Student Conference Proceedings
Vol 1, No 1, 2025, Article ID 2010, 4 Pages

Student Conference Lübeck

Proceedings Article

Integrating Physical and Logical Feedback Control on a Differential Drive Robot

Kilian Schweppe [a,*] · Satya Prakash Nayak [b] · Arabinda Ghosh [b] · Atef Tej [c] · Anne-Kathrin Schmuck [b,*]

[a] Student of Robotics and Autonomous Systems, Universität zu Lübeck, Lübeck, Germany
[b] Max Planck Institute for Software Systems, Kaiserslautern, Germany
[c] Student of Computer Science, RTPU, Kaiserslautern, Germany
[*] Corresponding author, email: kilian.schweppe@student.uni-luebeck.de; akschmuck@mpi-sws.org

Received 05 February 2025; Accepted 03 November 2025; Published online 21 November 2025

Abstract

This paper presents a two-layered control architecture for integrating logical and physical feedback control in dynamic environments with real-world imperfections. The higher layer uses game-based synthesis to determine strategic reactions to logical context switches, while the lower layer employs task-dependent feedback controllers to actuate these decisions. Unlike conventional methods, the proposed framework provides direct logical feedback to environment-induced context switches and incorporates dynamic barrier certificates to ensure real-time safety. We demonstrate the practical effectiveness of our approach through an end-to-end implementation on a Turtlebot4, validated in an office environment with dynamic context switches.

I. Introduction

The increasing autonomy of modern systems poses significant challenges in ensuring reliable and performant operation. Control software for cyber-physical systems must handle *real-world imperfections* (e.g., model uncertainty or measurement noise) and time-varying *safety constraints* (e.g., moving obstacles). Furthermore, logical decisions must be taken to adapt dynamically to changes in the *logical context* of the environment.

In order to concretize the outlined control challenges further, we consider a concrete example throughout this paper: A mobile robot navigates an office environment with unknown, potentially moving obstacles (e.g., humans, other robots, or chairs). Its task is to pick and place products in different parts of the office space in cooperation with other robots and in response to requests from office workers.

The workspace, depicted schematically in Fig. 1, consists of regions $Q_1, \ldots, Q_5$, a controlled robot R_c, and an environment robot R_e. For $1 \leq i \leq 5$, we consider Boolean variables $\mathtt{c}_i$ and $\mathtt{e}_i$ which are set to `true` if, respectively, robot R_c or R_e is in region Q_i. For example, in Fig. 1 only $\mathtt{c}_2$ and $\mathtt{e}_1$ are set. In addition, we set $\mathtt{human}_i$ to true if a (human) obstacle is present in region Q_i and $\mathtt{o}$ to true if robot R_c has collided with any obstacle.

Both robots can be requested to *leave* or *enter* a particular region Q_i. For example, $\mathtt{leave}^{\mathtt{e}}_{\mathtt{i}}$ is true if robot R_e is requested to leave region Q_i. The Boolean variables $\mathtt{leave}^{\mathtt{c}}_{\mathtt{i}}$, $\mathtt{enter}^{\mathtt{e}}_{\mathtt{i}}$ and $\mathtt{enter}^{\mathtt{c}}_{\mathtt{i}}$ are defined analogously. These variables can either be set to `true` by the other robot or by a human office worker. We assume that only one `enter`-request is active for each robot at a time.

We aim to synthesize a hybrid controller for robot R_c w.r.t. a logical specification over these Boolean variables. For illustration, we consider the following simple logical requirements:

(a) R_c must always avoid obstacles ($\mathtt{o}$ is always `false`),
(b) Q_1 can only be occupied by a single robot at a time,

10.18416/SCP.2025.2010

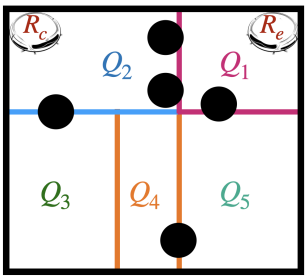

Figure 1: Example workspace of a robot in an office environment for the motivating example. The black circles are static obstacles.

(c) R_c can only set $\texttt{leave}_1^e$ to `true` if it is located in Q_2,

(d) R_c must leave and avoid Q_4 when entered by a (human) obstacle.

We assume that the (externally controlled) robot R_e reacts to requests $\texttt{leave}_i^e$ and $\texttt{enter}_i^e$ accordingly, ensuring these predicates eventually become false after being set. With this specification in mind, we can see that a single request for R_c to come to Q_1 may trigger a reactive chain of reach-while-avoid (RWA) control problems.

Game-based synthesis techniques have gained significant attention in recent years. However, existing approaches based on state-space discretization (e.g., [1]) suffer heavily from the "curse of dimensionality". To address this issue, two-layer control architectures have been proposed [2, 3], combining a high-level game with a low-level feedback controller. Yet, a common limitation of these approaches is their reliance on offline computations and inability to handle dynamic obstacles, restricting them to static environments.

We propose a novel two-layered control architecture, as outlined in Section II, which automatically synthesizes a logical controller realizing the specification. For the lower level, we utilize barrier certificates to enforce all avoid obligations of the individual RWA problems, even for unknown, moving obstacles. This will be demonstrated in Section III, where we show the realization of the reactive controller on a real TurtleBot4 platform.

II. Methods

The High Level Logical Controller

We consider logical specifications as temporal formulas over a finite set of *atomic propositions*, which are Boolean variables signaling important information to the logical controller. The atomic propositions are divided into three categories (see [3, Sec.II B]):

1. *state propositions* $\texttt{AP}_S$ are associated with subsets of the state space (e.g., $\texttt{c}_i$ for regions Q_i or $\texttt{o}$ for obstacle regions). A proposition in $\texttt{AP}_S$ is true at time t if the system's current state lies within the corresponding subset, e.g., $\big[x(t), y(t)\big]^\top \in Q_i \subset \mathbb{R}^2$.
2. *observation propositions* $\texttt{AP}_O$ represent information observed by the logical controller, for example $\texttt{e}_i$ and $\texttt{human}_i$ observes the current state of the environment robot and a human, respectively.
3. *control propositions* $\texttt{AP}_C$ denote controllable variables, such as $\texttt{leave}_\texttt{i}^\texttt{c}$ or $\texttt{enter}_\texttt{i}^\texttt{c}$.

The requirements specification over the atomic propositions is formulated in Linear Temporal Logic (LTL). For instance, requirement (a) can be expressed as $\square\neg\texttt{o}$ where the operator "$\square$" denotes *always* and "$\neg$" indicates that the proposition $\texttt{o}$ must be `false`. Similarly, requirement (c) can be written as $\square(\texttt{leave}_1^\texttt{e} \Rightarrow \texttt{c}_2)$, utilizing the *implication* operator $\Rightarrow$ to specify that if $\texttt{leave}_1^\texttt{e}$ is set to `true`, then $\texttt{c}_2$ must also be `true`.

Every LTL formula over some finite proposition set can be translated into an equivalent two-player game between the controller and the environment. Building on previous work [3], we can compute a *winning strategy* for the game, satisfying the specification by activating context-dependent reach-while-avoid (cRWA) objectives in response to observed logical context changes.

For instance, if robot R_e is in region Q_1 and R_c is requested to go to Q_1 from Q_2 (see Fig. 1), the context is $\kappa = \{\texttt{c}_2, \texttt{e}_1, \texttt{enter}_1^c\}$. The corresponding cRWA is $(\kappa, \mathscr{R}, \mathscr{A})$, where $\mathscr{R} = \{\texttt{c}_2\}$ denotes the target set (implying that R_c needs to move to Q_2) and $\mathscr{A} = \{\texttt{c}_1, \texttt{o}\}$ denotes the avoid set (indicating the obstacles and the region Q_1 that need to be avoided). Whenever the logical context changes – either due to a change in the truth value of observation predicates or because the reach-part of the currently active cRWA has been fulfilled – the internal state of the logical controller will update, triggering the activation of a different cRWA based on the reactive strategy. For a more formal definition and further details, we refer to [3]. Each cRWA is realized by a low-level feedback controller with safety guarantees as discussed in the next section.

The Lower Level Controller

In the following, we consider a differential-drive robot whose dynamics can be approximated by the standard unicycle model

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\cos\theta & 0\\ \sin\theta & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix},$$

where $p = [x, y]^T \in \mathbb{R}^2$ is the position of the robot, $\theta \in (-\pi, \pi]$ denotes the heading angle, $v \in \mathbb{R}$ and $\omega \in \mathbb{R}$ are the inputs and denote the linear and angular velocities respectively. The robot is equipped with a LIDAR which is positioned at the center z of the robot.

10.18416/SCP.2025.2010

Obstacle Detection and Overapproximation

We utilize a LIDAR scanner to compute the state subset which overapproximates the points in the state space which are classified as a (moving) obstacle. To process the data, we apply an Adaptive Breakpoint Detector [4] to cluster the points, and each cluster is overapproximated by a minimum volume ellipse. Let $\widehat{\Lambda}(t)$ be the set of ellipses at time t returned by the online obstacle detection. Additionally, let $\widetilde{\Lambda}(t)$ denote the overapproximation of the currently active avoid set $\mathscr{A}(t)$ by a finite set of ellipses. We combine these two sets into the finite set $\Lambda(t) = \widehat{\Lambda}(t) \cup \widetilde{\Lambda}(t)$ with changing (but finite) cardinality $N(t) = |\Lambda(t)|$. The next section shows how this set can be used to define a (dynamic) barrier certificate that results in a provably safe feedback control policy.

Dynamic Barrier Function Computation

First, we introduce the notion of control barrier functions as defined in [5]. Consider a set $\mathscr{C}$ defined as the superlevel set of a continously differentiable function $h : D \subset \mathbb{R}^n \to \mathbb{R}$, i.e., $\mathscr{C} = \{x \in D : h(x) \geq 0\}$. We refer to $\mathscr{C}$ as the *safe set* for which we want to guarantee safety (in the sense of forward invariance). For an affine control system $\dot{x} = f(x) + g(x)u(x)$ with set of admissible inputs U, the function h is called a *control barrier function (CBF)* if there exists an extended class $\mathscr{K}_\infty$ function α such that

$$\sup_{u \in U} \left[\nabla h^\top(x) f(x) + \nabla h^\top(x) g(x) u \right] \geq -\alpha(h(x)) \quad (1)$$

holds for all $x \in D$. Notably, for any control input $u \in U$ satisfying the inequality in (1) it is ensured that the system will remain in the safe set.

Now recall that at every time point $t \in [0, T)$ we have a finite set of ellipses $\Lambda(t)$ which overapproximates the regions of the state space that need to be avoided by the robot. Each ellipse $(a_i, b_i, x_i, y_i, \theta_i) \in \Lambda(t)$ is parameterized by its center coordinates $x_{o,i} = [x_i, y_i]^\top \in \mathbb{R}^2$, principal axis length $a_i, b_i \in \mathbb{R}$ and orientation $\theta_i \in (-\pi, \pi]$. Therefore, we can define a (quadratic) CBF for each of these obstacles as

$$h_i^t(x') = (x' - x_{o,i})^\top P_i (x' - x_{o,i}) - 1 \ \forall i \in N(t),$$

where

$$P_i = R(\theta_i) \Sigma_i R^\top(\theta_i), \ \Sigma_i = \begin{bmatrix} (a_i + D)^2 & 0 \\ 0 & (b_i + D)^2 \end{bmatrix}^{-1},$$

and $R(\theta)$ represents the counterclockwise rotation matrix

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

Note that we inflate each ellipse by the desired safety distance $D \in \mathbb{R}$.

Observe that the CBF depends only on the position p of the robot and not on its orientation θ. Since the angular velocity ω has no direct effect on the time derivative of p, we would lose control over ω by applying the safety controller directly to the system. Therefore, we consider the control methodology from [6]. That is, instead of considering the center of the robot, we consider the point $q = p + l R(\theta) \hat{e}_1$, where $\hat{e}_1 = [1, 0]^\top$, which is orthogonal to the wheel axis of the robot with (small) distance $l > 0$. To guarantee safety for the point q, we must ensure that

$$\nabla(h_i^t(q))^\top \dot{q} = 2 P_i (q - x_{o,i})^\top \dot{q} \geq -\alpha(h_i^t(q)). \quad (2)$$

Using the invertible mapping

$$u = \begin{bmatrix} v \\ \omega \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1/l \end{bmatrix}}_{L} \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \dot{q} = L R(-\theta) \dot{q}$$

the constraint (2) can be reformulated as

$$2(P_i(-x_{o,i} + l\,\hat{e}_1))^\top L^{-1} u \geq -\alpha(h_i^t(q)).$$

Note that this expresses the dynamics in the local frame of the robot, making global state estimation redundant. Note also that by considering a point at distance l from the center, the safety distance D must be chosen so that $D \geq r + l$, where r is the radius of the robot.

Given this, we can formulate a quadratic program (QP) to determine a collision-free control input, with one CBF constraint for each obstacle $i \in N(t)$:

$$\begin{aligned} u(t) = \quad & \arg\min_{u \in U} \left\| L^{-1}(u - u_t^{\text{ref}}) \right\|^2 \\ \text{s.t.} \quad & 2(P_i(-x_{o,i} + l\,\hat{e}_1))^\top L^{-1} u \\ \geq \quad & -\alpha\big((-x_{o,i} + l\,\hat{e}_1)^\top P_i (-x_{o,i} + l\,\hat{e}_1) - 1\big) \end{aligned}$$

In essence, this QP serves as a safety-filter that computes the control input that minimally deviates from a reference input u_t^{ref}, weighted by L^{-1}. The nominal input $u_t^{\text{ref}} = \left[v^{\text{ref}}, \omega^{\text{ref}} \right]^\top$ is calculated using a globally asymptotically stable control law [7], given by

$$\begin{aligned} v^{\text{ref}} &= k_1 \|p - \bar{p}\|_2 \cos\phi \\ \omega^{\text{ref}} &= -k_1 \cos\phi \sin\phi - k_2 \phi \end{aligned}$$

where p is the current position of the robot, $\bar{p}$ is the reference point, ϕ is the heading error and k_1, k_2 are control gains.

III. Experiments

The proposed approach was successfully tested on a LIDAR-equipped TurtleBot4 platform. We have considered two different scenarios for experimental validations: **Scenario 1:** The initial context is $\kappa_1 = \{\mathtt{c}_5, \mathtt{e}_1, \mathtt{enter}_1^c\}$, activating the RWA objective $\mathscr{R} = \{\mathtt{c}_2\}, \mathscr{A} = \{\mathtt{c}_1, \mathtt{o}\}$, leading to the context $\kappa_2 = \{\mathtt{c}_2, \mathtt{e}_1, \mathtt{enter}_1^c, \mathtt{leave}_1^e\}$. After

10.18416/SCP.2025.2010

Figure 2: Scenario 1: R_c is instructed to go to Q_1 (yellow) from Q_5 (red), but has to go to Q_2 (purple) instead to request R_e (marked with a red flag) to leave.

Figure 3: Scenario 2: R_c is instructed to move to Q_5 (red) from Q_3 (blue), but the shortest path through Q_4 (green) is blocked by a human. Hence it instead opts for an alternative route passing through Q_2 and Q_1 after requesting R_e to leave.

$\mathtt{e}_1$ is false, the robot can reach the final target context $\kappa_3 = \{\mathtt{c}_1, \mathtt{e}_5\}$. This scenario is illustrated in Figure 2, which shows an overlay of multiple time-steps.

Scenario 2: The initial context is $\kappa_1 = \{\mathtt{c}_3, \mathtt{e}_1, \mathtt{human}_4, \mathtt{enter}_5^c, \mathtt{leave}_1^e\}$. After reaching $\kappa_2 = \{\mathtt{c}_2, \mathtt{e}_1, \mathtt{human}_4, \mathtt{leave}_1^e, \mathtt{enter}_5^c\}$ and waiting for $\mathtt{e}_1$ to become false, the intermediate context $\kappa_3 = \{\mathtt{c}_2, \mathtt{e}_5, \mathtt{human}_4, \mathtt{enter}_5^c\}$ and the final target context $\kappa_4 = \{\mathtt{c}_5, \mathtt{e}_5\}$ can be reached. This progression is illustrated in Figure 3.

It is important to note that all strategic decisions described in the two outlined scenarios are automatically taken by the logical control layer based on the (changing) logical context. The context is given by the different truth values of the involved logical variables which are triggered by either the robot or the environment. To actuate these chains of strategic tasks, R_c navigates through the workspace with strong safety guarantees, ensuring it avoids obstacles by maintaining a safe distance. Thus, the lower feedback controller is able to handle dynamic constrains in real-time, while the higher layer is able to strategically react to context changes. A video of the experiments is available at: https://cloud.mpi-sws.org/index.php/s/kPLDWSNgnF3m9Kr

IV. Conclusion

We introduce a novel two-layered methodology for the automated synthesis of a hybrid controller that seamlessly integrates dynamic strategic decisions into logical control software. At the higher layer, we generate a sequence of RWA control objectives based on logical variables manipulated by the environment. We then implement each active RWA problem in the lower layer by a feedback controller that utilizes CBFs to enforce all avoid obligations, even for unknown, moving obstacles. Experimental validation conducted on a Turtlebot4 serves to confirm the practical efficacy of our proposed approach. In future work, we plan to incorporate a game-based technique for both agents, ensuring guaranteed reachability, and conduct further experimental validations.

Acknowledgments

The work has been carried out at the Max Planck Institute for Software Systems and co-supervised by Anne-Kathrin Schmuck and Georg Schildbach, Institute for Electrical Engineering in Medicine, Universität zu Lübeck.

Author's statement

Authors state no conflict of interest. AI tools were used for the linguistic fine-tuning of this paper.

References

[1] G. Reissig, A. Weber, and M. Rungger. Feedback refinement relations for the synthesis of symbolic controllers. *IEEE Transactions on Automatic Control*, 62(4):1781–1796, 2017, doi:10.1109/tac.2016.2593947.

[2] D. Gundana and H. Kress-Gazit. Event-based signal temporal logic synthesis for single and multi-robot tasks. *IEEE Robotics and Automation Letters*, 6(2):3687–3694, 2021, doi:10.1109/lra.2021.3064220.

[3] S. P. Nayak, L. N. Egidio, M. Della Rossa, A.-K. Schmuck, and R. M. Jungers. Context-triggered abstraction-based control design. *IEEE Open Journal of Control Systems*, 2:277–296, 2023, doi:10.1109/OJCSYS.2023.3305835.

[4] G. A. Borges and M.-J. Aldon. Line extraction in 2d range images for mobile robotics. *Journal of Intelligent and Robotic Systems*, 40(3):267–297, 2004, doi:10.1023/b:jint.0000038945.55712.65.

[5] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, Control barrier functions: Theory and applications, in *2019 18th European Control Conference (ECC)*, IEEE, 2019. doi:10.23919/ecc.2019.8796030.

[6] P. Glotfelter, I. Buckley, and M. Egerstedt. Hybrid nonsmooth barrier functions with applications to provably safe and composable collision avoidance for robotic systems. *IEEE Robotics and Automation Letters*, 4(2):1303–1310, 2019, doi:10.1109/LRA.2019.2895125.

[7] S.-O. Lee, Y.-J. Cho, M. Hwang-Bo, B.-J. You, and S.-R. Oh, A stable target-tracking control for unicycle mobile robots, in *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000) (Cat. No.00CH37113)*, ser. IROS-00, IEEE. doi:10.1109/iros.2000.895236.