University of Sheffield

# A ROS-Based Framework for Programming and Verifying Collision-Free Behaviours for Collaborative Robots – Literature Review


Artur Graczyk

# Literature Review

In this chapter we review the most essential software tools that we will be using in our work (Section 1) and the some of the relevant research in the area (Section 2).

## 1 Tools and technical concepts

We are building a versatile framework for guaranteeing collision avoidance, which will make use of a variety of tools and frameworks.

### 1.1 ROS

ROS (Robot Operating System) is a collection of libraries and tools for developing robot software. Although ROS is not an operating system in the full sense of the word, it provides a complete system of inter-process communication and support for low-level device control. In ROS, communication takes place by creating nodes that represent single processes that are acting with a great deal of autonomy. ROS also introduces an abstraction over the hardware layer. Nodes communicate with each other by continuously broadcasting and listening for messages of a specific type and subject. It provides a simulation environment and support for data visualization, which translates into quick and efficient testing of new robots [10].

Thanks to its powerful abstraction features which make it easy to learn and use, ROS has been widely deployed in academic research labs and government agencies as their robot development platform. This helped building a large, active community of enthusiasts, which is constantly developing new technologies and improving existing solutions by making new packages available [7]. ROS has begun to migrate beyond academia , being increasingly used in industry, where it is being deployed to control robots in everyday use in production or other warehouse operations [6].

ROS will be our main platform on which we develop our domain specific language and tools.

### 1.2 ROS-Industrial

ROS-Industrial [3] is a project that extends the ROS capabilities with software libraries and hardware that are tailored to the needs of industry, and advanced manufacturing in particular. Compared with those avilable for ROS, the ROS-industrial software responsible for path/motion planning (industrial_moveit) [2] are still in the experimental phase. As part of verification case study at the AMRC, we will look into extending our tools with ROS-industrial features, as suitable for the robotic devices involved in that case study.

### 1.3 The Gazebo simulator

Gazebo is an advanced simulation for robots that can be freely reconfigured and expanded. It is fully compatible with ROS. Simulation can be run from ROS, and its API can be used to control robots in simulations, via sending to and receiving data from them [4].

This software allows performing realistic simulations of physics affecting objects inside the virtual world. Robots can interact with the simulated environment. They can pick up and push objects, roll and slide on the ground, and bump into obstacles. The virtual environment can act on the robot too, using gravity. To make this possible, Gazebo uses multiple physics engines such as Open Dynamics Engine (ODE), Bullet, Symbody, or DART.

It is possible to create Robots by models in Unified Robot Description Format (URDF) and load them at runtime. In addition, it is possible to create simulation scenarios (worlds) that modify the characteristics of contact with the ground, obstacles, and gravity values in all three dimensions. Gazebo includes various plugs for adding sensors to a robot model and simulating them, covering aspects such as odometry, contact force, laser sensors, and stereo cameras. It also allows training of AI-based solutions.

Most individual functionalities introduced in the framework during our development process will be intensively tested using Gazebo simulations.

### 1.4 The RVIZ visualisation tool

RVIZ [8] is a 3D visualization tool. It uses data from all kinds of sensors installed on the robot, including be camera images, infrared distance measurements, sonar data. In addition, information can also come from robot systems from all related ROS topics. RVIZ displays a graphical representation of these values in real-time. It also enables the programming of the robot model's movements through the appropriate additional software [1].

This tool will be used extensively during the testing and software development for this study because one of the functionalities responsible for virtual obstacles added to the map of the surroundings in the navigation stack will only be visible through the robot's navigation system. One should interpret these objects in the same way as actual physical obstacles from the surroundings. Therefore, RVIZ will replace the Gazebo simulation, which can only display physical obstacles.

### 1.5 OpenCV computer vision library

OpenCV stands for Open Source Computer Vision [5]. It is an open-source cross-platform library of functions used in image, video processing and real-time computer vision, incorporating machine learning algorithms.

In our work, will be used to generate geometric figures according to strictly defined parameters provided by the path finder mechanism for the robot and added to its navigation map. In the later stages of our work, we will also use it to detect objects in the area mapped out for the robot's movement. This will provide the necessary feedback for the algorithm responsible for the safety mechanism. and help set up a safe route.

### 1.6 The Isabelle verifier

Isabelle is a general-purpose interactive theorem prover based on higher-order logic [42]. The implementation languages of Isabelle are Standard ML and Scala. It allows defining mathematical models for software and hardware systems and formulating properties about them, and proving these properties in a logical calculus using a combination of interactive and automatic methods.

Isabelle will the backbone of part of our framework that is responsible for verification. It will be connected with our programming language environment via a code generator.

## 2 Relevant Literature

The two pillars of our planned collision avoidance framework are a verification infrastructure and a safe navigation infrastructure. Concerning the latter, we discuss relevant literature on robot path planning (§2.1) and an architecture for safe navigation (§2.2). Concerning the former, we discuss existing work on robot verification (§2.4). We also discuss the similarities and differences between our work and the area of AI (task) planning (§2.3). We conclude with a summary of what we regard to be the novel contributions of our work (§**??**).

### 2.1 Robot path planning

Obstacle avoidance methods for mobile robots have been the target of extensive research as part of the broader area of path planning (also called motion planning,[1] or trajectory planning) [13, 20, 38, 47]. Latombe's influential monograph [34] formulates the following question as the fundamental question of path (motion) planning: "How can a robot decide what motions to perform in order to achieve goal arrangements of physical objects?" [34]. A particular case of this problem is determining paths to navigate the physical space in such a way that certain target positions are being reached, while avoiding given physical obstacles. Formally, the robot has to navigate a configuration space, where the points in the configuration characterize the spacial position of the robot, but also potentially other characteristics such as velocity and acceleration. The *basic version* of the path planning problem deals with positions only and ignores other aspects. Several approaches have been proposed to address this problem, including:

- Roadmap [34] (the creation of a "roadmap" consisting of one-dimensional curves that connect the obstacles' edges through the free space of the configuration space, which can guide the creation of paths)
- Cell Decomposition [34] (decomposing the configuration space into smaller navigation-homogeneous regions and navigating along their adjacency graph)
- Potential Field [32] (endowing the obstacles and the target position with some rejection and attraction potential whose overall resultant guides the navigation locally, as if moving a particle through a potential field)

[1] While "path planning" and "motion planning" are often used as synonyms, they are sometimes distinguished as follows: Path planning is concerned with constructing a path from a starting point to an endpoint given a full, partial or dynamic map, whereas motion planning is concerned with determining the set of actions needed for a robot in order follow the path.

Several other approaches and refinements of the above approaches are surveyed in [38, 47]. One can roughly distinguish between *global* and *local* path planning. With global path planning, one uses prior (total or partial) knowledge about the environment in which a robot is about to move, which can allow the system to choose offline paths that match various optimality criteria [15]. With local path planning, the robot analyses its direct surroundings based on sensor readings while moving and can react by adjusting the path to changes not taken into account when planning a global path, for example the change in position of obstacles or other events in a dynamic environment [16]. For example, the first two approaches mentioned above (Roadmap and Cell Decomposition) are best suited for global path planning, assuming that a complete model of the robot's environment is available. By contrast, the Potential Field approach is fundamentally local, making the best decision under the available "local field" influences. The Dynamic Window approach [24] (which has been the target of some verification work, see Section 2.4) also takes a local view to path planning. It focuses on the velocities available within a short time interval and making sure that the robot can stop before hitting the closest obstacle; this is more appropriate than global approaches when dealing with high speeds and fast obstacle avoidance. Path planning is a main component of robot navigation, which additionally includes self-localization and map building, and map interpretation.

In our work, we will *not* contribute new planning/navigation algorithms – but will rather deploy state of the art algorithms to ensure safe navigation in a fairly controlled setting, namely *map-based* navigation though "corridors" along trajectories that have already been deemed fundamentally safe in the static verification stage (as we detail in Chapter **??**). We assume that the high-level planning (the "what") has been handled by the robot program designer, and then we verify that this planning is indeed collision free, more precisely, free of what we call fundamental collisions; if we think of collaborative robots as threads in a multi-threaded program, than the lack of fundamental collisions can be compared to race freedom. Our novel contribution will consist in a global, map-based navigation scheme through pre-computed collision-free paths *whose construction is informed at the verification stage* (and delivered in the form of virtual corridors that are added to the robots' maps, as explained in Chapter **??**).

Another classification of path planning methods (also in [38]) distinguishes between mathematical-model based, bio-inspired, sampling based and docomposition based methods for path planning. In our initial experiments, we chose a sampling based method, employing the Rapidly Exploring Random Tree (RRT) algorithm [30, 31, 48, 14] for searching a path in 2D. This is an efficient algorithm, although it is incomplete (like all sampling based methods) and does not necessarily give an optimal solution w.r.t. distance—we are considering employing an improvement, RRT$^*$ [14], which guarantees optimal solutions and gives better results in 3D. However, our combined verification / safe navigation scheme is agnostic with respect to the particular method that we deploy for path finding. We envision a framework where different path planning algorithms can be deployed on a plug and play fashion, depending on which one fits the particular problem; the interaction between path planning and verification will proceed in exactly the same way.

Finally, other distinctions that are being made in the literature [38] are between (1) centralized and decentralized, and (2) real-time versus offline (pre-computation based) methods for robot management systems, and robot path planning in particular. The centralized

methods are usually based on providing one central computing server, which is then responsible for the operation of each individual robot based on the received data. Such systems have top-down information about each robot, as well as its goals, starting position and surroundings. Based on this knowledge, the algorithms can calculate collision-free paths for each robot. By contrast, decentralized methods count on the decisions of the internal systems of each robot separately, which in their operation do not know the goals and intentions of other robots in their environment; no communication is required between robots to achieve the assigned goals. In our work, the verification stage offers a *centralized* view that creates a more predictable environment, where we then enable *decentralized* execution for the individual robots – so we employ a hybrid between the centralized and decentralized methods for managing multi-robot systems. Moreover, real-time aspects are largely abstracted away from our framework, and the verification-informed calculations happen offline—at least before we move to factor in humans in the framework.

## 2.2 Safe Navigation Architecture

Macek et al. [37] propose a safe navigation architecture that distinguishes between a *Route Planner* for computing valid itineraries towards given goals, and a *Partial Motion Planner* for ensuring the safe, obstacle avoiding navigation along the itineraries (where the obstacles can be moving objects, pedestrians, other vehicles, etc.). While this architecture was primarily designed for autonomous vehicles operating within urban roads, its ideas are more general. The Route Planner component essentially corresponds to the high-level aspects (the "what" in Chapter **??**) of a robot behavior which we plan to verify, whereas the Partial Motion Planner component corresponds to what we plan to handle via quasi-dynamic safe navigation techniques (the "how" in Chapter **??**). A distinguishing feature of the scenarios we plan to cover in our work, which is not emphasized by this architecture, is the highly cooperative nature of the tasks – which can make the Route Planner component non-trivial. For example, a cooperative task at the factory floor will not be only about establishing some waypoints that must be visited in succession – but may require some synchronization between different robots; in addition, the sequence of waypoints to be visited by each individual robot may not be entirely predetermined.

Macek et al.'s work is important also because it introduces some useful collision-safety properties: *Passive Safety* and *Passive Friendly Safety*. Both properties refer to individual vehicles (and not the entire set of participating vehicles), and are ways of expressing that, should a collision occur, it will not be that vehicle's "fault". Passive Safety states this in a weak form: A collision can only occur while the vehicle is not moving. Passive Friendly Safety states a stronger (and more reasonable) property: Any collision is avoidable, i.e., it is within the braking (stopping) capabilities of the other vehicles to avoid the collision. These properties are of course not as strong as one that would guarantee overall collision freeness of the entire traffic, but represents a sensible compromise – given the difficulty of the problem. In our work, we will prove collision freeness of the entire system, which is feasible because of the more predictable nature of the required tasks.

## 2.3 AI Planning

Another area that has points of intersection with our work is AI (task) planning [29], which deals with high-level planning and scheduling of tasks that must be achieved by robotic systems. The planning problems are defined using formal languages such as STRIPS [23] and PDDL [25]. These languages can describe objects, relations between them, and actions, usually employing the formalism of first-order logic. A planning problem is defined by an initial state of the world, the available actions for changing the state, some constraints on the possible states, and the goal expressed as a condition on the end state. Solving a planning problem means coming up with a sequence of actions that respect the constraints and achieve the goal—dedicated systems called AI planners implement algorithms for tackling these problems. The above classical planning problem also has temporal and probabilistic extensions, which allow dealing with timing constraints and uncertainty.

Planning systems have been successfully integrated with robot software development systems [19, 39]; in particular, ROSPlan [17] is such an integration for ROS. The way most of these integrations work is by allowing the developer to associate concrete implementations to high-level actions specified in languages such as PDDL.

While the planning problem is essentially a *synthesis problem*, our work focuses on a *verification and analysis problem*. Our domain specific language has some similarities with planning languages such as PDDL in that it focuses on high-level aspects of the robots' actions (in our case, specifically on actions that are relevant w.r.t. collision) and abstracts away the particular details on how the actions are achieved. However, our language is different from a planning language in that it is imperative rather than declarative—which allows the programmer to indicate the desired sequence of actions (modulo some nondeterminism brought by concurrency). In other words, we do not tackle the (high-level) planning problem, but assume that this has already been resolved. This having been said, our formal bridge between ROS and Isabelle could be in the future combined with an embedding of PDDL in Isabelle to integrate the planning step as well in the certification chain.

## 2.4 Verification of robotic systems

The general area of robotic system safety assurance and verification is a large and rapidly growing area. Important subareas include the verification of autonomous robotic systems [21, 36] and of industrial collaborative robots [22, 26] using methods such as theorem proving, model checking and safety controller synthesis and monitoring. The employed formal models include timed automata [28] and process algebras [43]. Our work will mostly apply to robotic systems that are involved in pre-determined and largely pre-scripted collaborative tasks (where the "script", the robot program, is responsible for avoiding what we will call "fundamental" collision hazards), but still have a certain degree of autonomy (which will be used in avoiding what we will call "incidental" collision hazards).

Below we highlight some of the verification work that is closer to the work we plan here – either by the target or the method of verification.

**Verification using theorem provers / proof assistants** Theorem provers offer one of the highest forms of certification, since they produce formal proofs that certain desirable properties are true. However, there has been little work in theorem-prover-based verification

of robotic systems. Täubig et al. [46] and Mitsch et al. [40] formally prove the correctness of various aspects of the Dynamic Window algorithm [24]. Täubig et al. use the Isabelle/HOL prover combined with some paper proofs to verify that the C implementation of the algorithm meets its specification: a Passive-Safety-like property. By contrast, Mitsch et al. do not prove correct an implementation, but a more high-level description of the algorithm; on the other hand, they address not only the discrete digital part, but also model the continuous behavior of the systems – using the hybrid theorem prover KeYmaera based on Differential Dynamic Logic; they formalize and prove both Passive Safety and Passive Friendly Safety. Liming et al. perform the formal verification of a two-arm cooperative task [35] in the HOL4 theorem prover [45]. Our work will provide a systematic solution for supporting collision-freeness verification efforts on a larger scale. Our approach will be to abstract away as much as possible from the technical knowledge about the specifics of the robots *before* verification time, so that verification can proceed at a purely discrete level. For example, the verification stage will count on the pre-computation of some upper bounds on the space taken by a robot to perform certain tasks at certain locations, so that verification would have to make sure the over-approximated space does not overlap with the space taken by other robots. Admittedly, our approach trades some precision and verification coverage for feasability and automation (in that we will not verify these domain-specific pre-computations) and restricts the application scope to robotic tasks where these details *can* be abstracted away.

**Verification of collaborative robots** Previous work dealt with the verification of collaborative robots forming *swarms* (where all robots have identical behavior and are typically large in number) [44] or *teams* (where different robots may have distinct behaviors) [33]. The safety of human-robot collaboration has been addressed in the CSI:Cobot project led by the Universities of Sheffield and York – focusing specifically on safety controller synthesis [26, 22, 26, 27]. Part of this work has been performed within a digital twin framework [22], which facilitates the fine control over safety monitoring mechanisms. Our work will take place in a less sophisticated framework based on ROS, but the techniques we develop will be potentially applicable to that framework as well. We have initially investigated implementing our verification infrastructure on top of the CSI digital twin framework, but in the end we chose ROS due to the higher simplicity and more comprehensive documentation. However, the CSI framework will inspire our work when we will move to factor in humans in our collision avoidance framework; in particular, their approach of employing safety modes to switch to safer activities depending on whether a human is present in the work area can be incorporated into our ROS-based domain-specific language by distinguishing between different types of local activities. Another commonality between the CSI:Cobot line of work and ours is the focus on examples and case studies from manufacturing. It would be interesting to see how our framework can be deployed to some of their case studies, to combine the human safety guarantees with general-purpose collision-freeness guarantees.

**Frameworks for verifiable robotic software** BIP (Behavior Interaction Priority) [11] is a framework for the compositional modeling of real-time software, employing a finite state machine formalism. It has been integrated [9] with the G$^{\text{en}}$oM robot software architecture to synthesize robot control software that is correct by construction, and in particular to synthesize software for controlling a wheeled rover robot; and with the LAAS autonomous

robotic software, for which safety properties have been verified [12]. This BIP-based work employed model checking and fault injection as verification and testing techniques. By contrast, our framework will be centered on theorem proving and static analysis imported from the world of concurrency.

The aims of our framework has similarities to those of the RoboChart framework, with the difference that our scope is not as broad but is specialized to collision avoidance. RoboChart [41] is a specification language for robotic systems developed at the University of York as an extensive collaborative effort involving several researchers. It allows specifications of interactive state machines in an UML-style notation, which is given semantics using the CSP process algebra. RoboChart is supported by a collection of Eclipse plugins called RoboTool that provides a graphical and textual editor and export of models and properties to various model checkers, such as FDR and PRISM. RoboSim [18] is a simulation language, also based on state machines, that can be used in conjunction with RoboChart.

The following table shows a rough comparison between our planned work and the University of York line of work:

| | RoboChart/RoboTool/RoboSym | ROSCollA |
|---|---|---|
| Computation model | UML-based , CSP semanitcs | ROS-based DSL |
| Targeted properties | deadlock-freeness<br>determinism<br>divergence-freeness | collision avoidance |
| Verification/validation approach | static – refinement proofs<br>dynamic-simulation | static – collision-freedom proofs<br>dynamic-corridors |
| Verification tools | model checkers<br>Isabelle theorem prover (ongoing) | Isabelle theorem prover<br>static analysis tools<br>navigation & path finding tools |
| Time representation | explicit | implicit |

# Bibliography


[1] ROS robotics by example, 2016.

[2] Industrial_moveit - ROS Wiki, 2022. URL: http://wiki.ros.org/industrial_moveit.

[3] ROS-Industrial, 2022. URL: https://rosindustrial.org/about/description/.

[4] The Gazebo Website, 2022. URL: http://gazebosim.org.

[5] The OpnCV Website, 2022. URL: https://opencv.org.

[6] The ROS-Industrial Website, 2022. URL: https://rosindustrial.org/.

[7] The ROS Website, 2022. URL: https://www.ros.org.

[8] The RVIZ Visualizer, 2022. URL: https://sdk.rethinkrobotics.com/wiki/Rviz.

[9] Tesnim Abdellatif, Saddek Bensalem, Jacques Combaz, Lavindra de Silva, and Felix Ingrand. Rigorous design of robot software: A formal component-based approach. *Robotics and Autonomous Systems*, 60(12):1563–1578, 2012. URL: https://www.sciencedirect.com/science/article/pii/S0921889012001510, doi:https://doi.org/10.1016/j.robot.2012.09.005.

[10] Adnan Ademovic. An Introduction to Robot OS, 2022. URL: https://www.toptal.com/robotics/introduction-to-robot-operating-system.

[11] A. Basu, M. Bozga, and J. Sifakis. Modeling heterogeneous real-time components in bip. In *Fourth IEEE International Conference on Software Engineering and Formal Methods (SEFM'06)*, pages 3–12, 2006. doi:10.1109/SEFM.2006.27.

[12] Ananda Basu, Matthieu Gallien, Charles Lesire, Thanh-Hung Nguyen, Saddek Bensalem, Félix Ingrand, and Joseph Sifakis. Incremental component-based construction and verification of a robotic system. In *Proceedings of the 2008 Conference on ECAI 2008: 18th European Conference on Artificial Intelligence*, page 631–635, NLD, 2008. IOS Press.

[13] Francisco Bonin-Font, Alberto Ortiz, and Gabriel Oliver. Visual navigation for mobile robots: A survey. *J. Intell. Robotics Syst.*, 53(3):263–296, nov 2008. doi:10.1007/s10846-008-9235-4.

[14] João Braun, Thadeu Brito, José Lima, Paulo Costa, Pedro Costa, and Alberto Nakano. A comparison of a* and rrt* algorithms with dynamic and real time constraint scenarios for mobile robots. In *Proceedings of the 9th International Conference on Simulation and Modeling Methodologies, Technologies and Applications*, SIMULTECH 2019, page 398–405, Setubal, PRT, 2019. SCITEPRESS - Science and Technology Publications, Lda. doi:10.5220/0008118803980405.

[15] N Buniyamin, N Sariff, WAJ Wan Ngah, and Z Mohamad. Robot global path planning overview and a variation of ant colony system algorithm. *International journal of mathematics and computers in simulation*, 5(1):9–16, 2011.

[16] N Buniyamin, WAJ Wan Ngah, N Sariff, and Z Mohamad. A simple local path planning algorithm for autonomous mobile robots.

[17] Michael Cashmore, Maria Fox, Derek Long, Daniele Magazzeni, Bram Ridder, Arnau Carrera, Narcís Palomeras, Natàlia Hurtós, and Marc Carreras. ROSPlan: Planning in the Robot Operating System. In Ronen I. Brafman, Carmel Domshlak, Patrik Haslum, and Shlomo Zilberstein, editors, *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling, ICAPS 2015, Jerusalem, Israel, June 7-11, 2015*, pages 333–341. AAAI Press, 2015. URL: http://www.aaai.org/ocs/index.php/ICAPS/ICAPS15/paper/view/10619.

[18] Ana Cavalcanti, Augusto Sampaio, Alvaro Miyazawa, Pedro Ribeiro, Madiel Conserva Filho, André Didier, Wei Li, and Jon Timmis. Verified simulation for robotics. *Sci. Comput. Program.*, 174:1–37, 2019. doi:10.1016/j.scico.2019.01.004.

[19] Caroline Ponzoni Carvalho Chanel, Charles Lesire, and Florent Teichteil-Königsbuch. A robotic execution framework for online probabilistic (re)planning. In Steve A. Chien, Minh Binh Do, Alan Fern, and Wheeler Ruml, editors, *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2014, Portsmouth, New Hampshire, USA, June 21-26, 2014*. AAAI, 2014. URL: http://www.aaai.org/ocs/index.php/ICAPS/ICAPS14/paper/view/7928.

[20] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George Kantor, Wolfram Burgard, Lydia Kavraki, and Sebastian Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, May 2005.

[21] Clare Dixon. Verifying autonomous robots: Challenges and reflections (invited talk). In Emilio Muñoz-Velasco, Ana Ozaki, and Martin Theobald, editors, *27th International Symposium on Temporal Representation and Reasoning, TIME 2020, September 23-25, 2020, Bozen-Bolzano, Italy*, volume 178 of *LIPIcs*, pages 1:1–1:4. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.TIME.2020.1.

[22] James A. Douthwaite, Benjamin Lesage, Mario Gleirscher, Radu Calinescu, Jonathan M. Aitken, Rob Alexander, and James Law. A modular digital twinning framework for safety assurance of collaborative robotics. *Frontiers Robotics AI*, 8:758099, 2021. doi:10.3389/frobt.2021.758099.

[23] Richard E. Fikes and Nils J. Nilsson. STRIPS: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3):189–208, 1972.

[24] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, 4(1):23–33, 1997. doi:10.1109/100.580977.

[25] M. Ghallab, A. Howe, C. Knoblock, D. Mcdermott, A. Ram, M. Veloso, D. Weld, and D. Wilkins. PDDL—The Planning Domain Definition Language, 1998. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.37.212.

[26] Mario Gleirscher and Radu Calinescu. Safety controller synthesis for collaborative robots. In Yi Li and Alan Wee-Chung Liew, editors, *25th International Conference on Engineering of Complex Computer Systems, ICECCS 2020, Singapore, October 28-31, 2020*, pages 83–92. IEEE, 2020. doi:10.1109/ICECCS51672.2020.00017.

[27] Mario Gleirscher, Radu Calinescu, James A. Douthwaite, Benjamin Lesage, Colin Paterson, Jonathan M. Aitken, Rob Alexander, and James Law. Verified synthesis of optimal safety controllers for human-robot collaboration. *Sci. Comput. Program.*, 218:102809, 2022. doi:10.1016/j.scico.2022.102809.

[28] Raju Halder, José Proença, Nuno Macedo, and André Santos. Formal verification of ros-based robotic applications using timed-automata. In *2017 IEEE/ACM 5th International FME Workshop on Formal Methods in Software Engineering (FormaliSE)*, pages 44–50, 2017. doi:10.1109/FormaliSE.2017.9.

[29] James A. Hendler, Austin Tate, and Mark Drummond. Ai planning: Systems and techniques. *AI Magazine*, 11(2):61, Jun. 1990. URL: https://ojs.aaai.org/index.php/aimagazine/article/view/833, doi:10.1609/aimag.v11i2.833.

[30] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011. arXiv:https://doi.org/10.1177/0278364911406761, doi:10.1177/0278364911406761.

[31] Sertac Karaman, Matthew R. Walter, Alejandro Perez, Emilio Frazzoli, and Seth J. Teller. Anytime motion planning using the RRT. In *IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China, 9-13 May 2011*, pages 1478–1483. IEEE, 2011. doi:10.1109/ICRA.2011.5980479.

[32] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98, 1986. arXiv:https://doi.org/10.1177/027836498600500106, doi:10.1177/027836498600500106.

[33] Marius Kloetzer, Xu Chu Ding, and Calin Belta. Multi-robot deployment from LTL specifications with reduced communication. In *50th IEEE Conference on Decision and Control and European Control Conference, 11th European Control Conference, CDC/ECC 2011, Orlando, FL, USA, December 12-15, 2011*, pages 4867–4872. IEEE, 2011. doi:10.1109/CDC.2011.6160478.

[34] Jean-Claude Latombe. *Robot motion planning*, volume 124 of *The Kluwer international series in engineering and computer science*. Kluwer, 1991. doi:10.1007/978-1-4615-4022-9.

[35] Liming Li, Zhiping Shi, Yong Guan, Chunna Zhao, Jie Zhang, and Hongxing Wei. Formal verification of a collision-free algorithm of dual-arm robot in hol4. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1380–1385, 2014. doi:10.1109/ICRA.2014.6907032.

[36] Matt Luckcuck, M. Farrell, L.A. Dennis, C. Dixon, and M. Fisher. Formal specification and verification of autonomous robotic systems: A survey. *ACM Computing Surveys*, 52(5), September 2019. doi:10.1145/3342355.

[37] Kristijan Macek, Dizan Alejandro Vasquez Govea, Thierry Fraichard, and Roland Y. Siegwart. Towards Safe Vehicle Navigation in Dynamic Urban Scenarios. *Automatika - Journal for Control, Measurement, Electronics, Computing and Communications*, November 2009. URL: https://hal.inria.fr/inria-00447452.

[38] Ángel Madridano, Abdulla Al-Kaff, David Martín, and Arturo de la Escalera. Trajectory planning for multi-robot systems: Methods and applications. *Expert Syst. Appl.*, 173:114660, 2021. doi:10.1016/j.eswa.2021.114660.

[39] Conor McGann, Frederic Py, Kanna Rajan, Hans Thomas, Richard Henthorn, and Robert S. McEwen. A deliberative architecture for AUV control. In *2008 IEEE International Conference on Robotics and Automation, ICRA 2008, May 19-23, 2008, Pasadena, California, USA*, pages 1049–1054. IEEE, 2008. doi:10.1109/ROBOT.2008.4543343.

[40] Stefan Mitsch, Khalil Ghorbal, and André Platzer. On provably safe obstacle avoidance for autonomous robotic ground vehicles. In Paul Newman, Dieter Fox, and David Hsu, editors, *Robotics: Science and Systems IX, Technische Universität Berlin, Berlin, Germany, June 24 - June 28, 2013*, 2013. URL: http://www.roboticsproceedings.org/rss09/p14.html, doi:10.15607/RSS.2013.IX.014.

[41] Alvaro Miyazawa, Pedro Ribeiro, Wei Li, Ana Cavalcanti, Jon Timmis, and Jim Woodcock. Robochart: modelling and verification of the functional behaviour of robotic applications. *Softw. Syst. Model.*, 18(5):3097–3149, 2019. doi:10.1007/s10270-018-00710-z.

[42] Tobias Nipkow, Lawrence Paulson, and Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.

[43] Matthew O'Brien, Ronald C. Arkin, Dagan Harrington, Damian Lyons, and Shu Jiang. Automatic verification of autonomous robot missions. In Davide Brugali, Jan F. Broenink, Torsten Kroeger, and Bruce A. MacDonald, editors, *Simulation, Modeling, and Programming for Autonomous Robots*, pages 462–473, Cham, 2014. Springer International Publishing.

[44] Christopher A. Rouff, Amy Vanderbilt, Walter Truszkowski, James L. Rash, and Michael G. Hinchey. Formal methods for autonomic and swarm-based systems. In Tiziana Margaria, Bernhard Steffen, Anna Philippou, and Manfred Reitenspieß, editors, *International Symposium on Leveraging Applications of Formal Methods, ISoLA 2004, October 30 - November 2, 2004, Paphos, Cyprus. Preliminary proceedings*, volume TR-2004-6 of *Technical Report*, pages 100–102. Department of Computer Science, University of Cyprus, 2004.

[45] Konrad Slind and Michael Norrish. A brief overview of HOL4. In *TPHOLs*, pages 28–32, 2008.

[46] 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. *Auton. Robots*, 32(3):303–331, 2012. doi:10.1007/s10514-011-9271-y.

[47] Mohd. Nayab Zafar and J.C. Mohanta. Methodology for path planning and optimization of mobile robots: A review. *Procedia Computer Science*, 133:141–152, 2018. International Conference on Robotics and Smart Manufacturing (RoSMa2018). URL: https://www.sciencedirect.com/science/article/pii/S1877050918309621, doi:https://doi.org/10.1016/j.procs.2018.07.018.

[48] Christian Zammit and Erik-Jan van Kampen. Comparison between a* and RRT algorithms for 3d UAV path planning. *Unmanned Syst.*, 10(2):129–146, 2022. doi:10.1142/S2301385022500078.