r.t. the worst-case acceleration \( A \) and the distance needed for compensating its potential acceleration of \( A \) during that time with braking power \( b \), cf. \( \left( \frac{A}{2} + 1 \right) \left( \frac{A}{2} \varepsilon^2 + \varepsilon v_r \right) = \int_0^{\psi_{ps}} (v_r + A t) \, dt + f_{0\epsilon}^{\psi_{ps}} (v_r + A - b t) \, dt \) in (11). Third, during the worst-case time (\( \varepsilon \geq d \parallel A \varepsilon \)), the obstacle may approach the robot in the worst case on a straight line with maximum velocity \( V \), cf. (12).

To simplify the proof, we measure the distance between the robot’s position \( p_r \) and the obstacle \( p_o \) in the infinity-norm \( ||p_r - p_o||_\infty \) (i.e., both \( |p_r^x - p_o^x| \) and \( |p_r^y - p_o^y| \) must be safe) and thus over-approximate the Euclidean norm distance \( ||p_r - p_o||_2 = \sqrt{(|p_r^x - p_o^x|^2 + |p_r^y - p_o^y|^2)} \) by a factor of at most \( \sqrt{2} \). Finally, the continuous dynamics of the robot and the obstacle are defined in (13)–(17).

b) Verification: We verify the safety of the control algorithm modeled as a hybrid program in Model 1 using a formal proof calculus for dC \([18,20]\). The robot is at a safe distance from the obstacle, if it is able to brake to a complete stop at all times before the approaching obstacle reaches the robot. The following condition captures this requirement as a property that we prove to hold for all executions:

\[
\phi_{ps} \equiv (v_r = 0) \vee \left( ||p_r - p_o|| > \frac{v_r^2}{2b} + V \frac{v_r}{b} \right). \tag{18}
\]

The formula (18) states that the robot is stopped or the robot and the obstacle have different positions safely apart. This accounts for the robot’s braking distance \( \frac{v_r^2}{2b} \) while the obstacle is allowed to approach the robot with its worst-case travel distance \( V \frac{\sqrt{2}}{\varepsilon} \). We prove that the property (18) holds for all executions of Model 1 under the assumption that we start in a state satisfying the following conditions:\footnote{The formal proof uses the additional constraints stated earlier, \( v_r \geq 0, A \geq 0, V \geq 0, \Omega \geq 0, b > 0, \) and \( \varepsilon > 0 \), which we leave out for simplicity.}

\[
\psi_{ps} \equiv \phi_{ps} \wedge (||p_r - p_o|| > 0) \wedge (||d_r|| = 1). \tag{19}
\]

The first condition of the conjunction formalizes the fact that we are starting in a position that is passively safe, that is \( \phi_{ps} \) holds. The second conjunct states that the robot is not initially spinning. The last conjunct \( ||d_r|| = 1 \) says that the direction \( d_r \) is a unit vector.

**Theorem 1 (Passive safety):** If the robot starts in a state where \( \psi_{ps} \) (19) holds, then the control model \( dw_{ps} \) (Model 1) always guarantees the passive safety condition \( (\phi_{ps}) \), as expressed by the provable dC formula: \( \psi_{ps} \rightarrow [dw_{ps}] \phi_{ps} \).

We proved Theorem 1 for circular trajectories, spinning, and spiral trajectories using KeYmaera, a theorem prover for hybrid systems. In the next section, we explore the stronger requirements of passive friendly safety, where the robot will not only safely stop itself, but also allow for the obstacle to stop before a collision occurs.

### B. Passive Friendly Safety of Obstacle Avoidance

Passive friendly safety, as introduced above, requires the robot to take careful decisions that respect the features of moving obstacles. The definition of Maček et al. \([14]\) requires that the robot respects the worst-case braking time of the obstacle. In our model, the worst-case braking time is represented as follows. We assume an upper bound \( \tau \) on the reaction time of the obstacle and a lower bound \( b_0 \) on its braking capabilities. Then, \( \tau V \) is the worst-case distance that the obstacle can travel before actually reacting and \( \frac{v_r^2}{2b_0} \) is the distance for the obstacle to stop from the worst-case velocity \( V \) with an assumed minimum braking capability \( b_0 \).

c) Modeling: Model 2 uses the same obstacle avoidance algorithm as Model 1. The essential difference reflects what the robot considers to be a safe distance to an obstacle. As shown in (21), the distance not only accounts for the robot’s own braking distance, but also for the braking distance \( \frac{v_r^2}{2b} \) and reaction time \( \tau \) of the obstacle. The verification of passive friendly safety is more complicated than passive safety as it accounts for the behavior of the obstacle discussed below.

**Model 2 Dynamic window with passive friendly safety**

\[
dw_{pfs} \equiv (\text{ctrl: dyn})^* \text{ (see Model 1 for details)} \tag{20}
\]

\[
safe \equiv ||p_r - p_o||_\infty > \frac{v_r^2}{2b_0} + \frac{V^2}{2b_0} + \tau V \tag{21}
\]

\[
+ \left( \frac{A}{b} + 1 \right) \left( \frac{A}{2} \varepsilon^2 + \varepsilon v_r \right) + V \left( \varepsilon + \frac{v_r + A \varepsilon}{b} \right)
\]

\[
d) Verification: We verify the safety of the robot’s control choices as modeled in Model 2. Unlike the passive safety case, the passive friendly safety property \( \phi_{pfs} \) should guarantee that if the robot stops, moving obstacles (cf. Model 3) still have enough time and space to avoid a collision. This requirement can be captured by the following dC formula:

\[
\eta_{pfs} \equiv \left( (v_r = 0) \wedge (||p_r - p_o|| > \frac{V^2}{2b_0} + \tau V) \wedge (0 \leq v_o \leq V) \right) \rightarrow (\text{obstacle}) \left( (||p_r - p_o|| > 0) \wedge (v_o = 0) \right). \tag{22}
\]

Equation (22) says that, once the robot stops \( (v_r = 0) \), there exists an execution of the hybrid program \( \text{obstacle} \), (existence of a run is formalized by the diamond operator \( (\text{obstacle}) \)), that allows the obstacle to stop \( (v_o = 0) \) before a collision occurs \( (||p_r - p_o|| > 0) \). Passive friendly safety is now stated as \( \phi_{pfs} \equiv (\phi_{ps} \lor v_r = 0) \lor \eta_{pfs} \), where the property \( \phi_{pfs} \) is analogue to the passive safety property \( \phi_{ps} \) w.r.t. Model 2.

\[
\phi_{pfs} \equiv ||p_r - p_o|| > \frac{v_r^2}{2b} + \frac{V^2}{2b_0} + V \left( \frac{v_r}{b} + \tau \right).
\]

We study the passive friendly safety with respect to the initial feasible states satisfying the following property:

\[
\psi_{pfs} \equiv \phi_{pfs} \wedge (||p_r - p_o|| > 0) \wedge (||d_r|| = 1). \tag{23}
\]

Observe that, in addition to the condition \( \eta_{pfs} \), the difference with the passive safety condition is reflected in the special treatment of the case \( v_r = 0 \). Indeed, if the robot would start...
with a null translational velocity (which is passive safety) while not satisfying $\varphi_{\text{pfs}}$, then we cannot prove the passive friendly safety as the obstacle may be unsafely close already initially. Besides, we can see in $\varphi_{\text{pfs}}$ that we are required to prove $\eta_{\text{pfs}}$ even when the robot comes to a full stop.

In Model 2, the hybrid program $\text{ctrl}_o$ is a coarse model given by equation (3), which only constrains its non-negative velocity to be less than or equal to $V$. Such an obstacle could trivially prevent a collision by stopping instantaneously. In $\eta_{\text{pfs}}$, we consider a more interesting refined obstacle behavior modeled by the hybrid program given in Model 3.

**Model 3** Refined obstacle controls acceleration

$$\text{obstacle} \equiv (\text{ctrl}_o; \text{dyn}_o)^* \quad (24)$$

$$\text{ctrl}_o \equiv d_o := (\ast, \ast); ?\|d_o\| = 1; \quad (25)$$

$$a_o := \ast; \ ?v_o + a_o \varepsilon_o \leq V \quad (26)$$

$$\text{dyn}_o \equiv (t := 0; \ p_o'' = v_o d_o', \ p_o' = v_o d_o, \ v_o' = a_o, \ t' = 1 \ & \ t \leq \varepsilon_o \land v_o \geq 0) \quad (27)$$

The refined obstacle may choose any direction as described by the unit vector $d_o$ in (25) and any acceleration $a_o$, as long as it does not exceed the velocity bound $V$, cf. (26). The dynamics of the obstacle are straight-world transatlantic motion in the two-dimensional plane, see (27).

**Theorem 2 (Passive friendly safety):** If the property $\psi_{\text{pfs}}$ holds initially, then the control model $dw_{\text{pfs}}$ (Model 3) guarantees the passive friendly safety condition $\varphi_{\text{pfs}}$ in presence of obstacles per Model 3 as expressed by the provable $\Delta L$ formula: $\psi_{\text{pfs}} \rightarrow [dw_{\text{pfs}}] \varphi_{\text{pfs}}$.

We verified Theorem 2 in KeYmaera. The symbolic bounds on velocity, acceleration, braking, and time in the above models represent uncertainty implicitly (e.g., one may instantiate the braking power $b$ with the minimum specification of the robot’s brakes, or with the actual braking power achievable w.r.t. the current terrain). Dually, whenever knowledge about the current state is available, the bounds can be instantiated more aggressively to allow efficient robot behavior (e.g., in a rare worst case we may face a particularly fast obstacle, but right now there are only slow-moving obstacles around). Theorems 1 and 2 are verified for all those values. Other aspects of uncertainty need explicit changes in the models and proofs, as discussed in the next section.

**C. Safety of Obstacle Avoidance Despite Uncertainty**

Robots have to deal with uncertainty in almost every aspect of their interaction with the environment, ranging from sensor inputs (e.g., inaccurate localization, distance measurement) to actuator effects (e.g., wheel slip depending on the terrain). In this section, we show two examples of uncertainty explicitly handled by the models. First, we allow localization uncertainty (the robot knows its position only approximately). We then consider imperfect actuator commands (braking and acceleration) considered to be within an interval. Such intervals are instantiated, e.g., according to sensor or actuator specification (e.g., GPS error), or w.r.t. experimental measurements.\(^\text{10}\)

**Model 4** Safety despite uncertain position measurements

$$dw_{\text{ups}} \equiv (\text{locate}; \text{ctrl}; \text{dyn})^* \quad (28)$$

Model 4 introduces location $\hat{p}_r$ before the control decisions are made. This location measurement may deviate from the real position $p_r$ by no more than symbolic parameter $U_p$, cf. (28). The measured location $\hat{p}_r$ is used in all control decisions of the robot (e.g., in (30) to compute the curve’s center point and rotational velocity), while the robot’s physical motion is still computed on the symbolic real position $p_r$. We prove in KeYmaera that the robot is still safe (passive safety), if it accounts for the location uncertainty as stated in the safety constraint (31).

**Model 5** Safety despite actuator uncertainty

$$dw_{\text{aus}} \equiv (\text{ctrl}; \text{act}; \text{dyn})^* \quad (29)$$

Model 5 introduces uncertainty in actuation with an actuator perturbation between control and dynamics, cf. (29). Actuator perturbation affects the acceleration by a damping factor $\varepsilon_d$, which is assumed to be constant for $\varepsilon_d$ time units. In the worst case, the robot has full acceleration ($\varepsilon_d = 1$) but fully reduced braking ($\varepsilon_d = 0$). For instance, the robot accelerates on perfect terrain, but is on slippery terrain whenever it brakes. The robot considers this worst-case scenario during control in its safety constraint (34). Using KeYmaera, we prove that the robot is still safe (passive safety) when it follows Model 5.

**V. Conclusion and Future Work**

Robots are hybrid systems, because they share continuous physical motion with complicated computer algorithms controlling their behavior. We demonstrate that this understanding also helps proving robots safe. We develop hybrid system models of the dynamic window algorithm for autonomous ground vehicles and prove that the algorithm guarantees both passive safety and passive friendly safety in the presence of...
moving obstacles. All the proofs were achieved automatically with some manual guidance using the verification tool KeYmaera. We manually provided inductive and differential invariants (crucial), as well as reduced the number of terms and variables with simple interactions to reduce arithmetic complexity. 85% of the proof steps were automatic. Most interactive steps were simple arithmetic simplifications that KeYmaera’s current proof strategies do not yet automate. We augment these models and safety proofs with robustness for localization uncertainty and imperfect actuation. We observe that incremental revision of models and proofs helps reducing the verification complexity and understanding the impact of uncertainty on the behavior of the robot. Further, note that all symbolic bounds in our models—such as those for maximum obstacle velocity and sensor/actuator uncertainty—can be instantiated with probabilistic models (e.g., assume the 2σ confidence interval of the distribution of obstacle velocities as maximum obstacle velocity). In this case, our verified safety guarantees translate into a safety probability.

Future work includes refined kinematic capabilities (e.g., going sideways with omni-drive) and additional sources of uncertainties, such as distance measurements. Also, to derive less conservative safety bounds for specific environments, one could introduce specific obstacle models (e.g., pedestrians).

ACKNOWLEDGMENTS

This material is based upon work supported by NSF CAREER Award CNS-1054246 and NSF EXPEDITION CNS-0926181, by DARPA FA8750-12-2-0291, and by US DOT UTC TSET award # DTRT12GUTC11. This work was also supported by the Austrian BMVIT under grant FIT-IT 829598, FFG BRIDGE 838526, and FFG Basisprogramm 838181.

REFERENCES


