SpaTiaL

monitoring and planning of robotic tasks using spatio-temporal logic specifications

Pek, Christian; Schuppe, Georg Friedrich; Esposito, Francesco; Tumova, Jana; Kragic, Danica

DOI
10.1007/s10514-023-10145-1

Publication date
2023

Document Version
Final published version

Published in
Autonomous Robots

Citation (APA)
Pek, C., Schuppe, G. F., Esposito, F., Tumova, J., & Kragic, D. (2023). SpaTiaL: monitoring and planning of robotic tasks using spatio-temporal logic specifications. *Autonomous Robots*, *47*(8), 1439-1462. https://doi.org/10.1007/s10514-023-10145-1

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Autonomous Robots (2023) 47:1439–1462
https://doi.org/10.1007/s10514-023-10145-1

SpaTiaL: monitoring and planning of robotic tasks using spatio-temporal logic specifications

Christian Pek[1,2] · Georg Friedrich Schuppe[2] · Francesco Esposito[2] · Jana Tumova[2] · Danica Kragic[2]

Received: 31 December 2022 / Accepted: 26 September 2023 / Published online: 3 November 2023

Abstract
Many tasks require robots to manipulate objects while satisfying a complex interplay of spatial and temporal constraints. For instance, a table setting robot first needs to place a mug and then fill it with coffee, while satisfying spatial relations such as forks need to placed left of plates. We propose the spatio-temporal framework SpaTiaL that unifies the specification, monitoring, and planning of object-oriented robotic tasks in a robot-agnostic fashion. SpaTiaL is able to specify diverse spatial relations between objects and temporal task patterns. Our experiments with recorded data, simulations, and real robots demonstrate how SpaTiaL provides real-time monitoring and facilitates online planning. SpaTiaL is open source and easily expandable to new object relations and robotic applications.

Keywords Task and motion planning · Spatio-temporal logics · Monitoring · Object-centric planning

1 Introduction

Many real-world robot planning tasks require a complex interplay of spatial and temporal constraints on the objects to manipulate (Menghi et al., 2019; Billard and Kragic, 2019). For instance, in the illustrated kitchen scenario (see Fig. 1), the robot has to place the fork right of the plate; may not be allowed to place the kanelbulle on the plate until the plate is at its dedicated location; and it needs to serve the coffee within a given time bound, e.g., while it's hot In addition, the manipulated objects may have various shapes and their exact positions may be uncertain due to sensor noise. These requirements and properties make manipulation tasks considerably harder to solve.

Various approaches have been developed to streamline and standardize the specification, monitoring and planning of complex robotic tasks, e.g., using the Stanford Research Institute Problem Solver (STRIPS) (Fikes et al., 1972), the Planning Domain Definition Language (PDDL) (McDermott et al., 1998; Fox & Long, 2003) or symbolic planning (see Sect. 1.1). For instance, users can specify initial and goal states, predicates, objects and actions in PDDL. These specifications are powerful but usually become complex, since users are required to specify both what the robot needs to do and how the robot will do it, e.g., by specifying the robot's action space.

Temporal Logics (TLs) have become a popular tool to focus on specifying what robots need to do in a task without requiring information on how the robot will perform the task (Kress-Gazit et al., 2018). In contrast to classical specification languages, TLs are rich and concise, often closer to resemble natural language (Kress-Gazit et al., 2008) and can be used to specify various temporal events of tasks without explicitly modeling the robot's action space. Many TL solutions handle requirements only on the robot's state and not

Christian Pek and Georg Friedrich Schuppe contributed equally to this work.

✉ Christian Pek
c.pek@tudelft.nl

Georg Friedrich Schuppe
schuppe@kth.se

Francesco Esposito
fesp@kth.se

Jana Tumova
tumova@kth.se

Danica Kragic
dani@kth.se

[1] Department of Cognitive Robotics, Faculty of Mechanical Engineering, Delft University of Technology, 2628 CD Delft, The Netherlands

[2] Division of Robotics, Perception and Learning, School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, 114 28 Stockholm, Sweden

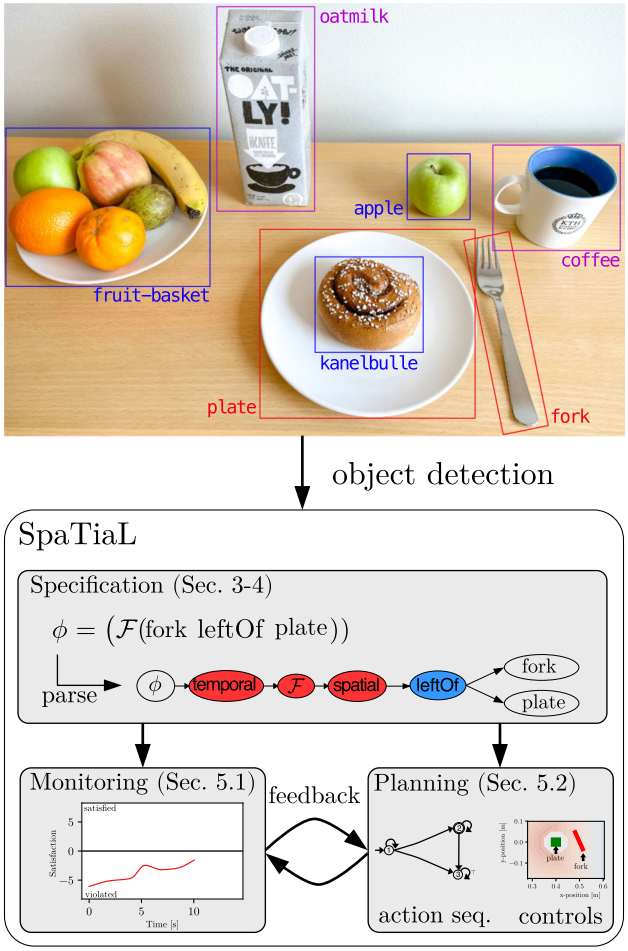


Fig. 1 The illustrated kitchen task requires a complex interplay of spatial and temporal constraints. The robot needs to arrange the objects in a predefined temporal order, e.g., first place the fork left of the plate. Moreover, it needs to consider time bounds such as placing the filled coffee mug within 10 s to the right side but close to the plate after the kanelbulle has been placed within the plate. This specification requires designers to deal with various object shapes, object detection uncertainties, and temporal orders. SpaTiaL parses user specifications and monitors their satisfaction in real-time, providing feedback to planners

on objects in the environment, target mobile robot domains, require environment abstractions, e.g., grid cells, and focus either on specifying/monitoring tasks or planning.

For complex manipulation tasks like the one shown in Fig. 1, we require a specification tool that (1) is expressive enough to cover the various spatial and temporal requirements of tasks in an object-centric fashion; (2) allows end users to easily specify tasks while being formal for automatic processing; and (3) can be used for specification, monitoring and planning. Ideally, this tool should support various object geometries, consider object detection uncertainties, and operates in a continuous world.

We present our framework SpaTiaL, that has been designed to specify complex robotic tasks in an object-centric fashion. It allows users to express various spatial and temporal relations between objects, e.g., an object is eventually left of and never close to another object (see Sect. 3 and Sect. 4). We demonstrate that SpaTiaL allows both for monitoring as well as high-level planning (see Sect. 5). SpaTiaL is declarative so that users only specify the task's goal configuration and constraints without defining high-level robot actions beforehand. We present a high-level, online planning method for SpaTiaL specifications in scenarios where the dynamics of the robot are largely unrestricted. It is able to plan next steps, monitor execution and replan in the case of failures. We illustrate the advantages of SpaTiaL in various experiments, e.g., pick-and-place tasks, in simulation and on robots.

1.1 Related work

In the following paragraphs, we briefly review classical and recent approaches for task and motion planning (TAMP) (Kress-Gazit et al., 2018; Karpas & Magazzeni, 2020) and provide an overview of the spatio-temporal relations in complex robotic tasks. We use the term specification to refer to the desired goal configurations (e.g., a fixed object arrangement) and constraints (e.g., objects should not be close to each other for a predefined time interval) of a planning task that a robot executes within an environment.

Task and motion planning Various description languages exist for different robotic applications. STRIPS is one of the earliest specification languages (Fikes et al., 1972). It models the planning problem as a state transition system (TS) with pre- and post-conditions for applying actions. The states in the TSs are composed of Boolean propositions to denote the current planning phase. STRIPS has been successfully applied to numerous tasks (Bylander, 1994; Jiménez et al., 2012) and updated over the years (Garrett et al., 2017; Aineto et al., 2018).

The Action Description Language (ADL) (Pednault, 1989) builds up on the success of STRIPS and allows users to remove the closed-world assumption (negative literals are allowed), specify goals with con- and disjunctions, and introduces conditional action effects. These additions enabled the expression of more complex robotic tasks. Although STRIPS and ADL are powerful, they usually require users to specify the whole domain and the problem instance from scratch for new applications.

PDDL (McDermott et al., 1998; Fox & Long, 2003; Garrett et al., 2020) separates domain and problem specific definitions. This choice makes planners comparable and allows users to apply domain definitions across various problem instances. PDDL can be seen as a generalized and standardized version of STRIPS. It has shown success in many applications and is supported by various systems, such as the Robot Operating System (Cashmore et al., 2015). PDDL is a powerful planning framework for robotic tasks,

but it is challenging to account for object detection uncertainties or low level control effects.

Manipulation primitives (MPs) and related concepts (Finkemeyer et al., 2005; Kröger et al., 2010; Pek et al., 2016; Sobti et al., 2021) consider low-level control effects in TAMP by interfacing robot programming and sensor-based motion control. MPs represent primitive motions that are executed with feedback control and available sensor data. By chaining MPs, one can design high-level robot programs that are automatically executed in the control architecture. These MP chains need to be carefully designed for the intended application. Similarly, Behavior Trees (BTs) are representations of a solution to a TAMP problem and follow a reactive execution approach (Ghzouli et al., 2020; Iovino et al., 2022; Colledanchise and Ögren, 2018). BTs form directed graphs where nodes correspond to predefined (sub-) tasks that are executed by (local) continuous controllers. They allow more fine-grained control of robotic systems due to their resemblance to hybrid control systems (Ögren, 2020; Sprague et al., 2018). Yet, users need to explicitly consider the problem-specific task properties to properly chain MPs or create BTs.

Recent symbolic planning approaches unify the low-level control benefits with the power of reasoning systems to automatically synthesize plans (Zhao et al., 2021; Silver et al., 2021; Loula et al., 2020), i.e., combined TAMP. Logic-geometric programming (LGP) (Toussaint, 2015; Toussaint et al., 2018), optimizes motions on a symbolic, interaction, and low level. The symbolic level decides the high-level actions, the interaction level encodes the resulting geometry of the world, and the low level takes care of the kinematic planning. LGP has shown remarkable achievements for various tasks by integrating differentiable physics and interaction modes of objects in the scene (Migimatsu & Bohg, 2020).

The aforementioned TAMP approaches focus on combining the specification and planning of tasks. Most of the approaches require dedicated knowledge of the robotic system and the available domain propositions. Yet, desired tasks may be executed by various types of systems (e.g., manipulators and drones) and should be specified without defining (discrete) propositions or actions beforehand. Moreover, these approaches cannot monitor how close the system is to satisfying or violating the specification in the presence of object detect uncertainties.

Temporal logics in robotics TLs have become increasingly popular for robotic applications (Katayama et al., 2020; Pan et al., 2021; Wells et al., 2021; Kshirsagar et al., 2019). They allow users to specify what the robot needs to do without dedicated knowledge about the robot's dynamics. Linear Temporal Logic (LTL) (Pnueli, 1977) is a modal logic over Boolean propositions, where the modality is interpreted as linearly progressing time. Originally developed to describe and analyze the behaviour of software programs, it is now widely used in robotics (Kress-Gazit et al., 2009, 2018; Li et al., 2021; Plaku & Karaman, 2016). LTLMoP (Finucane et al., 2010) is a toolkit to design, test and implement hybrid controllers generated from LTL specifications.

LTL of finite traces (LTL_f) (De Giacomo and Vardi , 2013) and co-safe LTL (scLTL) (Kupferman & Vardi, 2001) are an efficient fragment of LTL designed to analyze and verify finite properties. Since robotic tasks usually have a defined ending criterion, LTL_f and scLTL are well suited as a specification language in task planning (Wells et al., 2020; He et al., 2019; He et al., 2018; He et al., 2015; Vasilopoulos et al., 2021; Lacerda et al., 2014; Schillinger et al., 2018). All classical LTL definitions or fragments thereof rely on discrete time and Boolean predicates, but some define quantitative predicates (Li et al., 2017) or robustness metrics (Vasile et al., 2017; Tumova et al., 2013). These quantitative evaluations provide a continuous real value that measures the extent to which a plan satisfies or violates a specification. Many approaches require users to manually define predicates and quantitative evaluations while we use spatial relations in a general and compositional manner.

Metric Temporal Logic (MTL) (Koymans, 1990) enables expressing quantitative time properties by introducing bounded temporal operators. Metric Interval Temporal Logic (MITL) (Alur et al., 1996) is a fragment of MTL that bounds all temporal operators through intervals of time. In contrast to MTL, MITL is decidable, making it more suitable for planning (Alur et al., 1996). Instead of Boolean propositions as used in MITL and MTL, SpaTiaL reasons over real-valued propositions and naturally provides quantitative semantics (see Def. 16).

Signal Temporal Logic (STL) (Donzé et al., 2010) is defined over real-valued signals and continuous time, where all temporal operators are bounded by an interval. STL is well suited to analyze real-valued signals and has been applied in control (Lindemann & Dimarogonas, 2018) and motion planning (Raman et al., 2014). STL specifications can often directly be used within optimization problems by exploiting its quantitative semantics (Barbosa et al., 2019) or to define satisfying regions in the state space (Lindemann and Dimarogonas , 2017).

To summarize, TLs are well suited to cover temporal patterns in various robotic tasks. Yet, we notice that some logics (like LTL) are more user-friendly due to the use of discrete high-level predicates, whereas continuous logics (like STL) naturally provide evaluations of continuous signals and quantitative semantics. SpaTiaL unifies these advantages to synthesize high-level plans that can be executed through any user-defined low-level controller.

Spatial and temporal requirements Many robotic tasks involve a complex interplay of spatial and temporal patterns (Menghi et al., 2019). Already allegedly simple box stacking tasks require robots to consider both precise placement of boxes relative to each other, as well as temporal

constraints, all while incorporating object detection uncertainties. In (Menghi et al., 2019), the authors summarize common temporal patterns of robotic tasks. These patterns include surveillance, conditions, triggers or avoidance patterns among others. Specifically, manipulation tasks include temporal sequences to assemble parts, recurring triggers (such as continuously moving certain parts), and time-bounded actions (Correll et al., 2016).

Various approaches incorporate spatial relations between objects into robotic frameworks (Ramirez-Amaro et al., 2013; Ramirez-Amaro et al., 2017; Diehl et al., 2021; Yuan et al., 2022; Liu et al., 2022; Paxton et al., 2022), e.g., what it means when two boxes are touching. These spatial relations help robots to understand desired goal configurations of objects (Izatt and Tedrake , 2020) and to predict an action's outcome (Paus et al., 2020). To this end, many approaches perform object-centric planning (Devin et al., 2018; Sharma et al., 2020; Shridhar et al., 2022), where planning is performed in the task space of the object itself (Manuelli et al., 2019; Migimatsu & Bohg, 2020; Agostini and Lee, 2020).

Spatial relations between objects range from high-level geometric relations, such as an object being left of another (Jund et al., 2018; Guadarrama et al., 2013), to relations that describe which objects support another object within a 3D structure (Kartmann et al., 2018; Mojtahedzadeh et al., 2015; Panda et al., 2016). Learning spatial relations from data and demonstrations is difficult (Rosman & Ramamoorthy, 2011; Driess et al., 2021; Li et al., 2022) due to the variety of relations and the complexity of separating similar relations.

Specification frameworks for robotic tasks should be able to encode the richness of temporal patterns and incorporate spatial relations of various kinds. Moreover, they should measure how much the robot satisfies/violates desired spatial relations so that the robot can monitor its progress even in the presence of object detection uncertainties. To enhance usability and interpretability, these frameworks ideally define spatial relations close to natural language (Skubic et al., 2004; Nicolescu et al., 2019).

1.2 Contributions

Our goal is to create a formal language that allows users to specify, monitor, and plan various robotic tasks. To unify the advances in formal methods as well as robot motion planning, we present our spatio-temporal framework SpaTiaL that connects these two domain while maintaining the rigorousness and properties of temporal logics. More specifically, SpaTiaL:

1. is a formal object-centric, robot-agnostic specification language for tasks with spatial and temporal relations between objects in a continuous world,
2. provides a monitoring algorithm that answers how much a specification is satisfied/violated in real-time, and
3. facilitates online planning through determining sequences of high-level actions which are executed by user-define controllers and are monitored online.

We demonstrate the advantages of SpaTiaL on various tasks with real data, simulations, and on robots. SpaTiaL is openly available as a Python package and extensible for various applications.

1.3 Structure of the article

This article is structured as follows: in Sect. 2, we introduce the necessary mathematical notations of sets and objects in a scenario. Subsequently, we define spatial relations between objects, define the syntax of SpaTiaL and derive the computation of SpaTiaL's quantitative semantics in Sect. 3 and 4. In Sect. 5, we illustrate how SpaTiaL can be used to monitor and plan robotic tasks using an example scenario. In Sect. 6, we demonstrate SpaTiaL in various experiments ranging from monitoring pushing and pick-and-place tasks, monitoring social distances in drone surveillance videos, to planning pushing and pick-and-place tasks. We discuss the results and limitations of our approach in Sect. 7 and finish with conclusions in Sect. 8.

2 Preliminaries

We consider an environment which is modeled as a subset of the Euclidean space $\mathbb{R}^d, d \in \{2, 3\}$. This environment is occupied by $N \in \mathbb{N}_+$ objects, e.g., robots, cups, or cubes. Time discretizations are common in environmental models of various robotic systems due to the sampling rate of used sensors (e.g., the frame rate of a camera mounted to an end-effector). A perception module (e.g., a camera or laser scanner) provides the footprint $\mathcal{O}_i^t \subset \mathbb{R}^d$ and orientation vector $u_i \in \mathbb{R}^d$, $||u_i||_2 = 1$, of each object $i \in \{1, \dots, N\}$ in the environment at discrete points in time $t \in \mathbb{N}_{\geq 0}$, where Δt is the time step between two consecutive time steps $t_{i+1} - t_i$. We denote the time step $t_0 = 0$ as the initial time. We may also consider uncertainties in the object representation, e.g., by providing a larger footprint for object i or enlarging its footprint using the Minkowski sum and an error term E, i.e., $\mathcal{O}_{\text{enl},i}^t = \mathcal{O}_i \oplus E := \{p + e \mid p \in \mathcal{O}_i^t \wedge e \in E\}$. Furthermore, we consider that footprints $\mathcal{O}_i^t$ are compact and convex sets:

Definition 1 [Compact and Convex Set] A set $\mathcal{O} \subseteq \mathbb{R}^d$ is compact if it is bounded and closed. Moreover, this set $\mathcal{O}$ is convex if $\forall p_1, p_2 \in \mathcal{O} : \forall \alpha \in [0, 1] : (1-\alpha)p_1 + \alpha p_2 \in \mathcal{O}$.

The state $s_t = \left((\mathcal{O}_1^t, u_1^t), (\mathcal{O}_2^t, u_2^t), \dots\right)$ contains the information of all objects in the scenario at time step t. For brevity,

we omit the time step from an object $(\mathcal{O}_i^t, u_i^t)$ when the particular time step can be inferred from the context.

In certain tasks, objects may not be convex, e.g., a banana or a cup with a handle. We can either compute the convex hull of these objects, i.e., hull$(\mathcal{O})$, or decompose them into a set of convex objects (Deng et al. 2020). For instance, we may represent the inside and outside of a cup through two cylinders and the handle with a set of three cuboids. One can also specify phantom objects in SpaTiaL. Phantom objects are virtual objects that do not correspond to a physical object in the world and are used to specify goal regions, stay-in or stay-out regions for physical objects. With the introduced spatial relations, users can specify forbidden rectangular region through phantom objects and constrain objects to never enter this forbidden region.

The intersection between two sets $\mathcal{X}_1$ and $\mathcal{X}_2$ is denoted as $\mathcal{X}_1 \cap \mathcal{X}_2$ and the Cartesian product as $\mathcal{X}_1 \times \mathcal{X}_2$. The power set is written as $2^{\mathcal{X}}$. Our definitions of spatial and temporal relations follow the Backus-Naur form. The operators $\wedge$, $\vee$ and $\neg$ denote the Boolean and, or, and negation operations, respectively.

3 Spatial relations between objects

Many tasks require robots to arrange objects in desired configurations relative to each other. For instance, an object $\mathcal{O}_i$ needs to be placed close to and left of object $\mathcal{O}_j$. Our goal is to define spatial relations in continuous space to eliminate discretization effects (Wells et al., 2019; Dantam et al., 2016) and to provide quantitative semantics, i.e., a continuous measure about the extent that a scenario satisfies a spatial relation. In the following four subsections, we use signed distances, projections and cosine similarity to derive spatial relations between objects and introduce quantitative semantics to measure the satisfaction of a specification given a state s_t of the scenario (more details on monitoring can be found in Sect. 5).

3.1 Distance-based relations

To derive distance-based spatial relations, such as an $\mathcal{O}_i$ is close to $\mathcal{O}_j$, we rely on signed distance computations between convex objects that are used in computational geometry for robotic applications (Boyd et al., 2004; Oleynikova et al., 2016; Driess et al., 2022). These approaches define constraints between geometric shapes through distances between them, e.g., an object cannot be closer to another object by a predefined margin (Zucker et al., 2013). We derive spatial relations by exploiting the signed distance between objects. The signed distance is computed using the distance and penetration depth. The distance between two objects is defined as:

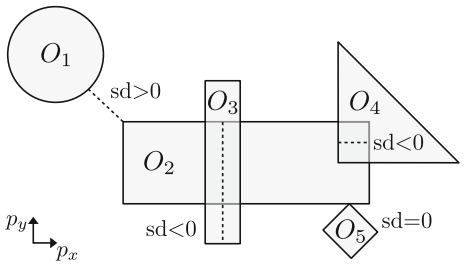


Fig. 2 Illustration of signed distances sd (dashed lines) between various objects: $sd > 0$ when objects do not intersect (e.g., sd$(\mathcal{O}_1, \mathcal{O}_2)$), $sd = 0$ when objects touch (e.g., sd$(\mathcal{O}_2, \mathcal{O}_5)$), and $sd < 0$ when the interior of objects intersects (e.g., sd$(\mathcal{O}_2, \mathcal{O}_3)$)

Definition 2 [Distance] The distance $\mathrm{d}(\mathcal{O}_i, \mathcal{O}_j)$ between two objects, $\mathcal{O}_i$ and $\mathcal{O}_j$, is the norm of the smallest translation so that both objects intersect:

$$\mathrm{d}(\mathcal{O}_i, \mathcal{O}_j) := \inf \left\{ \|T \in \mathbb{R}^d\| \mid (T \oplus \mathcal{O}_i) \cap \mathcal{O}_j \neq \emptyset \right\}.$$

Intuitively, the distance $\mathrm{d}(\mathcal{O}_i, \mathcal{O}_j)$ is larger than zero when $\mathcal{O}_i$ and $\mathcal{O}_j$ do not intersect and zero otherwise. In the case of $\mathcal{O}_i \cap \mathcal{O}_j \neq \emptyset$, we are interested in computing the distance required to get both objects out of contact, denoted as the penetration depth:

Definition 3 [Penetration Depth] The penetration depth $\mathrm{pd}(\mathcal{O}_i, \mathcal{O}_j)$ between two objects, $\mathcal{O}_i$ and $\mathcal{O}_j$, is the norm of the smallest translation so that they do not intersect:

$$\mathrm{pd}(\mathcal{O}_i, \mathcal{O}_j) := \inf \left\{ \|T \in \mathbb{R}^d\| \mid (T \oplus \mathcal{O}_i) \cap \mathcal{O}_j = \emptyset \right\}.$$

The distance and penetration depth can be efficiently computed with the Gilbert-Johnson-Keerthi algorithm (GJK) (Ericson, 2004, Sect. 9.5) in linear or logarithmic time complexity with respect to the number of vertices. GJK is a popular choice in real-time collision checking and uses Minkowski differences between convex polygons to compute distances in the polygon's configuration space. Def. 2 and Def. 3 are complementary to each other and can never simultaneously be non-zero. Finally, we define the signed distance as (Schulman et al., 2014):

Definition 4 [Signed Distance] The signed distance $\mathrm{sd}(\mathcal{O}_i, \mathcal{O}_j)$ between two objects, $\mathcal{O}_i$ and $\mathcal{O}_j$, is given by:

$$\mathrm{sd}(\mathcal{O}_i, \mathcal{O}_j) := \mathrm{d}(\mathcal{O}_i, \mathcal{O}_j) - \mathrm{pd}(\mathcal{O}_i, \mathcal{O}_j).$$

The signed distance between two objects is positive if they do not intersect, zero when they are touching, and negative when their interiors intersect. Figure 2 illustrates these three cases. The signed distance allow us to define spatial relations for describing if two objects are in a certain distance or overlap:

Proposition 1 *[Distance and Overlap Relations] Two objects $\mathcal{O}_i$ and $\mathcal{O}_j$ are in ϵ-range (*closeTo$_\epsilon$*) of each other if:*

$$\mathcal{O}_i \text{ closeTo}_\epsilon \mathcal{O}_j := \text{sd}(\mathcal{O}_i, \mathcal{O}_j) \leq \epsilon,$$

and overlapping (ovlp *) if:*

$$\mathcal{O}_i \text{ ovlp } \mathcal{O}_j := \text{sd}(\mathcal{O}_i, \mathcal{O}_j) \leq 0.$$

Proof The proof follows directly from the properties of the signed distance between two objects. □

By exploiting the convexity of the object shapes, we can also determine whether an object is enclosed in another object:

Proposition 2 *[Enclosed Within Relation] The shape of an object $\mathcal{O}_i$ is fully enclosed in (*enclIn*) the shape of object $\mathcal{O}_j$ if:*

$$\mathcal{O}_i \text{ enclIn } \mathcal{O}_j := \forall p \in \text{hull}(\mathcal{O}_i) : \text{sd}(p, \mathcal{O}_j) \leq 0$$

Proof When $\forall p \in \text{hull}(\mathcal{O}_i) : \text{sd}(p, \mathcal{O}_j) \leq 0$, then all points of the hull of $\mathcal{O}_i$ are enclosed in $\mathcal{O}_j$. Since our objects are convex, it follows that every line segment of $\mathcal{O}_i$ is also enclosed, i.e., $\forall p_1, p_2 \in \text{hull}(\mathcal{O}_i), \forall : \alpha \in [0, 1] : (1-\alpha)p_1 + \alpha p_2 \in \mathcal{O}_j$. □

Figure 3 illustrates the proof of Prop. 2 for two objects $\mathcal{O}_i$ and $\mathcal{O}_j$.

Definition 5 [Derived Distance-based Spatial Relations] The spatial relations *far from* (farFrom), *touch* (touch , with numeric parameter), *partial overlap* (partOvlp), and *closer to than* (closerTo) are derived from the distance-based relations:

$$\begin{aligned}
\mathcal{O}_i \text{ farFrom}_\epsilon \mathcal{O}_j &:= \neg \mathcal{O}_i \text{ closeTo}_\epsilon \mathcal{O}_j \\
\mathcal{O}_i \text{ touch } \mathcal{O}_j &:= \mathcal{O}_i \text{ closeTo}_\epsilon \mathcal{O}_j \wedge \\
&\qquad \mathcal{O}_i \text{ farFrom}_{-\epsilon} \mathcal{O}_j, \\
\mathcal{O}_i \text{ partOvlp } &:= \mathcal{O}_i \text{ ovlp } \mathcal{O}_j \wedge (\neg \mathcal{O}_i \text{ enclIn } \mathcal{O}_j) \\
\mathcal{O}_i \text{ closerTo } \mathcal{O}_j \text{ than } \mathcal{O}_k &:= \text{sd}(\mathcal{O}_i, \mathcal{O}_j) \leq \text{sd}(\mathcal{O}_j, \mathcal{O}_k)
\end{aligned}$$

3.2 Projection-based relations

The distance-based spatial relations already allow us to specify a variety of object configurations and to derive additional relations, such as an object being far away or in a certain range of distance values to another object. To place objects more precisely, we are further interested in specifying directions, e.g., an object is left of another one. We can derive such relations by projecting the objects onto a common coordinate system axis and determine their order.

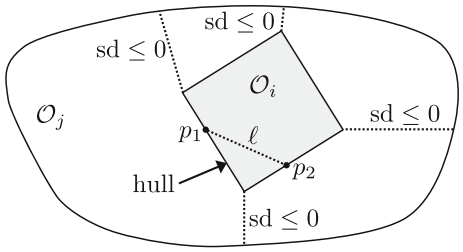


Fig. 3 The objects $\mathcal{O}_i$ and $\mathcal{O}_j$ are convex, i.e., every line segment ℓ between any two points on the hull is also enclosed. Therefore, $\mathcal{O}_i$ is enclosed in $\mathcal{O}_j$ if the signed distance of every point on the hull is negative or zero

Definition 6 [Projection] The projection operator $\text{proj}_{\text{ax}}(\mathcal{O})$ projects the shape of an object $\mathcal{O}$ onto the axis ax.

Since our objects are compact and convex sets, the projection operator returns a closed interval. We use interval orders to determine the left-to-right precedence relation between two given intervals (Allen, 1983):

Definition 7 [Interval Precedence] An interval $\mathcal{I}_1 \subset \mathbb{R}$ partially precedes another interval $\mathcal{I}_2 \subset \mathbb{R}$ if $\min(\mathcal{I}_1) \leq \min(\mathcal{I}_2)$, and fully precedes it if $\max(\mathcal{I}_1) \leq \min(\mathcal{I}_2)$.

Thus, when we project two objects onto a given axis, we can determine whether $\mathcal{O}_i$ is partially or fully left of $\mathcal{O}_j$ with respect to axis ax:

Definition 8 [Precedence Relations] An object $\mathcal{O}_i$ partially precedes (partPrec) an object $\mathcal{O}_j$ with respect to axis ax if

$$\mathcal{O}_i \text{ partPrec}_{\text{ax}} \mathcal{O}_j := \min\left(\text{proj}_{\text{ax}}(\mathcal{O}_i)\right) - \min\left(\text{proj}_{\text{ax}}(\mathcal{O}_j)\right) \leq 0,$$

and fully precedes (prec) $\mathcal{O}_j$ if

$$\mathcal{O}_i \text{ prec}_{\text{ax}} \mathcal{O}_j := \max\left(\text{proj}_{\text{ax}}(\mathcal{O}_i)\right) - \min\left(\text{proj}_{\text{ax}}(\mathcal{O}_j)\right) \leq 0.$$

With the precedence relations, users can derive spatial relations for a given application by choosing different axes, e.g., determining if an object is above another one when projecting it to the height-axis. Figure 4 illustrates how we can use the projection and interval order to determine whether an object is left of another object with respect to a given axis.

Finally, we derive additional projection-based spatial relations. Users can define various coordinate frames to express the projection-based spatial relations, given as a parameter. Without loss of generality, we consider a 2D environment with a single reference frame with axes p_x and p_y in the following paragraphs.

Definition 9 [Derived Projection-based Spatial Relations] The spatial relations *left of* (leftOf), *right of* (rightOf), *below of* (below), *above of* (above), and *between* (between) are derived from the projection relations:

$$\mathcal{O}_i \text{ leftOf } \mathcal{O}_j := \mathcal{O}_i \text{ prec}_{p_x} \mathcal{O}_j$$

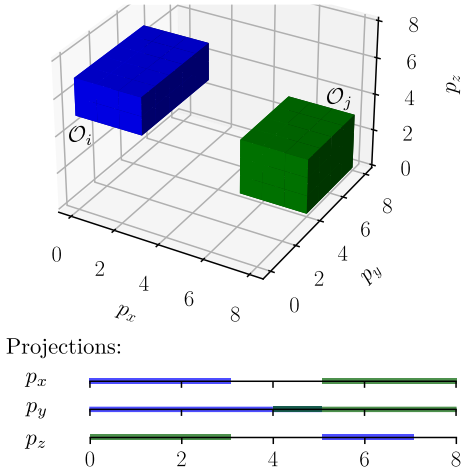


Fig. 4 We can determine the order relation of two objects $\mathcal{O}_i$ and $\mathcal{O}_j$ by projecting them onto an axis ax $\in \{p_x, p_y, p_z\}$ and determining the order of the intervals. Here, $\mathcal{O}_i$ is left of $\mathcal{O}_j$ for the axes p_x and p_y and $\mathcal{O}_j$ left of $\mathcal{O}_i$ for axis p_z

$$\mathcal{O}_i \text{ rightOf } \mathcal{O}_j := \neg\mathcal{O}_i \text{ prec}_{p_x} \mathcal{O}_j$$
$$\mathcal{O}_i \text{ below } \mathcal{O}_j := \mathcal{O}_i \text{ prec}_{p_y} \mathcal{O}_j$$
$$\mathcal{O}_i \text{ above } \mathcal{O}_j := \neg\mathcal{O}_i \text{ prec}_{p_y} \mathcal{O}_j$$
$$\mathcal{O}_i \text{ between}_{\text{ax}} \mathcal{O}_j \text{ and } \mathcal{O}_k := \mathcal{O}_j \text{ prec}_{\text{ax}} \mathcal{O}_i \wedge \mathcal{O}_i \text{ prec}_{\text{ax}} \mathcal{O}_k$$

Note that the partial versions of the relations are obtained by using the partial precedence relation. These versions allow one to check, e.g., whether a part of an object is left of another object.

3.3 Angle-based relations

In some applications, we want to orient objects in a desired way, e.g., a knife needs to point upwards. This requires us to compare the orientation vector u_i of an object $\mathcal{O}_i$ with the desired orientation vector u_d. In data analysis, the cosine similarity measures the relative angle between two vectors. For computational efficiency and to enforce a positive measure, we use the Euclidean distance approximation of the cosine distance:

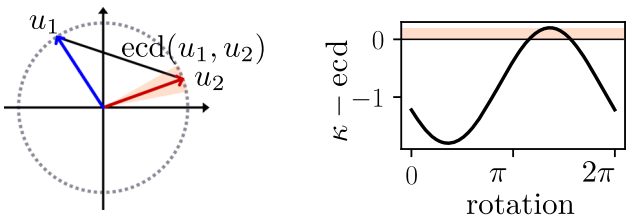


Fig. 5 The Euclidean cosine distance ecd provides a measure for the angle between two unit vectors, u_1 and u_2. The shaded area corresponds to being $\kappa-$close to the desired angle of u_2. The right figure shows the orientation relation when rotating u_1

Definition 10 [Euclidean Cosine Distance] The Euclidean approximation ecd of the cosine distance between two vectors u_1 and u_2 is given by

$$\text{ecd}(u_1, u_2) = (||u_1 - u_2||_2^2)/2,$$

where $||u_1||_2 = ||u_2||_2 = 1$.

This definition allows us to define a relation that describes whether two objects are aligned in the same orientation:

Definition 11 [Orientation Relation] An object $\mathcal{O}_i$ with angle vector u_i is oriented as object $\mathcal{O}_j$ with angle vector u_j if

$$\mathcal{O}_i \text{ oriented } \mathcal{O}_j := \text{ecd}(u_i, u_j) \leq \kappa,$$

where κ is a user-defined parameter to allow small numerical deviation from the desired orientation.

Figure 5 illustrates the ecd between two angle vectors u_1 and u_2 and how we use the parameter κ to account for numerical deviations.

3.4 Spatial compositions and quantitative semantics

Users can create complex spatial specifications by composing our spatial relations with Boolean operators:

Definition 12 [Spatial Relation Grammar] Spatial relation specifications are recursively defined using the following grammar:

$$\mathcal{R} := \top | \mathcal{O}_1 \text{rel}_2 \mathcal{O}_2 | \mathcal{O}_1 \text{rel}_3 \mathcal{O}_2, \mathcal{O}_3 | \neg\mathcal{R} | \mathcal{R}_1 \wedge \mathcal{R}_2 | \mathcal{R}_1 \vee \mathcal{R}_2,$$

where rel_2 and rel_3 are spatial relations between two or three objects, respectively. The vertical bar | represents a choice between expressions and $\top$ represents unconditional truth. Through the recursive definition, spatial relations can be nested, negated or combined through logical conjunction and disjunction in arbitrary fashion.

Similar to the quantitative semantics of STL (Donzé et al., 2010), we define quantitative semantics for our spatial relations by converting their constraint formulation (i.e., a form of $g \leq c$, where g is a function and c a constant) into a satisfaction formulation. We consider the state s_t (see Sect. 2) of all objects at time t to compute the satisfaction.

Definition 13 [Quantitative Semantics for Spatial Relations] We define the quantitative semantics $\rho(s_t, \mathcal{R})$ of a spatial relation specification $\mathcal{R}$ as:

$$\rho(s_t, \top) = \infty$$
$$\rho(s_t, \neg\mathcal{R}) = -\rho(s_t, \mathcal{R})$$
$$\rho(s_t, \mathcal{R}_1 \wedge \mathcal{R}_2) = \min(\rho(s_t, \mathcal{R}_1), \rho(s_t, \mathcal{R}_2))$$

$$\rho(s_t, \mathcal{R}_1 \vee \mathcal{R}_2) = \max(\rho(s_t, \mathcal{R}_1), \rho(s_t, \mathcal{R}_2))$$

We define ρ for single spatial relations:

$$\begin{aligned}
\rho(s_t, \mathcal{O}_i \text{ closeTo}_\epsilon\ \mathcal{O}_j) &= \epsilon - \text{sd}(\mathcal{O}_i^t, \mathcal{O}_j^t) \\
\rho(s_t, \mathcal{O}_i \text{ ovlp}\ \mathcal{O}_j) &= -\text{sd}(\mathcal{O}_i, \mathcal{O}_j) \\
\rho(s_t, \mathcal{O}_i \text{ closerTo}\ \mathcal{O}_j \text{ than } \mathcal{O}_k) &= \\
&\quad \text{sd}(\mathcal{O}_j, \mathcal{O}_k) - \text{sd}(\mathcal{O}_i, \mathcal{O}_j) \\
\rho(s_t, \mathcal{O}_i \text{ enclIn}\ \mathcal{O}_j) &= -\max_{p \in \text{hull}(\mathcal{O}_i)} \text{sd}(p, \mathcal{O}_j) \\
\rho(s_t, \mathcal{O}_i \text{ partPrec}_{\text{ax}}\ \mathcal{O}_j) &= \min\left(\text{proj}_{\text{ax}}(\mathcal{O}_j)\right) \\
&\quad - \min\left(\text{proj}_{\text{ax}}(\mathcal{O}_i)\right) \\
\rho(s_t, \mathcal{O}_i \text{ prec}_{\text{ax}}\ \mathcal{O}_j) &= \min\left(\text{proj}_{\text{ax}}(\mathcal{O}_j)\right) \\
&\quad - \max\left(\text{proj}_{\text{ax}}(\mathcal{O}_i)\right)
\end{aligned}$$

The quantitative semantics for the other spatial relations are computed analogously by recursively applying our introduced semantics.

We can use our geometry-based spatial relations also between objects over time, e.g., to describe the motion of an object. As an example, we consider the composition of a new unary spatial relation that describes if an object is moving side-wards over time.

Definition 14 [Time-Relative Relations] We define an object $\mathcal{O}_i^t$ moving to the right side as

$$\mathcal{O}_i^t \text{ movingRight} \Leftrightarrow \mathcal{O}_i^{t-1} \text{ leftOf } \mathcal{O}_i^t.$$

This time-relative property can be defined for all spatial relations and allows users to compose highly complex relations. Since we operate on geometric objects, we can also create phantom objects, such as goal or forbidden regions, to specify desired goal configurations or constraints.

4 Temporal behaviour of spatial relations

With the temporal operators from logics such as LTL or MITL, we further allow users to specify the interplay of spatial relations in time.

Table 1 Temporal operators in SpaTiaL

Symbol	Operator
$\mathcal{X}\phi$	ϕ holds next time step
$\phi_1 \mathcal{U} \phi_2$	ϕ_1 holds until ϕ_2
$\mathcal{F}\phi$	ϕ holds eventually
$\mathcal{G}\phi_1$	ϕ always holds

Definition 15 [Syntax of Temporal Relations] Temporal behaviour of spatial relations is recursively defined using the following syntax:

$$\phi := \top \mid \mathcal{R} \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \mathcal{X}\phi \mid \phi_1 \mathcal{U} \phi_2 \mid \phi_1 \mathcal{U}_{[a,b]} \phi_2,$$

where $0 \leq a < b$.

Similar to Def. 12, the vertical bar | represents a choice between expressions and $\top$ represents unconditional truth. Through the recursive definition, SpaTiaL formulas can be nested, negated or combined through logical conjunction and disjunction in arbitrary fashion. The temporal operators are summarized in Table 1. The *next*-operator $\mathcal{X}\phi$ requires a spatial relation to hold in the next time step. $\phi_1 \mathcal{U} \phi_2$ requires ϕ_1 to hold *until* ϕ_2 holds. Using the operators $\neg$ and $\wedge$, we obtain the full power of propositional logic. The operators $\vee, \rightarrow, \leftrightarrow$ are defined as usual (Baier and Katoen, 2008). Furthermore, $\mathcal{F}\phi = \top \mathcal{U} \phi$ represents ϕ holding true *eventually* in the future and $\mathcal{G}\phi = \neg\mathcal{F}\neg\phi$ *always* holding from now on. Bounded temporal operators additionally use a time interval $[a, b]$ to define a time window in which formulas have to hold. Similar to their unbounded variants, $\mathcal{F}_{[a,b]}\phi = \top \mathcal{U}_{[a,b]} \phi$ and $\mathcal{G}_{[a,b]}\phi = \neg\mathcal{F}_{[a,b]}\neg\phi$.

We interpret a temporal formula ϕ over finite sequences of states. We denote a sequence of states from time t to $t + k$ as $s_{t:t+k} = s_t s_{t+1} \ldots s_{t+k}$.

Definition 16 [Quantitative Semantics of Temporal Relations] We define $\rho(s_{t:t+k}, \phi)$ as:

$$\begin{aligned}
\rho(s_{t:t+k}, \top) &= \infty \\
\rho(s_{t:t+k}, \mathcal{R}) &= \rho(s_t, \mathcal{R}) \\
\rho(s_{t:t+k}, \neg\phi) &= -\rho(s_{t:t+k}, \phi) \\
\rho(s_{t:t+k}, \phi \wedge \psi) &= \min(\rho(s_{t:t+k}, \phi), \rho(s_{t:t+k}, \psi)) \\
\rho(s_{t:t+k}, \mathcal{X}\phi) &= \rho(s_{t+1:t+k}, \phi), k > 0 \\
\rho(s_{t:t+k}, \phi_1 U \phi_2) &= \max_{t' \in [t, t+k]} (\min(\rho(s_{t':t+k}, \phi_1), \\
&\qquad \min_{t'' \in [t, t']} (\rho(s_{t'':t'}, \phi_2)))), \\
\rho(s_{t:t+k}, \phi_1 U_{[a,b]} \phi_2) &= \rho(s_{(t+a):\min(t+k, t+b)}, \phi_1 U \phi_2),
\end{aligned}$$

where $0 \leq a \leq b$ and $a \leq k$.

From this definition, we can derive additional semantics for the derived operators:

$$\begin{aligned}
\rho(s_{t:t+k}, \phi_1 \vee \phi_2) &= \max(\rho(s_{t:t+k}, \phi_1), \rho(s_{t:t+k}, \phi_2)), \\
\rho(s_{t:t+k}, \mathcal{F}\phi) &= \max_{t' \in [t, t+k]} (\rho(s_{t':t+k}, \phi)), \\
\rho(s_{t:t+k}, \mathcal{G}\phi) &= \min_{t' \in [t, t+k]} (\rho(s_{t':t+k}, \phi)).
\end{aligned}$$

A sequence of states $s_{t:t+k}$ satisfies a formula ϕ, denoted as $s_{t:t+k} \models \phi$, if $\rho(s_{t:t+k}, \phi) \geq 0$. We denote the set of all traces that satisfy ϕ as $L(\phi)$.

Remark 1 [Comparison to other TLs] SpaTiaL takes concepts and operators from established temporal logics, such as LTL_f, MITL and STL. As spatial relations are real-valued functions, our language expresses a fragment of STL with robust semantics (Donzé et al., 2013) over discrete intervals of time. The syntax and semantics of unbounded temporal relations are in line with LTL_f (De Giacomo and Vardi, 2013) and TLTL (Li et al., 2017), which express temporal properties over finite intervals of time. Compared to MITL (Koymans, 1990), our bounded temporal operators are defined over discrete time instead of continuous time. Applying timed automata-based planning approaches to SpaTiaL remains an interesting avenue for future research.

5 Monitoring and planning with SpaTiaL

The quantitative semantics of SpaTiaL allow users to monitor the satisfaction of a given specification on a sequence of states. This monitoring approach can be applied to video data with tracked objects, as we demonstrate in Sect. 6.2 and Sect. 6.3. In these experiments, we use SpaTiaL to monitor both the completion of an object manipulation task with complex spatial constraints, as well as real-time monitoring of social distancing behaviour, where we showcase temporal constraints. Beyond monitoring, SpaTiaL can also be used to derive a plan to satisfy a specification. In this section, we outline how to use SpaTiaL for online monitoring and present a heuristic planning method that is independent of the underlying robot dynamics.

5.1 Online monitoring

SpaTiaL can evaluate the satisfaction of a specification online with respect to the information about objects through its quantitative semantics. These semantics measure the extent to which a scenario satisfies or violates a specification and can be used, e.g., to obtain fine-grained information on the robot's task progress during operation. Let us demonstrate the monitoring on an example:

Example 1 [Monitoring an Object's Motion] Consider a workspace with a cube $\mathcal{O}_{\text{cube}}$ that needs to be moved to the right into a pre-defined goal region $\mathcal{O}_{\text{goal}}$. The spatial relation specification for this example is $\mathcal{R}_{\text{bp}} = (\mathcal{O}_{\text{cube}}\ \text{enclIn}\ \mathcal{O}_{\text{goal}})$.

Figure 6a illustrates the example and the motion of the object $\mathcal{O}^t_{\text{cube}}$ over time steps t. The computed satisfaction value for the specification ϕ_{bp} is shown in Fig. 6b (blue graph). While moving towards the goal region, the satisfaction value increases. Values below zero indicate a violation of the specification while values greater or equal zero indicate satisfaction. As soon as $\mathcal{O}_{\text{cube}}$ is in the goal region, the satisfaction value is positive. After leaving the goal region, the specification is violated again. In Fig. 6b, we also show the satisfaction values when replacing the enclIn relation with other spatial relations.

The computational complexity of the monitoring scales similary as the robust evaluation of STL as described in (Donzé et al., 2013). To this end, the evaluation is linearly in the number of nodes N_{tree} in the parsed syntax tree (evaluation of SpaTiaL relations) for each time step evaluation. If more objects are contained in the specification, e.g., due to the decomposition of non-convex objects, the syntax tree contains more nodes. The evaluation of relations by using the GJK algorithm scales in the worst case linearly in the number of vertices in the polygons.

When evaluating specifications over time, the complexity increases to $O(N_{\text{tree}} \cdot N_{\text{states}} \cdot D^h)$, where N_{states} denotes the number of states in the time history, h the maximum length of a path in the syntax tree, and D a positive constant denoting the number of samples needed to evaluate the bounded time operators. Thus, the evaluation of complex specifications may suffer from some degree of exponential complexity.

To address this complexity, we have implemented two strategies: (1) we use a dynamic programming approach to cache previously evaluated nodes in the syntax tree, reducing

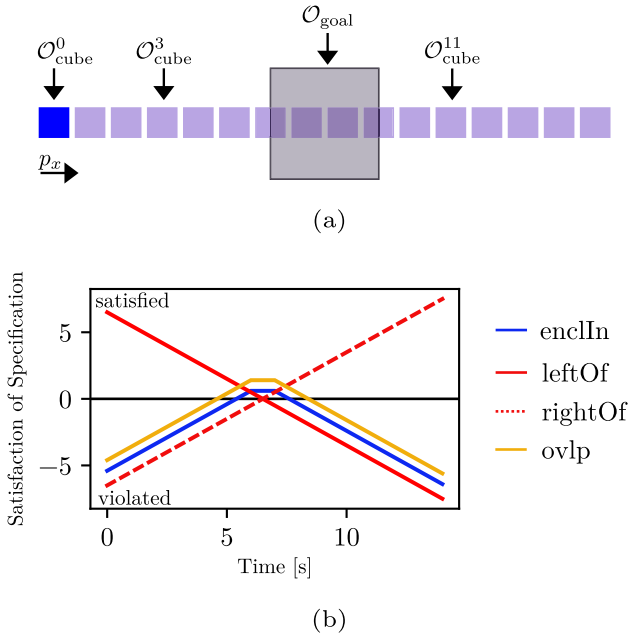


Fig. 6 The blue block $\mathcal{O}_{\text{cube}}$ needs to be pushed into the goal region $\mathcal{O}_{\text{goal}}$, encoded as $\phi_{\text{bp}} = (\mathcal{O}_{\text{cube}}\ \text{enclIn}\ \mathcal{O}_{\text{goal}})$. (a) shows the motion of $\mathcal{O}^t_{\text{cube}}$ over time. (b) shows the computed satisfaction value for ϕ_{bp} (in blue), when using a leftOf relation, rightOf, and ovlp relation (Color figure online)

the complexity to a constant lookup when looking back in time; and (2) the evaluation of the specification over a sliding time window similar to Donzé et al. (2013). This sliding window approach bounds the number of temporal evaluations and can be used in many robotic tasks, since one is not necessarily interested in all information since the start of the robotic system.

To monitor individual parts of complex specifications, we automatically parse every SpaTiaL specification into a tree structure, since SpaTiaL uses a Backus-Naur grammar to compose specifications. This is done through the Python libary *lark*. Figure 7 shows an example tree generated from the specification ϕ_{ex}. By color coding and overlaying satisfaction values, users can precisely inspect which individual parts of the specification are satisfied or violated. For instance, the negation branch of the specification tree in Fig. 7 is violated, since the object $\mathcal{O}_A$ is still positioned above $\mathcal{O}_B$. Moreover, this tree visualization provides an easier way to interpret the specification and allows users further to compose specifications by concatenating trees.

5.2 Automaton-based online planning

The object-centric nature of SpaTiaL can be used to facilitate high-level planning agnostic of the underlying robot platform. In this section, we demonstrate how SpaTiaL can be used for planning, execution and monitoring of complex tasks. For this, we present a greedy planning algorithm in Alg. 1 that monitors the execution of the plan and is able to replan if the execution of a step fails or if a step turns out to be impossible to execute. We apply automaton-based methods from temporal logic-based strategy synthesis to create a greedy planning method that aims to satisfy a syntactic subset of SpaTiaL specifications. To achieve this, we translate SpaTiaL specification into an LTL$_f$ formula by abstracting spatial relations into atomic propositions. From there, we automatically construct a *Deterministic Finite Automaton* (DFA) to represent the temporal structure. Every LTL$_f$ formula can be represented by a DFA (Baier and Katoen, 2008; De Giacomo and Vardi , 2013) by off-the-shelf tools such as MONA (Henriksen et al., 1995). We use DFAs to repeatedly plan the next high-level action, e.g., move object $\mathcal{O}_A$ left of $\mathcal{O}_B$. Through the quantitative semantics of SpaTiaL, we translate the abstract high-level action of transitioning between states in the DFA into a formulation more suited for low-level execution. The high-level actions are planned independently of the underlying robot dynamics. This makes the plan agnostic of the executing robot. In our experiments in Sect. 6.4, we use gradient maps as an example on how to complete those high-level actions. We analyze the resulting method and find it to be sound, but not optimal or complete due to separation of robot dynamics from the high-level planning.

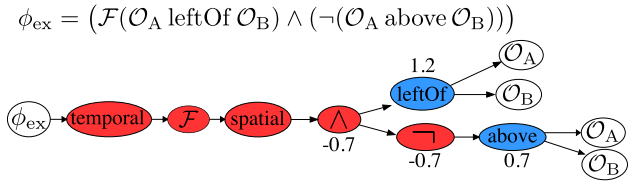


Fig. 7 Automatically generated tree from the specification ϕ_{ex} by exploiting the Backus-Naur form of SpaTiaL. Through color coding, we allow users to see which part's of a specification are violated (red) and satisfied (blue) (Color figure online)

Definition 17 [DFA] A DFA is a tuple $A = (Q, \Sigma, \delta, q_0, F)$ where Q is the set of states, Σ the alphabet, $\delta : Q \times \Sigma \to Q$ the transition function, q_0 the initial state and F the set of accepting states.

Provided a sequence of symbols from the alphabet Σ, a DFA generates a *run*.

Definition 18 [Run] Given a trace $\tau = \tau_0\tau_1 \dots \tau_n$, where $\tau_i \in \Sigma$, a finite run of a DFA is the sequence of states $q_0q_1 \dots q_{n+1}$ such that $q_{i+1} = \delta(q_i, \rho_i)$ for all $0 \leq i \leq n$. This run is accepting if $q_{i+1} \in F$.

When a run is accepting, the sequence of symbols provided to the DFA satisfies the specification (Baier and Katoen 2008). In other words, when we reach an accepting state in the DFA, we have satisfied the specification.

In order to facilitate planning, we omit the usage of bounded temporal operators and the *next*-operator $\mathcal{X}$. When omitting bounded temporal operators, every SpaTiaL formula ϕ can be translated into an LTL$_f$ formula ϕ_t by mapping every spatial relation $\mathcal{R}$ to an atomic proposition.

Example 2 Consider the SpaTiaL formula

$$\phi = \mathcal{F}(\mathcal{O}_0\ \text{closeTo}_\epsilon\ \mathcal{O}_1) \wedge \mathcal{G}\neg(\mathcal{O}_0\ \text{touch}\ \mathcal{O}_2) \tag{1}$$

with spatial relations

$$r_1 = (\mathcal{O}_0\ \text{closeTo}_\epsilon\ \mathcal{O}_1),$$
$$r_2 = (\mathcal{O}_0\ \text{touch}\ \mathcal{O}_2).$$

The temporal structure of ϕ can be expressed by

$$\phi_t = \mathcal{F}goal \wedge \mathcal{G}\neg collide \tag{2}$$

over atomic propositions $AP = \{goal, collide\}$ and a mapping function $m : AP \to \mathcal{R}, m(goal) = r_1, m(collide) = r_2$.

Additionally, through omitting the *next*-operator $\mathcal{X}$, we obtain *stutter-insensitive* temporal property.

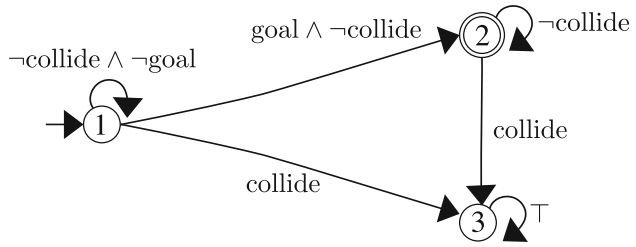


Fig. 8 DFA for Ex. 2 where state 1 and state 2 are the initial and accepting states, respectively

Lemma 1 *[Stutter-Insensitive Properties] Every LTL$_f$ formula without $\mathcal{X}$ can be translated into a DFA $A = (Q, \Sigma = 2^{AP}, \delta, q_i, F)$ where for every state $q \in Q \setminus q_i$, there exists a* self-loop *transition such that $q = \delta(q, \sigma)$ for some $\sigma \in \Sigma$ (Baier and Katoen, 2008; Kantaros et al., 2020).*

Remark 2 [Properties without temporal operators] Formulae where spatial relations occur without temporal operators, e.g. $\phi = \mathcal{O}_0$ touch $\mathcal{O}_2$ correspond to A_ϕ where q_i does not have a self-loop. An initial observation provides the truth value of such properties and enables a transition into a state with a self-loop before we plan our first step.

We translate the corresponding LTL$_f$ formula ϕ_t into a DFA A_{ϕ_t} that precisely accepts $L(\phi_t)$, e.g., using the tool MONA (Henriksen et al., 1995). The resulting automaton for (1) is depicted in Fig. 8.

Utilizing the structure of A_{ϕ_t} following from Lemma 1, we determine a sequence of actions aiming to satisfy ϕ. With A_{ϕ_t} handling the temporal structure of ϕ, we are able to satisfy ϕ by determining a path to an accepting state in the DFA. This path indicates which relations need to be satisfied sequentially. After an initial observation, every state has a self-loop. We use the conditions of the self-loop as constraints for the next execution step. This way, we avoid triggering unwanted transitions. The procedure is outlined in algorithm 1. First, we construct A_{ϕ_t} and keep track of the current state (up to line 6). Using standard methods, we find a shortest path to an accepting state inside the DFA. The path length is defined through the number of required transitions. From this path, we select a transition (q_c, q_{next}) that we wish to enable in a future step (line 10). If we are in an accepting state, i.e., $q_c \in F$, then we stay in that state, i.e., $q_{next} = q_c$ (line 8). Let

$$\Sigma_t \subseteq \Sigma = \{\sigma \in \Sigma \,|\delta(q_c, \sigma) = q_{next}\} \tag{3}$$

be the *progress set*. By satisfying any $\sigma \in \Sigma_p$, we transition into the planned state, bringing us closer to an accepting state. A symbol σ is *satisfied*, if and only if $\rho(m(x)) > 0$ for all $x \in \sigma$. We define

$$\Sigma_c \subseteq \Sigma = \{\sigma \in \Sigma \mid \delta(q_c, \sigma) \neq q_{next} \wedge \delta(q_c, \sigma) \neq q_c\} \tag{4}$$

Algorithm 1 Automaton-Based Monitoring and Planning

Data: SpaTiaL formula ϕ
1 construct DFA $A_{\phi_t} = (Q, \Sigma, \delta, q_0, F)$
2 prune infeasible transitions offline (see Rem 3)
3 $q_c = q_0$; `// set current state to initial state`
4 **while** $q_c \notin F$ **do**
5 $\quad \sigma_i = \{r \in AP \mid \rho(m(r), i) \geq 0\}$
6 $\quad q_c = \delta(q_c, \sigma_i)$; `// update current state`
7 $\quad$ find shortest path $q_1 q_2 \ldots q_n \in Q^*$ s.t. $q_n \in F, q_1 = q_c$ and $\forall i \exists \sigma : (q_i, \sigma, q_{i+1}) \in \delta$
8 $\quad q_{next} = q_2$
9 $\quad$ **if** *No path exists* **then**
10 $\quad\quad$ break
11 $\quad \Sigma_p = \{\sigma \in \Sigma \mid \delta(q_c, \sigma) = q_{next}\}$
12 $\quad \Sigma_c = \{\sigma \in \Sigma \mid \delta(q_c, \sigma) \neq q_{next} \wedge \delta(q_c, \sigma) \neq q_c\}$
13 $\quad$ execute (Σ_p, Σ_c) (see Lemma 2)
14 $\quad$ **if** *unable to execute* (Σ_p, Σ_c) **then**
15 $\quad\quad$ set $\delta(q_c, \sigma) = \varnothing, \forall \sigma \in \Sigma_p$; `// prune edge`

as the *constraint set*. By never satisfying any $\sigma \in \Sigma_c$, we guarantee staying in q_c until we satisfy any $\sigma \in \Sigma_p$. This is always possible since there always exist some σ such that $(q_c, \sigma, q_c) \in \delta$ due to Lemma 1.

Guided by the DFA, we have reduced the problem of satisfying ϕ to the problem of sequentially aiming to satisfy any $\sigma_p \in \Sigma_p$ while avoiding to satisfy any $\sigma_c \in \Sigma_c$. This planning approach is similar to the approach presented in (Kantaros et al., 2020) without the domain- and problem-specific constraints and pruning rules.

Lemma 2 *[Satisfying Spatial Relations] Given a progress set Σ_p and a constraint set Σ_c, we can plan to transition to the desired next state of the DFA by satisfying*

$$\max_{\sigma_p \in \Sigma_p} (\rho(\sigma_p)) \geq 0$$

with subject to the constraint

$$\min_{\sigma_c \in \Sigma_c} (\rho(\sigma_c)) < 0$$

where $\rho(\sigma) = \min_{r \in \sigma}(\rho(m(r)))$.

Lemma 2 aims at satisfying spatial relations that trigger a planned transition while avoiding triggering any unwanted transitions. In general, finding a solution to satisfy the equations presented in Lemma 2 depends on the dynamics of the executing robot. However, by abstracting between high-level planning and low-level control, we allow users to apply their own methods to execute the planned next action.

Search-based planning using gradient maps We demonstrate a solution that is straightforward to implement for the case where satisfying the equations in Lemma 2 is possible by moving a single object only. By choosing an object $\mathcal{O}$ and virtually moving it on a grid of the workspace, we compute

a *gradient map* by evaluating spatial subformulae in Σ_p and Σ_c for all positions of $\mathcal{O}$.

Definition 19 [Gradient Maps] Given Σ_p and Σ_c, an object $\mathcal{O}$, and a point $p \in \mathbb{R}^2$, we write $\widetilde{\mathcal{O}}$ for the the object $\mathcal{O}$ with its center displaced to p. We write $\tilde{s}_t$ for a state s_t with $\mathcal{O}$ replaced by $\widetilde{\mathcal{O}}$.

$$\mathcal{G}_{\Sigma_p,\Sigma_c,\mathcal{O}}(p) = \begin{cases} -\infty, & \text{if } \max_{\sigma_c \in \Sigma_c}(\rho(\tilde{s}_t, \sigma_c)) \geq 0 \\ \max_{\sigma_p \in \Sigma_p}(\rho(\tilde{s}_t, \sigma_p)), & \text{otherwise.} \end{cases}$$

Gradient maps are a uniform sampling approach in the task-space of a selected object to determine the gradient of satisfying a given spatial relation with respect to the object's position. Varying rotations can be integrated by computing gradient maps for different rotations, similar as in (Lozano-Perez, 1990).

Example 3 [Gradient Maps] Consider a rectangular workspace with three colored blocks. Each block is described by its color, its position and its orientation. Given a progress set $\Sigma_p \subseteq \Sigma$ induced by $\phi_p = (green \text{ rightOf } red) \wedge (\text{d}(green, red) \leq 0.05)$ and a constraint set characterized by $\phi_c = \bigwedge_{i \neq j}(\text{d}(i, j) < 0.01), i, j \in \{green, red, blue\}$, we depict gradient maps for the green block and for the red block in Fig. 9. The white cut-out regions around the blocks are the result of the constraint set and represent the area where ϕ_c would be satisfied if we moved the currently considered object into that region. Since ϕ_p can be satisfied both by moving the green block right of the red block or by moving the red block left of the green block, both gradient maps have positive values. The gradient map for the blue block (not depicted) does not have any positive values.

If for a given Σ_p, Σ_c no gradient map has any positive values for any considered object, the corresponding transition is deemed infeasible and pruned. The steps of algorithm 1 and the construction of gradient maps can be repeated until a goal state is reached. Depending on the application, the gradient maps can be used to find either a continuous path or a new position for a single object in order to transition to the desired next state of the DFA. Two examples using this procedure are demonstrated in Sect. 6.4.

Remark 3 [Pruning the DFA] Some symbols σ and therefore transitions might be infeasible, especially when physical constraints are not represented in the specification. An example for such a transition might be requiring an object to be both close and far from another object at the same time. Additional offline pruning of infeasible transitions might be done through specific domain knowledge, depending on the application. In the following experiments, we employ only online pruning of edges that turn out to be infeasible through Lemma 2 (see Alg. 1 line 15).

5.2.1 Analysis of the planning approach

In the following paragraphs, we analyze the termination, optimality, completeness, and soundness properties of our proposed planning approach.

Termination Assuming the executing robot can either complete or determine infeasibility of a planned step (Σ_p, Σ_c) in finite time, the planning algorithm Alg. 1 is guaranteed to terminate in finite time. This follows directly from the finite amount of transitions in the DFA and the iterative pruning of infeasible transitions.

Optimality The chosen path is greedily chosen with respect to the least amount of transitions in the DFA. Without additional reasoning on the dynamics of the executing robot, it is not possible to infer further information about the quality of the solution. Within the scope of our experiments, reaching an accepting state with the least amount of transitions led to solutions where the robot performs the least amount of primitive actions possible, i.e. moving the fewest elements in the scenario.

Completeness If any solution exists to satisfy a given specification ϕ, it can be expressed as a sequence of Σ_p and Σ_c (see Alg. 1 and Lemma 2). This follows directly from the observation that the DFA is without loss of generality complete, meaning it covers all combinatorial evaluations of spatial relations in the specification ϕ. Since the planning algorithm does not take the low-level dynamics of the executing robot into consideration, the algorithm cannot be complete. In situations with complex restrictions on the robot's dynamics, planning approaches that take these restrictions into account are preferable to the presented robot-agnostic approach.

Soundness If a sequence of executable actions that lead to an accepting state of A_{ϕ_t} can be found, the generated trace of the execution is satisfying ϕ_t and transitively satisfying the specification ϕ. This follows directly from known results on the translation of LTL$_f$ formulae to DFA (Baier and Katoen, 2008; De Giacomo and Vardi , 2013).

6 Experiments

To demonstrate the capabilities of SpaTiaL, we designed three different experiments with recorded data, simulations, and a robot. Each experiment focuses on different aspects of SpaTiaL (see contributions in Sect. 1.2):

1. in Sect. 6.2, we monitor object pushing and pick-and-place tasks. This experiment highlights the creation of specifications, the use of quantitative semantics, and accounting for object detection uncertainties;
2. in Sect. 6.3, we monitor social distancing of agents in the Stanford drone dataset. This experiment highlights the usage of temporal properties with time bounds;

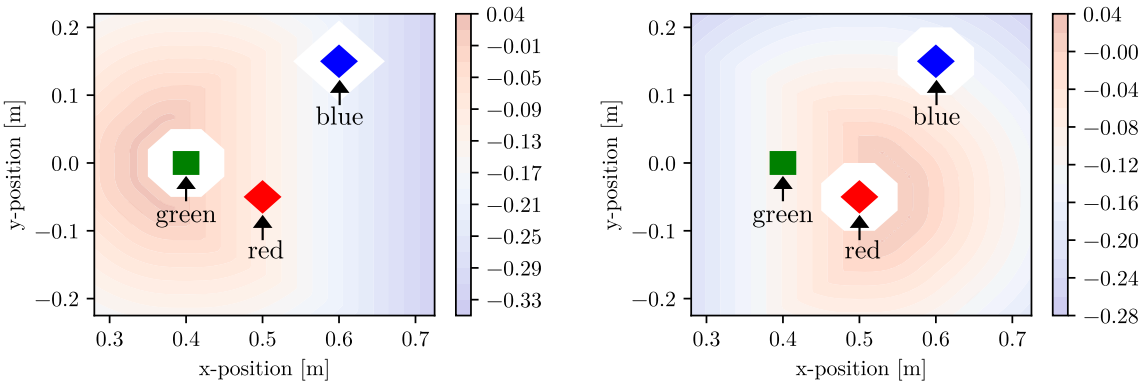


Fig. 9 Gradient Maps for the spatial relation $\phi_p = (green\ \text{rightOf}\ red) \wedge (\text{d}(green, red) \leq 0.05)$ with spatial constraint $\phi_c = \bigwedge_{i \neq j}(\text{d}(i, j) < 0.01), i, j \in \{green, red, blue\}$ from the perspective of the green cube and the red cube (Color figure online)

3. in Sect. 6.4, we use SpaTiaL for planning in robotic tasks. This experiment illustrates the high-level temporal and low-level control abstractions.

Additional experiments are in our code repository and videos of our experiments in the article's media attachment.

6.1 Implementation

We implemented a SpaTiaL parser and interpreter in Python 3, using Lark (Shinan, 2021) to parse the logic, Shapely (Gillies et al., 2007) for geometric computations, and MONA (Henriksen et al., 1995) with LTLf2DFA (Fuggitti, 2021) to construct DFAs for planning. Our software is published under the MIT license and available at https://github.com/KTH-RPL-Planiacs/SpaTiaL. We provide an API to parse SpaTiaL specifications, define objects through sets of convex polygons, and to automatically interpret the formula to obtain satisfaction values, as well as DFA planning functionalities. An API overview is available at https://kth-rpl-planiacs.github.io/SpaTiaL/. We have also released the full code for all experiments, including gradient map computation and our simulation setup in PyBullet (Coumans and Bai, 2022).

Table 2 Mean and standard deviation of the computation time per frame in our monitoring experiments over five runs

Experiment	Framerate	Mean	StdDev
Block pushing	30 fps	32.12 ms	0.2 ms
	10 fps	12.7 ms	0.44 ms
Pick-and-place	30 fps	71.15 ms	0.57 ms
	10 fps	16 ms	0.86 ms

6.2 Monitoring robotics tasks

SpaTiaL allows user to monitor the satisfaction of specifications in real time for various applications. In our first two experiments, we monitor a block pushing and a pick-and-place task. We recorded both tasks using a camera with 30fps and localized objects in the scene using AprilTags (Wang et al., 2016). The computation times per frame of both experiments can be found in Table 2 when analyzing the video with 30fps and 10fps (i.e., skipping every 2 frames). We used a machine with macOS, an Intel i5 2GHz Quadcore CPU, 32GB of DDR4 memory, and Python 3.8.

Block pushing task In this task, a human is pushing small blocks on an even surface towards a goal configuration while not violating constraints (see setup in Fig. 10). We placed 2 red, 2 green, and 2 blue blocks in the scene and specify additional phantom objects that serve as goal areas. The specification requires the human to push the red and green blocks to their corresponding goal regions *redGoal* and *greenGoal*, respectively. At the same time, we impose a task constraint: when moving red/green blocks to the other side, the human needs to push the blocks through the passage that is formed by the two blue blocks. This specification ϕ_{blocks} is encoded using the eventually and always operators as shown in Table 3.

Figure 10 illustrates selected camera frames. At $t = 7s$, the human is pushing a red block through the passage to not violate the mission constraint. However, even though the human is moving one red block towards the goal, the satisfaction value is not yet changing. This observation is due to the fact that the computed satisfaction value corresponds to the worst violation of the subparts of ϕ_{blocks}, i.e., every *and* operator in the quantitative semantics is essentially a minimum operation. In this scenario, the human did not move the red block that is further away from its goal region. Our monitoring algorithm monitors each subpart of the syntax tree parsed from the specification. Sect. 5.1 outlines how the

Table 3 Considered specifications in the experiments of Sect. 6.2 and 6.4

Specification	Definition
ϕ_{blocks}	$(\mathcal{F}(red\ \text{enclIn}\ redGoal)) \wedge (\mathcal{F}(green\ \text{enclIn}\ greenGoal)) \wedge (\mathcal{G}((red1\ \text{closeTo}\ blue1) \rightarrow (red1\ \text{below}\ blue1)))$ $\wedge(\mathcal{G}(red2\ \text{closeTo}\ blue2) \rightarrow (red2\ \text{above}\ blue2))) \wedge (\mathcal{G}((green1\ \text{closeTo}\ blue1) \rightarrow (green1\ \text{below}\ blue1)))$ $\wedge(\mathcal{G}((green2\ \text{closeTo}\ blue2) \rightarrow (green1\ \text{above}\ blue2)))$
ϕ_{pnp}	$(\mathcal{F}((mug\ \text{leftOf}\ plate) \wedge (mug\ \text{ovlp}\ plate))) \wedge ((\neg(plate\ \text{moved}\ plate[-1]))U\ ((cookies\ \text{above}\ plate)$ $\wedge(cookies\ \text{ovlp}\ plate) \wedge (orange\ \text{above}\ plate) \wedge (orange\ \text{ovlp}\ plate)) \wedge (\mathcal{F}(coffee\ \text{rightOf}\ plate) \wedge (milk\ \text{closeTo}\ coffee)$ $\wedge(milk\ \text{rightOf}\ coffee)\wedge))(\mathcal{F}(jug\ \text{leftOf}\ coffee) \wedge (jug\ \text{closeTo}\ coffee))$
$\phi_{\text{plan,pnp}}$	$\mathcal{F}(kanelbulle\ \text{enclIn}\ plate)$ $\wedge\mathcal{F}(0.1 \leq banana\ \text{d}\ plate \leq 0.3 \wedge banana\ \text{leftOf}\ plate \wedge banana\ \text{below}\ plate)$ $\wedge\mathcal{F}(0.1 \leq mug\ \text{d}\ plate \leq 0.3 \wedge mug\ \text{leftOf}\ plate \wedge mug\ \text{above}\ plate)$ $\wedge\mathcal{F}(0.1 \leq bottle\ \text{d}\ plate \leq 0.3 \wedge bottle\ \text{leftOf}\ plate \wedge bottle\ \text{above}\ plate)$ $\wedge\mathcal{F}(sugarbox\ \text{d}\ plate \geq 0.4 \wedge sugarbox\ \text{d}\ crackerbox \leq 0.2)$
ϕ_{ABB}	$(\mathcal{G}((bowl\ \text{d}\ fork \geq 0.001) \wedge (bowl\ \text{d}\ knife \geq 0.001) \wedge (bowl\ \text{d}\ mug \geq 0.001) \wedge (mug\ \text{d}\ knife \geq 0.001)$ $\wedge(mug\ \text{d}\ fork \geq 0.001) \wedge (knife\ \text{d}\ fork \geq 0.001))) \wedge (\mathcal{F}(((bowl\ \text{d}\ center \leq 0.01) \wedge (\mathcal{F}((fork\ \text{leftOf}\ center)$ $\wedge(fork\ \text{d}\ center \leq 0.10))) \wedge (\mathcal{F}((mug\ \text{above}\ center) \wedge (mug\ \text{rightOf}\ center)$ $\wedge(mug\ \text{d}\ center \leq 0.20))) \wedge (\mathcal{F}((knife\ \text{rightOf}\ center) \wedge (knife\ \text{d}\ bowl \leq 0.10))))))$

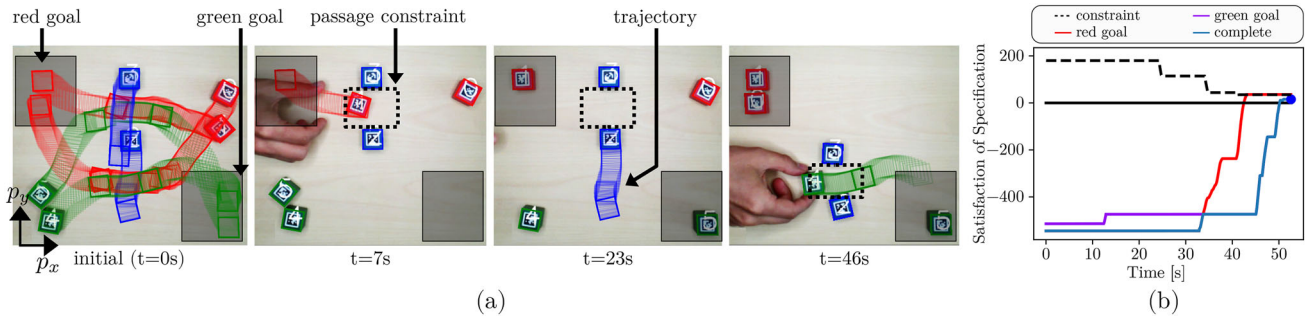


Fig. 10 Results of our experiment in which we monitor a pushing task. **a** shows the camera frames and trajectories (full trajectory for $t = 0s$ and partial ones with length $3s$ otherwise) of the cubes for selected time steps. The trajectories show the future positions of the cubes. **b** illustrates the satisfaction value of the specification and its sub-parts over the time of the recording

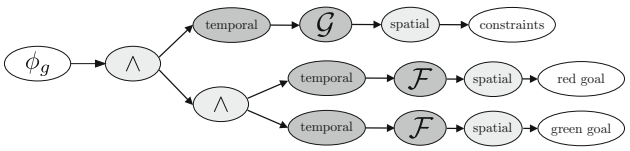


Fig. 11 Tree representation of the parsed specification ϕ_{blocks}. The recursive computation of the quantitative semantics allows us to also monitor individual subtrees

evaluated syntax tree allows one to obtain this fine-grained information. Figure 11 shows the parsed tree which represents the three individual components that specify the red goal, green goal, and the constraints. For instance, when the human moves the green block at around $12s$, the subpart of the specification that considers the green blocks is changing (see purple line in Fig. 10b).

All spatial relations are evaluated in continuous space while considering the currently available environment information. At $t = 23s$, the human is moving the blue blocks of the passage. SpaTiaL adapts to this change so that the remaining red and green block also need to be moved through the new passage. From $t = 33s$ to $t = 43s$, the last red block is successfully pushed into its goal region, as indicated by the satisfaction graph of the red goal in Fig. 10b. Finally, the specification ϕ_{blocks} is satisfied at $t = 50.1s$ with a value of 15.35.

Object pick and place task In our second monitoring experiment, we have a closer look at the temporal operator *until* U and evaluations of spatial relations over time. Therefore, we consider a pick-and-place task in which kitchen items need to be placed according to a given specification (similar to the task shown in Fig. 1). In our example, the plate needs to be moved close to the coffee mug; however, the plate is only allowed to be moved when the objects *cookies* and *orange* have been placed on the plate. This requirement is encoded in the specification ϕ_{pnp} using the until operator in Table 3. Moreover, ϕ_{pnp} specifies how the other objects *jug*, *coffee*,

mug, and *milk* need to be placed. SpaTiaL allows users to easily define new relations. In this experiment, we defined a new spatial relation *moved* that indicates if an object has been moved over time using time-relative spatial relations:

$$\mathcal{O}\,\text{moved}\,\mathcal{O}[-1] := \mathcal{O}\,\text{enclIn}\,\mathcal{O}[-1] \oplus \mathcal{C}_\epsilon, \tag{5}$$

where $\mathcal{C}_\epsilon$ is a circle with radius $\epsilon = 25$ pixel to enlarge an object to account for object detection uncertainties of the camera detection. The relation in (5) checks whether the object has moved with respect to its occupancy in the previous time step.

Figure 12a–e illustrate selected time steps of our pick-and-place experiment with partial trajectories (length: $3s$) of the objects. The computed satisfaction value is shown in Fig. 12f over the course of the experiment with a duration of $35s$. As soon as the required objects *cookies* and *orange* have been correctly placed on the plate, the human is allowed to move the plate (see images for $t = 11s$ and $t = 16s$). In contrast, we also tested a modified version of the specification in which we require that the human places the *orange* left of the *cookies* on the plate. This modified specification is violated since the human is moving the plate without reaching the required spatial relation (see orange line in Fig. 12f).

6.3 Monitoring agents in the environment

We can also use SpaTiaL to monitor agents in an environment. Bounded temporal operators allow users to specify that certain spatial relations need to hold or not hold within a given time interval. We choose a surveillance application in which a drone is recording an environment to demonstrate the bounded time operators and the nesting of temporal operators. In our case, we focus on detecting violations of social distancing using the Stanford Drone dataset (Robicquet et al., 2016). We select the bookstore data where a drone is hovering and recording the bounding boxes of pedestrians, cyclists, and other objects. With this experiment, we show that we can specify constraints on an agent's state by grounding the state on geometrical representations, e.g., bounding boxes.

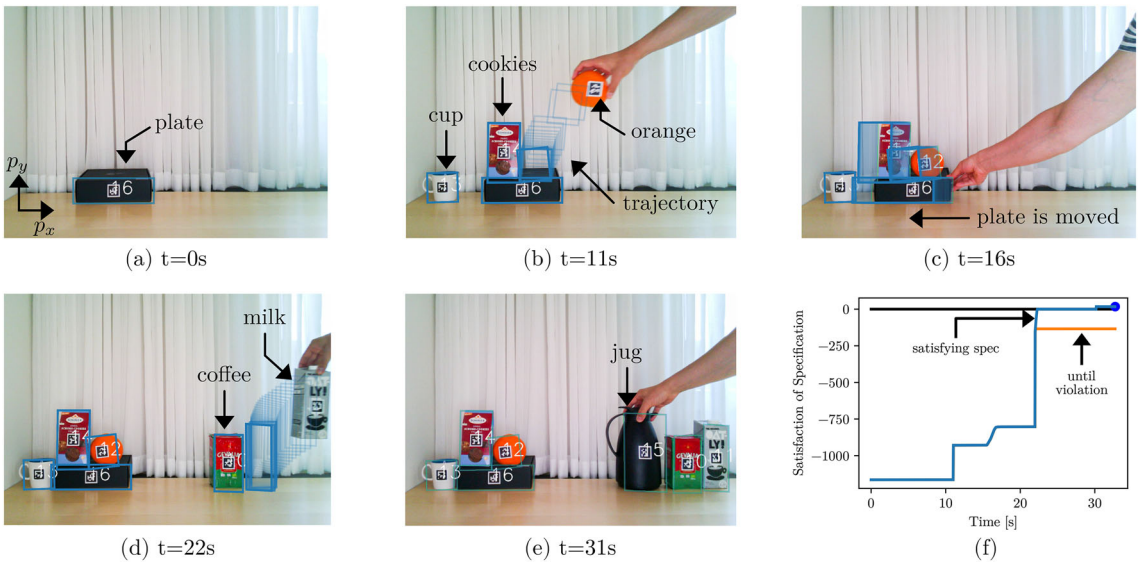
We consider four different social distancing specifications which use different temporal operators and time bounds. They are based on checking whether an object is coming too close to another object in the current scene. To identify when objects are too close, we consider two variations, close_1 and close_2:

$$\text{close}_1 := (\textit{ego}\ \text{ovlp}\ \textit{others}), \tag{6}$$

$$\text{close}_2 := (\textit{ego}\ \text{closeTo}_{\epsilon_\text{d}}\ \textit{others}), \tag{7}$$

where *ego* denotes the currently selected object for investigation, *others* corresponds to all other objects in the scene, and $\epsilon_\text{d} = 15$ is the preferred minimum distance between bounding boxes of the objects (chosen empirically using distance measurements in the pixel space of the image). Our considered social distancing specifications are:

$$(\phi_1)\ \big(\mathcal{G}\ \text{close} \rightarrow (\mathcal{G}_{[30,60]}\neg\text{close})\big),$$
$$(\phi_2)\ \big(\mathcal{G}\ \text{close} \rightarrow (\mathcal{G}_{[90,180]}\neg\text{close})\big),$$
$$(\phi_3)\ \big(\mathcal{G}\ \text{close} \rightarrow (\mathcal{F}_{[0,60]}\neg\text{close})\big),$$



(a) t=0s (b) t=11s (c) t=16s

(d) t=22s (e) t=31s (f)

Fig. 12 Results of our experiment in which we monitor a pick-and-place task. **a**–**e** show the camera frames and trajectories (partial ones with length of $3s$) of the objects in the scene for selected time steps. The trajectories show the future positions of the objects. **f** illustrates the satisfaction value of the original specification and the modified one with an *until* violation over the time of the recording

$$(\phi_4)\ \big(\mathcal{G}\ \text{close} \rightarrow (\mathcal{F}_{[0,150]}\neg\text{close})\big),$$

where close $\in \{\text{close}_1, \text{close}_2\}$. Specifications ϕ_1 and ϕ_2 encode that whenever an object is too close to another, it needs to be always distant to other objects during 1 to 2 s or 3 to 6 s after contact (since the video has been recorded with 30fps). Specifications ϕ_3 and ϕ_4 specify that whenever an object is too close to another, it needs to be eventually distant to other objects within 2 s or 5 s after contact.

Figure 13 illustrates the evaluation of the four specifications on frame number 9000. We colored the bounding boxes to encode whether the object is satisfying (blue color) or violating (red color) the specification. The temporal operator *always* or *eventually* allow us to encoded different levels of strictness of social distancing. This observation can be seen from the fact that the specifications with the *eventually* operator have a higher percentage of satisfying objects in the scene. Table 4 provides a more detailed analysis of the number of satisfying or violating objects. Finally, Fig. 14 shows the satisfaction value for the object with ID 224 over time.

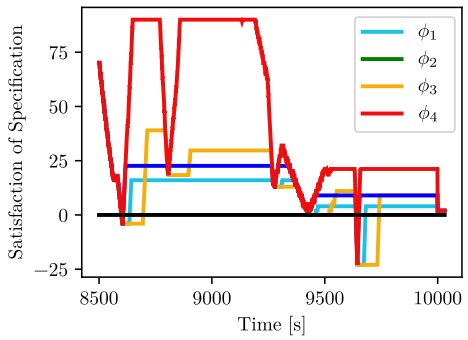


Fig. 14 Satisfaction values of the object with ID 224 with respect to the four specifications ϕ_1 to ϕ_4 and the closeness definition close_1, where two objects are close when their bounding boxes overlap

6.4 Task and motion planning

In the following three experiments, we demonstrate the online monitoring and planning technique outlined in Sect. 5.2. The first experiment features a simulated robot arm pushing blocks and illustrates the individual steps of DFA planning and the gradient maps to find continuous paths to push objects along. The second experiment features a pick-and-place specification, requiring a simulated robot to arrange a table. The third experiment demonstrates a table setting task on ABB YuMi robots.

Pushing task and motion planning We consider a robot arm mounted on a planar surface as depicted in Fig. 15. The robots is required to push three colored blocks within a rectangular

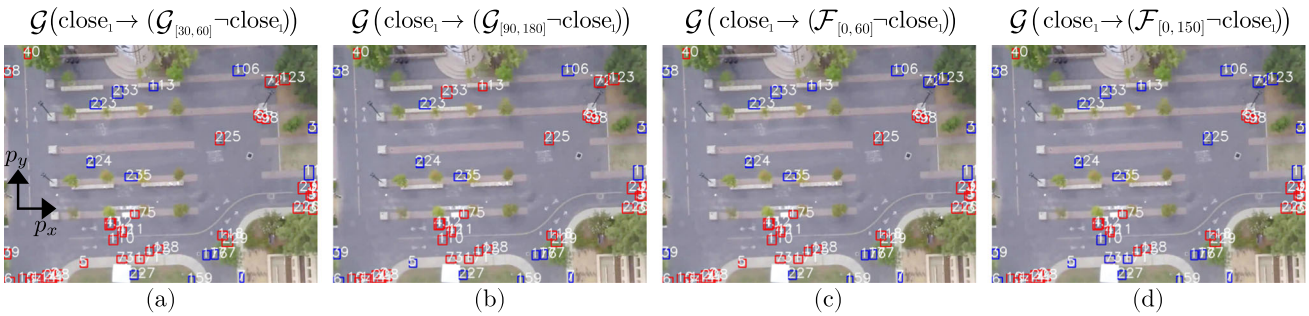


Fig. 13 Excerpts of our experiment where we monitor which agents in the environment respect social distancing to other objects. The images show the frame 9000 (5 min) of the dataset for our four specifications ϕ_1 to ϕ_4 and the too close definition $close_1$. Blue and red colored bounding boxes denote satisfying or violating the specification, respectively. The visualized IDs of the objects correspond to the IDs in the dataset

Table 4 Evaluation results of the four social distancing specifications ϕ_1 to ϕ_4 for the two too close definitions

Predicate	Specification	Satisfying Obj	Violating Obj	Worst Violation	Highest Satisfaction
ovlp	$\mathcal{G}(\text{close}_1 \rightarrow (\mathcal{G}_{[30,60]}\neg\text{close}_1))$	35 (53%)	31 (47%)	ID 40, value=−47.0	ID 113, value=296.0
	$\mathcal{G}(\text{close}_1 \rightarrow (\mathcal{G}_{[90,180]}\neg\text{close}_1))$	41 (62%)	25 (38%)	ID 40, value=−47.0	ID 113, value=293.8
	$\mathcal{G}(\text{close}_1 \rightarrow (\mathcal{F}_{[0,60]}\neg\text{close}_1))$	43 (65%)	23 (35%)	ID 40, value=−47.0	ID 113, value=298.1
	$\mathcal{G}(\text{close}_1 \rightarrow (\mathcal{F}_{[0,150]}\neg\text{close}_1))$	49 (74%)	18 (26%)	ID 40, value=−47.0	ID 113, value=298.2
closeTo	$\mathcal{G}(\text{close}_2 \rightarrow (\mathcal{G}_{[30,60]}\neg\text{close}_2))$	25 (38%)	41 (62%)	ID 40, value=−62.0	ID 113, value=281.0
	$\mathcal{G}(\text{close}_2 \rightarrow (\mathcal{G}_{[90,180]}\neg\text{close}_2))$	30 (45%)	36 (55%)	ID 40, value=−62.0	ID 113, value=278.8
	$\mathcal{G}(\text{close}_2 \rightarrow (\mathcal{F}_{[0,60]}\neg\text{close}_2))$	31 (47%)	35 (53%)	ID 40, value=−62.0	ID 113, value=283.2
	$\mathcal{G}(\text{close}_2 \rightarrow (\mathcal{F}_{[0,150]}\neg\text{close}_2))$	40 (61%)	26 (39%)	ID 40, value=−62.0	ID 113, value=283.2

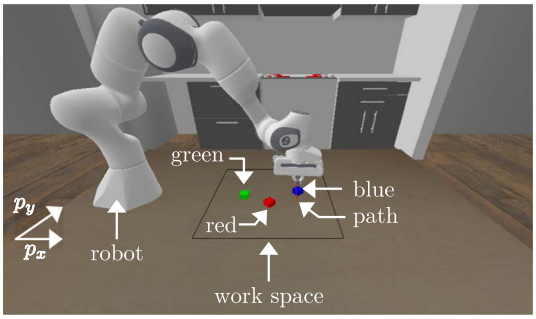


Fig. 15 Simulator environment for the pushing TAMP experiment

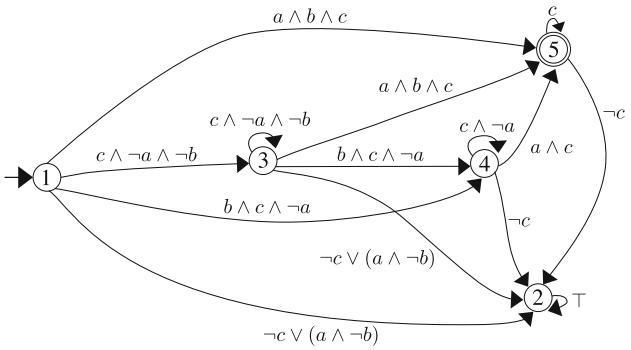


Fig. 16 The generated DFA for the pushing task. Except for the initial state 1, each state has a self-loop. The initial state not having a self-loop is an implementation detail that is circumvented by providing an initial observation

workspace to fulfill the SpaTiaL specification:

$$\begin{aligned}\phi =& \mathcal{F}(g \text{ rightOf } r \wedge g \text{ rightOf } b)\\ &\wedge\neg(g \text{ rightOf } r \wedge g \text{ rightOf } b)\mathcal{U}(r \text{ above } b)\\ &\wedge\mathcal{G}\bigwedge_{i\neq j} \mathrm{d}(i,j) \geq 0.03, i, j \in \{r, g, b\},\end{aligned}$$

where r, g and b denote the red, green, and blue blocks, respectively. Intuitively speaking, this specification requires the green block to eventually be right of both the red and the blue block at the same time. We disallow the green block to be right of the other blocks until the red block is above the blue block (which is not true in the initial configuration). Additionally, no block should ever come too close to another block. From this specification, we automatically generate a DFA representing the temporal structure as described in Sect. 5.2. The resulting automaton is depicted in Fig. 16, with the following mapping from atomic propositions to spatial subformulae:

$$\begin{aligned}a &:= g \text{ rightOf } r \wedge g \text{ rightOf } b\\ b &:= r \text{ above } b\\ c &:= \bigwedge_{i\neq j} \mathrm{d}(i,j) \geq 0.03, i, j \in \{r, g, b\}\end{aligned}$$

Given the initial configuration of objects depicted in Figs. 15 and 17, the planning algorithm starts in DFA state 3. The shortest path to a satisfying configuration is by taking the transition $3 \rightarrow 5$ to the accepting state, requiring all the spatial subformulae to hold true at the same time. Since it is not possible to satisfy all subformulae directly by moving only a single object, the transition is assessed as infeasible and pruned. The next (and only) path to an accepting state is $3 \rightarrow 4 \rightarrow 5$.

Figure 17a shows the gradient map for the blue block generated from the goal and constraint set for transition $3 \rightarrow 4$. The dashed line indicates the border to a satisfying value. The shortest pushing path that avoids all constrained areas and triggers the desired transition is computed through Jump-Point-Search (Harabor and Grastien , 2011). The path avoids all areas that violate the constraints in Σ_c, depicted in white. Moving the blue block below the red block satisfies the Until-requirement of the specification. The robot executes the plan and pushes the blue block along the computed path. Our implemented pushing controller uses the feedback of our monitoring algorithm in continuous space to adjust control inputs and account for low level control effects where blocks are not pushed properly into satisfying regions. Note that we do not need to specify the exact order of which block needs to be pushed; the planning algorithm has the freedom to decide suitable actions on the fly. For example, the planning algorithm may decide to push the red block above the blue block instead, as this action also leads to specification satisfaction. Afterwards, the DFA state is updated through observation and the current state evolves to 4. From here, the planner can reach the satisfying state 5 by pushing the green block right of both other blocks, depicted in Fig. 17b. After execution, the specification is satisfied and the algorithm terminates.

Pick-and-place task and motion planning The second experiment considers a similar setup, where a robot arm is mounted on a planar surface. The robot is tasked to arrange the different objects on the table. The robot can pick and place objects in a radial area around the base. The objects are identified by the labels: *bottle*, *banana*, *mug*, *gelatin*, *sugarbox*, *can*, *crackerbox*, *kanelbulle* and *plate*. The SpaTiaL specification $\phi_{\text{plan,pnp}}$ is given in Table 3, and requires the robot, e.g., to place the Swedish cinnaman bun *kanelbulle* within the *plate*. The resulting DFA has 33 states and 275 transitions. The shortest paths to an accepting state are unsatisfiable through manipulation of a single object and are pruned online. When the computed gradient maps contain positive values, the robot chooses to place the considered object at the position which maximises the value of the gradient map. After executing the action, the DFA is updated with the latest observations. Since we monitor the evolution of the workspace during execution, our planning method is able to detect failing grasps

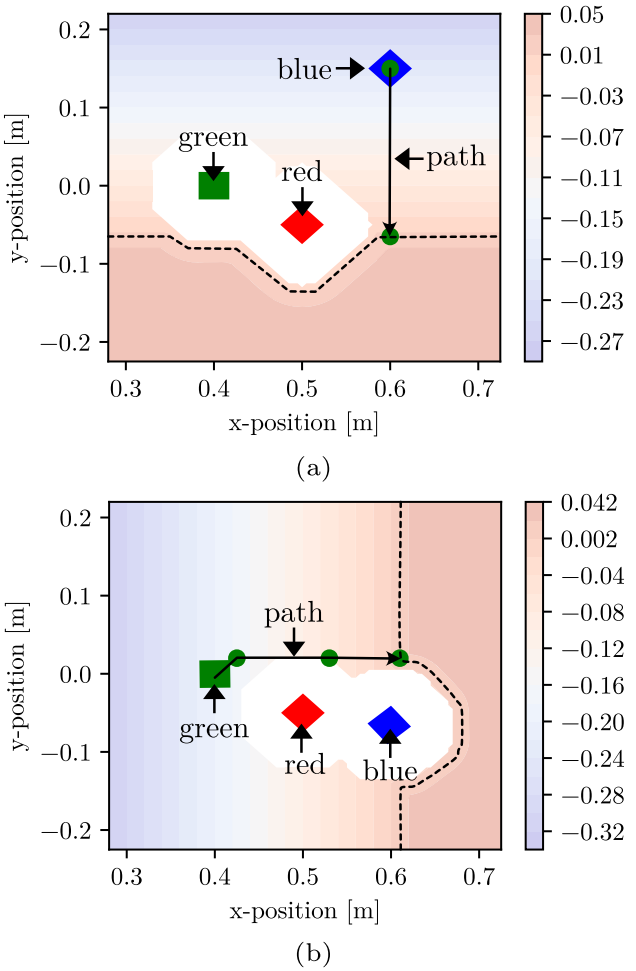


Fig. 17 In the pushing task experiment, the robot has to satisfy a given specification by pushing the blocks around the workspace

or misplaced objects. Execution failures in the simulation range from tipping over neighbouring objects or losing grasp of the current object. After failed executions, a new path to an accepting state is found and execution continues with an alternative next step or another try of the same execution step. An exemplary execution progression is depicted in Fig. 18. Since the specification does not impose any constraint on the order in which the desired spatial relationships are fulfilled, each execution can slightly differ.

Our planning experiments illustrate how to facilitate TAMP over SpaTiaL specifications through repeated observation and planning of the next step through the DFA and gradient maps.

Experiments with an ABB YuMi Robot After our successful simulation experiments, we implemented our approach on an ABB YuMi Robot to validate planning with real sensor data and low-level controllers. The robot's table setting task specification ϕ_{ABB} can be found in Table 3. The objects are detected using camera images.

Grasping objects is a difficult problem and depends on the robots kinematics, end-effector dynamics and shape of the object. We deliberately designed SpaTiaL to not consider those dynamical effects and focus on the object placement itself. Yet, we can consider the grasping of objects in our ABB YuMi experiments: we use a Behavior tree approach to combine symbolic planning and geometric reasoning. As in the other experiments, our planning automaton first determines which spatial relation should be satisfied next. This triggers the behavior tree controller of the robot. The gradient maps provide the behavior tree the desired object pose. The picking pose is determined by another behavior tree controller when the tree determines the behavior to pick or release an object. In this way, we can be controller-agnostic and allow users to use their designated controllers to determine the picking pose while SpaTiaL solely provides feedback in continuous space about how objects should be arranged.

Figure 19 shows two snapshots from our two robot experiments. In the first experiment (see Fig. 19a), the robot executes the desired table setting task by rearranging the objects according to ϕ_{ABB}, e.g., placing the knife right of the plate with orientation pointing away from the robot. In the second experiment (see Fig. 19b), we use a mobile YuMi robot for the same specification ϕ_{ABB}. However, we place the knife on another table in this scenario. Our satisfaction values guide the behavior tree-based controller of our mobile robot to first pick up the knife and then to transport it to goal table. The online monitoring in our example planning approach automatically reflects the successful placement of the knife to continue rearranging the remaining objects. Videos of the two experiments can be found in the paper's media attachment.

7 Discussion and limitations

Our results showed that we can use SpaTiaL to specify, monitor and plan rich spatio-temporal robotic tasks, such as object pushing, pick and place, and agent monitoring tasks.

We propose various spatial relations, such as leftOf, rightOf and above, to specify object placements. Currently, users can only address tasks which can be described with the existing set of spatial relations. However, this set can be extended by defining new spatial relations, such as presented in our experiments. In the future, we would like to add more spatial relations and support for occlusions and partial observability.

Our quantitative semantics only capture spatial and temporal aspects. For many robotics tasks, it may be important to include additional object properties in the specification, e.g., an object's color, weight, or temperature. Since SpaTial is a subset of STL, these properties can be translated into a

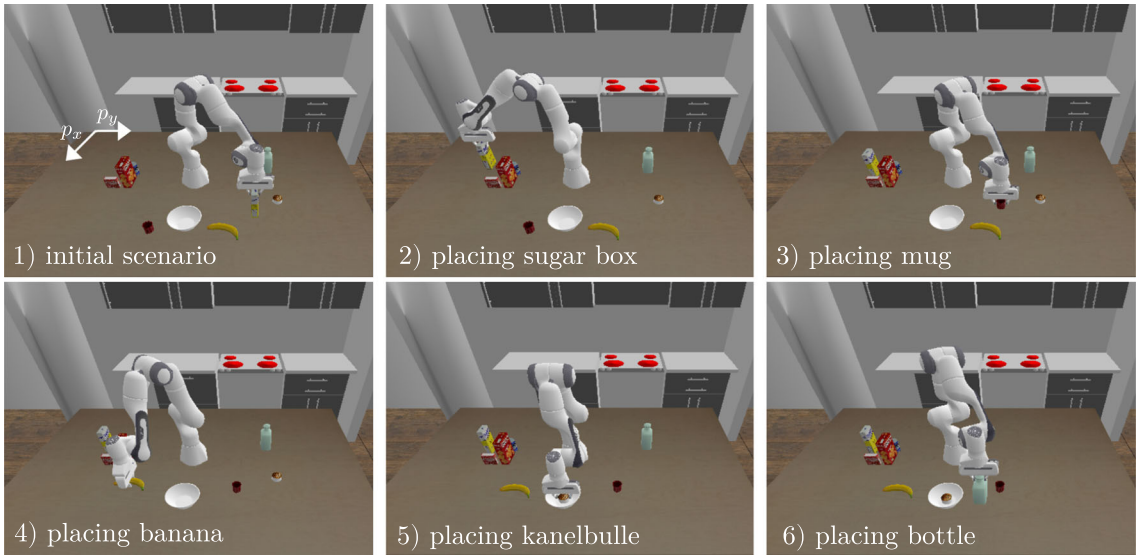
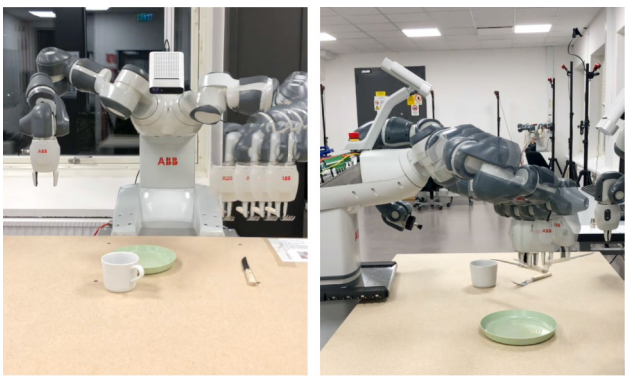


Fig. 18 Several snapshots of the simulator environment during execution of the pick-and-place TAMP experiment. For visibility, the camera is offset by a 90 deg angle in comparison to the workspace orientation



(a) (b)

Fig. 19 Experiments with an ABB YuMI robot. **a** the robot places a fork left of the plate. **b** the robot orientates and laces a knife

signal function that provides a satisfaction value for every timestep and could be incorporated.

Compared to classical TAMP approaches such as PDDLstream (Garrett et al., 2020), SpaTiaL focuses on object-centric tasks, allowing users to be robot-agnostic and disregard robot dynamics for simplicity. This comes with the limitation that SpaTiaL cannot synthesize plans directly as opposed to the reasoning system in PDDL. Moreover, SpaTiaL specifications can only define pre-conditions on the robot's state that can be grounded on geometrical representations and not yet identify dynamical illogical specifications. Yet, SpaTiaL operates in continuous space and can be combined with almost any planner and controller. By determining the desired object configuration, a dedicated planner and controller can take care of the successful satisfaction of the specification.

This combination with a planner and controller, however, cannot guarantee completeness (see also Sect. 5.2.1) as opposed to integrated symbolic and control approaches such as logic geometric programming (Toussaint, 2015). These integrated approaches can also consider object physics and plan object grasps. On the other hand, SpaTiaL provides a larger spatial and temporal expressivity. Users can specify a wide variety of tasks with the rich set of spatial relations, easily incorporate new relations, and use bounded and unbounded temporal operators. This makes it possible to use SpaTiaL for a wide variety of tasks, alone as a specification language or monitoring approach.

To improve the panning with SpaTiaL, we will expand the planning algorithm by providing general pruning rules for infeasible DFA transitions. Currently, the tool MONA can generate the DFAs from SpaTiaL specifications in milliseconds. However, finding object target positions to satisfy Lemma 2, e.g., by sampling, is more costly and can take from seconds to minutes. To address the computation times, we will investigate optimization-based approaches to satisfy spatial relations in our planning approach. Lastly, we like to develop approaches to learn specifications from demonstrations and to leverage users to correct specifications (Kress-Gazit et al., 2008, 2021; van Waveren et al., 2021; Kent et al., 2017; Zhang et al., 2021; van Waveren et al., 2022).

8 Conclusions

We presented SpaTiaL, a spatio-temporal framework that combines techniques from computational geometry and tem-

poral logics to provide a unified framework to specify, monitor and plan rich spatio-temporal tasks for robotic applications. It operates in an object-centric fashion and provides various spatial relations between objects, such as an object is left of or overlaps with another object. These relations resemble natural language which increases the usability by end users (Skubic et al., 2004; Nicolescu et al., 2019; Kress-Gazit et al., 2008). SpaTiaL can monitor the satisfaction of spatial relations in continuous space through its quantitative semantics. Users can easily compose new relations from existing ones. Spatial relations can be composed with (bounded) temporal operators to specify complex temporal patterns, e.g., constraints or sequences of configurations.

In various experiments, we used SpaTiaL to monitor tasks in real-time and for online planning. For the latter, we can decompose a specification into temporal and spatial abstractions by generating automata and computing the specification's robustness value. This decomposition has the advantage that high-level actions and low-level control can be treated separately. Furthermore, the low-level control level can make use of existing optimal controllers (e.g., MPC) and planners (e.g., CHOMP). SpaTiaL is open source and can be modified and extended by anyone.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10514-023-10145-1.

Acknowledgements We thank the WASP Research Arena Robotics and ABB for using their research platform. We also thank Ioanna Mitsioni, Fernando S. Barbosa, Sanne van Waveren, and Alexis Linard for their feedback.

Author Contributions CP and GFS developed the concepts and algorithms, implemented SpaTiaL, and designed as well as conducted the experiments. FE implemented the simulation environment. JT and DK helped conceptualizing the algorithms and research direction and supported the research project. The article was written by CP and GFS. All authors revised the article for publication.

Funding This research was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation and the Swedish Research Council (VR) (Project No. 2017-05102). The authors are also affiliated with Digital Futures.

Declarations

Ethics approval Danica Kragic is an Editorial Board Member of Springer Autonomous Robots. The experiments in this article have been conducted in compliance with ethical standards.

References

Agostini, A., & Lee, D. (2020). Efficient state abstraction using object-centered predicates for manipulation planning. Preprint retrieved from preprint arXiv:2007.08251

Aineto, D., Jiménez, S., & Onaindia, E. (2018). Learning STRIPS action models with classical planning. In *Proceeding of the international conference on automated planning and scheduling*.

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM, 26*(11), 832–843.

Alur, R., Feder, T., & Henzinger, T. A. (1996). The benefits of relaxing punctuality. *Journal of the ACM, 43*(1), 116–146.

Baier, C., & Katoen, J. P. (2008). Principles of model checking. MIT Press.

Barbosa, F. S., Duberg, D., Jensfelt, P., & Tumova, J. (2019). Guiding autonomous exploration with signal temporal logic. *IEEE Robotics and Automation Letters, 4*(4), 3332–3339.

Billard, A., & Kragic, D. (2019). Trends and challenges in robot manipulation. *Science, 364*(6446), eaat8414.

Boyd, S., Boyd, S. P., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

Bylander, T. (1994). The computational complexity of propositional strips planning. *Artificial Intelligence, 69*(1–2), 165–204.

Cashmore, M., Fox, M., Long, D., Magazzeni, D., Ridder, B., Carrera, A., Palomeras, N., Hurtos, N., & Carreras,M. (2015). ROSplan: Planning in the robot operating system. In *Proceedings of the international conference on automated planning and scheduling*.

Colledanchise, M., & Ögren, P. (2018). Behavior trees in robotics and AI: An introduction. CRC Press.

Correll, N., Bekris, K. E., Berenson, D., Brock, O., Causo, A., Hauser, K., Okada, K., Rodriguez, A., Romano, J. M., & Wurman, P. R. (2016). Analysis and observations from the first amazon picking challenge. *IEEE Transactions on Automation Science and Engineering, 15*(1), 172–188.

Coumans, E., & Bai, Y. (2022). PyBullet, a Python module for physics simulation for games, robotics and machine learning. https://pybullet.org.

Dantam, N. T., Kingston, Z. K., Chaudhuri, S., & Kavraki, L. E. (2016). Incremental task and motion planning: A constraint-based approach. *In Robotics: Science and systems, 12*, 1–11.

De Giacomo, G., & Vardi, M. Y. (2013). Linear temporal logic and linear dynamic logic on finite traces. In *Proceedings of the ACM international conference on artificial intelligence* (pp. 854–860).

Deng, B., Genova, K., Yazdani, S., Bouaziz, S., Hinton, G., & Tagliasacchi, A. (2020). CvxNet: Learnable convex decomposition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 31–44).

Devin, C., Abbeel, P., Darrell, T., & Levine, S. (2018). Deep object-centric representations for generalizable robot learning. In *Proceedings of the IEEE international conference on robotics and automation* (pp. 7111–7118).

Diehl, M., Paxton, C., & Ramirez-Amaro, K. (2021). Automated generation of robotic planning domains from observations. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems* (pp. 6732–6738).

Donzé, A., Ferrere, T., Maler, O. (2013). Efficient robust monitoring for STL. In *International conference on computer aided verification* (pp. 264–279). Springer.

Donzé, A., & Maler, O. (2010). Robust satisfaction of temporal logic over real-valued signals. In *Proceedings of the international conference on formal modeling and analysis of timed systems* (pp. 92–106).

Driess, D., Ha, J. S., Tedrake, R., & Toussaint, M. (2021). Learning geometric reasoning and control for long-horizon tasks from visual input. In *Proceedings of the IEEE international conference on robotics and automation* (pp. 14298–14305).

Driess, D., Ha, J. S., Toussaint, M., & Tedrake, R. (2022). Learning models as functionals of signed-distance fields for manipulation planning. In *Conference on robot learning* (pp. 245–255).

Ericson, C. (2004). *Real-time collision detection*. CRC Press.

Fikes, R. E., Hart, P. E., & Nilsson, N. J. (1972). Learning and executing generalized robot plans. *Artificial intelligence, 3*, 251–288.

Finkemeyer, B., Kröger, T., & Wahl, F. M. (2005). Executing assembly tasks specified by manipulation primitive nets. *Advanced Robotics, 19*(5), 591–611.

Finucane, C., Jing, G., & Kress-Gazit, H. (2010). LTLMoP: Experimenting with language, temporal logic and robot control. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems* (pp. 1988–1993).

Fox, M., & Long, D. (2003). PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research, 20*, 61–124.

Fuggitti, F. (2021). LTLf2DFA Github repository. Retrieved July 20, 2021, from https://github.com/whitemech/LTLf2DFA

Garrett, C. R., Lozano-Pérez, T., & Kaelbling, L. P. (2017). STRIPS planning in infinite domains (pp. 1–11). Preprint retrieved from arXiv:1701.00287

Garrett, C. R., Lozano-Pérez, T., & Kaelbling, L. P. (2020). PDDLStream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. In *Proceedings of the international conference on automated planning and scheduling* (pp. 440–448).

Ghzouli, R., Berger, T., Johnsen, E. B., Dragule, S., & Wasowski, A. (2020). Behavior trees in action: A study of robotics applications. In *Proceedings of the ACM SIGPLAN international conference on software language engineering* (pp. 196–209).

Gillies, S., Bierbaum, A., Lautaportti, K., & Tonnhofer, O. (2007). Shapely: Manipulation and analysis of geometric objects.

Guadarrama, S., Riano, L., Golland, D., Go, D., Jia, Y., Klein, D., Abbeel, P., Darrell, T. (2013). Grounding spatial relations for human-robot interaction. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems* (pp. 1640–1647).

Harabor, D., & Grastien, A. (2011). Online graph pruning for pathfinding on grid maps. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 25.

He, K., Lahijanian, M., Kavraki, L. E., & Vardi, M. Y. (2015). Towards manipulation planning with temporal logic specifications. In *Proceedings of the IEEE international conference on robotics and automation* (pp. 346–352).

He, K., Lahijanian, M., Kavraki, L. E., & Vardi, M. Y. (2018). Automated abstraction of manipulation domains for cost-based reactive synthesis. *IEEE Robotics and Automation Letters, 4*(2), 285–292.

He, K., Wells, A. M., Kavraki, L. E., & Vardi, M. Y. (2019). Efficient symbolic reactive synthesis for finite-horizon tasks. In *Proceedings of the international conference on robotics and automation* (pp. 8993–8999).

Henriksen, J. G., Jensen, J., Jørgensen, M., Klarlund, N., Paige, R., Rauhe, T., & Sandholm, A. (1995). Mona: Monadic second-order logic in practice. In *International workshop on tools and algorithms for the construction and analysis of systems* (pp. 89–110). Springer.

Iovino, M., Scukins, E., Styrud, J., Ögren, P., & Smith, C. (2022). A survey of behavior trees in robotics and AI. *Robotics and Autonomous Systems, 154*, 104096.

Izatt, G., & Tedrake, R. (2020). Generative modeling of environments with scene grammars and variational inference. In *Proceedings of the IEEE international conference on robotics and automation* (pp. 6891–6897).

Jiménez, S., De La Rosa, T., Fernández, S., Fernández, F., & Borrajo, D. (2012). A review of machine learning for automated planning. *The Knowledge Engineering Review, 27*(4), 433–467.

Jund, P., Eitel, A., Abdo, N., & Burgard, W. (2018). Optimization beyond the convolution: Generalizing spatial relations with end-to-end metric learning. In *Proceedings of the IEEE international conference on robotics and automation* (pp. 4510–4516).

Kantaros, Y., Malencia, M., Kumar, V., & Pappas, G .J. (2020). Reactive temporal logic planning for multiple robots in unknown environments. In *Proceedigns of the IEEE international conference on robotics and automation* (pp. 11479–11485).

Karpas, E., & Magazzeni, D. (2020). Automated planning for robotics. *Annual Review of Control, Robotics, and Autonomous Systems, 3*, 417–439.

Kartmann, R., Paus, F., Grotz, M., & Asfour, T. (2018). Extraction of physically plausible support relations to predict and validate manipulation action effects. *IEEE Robotics and Automation Letters, 3*(4), 3991–3998.

Katayama, M., Tokuda, S., Yamakita, M., & Oyama, H. (2020). Fast LTL-based flexible planning for dual-arm manipulation. In *Proceeedings of the IEEE/RSJ international conference on intelligent robots and systems* (pp. 6605–6612).

Kent, D., Saldanha, C., & Chernova, S. (2017). A comparison of remote robot teleoperation interfaces for general object manipulation. In *Proceedings of the ACM/IEEE international conference on human-robot interaction* (pp. 371–379).

Koymans, R. (1990). Specifying real-time properties with metric temporal logic. *Real-Time Systems, 2*(4), 255–299.

Kress-Gazit, H., Eder, K., Hoffman, G., Admoni, H., Argall, B., Ehlers, R., Heckman, C., Jansen, N., Knepper, R., Křetínskỳ, J., et al. (2021). Formalizing and guaranteeing human-robot interaction. *Communications of the ACM, 64*(9), 78–84.

Kress-Gazit, H., Fainekos, G. E., & Pappas, G. J. (2008). Translating structured English to robot controllers. *Advanced Robotics, 22*(12), 1343–1359.

Kress-Gazit, H., Fainekos, G. E., & Pappas, G. J. (2009). Temporal-logic-based reactive mission and motion planning. *IEEE Transactions on Robotics, 25*(6), 1370–1381.

Kress-Gazit, H., Lahijanian, M., & Raman, V. (2018). Synthesis for robots: Guarantees and feedback for robot behavior. *Annual Review of Control, Robotics, and Autonomous Systems, 1*, 211–236.

Kröger, T., Finkemeyer, B., & Wahl, F. M. (2010). *Manipulation primitives-a universal interface between sensor-based motion control and robot programming, Robotic systems for handling and assembly* (pp. 293–313). Springer.

Kshirsagar, A., Kress-Gazit, H., & Hoffman, G. (2019). Specifying and synthesizing human-robot handovers. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems* (pp. 5930–5936).

Kupferman, O., & Vardi, M. Y. (2001). Model checking of safety properties. *Formal Methods in System Design, 19*(3), 291–314.

Lacerda, B., Parker, D., & Hawes, N. (2014). Optimal and dynamic planning for Markov decision processes with co-safe LTL specifications. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems* (pp. 1511–1516).

Li, S., Park, D., Sung Y., Shah, J. A., & Roy, N. (2021). Reactive task and motion planning under temporal logic specifications. In *Pro-*

ceedings of the IEEE international conference on robotics and automation (pp. 12618–12624).
Li, X., Guo, D., Liu, H., & Sun, F. (2022). Embodied semantic scene graph generation. In *Conference on robot learning* (pp. 1585–1594).
Li, X., Vasile, C. I., & Belta, C. (2017). Reinforcement learning with temporal logic rewards. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems* (pp. 3834–3839).
Lindemann, L., & Dimarogonas, D. V. (2017). Robust motion planning employing signal temporal logic. In *Proceedings of the American control conference* (pp. 2950–2955).
Lindemann, L., & Dimarogonas, D. V. (2018). Control barrier functions for signal temporal logic tasks. *IEEE Control Systems Letters, 3*(1), 96–101.
Liu, W., Paxton, C., Hermans, T., & Fox, D. (2022). StructFormer: Learning spatial structure for language-guided semantic rearrangement of novel objects. In *Proceedings of the international conference on robotics and automation* (pp. 6322–6329).
Loula, J., Allen, K., Silver, T., & Tenenbaum, J. (2020). Learning constraint-based planning models from demonstrations. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems* (pp. 5410–5416).
Lozano-Perez, T. (1990). *Spatial planning: A configuration space approach, Autonomous robot vehicles* (pp. 259–271). Springer.
Manuelli, L., Gao, W., Florence, P. R., & Tedrake, R. (2019). KPAM: Keypoint affordances for category-level robotic manipulation. In *Proceedings of the international symposium on robotics research*, Vol. 20 (pp. 132–157). Springer.
McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., & Wilkins, D. (1998). PDDL-the planning domain definition language. Technical report, Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational.
Menghi, C., Tsigkanos, C., Berger, T., & Pelliccione, P. (2019). PsALM: Specification of dependable robotic missions. In *Proceedings of the IEEE/ACM international conference on software engineering* (pp. 99–102).
Menghi, C., Tsigkanos, C., Pelliccione, P., Ghezzi, C., & Berger, T. (2019). Specification patterns for robotic missions. *IEEE Transactions on Software Engineering, 47*(10), 2208–2224.
Migimatsu, T., & Bohg, J. (2020). Object-centric task and motion planning in dynamic environments. *IEEE Robotics and Automation Letters, 5*(2), 844–851.
Mojtahedzadeh, R., Bouguerra, A., Schaffernicht, E., & Lilienthal, A. J. (2015). Support relation analysis and decision making for safe robotic manipulation tasks. *Robotics and Autonomous Systems, 71*, 99–117.
Nicolescu, M., Arnold, N., Blankenburg, J., Feil-Seifer, D., Banisetty, S., Nicolescu, M., Palmer, A., Monteverde, T. (2019). Learning of complex-structured tasks from verbal instruction. In *Proceedings of the IEEE-RAS international conference on humanoid robots* (pp. 747–754).
Ögren, P. (2020). Convergence analysis of hybrid control systems in the form of backward chained behavior trees. *IEEE Robotics and Automation Letters, 5*(4), 6073–6080.
Oleynikova, H., Millane, A., Taylor, Z., Galceran, E., Nieto, J., & Siegwart, R. (2016). Signed distance fields: A natural representation for both mapping and planning. In *RSS workshop: Geometry and beyond-representations, physics, and scene understanding for robotics*.
Pan, T., Wells, A. M., Shome, R., & Kavraki, L. E. (2021). A general task and motion planning framework for multiple manipulators. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems* (pp. 3168–3174).
Panda, S., Hafez, A. A., & Jawahar, C. (2016). Single and multiple view support order prediction in clutter for manipulation. *Journal of Intelligent & Robotic Systems, 83*(2), 179–203.
Paus, F., Huang, T., & Asfour, T. (2020). Predicting pushing action effects on spatial object relations by learning internal prediction models. In *Proceedings of the IEEE international conference on robotics and automation* (pp. 10584–10590).
Paxton, C., Xie, C., Hermans, T., & Fox, D. (2022). Predicting stable configurations for semantic placement of novel objects. In *Conference on robot learning* (pp. 806–815).
Pednault, E. P. (1989). ADL: Exploring the middle ground between STRIPS and the situation calculus. In *Proceedings of the international conference on principles of knowledge representaion and reasoning* (pp. 324–332).
Pek, C., Muxfeldt, A., & Kubus, D. (2016). Simplifying synchronization in cooperative robot tasks-an enhancement of the manipulation primitive paradigm. In *Proceedings of the IEEE international conference on emerging technologies and factory automation* (pp. 1–8).
Plaku, E., & Karaman, S. (2016). Motion planning with temporal-logic specifications: Progress and challenges. *AI Communications, 29*(1), 151–162.
Pnueli, A. (1977). The temporal logic of programs. In *Proceedings of the annual symposium on foundations of computer science* (pp. 46–57).
Raman, V., Donzé, A., Maasoumy, M., Murray, R. M., Sangiovanni-Vincentelli, A., & Seshia, S. A. (2014). Model predictive control with signal temporal logic specifications. In *Proceedings of the IEEE conference on decision and control* (pp. 81–87).
Ramirez-Amaro, K., Beetz, M., & Cheng, G. (2017). Transferring skills to humanoid robots by extracting semantic representations from observations of human activities. *Artificial Intelligence, 247*, 95–118.
Ramirez-Amaro, K., Kim, E. S., Kim, J., Zhang, B. T., Beetz, M., & Cheng, G. (2013). Enhancing human action recognition through spatio-temporal feature learning and semantic rules. In *Proceedings of the IEEE-RAS international conference on humanoid robots* (pp. 456–461).
Robicquet, A., Sadeghian, A., Alahi, A., & Savarese, S. (2016). Learning social etiquette: Human trajectory understanding in crowded scenes. In *Proceedings of the European conference on computer vision* (pp. 549–565). Springer.
Rosman, B., & Ramamoorthy, S. (2011). Learning spatial relationships between objects. *The International Journal of Robotics Research, 30*(11), 1328–1342.
Schillinger, P., Bürger, M., & Dimarogonas, D. V. (2018). Decomposition of finite LTL specifications for efficient multi-agent planning. In *Distributed autonomous robotic systems* (pp. 253–267).
Schulman, J., Duan, Y., Ho, J., Lee, A., Awwal, I., Bradlow, H., Pan, J., Patil, S., Goldberg, K., & Abbeel, P. (2014). Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research, 33*(9), 1251–1270.
Sharma, M., Liang, J., Zhao, J., LaGrassa, A., & Kroemer, O. (2020). Learning to compose hierarchical object-centric controllers for robotic manipulation. In *Proceedings of the conference on robot learning*, Vol. 155 (pp. 822–844).
Spawn-Lee, S. A., Lark, T. J., Gibbs, H. K., Houghton, R. A., Kucharik, C. J., Malins, C., Pelton, R. E., Robertson, G. P. (2021). Lark Github repository. Retrieved July 20, 2021, from https://github.com/lark-parser/lark
Shridhar, M., Manuelli, L., & Fox, D. (2022). CLIPort: What and where pathways for robotic manipulation. In *Conference on robot learning* (pp. 894–906).
Silver, T., Chitnis, R., Tenenbaum, J. B., Kaelbling, L. P., & Lozano-Pérez, T. (2021). Learning symbolic operators for task and motion

planning. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems* (pp. 3182–3189).
Skubic, M., Perzanowski, D., Blisard, S., Schultz, A., Adams, W., Bugajska, M., & Brock, D. (2004). IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews). *Spatial language for human-robot dialogs, 34*(2), 154–167.
Sobti, S., Shome, R., Chaudhuri, S., & Kavraki, L. E. (2021). A sampling-based motion planning framework for complex motor actions. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems* (pp. 6928–6934).
Sprague, C. I., Özkahraman,Ö., Munafo, A., Marlow, R., Phillips, A., & Ögren, P. (2018). Improving the modularity of AUV control systems using behaviour trees. In *Proceedings of the IEEE/OES autonomous underwater vehicle workshop* (pp. 1–6).
Toussaint, M. (2015). Logic-geometric programming: An optimization-based approach to combined task and motion planning. In *Proceedings of the international joint conference on artificial intelligence*.
Toussaint, M. A., Allen, K. R., Smith, K. A., & Tenenbaum, J. B. (2018). *Differentiable physics and stable modes for tool-use and manipulation planning*. In Robotics: Science and systems foundation.
Tumova, J., Hall, G. C., Karaman, S., Frazzoli, E., & Rus, D. (2013). Least-violating control strategy synthesis with safety rules. In *Proceedings of the international conference on hybrid systems: computation and control* (pp. 1–10).
van Waveren, S., Carter, E. J., Örnberg, O., & Leite, I. (2021). Exploring non-expert robot programming through crowdsourcing. *Frontiers in Robotics and AI, 8*, 242.
van Waveren, S., Pek, C., Tumova, J., & Leite, I. (2022). Correct me if I'm wrong: Using non-experts to repair reinforcement learning policies. In *Proceedings of the ACM/IEEE international conference on human-robot interaction* (pp. 1–9).
Vasile, C. I., Tumova, J., Karaman, S., Belta, C., & Rus, D. (2017). Minimum-violation scLTL motion planning for mobility-on-demand. In *Proceedings of the IEEE international conference on robotics and automation* (pp. 1481–1488).
Vasilopoulos, V., Kantaros, Y., Pappas, G. J., & Koditschek, D. E. (2021). Reactive planning for mobile manipulation tasks in unexplored semantic environments. In *Proceedings of the IEEE international conference on robotics and automation* (pp. 6385–6392).
Wang, J., & Olson, E. (2016). Apriltag 2: Efficient and robust fiducial detection. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems* (pp. 4193–4198).
Wells, A. M., Dantam, N. T., Shrivastava, A., & Kavraki, L. E. (2019). Learning feasibility for task and motion planning in tabletop environments. *IEEE Robotics and Automation Letters, 4*(2), 1255–1262.
Wells, A. M., Lahijanian, M., Kavraki, L. E., & Vardi, M. Y. (2020). LTLf synthesis on probabilistic systems. In *Proceedings of the international symposium on games, automata, logics, and formal verification*, Vol. 326 (pp. 166–181).
Wells, M., Kingston, Z., Lahijanian, M., Kavraki, L. E., & Vardi, M. Y. (2021). Finite-horizon synthesis for probabilistic manipulation domains. In *Proceedings of the IEEE international conference on robotics and automation* (pp. 6336–6342).
Yuan, W., Paxton, C., Desingh, K., & Fox, D. (2022). SORNet: Spatial object-centric representations for sequential manipulation. In *Conference on Robot Learning* (pp. 148–157).
Zhang, H., Lu, Y., Yu, C., Hsu, D., La, X., & Zheng, N. (2021). Invigorate: Interactive visual grounding and grasping in clutter, pp. 1–11. Preprint retrieved from arXiv:2108.11092
Zhao, Z., Zhou, Z., Park, M., & Zhao, Y. (2021). SyDeBO: Symbolic-decision-embedded bilevel optimization for long-horizon manipulation in dynamic environments. *IEEE Access, 9*, 128817–128826.
Zucker, M., Ratliff, N., Dragan, A. D., Pivtoraiko, M., Klingensmith, M., Dellin, C. M., Bagnell, J. A., & Srinivasa, S. S. (2013). CHOMP: Covariant hamiltonian optimization for motion planning. *The International Journal of Robotics Research, 32*(9–10), 1164–1193.

Christian Pek is an assistant professor in the Department of Cognitive Robotics at Delft University of Technology since 2023. Before joining TU Delft, he was a postdoctoral researcher in the Division of Robotics, Perception and Learning at KTH Royal Institute of Technology and a PhD student at the Technical University of Munich and the BMW Group. In his research, Chris develops algorithmic foundations to ensure high degrees of safety, autonomy, and robustness of robots that operate and learn in complex real-world environments around humans.

Georg Friedrich Schuppe is a Researcher at SEBx and a PhD student in the Division of Robotics, Perception and Learning at KTH Royal Institute of Technology. Under supervision of Jana Tumova, he applies Formal Verification to the domain of Multi-Robot Systems by synthesizing correct-by-design strategies over Linear Time Logic specifications. Strategy negotation through assumption exchange between robots and humans are the overarching theme of his work. He received a Master of Science degree in computer science from the Leibniz University Hannover, Germany.

Francesco Esposito received his M.Sc. degree in automation engineering from University Federico II in Naples, Italy in 2019. From 2020 to 2021, he has been a Ph.D. student in the Division of Robotics, Perception and Learning at KTH Royal Institute of Technology. His research interests include robotics, machine learning and formal methods.

Jana Tumova (M'14) is an associate professor at the School of Electrical Engineering and Computer Science at KTH Royal Institute of Technology. She received PhD in computer science from Masaryk University and was awarded ACCESS postdoctoral fellowship at KTH in 2013. She was also a visiting researcher at MIT, Boston University, and Singapore-MIT Alliance for Research and Technology. Her research interests include formal methods applied in decision making, motion planning, and control of autonomous systems. She is a recipient of a Swedish Research Council Starting Grant and an Early Career Spotlight award from Robotics: Science and Systems foundation.

Danica Kragic (F'16) received MSc in mechanical engineering from Technical University of Rijeka, Croatia, in 1995 and the PhD in computer science from the Royal Institute of Technology, KTH, Sweden, in 2001. She is a Professor at the School of Electrical Engineering and Computer Science, Royal Institute of Technology, KTH. She had been a Visiting Researcher at Columbia University, Johns Hopkins University, and INRIA Rennes. Her research is in robotics, computer vision, and machine learning. Dr. Kragic received the 2007 IEEE Robotics and Automation Society Early Academic Career Award. She is a member of the Royal Swedish Academy of Sciences and Swedish Academy of Engineering Sciences. She holds an Honorary Doctorate from Lappeenranta University of Technology, Finland. She received an ERC Starting and Advanced Grants.