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 moving 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 [7] or driverless cars on regular Californian roads[1]—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.[2]

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 III recalls differential dynamic logic that we use as a modeling formalism for the hybrid dynamics of a robot and the safety constraints. In Sect. IV, 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. V 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].

PassAvoid [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 PassAvoid 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. [1] 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.

Seward et al. [24] try to avoid potentially hazardous situations by using Partially Observable Markov Decision Processes. Their focus, however, is on a user-definable trade-off between safety and progress towards a goal. Hence, safety is not guaranteed under all circumstances.

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), 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.

TABLE I: Hybrid program representations of hybrid systems.

Statement	Effect
$\alpha;\ \beta$	sequential composition, first run α, then β
$\alpha \cup \beta$	nondeterministic choice, following either α or β
α^*	nondeterministic repetition, repeats α $n \geq 0$ times
$x := \theta$	assign value of term θ to variable x (discrete jump)
$x := *$	assign arbitrary real number to variable x
$(x_1' = \theta_1, \ldots,$ $x_n' = \theta_n \ \&\ F)$	evolve x_i along differential equation system $x_i' = \theta_i$ restricted to maximum evolution domain F

In order to verify safe obstacle avoidance, we use *differential dynamic logic* dL [18, 20, 21], which has a notation for hybrid systems as *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. dL 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.

Sequential composition $\alpha;\ \beta$ says that β starts after α finishes (e. g., first let the robot choose acceleration, then steering angle). The nondeterministic choice $\alpha \cup \beta$ follows either α or β (e. g., let the robot decide nondeterministically between remaining stopped or accelerating). The nondeterministic repetition operator α^* repeats α zero or more times (e. g., let the robot repeatedly control and drive). Assignment $x := \theta$ instantaneously assigns the value of the term θ 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). $x' = \theta \ \&\ F$ 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 dL formulas is generated by the following EBNF grammar (where $\sim \in \{<, \leq, =, \geq, >\}$ and θ_1, θ_2 are arithmetic expressions in $+, -, \cdot, /$ over the reals):

$$\phi ::= \theta_1 \sim \theta_2 \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \phi \rightarrow \psi \mid \forall x \phi \mid [\alpha]\phi \mid \langle\alpha\rangle\phi$$

Further operations, such as Euclidian norm $\|\theta\|$ and infinity norm $\|\theta\|_\infty$ of a vector θ, are definable.

To specify the desired correctness properties of hybrid programs, dL formulas of the form $F \rightarrow [\alpha]G$ mean that all executions of the hybrid program α, 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 α 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 [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 [10], which focuses on piecewise linear systems. KeYmaera implements automatic proof strategies that decompose hybrid systems symbolically [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 dL [18, 21] indicating how this composition principle can be successful.

IV. Robotic Ground Vehicle Navigation

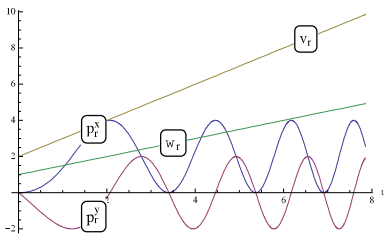
The robotics community has come up with a large variety of robot designs, which differ not only in their tool equipment, but also (and more importantly for the discussion in this paper) in their kinematic capabilities. We focus on wheel-based vehicles. In order to make our models applicable to a large 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.[3] Such trajectories can be realized by robots with single wheel drive, differential drive, Ackermann drive, synchro drive, or omni drive [4].

Many different navigation and obstacle avoidance algorithms have been proposed for such robots, e. g. *dynamic window* [9], *potential fields* [11], or *velocity obstacles* [8]. For an introduction to various navigation approaches for mobile robots, see [2, 6]. In this paper, we focus on the dynamic window algorithm [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 [8], are interesting for further work on provably safe moving obstacle avoidance.

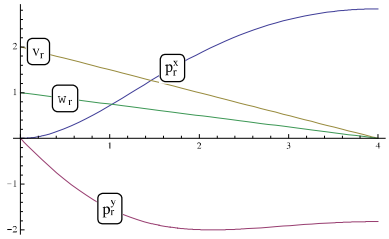
We want to prove motion safety of a robot that avoids obstacles by dynamic window navigation. Under *passive safety* [14], the vehicle is in a safe state if it is able to come to a full stop before making contact with an obstacle (i. e., the vehicle does not itself collide with obstacles, so if a collision 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* [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.[4]

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

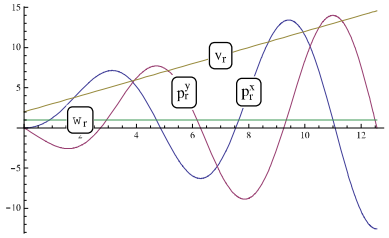
[3] If the radius of such a circle is infinite, the robot drives (forward) on a straight line. [4] The robot ensures that there is enough room for the obstacle to stop before a collision occurs. If the obstacle decides not to, the obstacle is to blame and our robot is still considered safe.



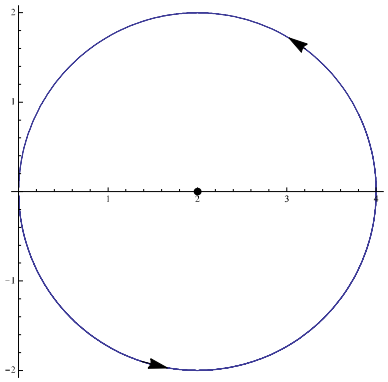
(a) Position (p_r^x, p_r^y), translational velocity v_r and rotational velocity ω_r for positive acceleration on a circle.



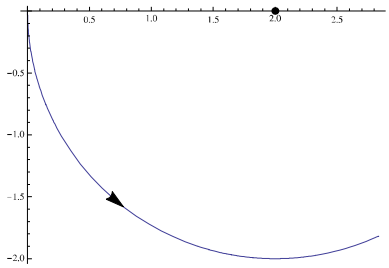
(b) Position (p_r^x, p_r^y), translational velocity v_r and rotational velocity ω_r for braking to a complete stop on a circle.



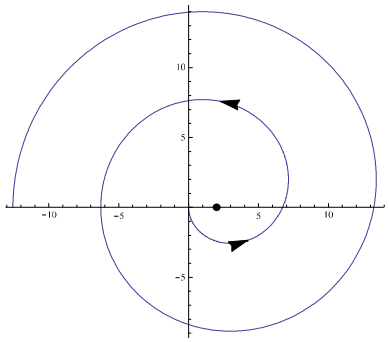
(c) Position (p_r^x, p_r^y), translational velocity v_r and rotational velocity ω_r for translational acceleration on a spiral.



(d) (p_r^x, p_r^y) motion plot for acceleration (a).



(e) (p_r^x, p_r^y) motion plot for braking (b).



(f) (p_r^x, p_r^y) motion plot for (c).

Fig. 1: Trajectories of the robot.

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, ω). 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) Modeling: We develop a model of the principles in the dynamic window algorithm as a hybrid program in dL. The dynamic window algorithm safely abstracts the robot's shape to a single point, since other shapes reduce to adjusting the (virtual) shapes of the obstacles (cf. [15] for an approach to attribute robot shape to obstacles). We also use this abstraction to reduce the verification complexity. The robot has state variables describing its current position $p_r = (p_r^x, p_r^y)$, translational velocity $v_r \geq 0$, translational acceleration a_r, a direction vector $d_r = (\cos\theta, \sin\theta)$,[5] and angular velocity $\theta' = \omega_r$.[6] The translational and rotational velocities are linked w.r.t. the rigid body planar motion by the formula $\|p_r - p_c\|\omega_r = v_r$, where the radius $\|p_r - p_c\|$ is the distance between the robot and the center of its current curve $p_c = (p_c^x, p_c^y)$. Following [19], we encode sine and cosine functions in the dynamics using the extra variables $d_r^x = \cos\theta$ and $d_r^y = \sin\theta$ to avoid undecidable arithmetic. The continuous dynamics for the dynamic window algorithm [9] can be described by the differential equation system of ideal-world dynamics of the planar rigid body motion $(p_r^{x\prime} = v_r d_r^x,\ p_r^{y\prime} = v_r d_r^y,\ v_r' = a_r,\ (\|p_r - p_c\|\omega_r)' = a_r,\ d_r^{x\prime} = -\omega_r d_r^y,\ d_r^{y\prime} = \omega_r d_r^x)$ where the condition $(\|p_r - p_c\|\omega_r)' = a_r$ encodes the rigid body planar motion $\|p_r - p_c\|\omega_r = v_r$ that we consider. The dynamic window algorithm assumes direct control of the translational velocity v_r. We, instead, control acceleration a_r and do not perform instant changes of the velocity. Our model is closer to the actual dynamics of a robot, which cannot really change its velocity instantly. The realizable velocities, then, follow from the differential equation system.

Figure 1(a) depicts the position and velocity changes of a robot accelerating on a circle around a center point $p_c = (2, 0)$. The robot starts at $p_r = (0, 0)$ as initial position, with $v_r = 2$ as initial translational velocity and $\omega_r = 1$ as initial rotational velocity; Figure 1(d) shows the resulting circular trajectory. Figure 1(b) and Figure 1(e) show the resulting

[5] As stated earlier, we study unidirectional motion: the robot moves along its direction, that is the vector d_r gives the direction of the velocity vector.
[6] The derivative with respect to time is denoted by superscript 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/ω_r^2). The corresponding plots are shown in Figures 1(c) and 1(f).

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 Ω on the rotational velocity ω_r. We use ε 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 ε. 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 $p_o = (p_o^x, p_o^y)$ and velocity $v_o = (v_o^x, v_o^y)$. The obstacle model is very liberal. The only restriction about the dynamics is, that within ε 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 ε, moving obstacles can make quite a difference, e. g., when fast obstacles meet communication-based virtual sensors as in RoboCup.[7]

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 $(\|p_r - p_c\|\omega_r)' = a_r$ 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

[7] http://www.robocup.org/

Model 1 Dynamic window with passive safety

$$dw_{\text{ps}} \equiv (ctrl; dyn)^* \tag{1}$$
$$ctrl \equiv ctrl_o \;||\; ctrl_r \tag{2}$$
$$ctrl_o \equiv v_o := (*, *);\ ?\|v_o\| \leq V \tag{3}$$
$$ctrl_r \equiv (a_r := -b) \tag{4}$$
$$\cup\ (?v_r = 0;\ a_r := 0;\ \omega_r := 0) \tag{5}$$
$$\cup\ (a_r := *;\ ?-b \leq a_r \leq A; \tag{6}$$
$$\omega_r := *;\ ? - \Omega \leq \omega_r \leq \Omega; \tag{7}$$
$$p_c := (*, *);\ d_r := (*, *); \tag{8}$$
$$p_o := (*, *);\ ?curve \wedge safe) \tag{9}$$
$$curve \equiv \|p_r - p_c\| > 0 \wedge \omega_r\|p_r - p_c\| = v_r \tag{10}$$
$$\wedge\ d_r = \frac{(p_r - p_c)^\perp}{\|p_r - p_c\|}$$
$$safe \equiv \|p_r - p_o\|_\infty > \frac{v_r^2}{2b} + \left(\frac{A}{b} + 1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon v_r\right) \tag{11}$$
$$+ V\left(\varepsilon + \frac{v_r + A\varepsilon}{b}\right) \tag{12}$$
$$dyn \equiv (t := 0;\ p_r^{x\prime} = v_r d_r^x,\ p_r^{y\prime} = v_r d_r^y, \tag{13}$$
$$d_r^{x\prime} = -\omega_r d_r^y,\ d_r^{y\prime} = \omega_r d_r^x, \tag{14}$$
$$p_o^{x\prime} = v_o^x,\ p_o^{y\prime} = v_o^y, \tag{15}$$
$$v_r' = a_r,\ \omega_r' = \frac{a_r}{\|p_r - p_c\|},\ t' = 1 \tag{16}$$
$$\&\ v_r \geq 0 \wedge t \leq \varepsilon) \tag{17}$$

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 III: 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 (4) 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_r, ω_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 ω_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 ω_r as defined by $\omega_r\|p_r - p_c\| = v_r$ in (10).[8]

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

[8] $d_r = \frac{(p_r - p_c)^\perp}{\|p_r - p_c\|}$ means component-wise equality of the orthonormal vector, normalized by the radius, as in $d_r^x = -\frac{p_r^y - p_c^y}{\|p_r - p_c\|} \wedge d_r^y = \frac{p_r^x - p_c^x}{\|p_r - p_c\|}$.

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 stand still, cf. $\frac{v_r^2}{2b} = \int_0^{v_r/b}(v_r - bt)dt$ in (11). Second, the robot may not react before ε 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. $\left(\frac{A}{b}+1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon v_r\right) = \int_0^{\varepsilon}(v_r + At)dt + \int_0^{A\varepsilon/b}(v_r + A\varepsilon - bt)dt$ in (11). Third, during the worst-case time $(\varepsilon + \frac{v_r + A\varepsilon}{b})$ entailed by (11), 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's position 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 dL [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:

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

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 $V\frac{v_r}{b}$. 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:[9]

$$\psi_{\text{ps}} \equiv \phi_{\text{ps}} \wedge (\|p_r - p_c\| > 0) \wedge (\|d_r\| = 1) \quad . \tag{19}$$

The first condition of the conjunction formalizes the fact that we are starting in a position that is passively safe, that is ϕ_{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 ψ_{ps} (19) holds, then the control model dw_{ps} (Model 1) always guarantees the passive safety condition (ϕ_{ps}), as expressed by the provable dL formula: $\psi_{\text{ps}} \rightarrow [dw_{\text{ps}}]\phi_{\text{ps}}$.

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.

[9] 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.

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 τ on the reaction time of the obstacle and a lower bound b_o on its braking capabilities. Then, τV is the worst-case distance that the obstacle can travel before actually reacting and $\frac{V^2}{2b_o}$ is the distance for the obstacle to stop from the worst-case velocity V with an assumed minimum braking capability b_o.

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 $\frac{V^2}{2b_o}$ and reaction time τ 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

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

$$\begin{aligned} safe \equiv \|p_r - p_o\|_\infty > \frac{v_r^2}{2b} + \frac{V^2}{2b_o} + \tau V \\ + \left(\frac{A}{b}+1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon v_r\right) + V\left(\varepsilon + \frac{v_r + A\varepsilon}{b}\right) \end{aligned} \tag{21}$$

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 ϕ_{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 dL formula:

$$\begin{aligned} &\eta_{\text{pfs}} \equiv \\ &\left((v_r = 0) \wedge \left(\|p_r - p_o\| > \frac{V^2}{2b_o} + \tau V\right) \wedge (0 \leq v_o \leq V)\right) \\ &\qquad \rightarrow \langle obstacle \rangle \left((\|p_r - p_o\| > 0) \wedge (v_o = 0)\right) . \end{aligned} \tag{22}$$

Equation (22) says that, once the robot stops ($v_r = 0$), there exists an execution of the hybrid program *obstacle*, (existence of a run is formalized by the diamond operator $\langle obstacle \rangle$), 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 $\phi_{\text{pfs}} \equiv (\varphi_{\text{ps}} \vee v_r = 0) \wedge \eta_{\text{pfs}}$, where the property φ_{ps} is analogue to the passive safety property ϕ_{ps} w.r.t. Model 2:

$$\varphi_{\text{ps}} \equiv \|p_r - p_o\| > \frac{v_r^2}{2b} + \frac{V^2}{2b_o} + V\left(\frac{v_r}{b} + \tau\right) \quad .$$

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

$$\psi_{\text{pfs}} \equiv \varphi_{ps} \wedge (\|p_r - p_c\| > 0) \wedge (\|d_r\| = 1) \quad . \tag{23}$$

Observe that, in addition to the condition η_{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 φ_{ps}, then we cannot prove the passive friendly safety as the obstacle may be unsafely close already initially. Besides, we can see in ϕ_{pfs} that we are required to prove η_{pfs} even when the robot comes to a full stop.

In Model 2 the hybrid program $ctrl_o$ 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 η_{pfs}, we consider a more interesting refined obstacle behavior modeled by the hybrid program given in Model 3.

Model 3 Refined obstacle controls acceleration

$$obstacle \equiv (ctrl_{or}; dyn_{or})^* \tag{24}$$
$$ctrl_{or} \equiv d_o := (*,*);\ ?\|d_o\| = 1; \tag{25}$$
$$a_o := *;\ ?v_o + a_o\varepsilon_o \leq V \tag{26}$$
$$dyn_{or} \equiv (t := 0;\ p_o^{x\prime} = v_o d_o^x,\ p_o^{y\prime} = v_o d_o^y, \tag{27}$$
$$v_o' = a_o,\ t' = 1\ \&\ t \leq \varepsilon_o \wedge v_o \geq 0)$$

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 ideal-world translational motion in the two-dimensional plane, see (27).

Theorem 2 (Passive friendly safety): If the property ψ_{pfs} (23) holds initially, then the control model dw_{pfs} (Model 2), guarantees the passive friendly safety condition ϕ_{pfs} in presence of obstacles per Model 3, as expressed by the provable dL formula: $\psi_{\text{pfs}} \rightarrow [dw_{\text{pfs}}]\phi_{\text{pfs}}$.

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.[10]

[10] Instantiation with probabilistic bounds means that the symbolically guaranteed safety is traded for a safety probability.

Model 4 Safety despite uncertain position measurements

$$dw_{\text{ups}} \equiv (locate; ctrl; dyn)^* \text{ (see Model 1 for details)} \tag{28}$$
$$locate \equiv \tilde{p}_r := (*,*);\ ?\|\tilde{p}_r - p_r\| \leq U_p \tag{29}$$
$$curve \equiv \|\tilde{p}_r - p_c\| > 0 \wedge \omega_r\|\tilde{p}_r - p_c\| = v_r \tag{30}$$
$$\wedge\ d_r = \frac{(\tilde{p}_r - p_c)^\perp}{\|\tilde{p}_r - p_c\|}$$
$$safe \equiv \|\tilde{p}_r - p_o\|_\infty > \frac{v_r^2}{2b} + \left(\frac{A}{b} + 1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon v_r\right) \tag{31}$$
$$+ V\left(\varepsilon + \frac{v_r + A\varepsilon}{b}\right) + U_p$$

Model 4 introduces location uncertainty. It adds a location measurement $\tilde{p}_r$ 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. (29). The measured location $\tilde{p}_r$ 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

$$dw_{\text{uas}} \equiv (ctrl; act; dyn)^* \text{ (see Model 1 for details)} \tag{32}$$
$$act \equiv u_m := *;\ \tilde{a}_r := u_m a_r;\ ?0 < U_m \leq u_m \leq 1 \tag{33}$$
$$safe \equiv \|p_r - p_o\|_\infty > \frac{v_r^2}{2bU_m} + \left(\frac{A}{bU_m} + 1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon v_r\right)$$
$$+ V\left(\varepsilon + \frac{v_r + A\varepsilon}{bU_m}\right) \tag{34}$$
dyn with a_r replaced by $\tilde{a}_r$

Model 5 introduces uncertainty in actuation with an actuator perturbation between control and dynamics, cf. (32). Actuator perturbation affects the acceleration by a damping factor u_m, known to be within $[U_m, 1]$, cf. (33). Note, that this damping factor can change arbitrarily often, but is assumed to be constant for ε time units. In the worst case, the robot has full acceleration ($u_m = 1$) but fully reduced braking ($u_m = U_m$). 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

[1] Daniel Althoff, James J. Kuffner, Dirk Wollherr, and Martin Buss. Safety assessment of robot trajectories for navigation in uncertain and dynamic environments. *Autonomous Robots*, 32:285–302, 2012.

[2] Francisco Bonin-Font, Alberto Ortiz, and Gabriel Oliver. Visual Navigation for Mobile Robots: A Survey. *J. Intelligent Robotics Systems*, 53(3):263–296, 2008.

[3] Sara Bouraine, Thierry Fraichard, and Hassen Salhi. Provably safe navigation for mobile robots with limited field-of-views in dynamic environments. *Autonomous Robots*, 32(3):267–283, 2012.

[4] Thomas Bräunl. Driving robots. In *Embedded Robotics: Mobile Robot Design and Applications with Embedded Systems*, pages 97–111. Springer, 2006.

[5] Oliver Brock and Oussama Khatib. High-speed navigation using the global dynamic window approach. In *Robotics and Automation*, pages 341–346. IEEE, 1999.

[6] Howie Choset, Kevin Lynch, Seth Hutchinson, George Kantor, Wolfram Burgard, Lydia Kavraki, and Sebastian Thrun. *Principles Of Robot Motion*. MIT Press, 2005.

[7] Paolo Fiorini and Erwin Prassler. Cleaning and Household Robots: A Technology Survey. *Autonomous Robots*, 9:227–235, 2000.

[8] Paolo Fiorini and Zvi Shiller. Motion Planning in Dynamic Environments Using Velocity Obstacles. *J. Robotics Research*, 17(7):760–772, 1998.

[9] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *Robotics Automation Magazine, IEEE*, 4(1):23–33, 1997.

[10] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. SpaceEx: Scalable verification of hybrid systems. In S. Qadeer G. Gopalakrishnan, editor, *CAV*, LNCS. Springer, 2011.

[11] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Robotics and Automation*, pages 500–505, 1985.

[12] Sarah M. Loos, André Platzer, and Ligia Nistor. Adaptive Cruise Control: Hybrid, Distributed, and Now Formally Verified. In *FM*, pages 42–56. Springer, 2011.

[13] Sarah M. Loos, David Renshaw, and André Platzer. Formal verification of distributed aircraft controllers. In *HSCC*. ACM, 2013.

[14] Kristijan Maček, Dizan Alejandro Vasquez Govea, Thierry Fraichard, and Roland Siegwart. Towards Safe Vehicle Navigation in Dynamic Urban Scenarios. *Automatika*, 50(3–4):184–194, 2009.

[15] Javier Minguez, Luis Montano, and José Santos-Victor. Abstracting Vehicle Shape and Kinematic Constraints from Obstacle Avoidance Methods. *Autonomous Robots*, 20(1):43–59, 2006.

[16] Stefan Mitsch, Sarah M. Loos, and André Platzer. Towards formal verification of freeway traffic control. In *ICCPS*, pages 171–180. IEEE, 2012.

[17] Jia Pan, L. Zhang, and D. Manocha. Collision-free and smooth trajectory computation in cluttered environments. *J. Robotics Research*, 31(10):1155–1175, 2012.

[18] André Platzer. Differential Dynamic Logic for Hybrid Systems. *J. Autom. Reas.*, 41(2):143–189, 2008.

[19] André Platzer. Differential-algebraic Dynamic Logic for Differential-algebraic Programs. *J. Log. Comput.*, 20(1): 309–352, 2010.

[20] André Platzer. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Springer, 2010.

[21] André Platzer. The Complete Proof Theory of Hybrid Systems. In *LICS*, pages 541–550. IEEE, 2012.

[22] André Platzer and Edmund M. Clarke. Formal Verification of Curved Flight Collision Avoidance Maneuvers: A Case Study. In *FM*, pages 547–562. Springer, 2009.

[23] Shahar Sarid, Bingxin Xu, and Hadas Kress-Gazit. Guaranteeing High-Level Behaviors while Exploring Partially Known Maps. In *Robotics: Science and Systems*, 2012.

[24] Derek Seward, Conrad Pace, and Rahee Agate. Safe and effective navigation of autonomous robots in hazardous environments. *Autonomous Robots*, 22:223–242, 2007.

[25] Holger Täubig, Udo Frese, Christoph Hertzberg, Christoph Lüth, Stefan Mohr, Elena Vorobev, and Dennis Walter. Guaranteeing functional safety: design for provability and computer-aided verification. *Autonomous Robots*, 32:303–331, 2012.

[26] Jur van den Berg, Pieter Abbeel, and Ken Goldberg. LQG-MP: Optimized Path Planning for Robots with Motion Uncertainty and Imperfect State Information. In *Robotics: Science and Systems*, 2010.

[27] Albert Wu and Jonathan P. How. Guaranteed infinite horizon avoidance of unpredictable, dynamically constrained obstacles. *Autonomous Robots*, 32:227–242, 2012.