SpaTiaL: Monitoring and Planning of Robotic Tasks Using Spatio-Temporal Logic Specifications

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

***Abstract*—Various real-world robotic tasks are difficult to monitor and plan for due to the complex interplay of spatial and temporal requirements. For instance, arranging a dining table requires manipulating objects of different shapes in a predefined temporal order, e.g., first placing the mug and then filling it with coffee, while satisfying spatial relations such as forks need to placed left of plates. Existing solutions either do not consider tasks in an object-centric fashion, depend on manual definition of object relations, or are not expressive enough to capture a rich variety of tasks. To address these shortcomings, we propose a spatio-temporal framework SpaTiaL that unifies the specification, monitoring, and planning of various real-world object-oriented robotic tasks. SpaTiaL is expressive enough to capture the diverse spatial relations between objects and temporal patterns of tasks while providing real-time monitoring and online planning. We demonstrate the benefits of SpaTiaL in multiple real scenarios as well as in simulated object pushing and pick-and-place tasks. SpaTiaL is open source and easily expandable to new applications.**

Fig. 1. The illustrated kitchen task requires a complex interplay of spatial and temporal constraints. The robot needs to place the objects to predefined goal regions considering a temporal order. Moreover, it needs to consider time bounds such as placing the filled coffee mug within 10 seconds to the right side but close to the plate after the kanelbulle has been placed within the plate. This specification requires dealing with various object shapes, sensor uncertainties, temporal orders and monitoring in continuous space.

I. Introduction

Manually specifying the placement of objects and monitoring the robot's progress (towards the goal configuration and the satisfaction of constraints) is often manageable when planning simple robotic tasks. However, many real-world planning tasks require a complex interplay of spatial and temporal constraints [1], [2]. For instance, in the illustrated kitchen scenario (see Fig. 1), the robot should 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. The manipulated objects may further have various shapes and their exact positions might be uncertain due to sensor noise. Considering all requirements and properties of the scene makes manipulation tasks considerably harder to solve.

Various approaches have been developed to streamline and standardize the specification, monitoring and planning of robotic tasks, e.g., using the Stanford Research Institute Problem Solver (STRIPS) [3], the Planning Domain Definition Language (PDDL) [4], [5] or symbolic planning (see Sec. I-A) For instance, with the popular PDDL, designers can specify initial and goal states, predicates, objects and actions. However, the created specifications are usually quite complex, since designers need to specify the task domain and the action space to achieve the goal specification. Over the years, Temporal Logics (TLs) have become a popular method for integrated monitoring and planning of robotic tasks [6]. In contrast to the classical specification languages, TLs are rich and concise, often closer to resemble natural language and can be used to specify various temporal events of tasks. Yet, proposed TL solutions are often targeted towards mobile robot domains and require an abstraction of the environment; are complex to use for non-experts; or focus either on specification and monitoring 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 the task; 2) is simple to use so that non-experts can design specifications while being formal for automatic processing; and 3) can be used for all applications, namely specification, monitoring and planning. This approach further needs to support various object shapes, consider uncertainties and operate in continuous space to avoid discretization effects.

In this article, we present our framework SpaTiaL, that has been designed from the ground up to specify complex robotic tasks in an object-centric fashion. SpaTiaL provides various spatial relations between objects, e.g., an object is above, close and left of another object. We demonstrate that SpaTiaL is easy to use and allows both for concise monitoring as well as planning. SpaTiaL is declarative so that designers only specify the goal configuration and constraints of the task without being required to define high-level robot actions beforehand.

* The authors contributed equally to this work.
All of the authors are with the Division of Robotics, Perception and Learning, School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, 114 28 Stockholm, Sweden. Mail addresses: {pek2, schuppe, fesp, tumova, dani}@kth.se

A. Related Work

In the following paragraphs, we briefly review classical and recent approaches for task and motion planning (TAMP) [6], [7] and provide an overview of the spatio-temporal relations in complex planning 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.

a) Task and Motion Planning: Various description languages have been developed for different robotic applications. The Stanford Research Institute Problem Solver (STRIPS) is one of the earliest specification languages [3]. 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 [8], [9] and updated over the years [10], [11]. Yet, STRIPS specifications are not expressive enough for applications that involve conditional actions.

For this reason, the Action Description Language (ADL) [12] has been developed as an extension of STRIPS. ADL eliminates some shortcomings of STRIPS and allows one to remove the closed-world assumption (negative literals are allowed), specify goals with con- and disjunctions, and introduces conditional effects of actions. The introduction of the latter, enabled the expression of more complicated robotic tasks. Both, STRIPS and ADL, require a great effort to specify tasks, since designers need to specify the whole domain and the problem instance from scratch.

The Planning Domain Definition Language (PDDL) [4], [5], [13] separates domain and problem specific definitions. This choice was made to make planners comparable and to reuse domain definitions across various problem instances. PDDL can be seen as a generalized and standardized version of STRIPS. It can be found in many applications and is supported by various systems, such as the robot operating system [14]. Despite PDDL's success, it still requires a dedicated reasoning system to synthesize plans. Moreover, the previous approaches focus more on the specification and high level planning of tasks and not on uncertainties or low level control effects, which makes TAMP much harder to solve.

To also consider low lever control effects in TAMP, Manipulation primitives (MPs) [15]–[17] aim at 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. Yet, MPs need to be carefully designed for the intended purpose. Behavior Trees (BTs) follow a similar approach [18] to solve TAMP problems. 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 [19], [20]. Nevertheless, designers need to explicitly consider the problem-specific properties of the task to properly chain MPs or create BTs.

To integrate both high level reasoning and low level control, recent symbolic planning approaches unify the low-level control benefits with the power of reasoning systems to automatically synthesize plans [21]–[23], i.e., combined TAMP. In logic-geometric programming (LGP) [24], [25], an optimizer determines 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 [26].

The aforementioned TAMP approaches focus on combining the specification and planning of tasks. Most of these approaches require dedicated knowledge about 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, it is difficult to determine how close the system is to satisfying or violating the specification even in the presence of uncertainties.

b) Temporal logics for robotics: TLs have become increasingly popular for robotic applications since they allow one to focus on the high-level plan without dedicated knowledge about the system dynamics. Linear Temporal Logic (LTL) [27] is a modal logic over Boolean propositions, where the modality is interpreted as linearly progressing time. It was originally developed to describe and analyze the behaviour of software programs, but its use across different disciplines, such as robotics, has been a topic of research in the last years [28] [6]. LTLMoP [29] is a toolkit to design, test and implement hybrid controllers generated from LTL specifications.

LTL of finite traces (LTL_f) [30] and co-safe LTL (scLTL) [31] 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 [32], [33]. All classical LTL definitions or fragments thereof rely on discrete time and Boolean predicates, but some define quantitative predicates [34] or robustness metrics [35], [36]. Quantitative evaluations allow for computing a degree of satisfaction or violation of a trace and a specification. However, they manually define functions for atomic propositions while our approach uses spatial relations in a general and compositional manner.

Metric Temporal Logic (MTL) [37] enables expressing quantitative time properties by introducing bounded temporal operators. Metric Interval Temporal Logic (MITL) [38] 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 [38]. Instead of Boolean propositions as used in MITL and MTL, SpaTL reasons over real-valued propositions, allowing us to define quantitative semantics (see Sec. 14).

Signal Temporal Logic (STL) [39] 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 [40] and

motion planning [41]. STL specifications can often directly be used within optimization problems [42] or to define satisfying regions in the state space [43].

To summarize, temporal logics are well suited to cover temporal patterns in various tasks. Yet, some logics (like LTL) are more user-friendly due to the use of discrete high-level predicates, whereas continuous logics (like STL) offer evaluations of continuous signals while providing quantitative semantics. Unifying these advantages allows one to plan high-levels actions while providing low-level control.

c) Spatial and temporal requirements: Many robotic tasks involve a complex interplay of spatial and temporal patterns. 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 sensor uncertainties. In [1], [44], 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 move parts), and time-bounded actions [45].

Over the years, various approaches have been proposed to incorporate semantics or spatial relations between objects into robotic frameworks [46]–[48], e.g., what it means when two boxes are touching. These spatial relations provide robots with an understanding of the desired goal configuration of objects [49] and the ability to predict the outcome of their actions [50]. In this regard, many approaches perform object-centric planning, where the plan is obtained in the task space of the object itself [26], [51]. Spatial relations between objects range from high-level geometric relations, such as an object being left of another [52], [53], to relations that describe which objects support another object within a 3D structure [54]–[56]. Learning spatial relations from data and demonstrations is difficult [57], [58] due to the variety of relations and the complexity of separating similar relations.

Specification frameworks need to be able to encode the richness of temporal patterns while being able to incorporate spatial relations of various kinds. Moreover, they should provide a measure of how much a desired spatial relations is violated or satisfied so that robots can plan desired actions even in the presence of uncertainties. To enhance usability by non-expert users, it is beneficial when spatial relations can be defined close to natural language [59], [60].

B. Contributions

To unify the specification, monitoring, and planning of various robotic tasks, we present our spatio-temporal framework SpaTiaL that:

1) allows one to specify spatial and temporal relations in a continuous and object-centric fashion (i.e., without the need for dedicated robotic models);
2) provides quantitative semantics to monitor the satisfaction of a specification during runtime, enabling uniform sampling of satisfying scene configurations;
3) is able to monitor scenes in real-time while incorporating object uncertainties, which we demonstrate in various experiments with real data;
4) allows online planning by providing high-level temporal and low-level control abstractions, which we highlight in simulated object pushing and pick-and-place tasks;
5) leverages the quantitative properties of our spatial relations to provide concise low-level control for arranging objects; and
6) is openly available as a Python package and expendable by developers for various applications.

C. Structure of the article

The remainder of this article is structured as follows: in Sec. II, we introduce the necessary mathematical notations of sets and objects in the scene. Subsequently, we define spatial relations between objects, define the syntax of SpaTiaL and derive the computation of SpaTiaL's quantitative semantics in Sec. III. In Sec. V, we illustrate how SpaTiaL can be used to monitor and plan robotic tasks. In Sec. VI, 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 finish with conclusions in Sec. VII.

II. 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, and is in discrete-time. A perception module (e.g., a camera or laser scanner) provides us with the footprint $\mathcal{O}_i^t \subset \mathbb{R}^d$ of each obstacle $i \in \{1,\dots,N\}$ in the environment at a given point in time $t \in \mathbb{N}_{\geq 0}$. In this representation, we may also consider uncertainties, e.g., by providing a larger footprint for obstacle i or enlarging the footprint using the Minkowski sum and an error term E, i.e., $\mathcal{O}_{\text{enl},i}^t = \mathcal{O}_i \oplus E := \{p+e \,|\, p \in \mathcal{O}_i^t \wedge e \in E\}$. 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 $\forall p_1, p_2 \in \mathcal{O} : \forall \alpha \in [0,1] : (1-\alpha)p_1 + \alpha p_2 \in \mathcal{O}$.

In certain tasks, objects might not be convex, e.g., a banana or a cup. We can either compute the convex hull of these objects, i.e., $\text{hull}(\mathcal{O})$, or decompose them it into a set of convex objects. For instance, we might represent the inside and outside of a cup through cylinders and the handle with a set of cuboids. One can also specify phantom objects in our framework. Phantom objects are virtual objects that can be used to specify goal regions or forbidden regions for objects. With the introduced spatial relations, one can specify, e.g., whether an object is within a goal 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 $2^{\mathcal{X}}$. The state $s_t = (\mathcal{O}_1^t, \mathcal{O}_2^t, \dots)$ contains the information of all objects in the scene at time step $t \in \mathbb{N}$. For brevity, we omit the time step from an object $\mathcal{O}_i^t$ when the particular time step can be inferred from the context or is not relevant to explain a matter. Our definitions of spatial and temporal relations follow the Backus-Naur form.

III. Spatial Relations between Objects

Many robotic tasks involve arranging 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 $\mathcal{O}_j$. Ideally, spatial relations are defined in continuous space to eliminate discretization effects and to provide quantitative semantics, i.e., a continuous measure about the satisfaction of a spatial relation. In the following three subsections, we use signed distances and projections to derive spatial relations between objects and define quantitative semantics.

A. Distance-based relations

To derive distance-based spatial relations, we take inspiration from computational geometry and geometric optimization for robotic applications [61], [62]. Constraints between geometric shapes can be defined through constraining the distance between them, e.g., an object cannot be closer to another object by a predefined margin [63]. We derive various 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:

Definition 2 (Distance)
The distance $\mathrm{d}(O_i, O_j)$ *between two objects,* $O_i \subset \mathbb{R}^d$ *and* $O_j \subset \mathbb{R}^d$*, is the norm of the smallest translation so that both objects intersect:*

$$\mathrm{d}(O_i, O_j) := \inf \left\{ \|T\| \,|\, T \in \mathbb{R}^d \wedge (T \oplus O_i) \cap O_j \neq \emptyset \right\}.$$

Intuitively, the distance $\mathrm{d}(O_i, O_j)$ is larger than zero when O_i and O_j do not intersect and zero otherwise. In the case of $O_i \cap O_j \neq \emptyset$, 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 $\mathrm{pd}(O_i, O_j)$ *between two objects,* $O_i \subset \mathbb{R}^d$ *and* $O_j \subset \mathbb{R}^d$*, is the norm of the smallest translation so that the objects do not intersect:*

$$\mathrm{pd}(O_i, O_j) := \inf \left\{ \|T\| \,|\, T \in \mathbb{R}^d \wedge (T \oplus O_i) \cap O_j = \emptyset \right\}.$$

The distance and penetration depth can be efficiently computed using the Gilbert-Johnson-Keerthi algorithm and Minkowski differences [64, Sec. 9.5]. Def. 2 and Def. 3 are complementary to each other and can never simultaneously be non-zero. Finally, we define the signed distance as [65]:

Definition 4 (Signed Distance)
The signed distance $\mathrm{sd}(O_i, O_j)$ *between two objects,* $O_i \subset \mathbb{R}^d$ *and* $O_j \subset \mathbb{R}^d$*, is given by:*

$$\mathrm{sd}(O_i, O_j) := \mathrm{d}(O_i, O_j) - \mathrm{pd}(O_i, O_j).$$

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 properties of the signed distance allow us to define spatial relations for describing if two objects are in a certain distance, touch or overlap:

Fig. 2. Illustration of signed distances sd (dashed lines) between various objects: $sd > 0$ when objects do not intersect (e.g., $\mathrm{sd}(O_1, O_2)$), $sd = 0$ when objects touch (e.g., $\mathrm{sd}(O_2, O_5)$), and $sd < 0$ when the interior of objects intersects (e.g., $\mathrm{sd}(O_2, O_3)$).

Proposition 1 (Distance and Overlap Relations)
Using the signed distance sd*, two objects* $\mathcal{O}_i$ *and* $\mathcal{O}_j$ *are in* ϵ*-range (*$\mathrm{closeTo}_\epsilon$*) of each other if:*

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

and overlapping (ovlp *) if:*

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

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:*

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

Proof: When $\forall p \in \mathrm{hull}(\mathcal{O}_i) : \mathrm{sd}(p, \mathcal{O}_j) \leq 0$, 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., $\forall p_1, p_2 \in \mathrm{hull}(\mathcal{O}_i) \forall : \alpha \in [0, 1] : (1 - \alpha)p_1 + \alpha p_2 \in \mathcal{O}_j$. ■

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:*

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

B. Projection-based relations

The distance-based spatial relations already allow us to specify a variety of object configuration and to derive additional relations, such as an object being far away or in a certain range of distance values to another object. However, to place objects more precisely, we are further interested in specifying

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

directions, e.g., an object is left of another one. Therefore, let us introduce the projection operator:

Definition 6 (Projection)
The projection operator $\mathrm{proj}_{\mathrm{ax}}(\mathcal{O})$ *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 [66]:

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 projecting two objects onto a given axis, we can determine whether an object is partially or fully left of another object with respect to the chosen axis:

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

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

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

$$\mathcal{O}_i \,\mathrm{prec}_{\mathrm{ax}}\, \mathcal{O}_j := \max\big(\mathrm{proj}_{\mathrm{ax}}(\mathcal{O}_i)\big) - \min\big(\mathrm{proj}_{\mathrm{ax}}(\mathcal{O}_j)\big) \leq 0.$$

The precedence relations can be used to 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. Without loss of generality, we consider a 2D environment with axes p_x and p_y.

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:*

$$\begin{aligned}
\mathcal{O}_i \,\mathrm{leftOf}\, \mathcal{O}_j &:= \mathcal{O}_i \,\mathrm{prec}_{p_x}\, \mathcal{O}_j \\
\mathcal{O}_i \,\mathrm{rightOf}\, \mathcal{O}_j &:= \neg\mathcal{O}_i \,\mathrm{prec}_{p_x}\, \mathcal{O}_j \\
\mathcal{O}_i \,\mathrm{below}\, \mathcal{O}_j &:= \mathcal{O}_i \,\mathrm{prec}_{p_y}\, \mathcal{O}_j \\
\mathcal{O}_i \,\mathrm{above}\, \mathcal{O}_j &:= \neg\mathcal{O}_i \,\mathrm{prec}_{p_y}\, \mathcal{O}_j \\
\mathcal{O}_i \,\mathrm{between}_{\mathrm{ax}}\, \mathcal{O}_j \text{ and } \mathcal{O}_k &:= \mathcal{O}_j \,\mathrm{prec}_{\mathrm{ax}}\, \mathcal{O}_i \wedge \\
&\quad\ \mathcal{O}_i \,\mathrm{prec}_{\mathrm{ax}}\, \mathcal{O}_k
\end{aligned}$$

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.

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.

C. Spatial compositions and quantitative semantics

Complex spatial specifications can be designed by composing our spatial relations with Boolean operators:

Definition 10 (Spatial Relation Grammar)
Spatial relation specifications are defined using the following grammar:

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

where rel_2 *and* rel_3 *are spatial relations between two or three objects, respectively.*

Inspired by the quantitative semantics of STL, we define a 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.

Definition 11 (Quantitative Semantics for Spatial Relations)
We define the quantitative semantics $f(s_t, \mathcal{R})$ *of a spatial*

relation specification $\mathcal{R}$ as:

$$\begin{aligned}\rho(s_t, \top) &= \rho_{\max}\\ \rho(s_t, \neg\mathcal{R}) &= -\rho(s_t, \mathcal{R})\\ \rho(s_t, \mathcal{R}_1 \wedge \mathcal{R}_2) &= \min(\rho(s_t, \mathcal{R}_1), \rho(s_t, \mathcal{R}_2))\\ \rho(s_t, \mathcal{R}_1 \vee \mathcal{R}_2) &= \max(\rho(s_t, \mathcal{R}_1), \rho(s_t, \mathcal{R}_2))\end{aligned}$$

where ρ_{max} represents the maximum satisfaction value. We define ρ for single spatial relations:

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

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

Our geometry based spatial relations can also be used 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 12 (Time-Relative Spatial Relations)
An object $\mathcal{O}_i^t$ is moving to the right side if

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

Time-relative spatial relations can be defined for arbitrary spatial relations and allow the composition of 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.

IV. Temporal behaviour of spatial relations

Using temporal operators known from temporal logics such as LTL, we allow designers further to specify the interplay of spatial relations in time. More specifically, temporal relations are LTL formulas interpreted over finite traces with spatial relations as atomic propositions.

Definition 13 (Syntax of Temporal Relations)
Temporal behaviour of spatial relations is defined using the following syntax:

$$\phi := \top \,\big|\, \mathcal{R} \,\big|\, \neg\phi \,\big|\, \phi_1 \wedge \phi_2 \,\big|\, \phi_1 \mathcal{U} \phi_2 \,\big|\, \phi_1 \mathcal{U}_{[a,b]} \phi_2, \quad 0 \le a \le b \mathcal{X}\phi$$

$\mathcal{X}\phi$ requires a spatial relation to hold in the *next* time step. $\phi_1 U \phi_2$ requires ϕ_1 to hold *until* ϕ_2 holds. Using the operators $\neg$ and $\wedge$, the full power of propositional logic is obtained. Operators $\vee, \rightarrow, \leftrightarrow$ are defined as usual [67]. Furthermore, $\mathcal{F}\phi = \top U \phi$ represents ϕ holding true *eventually* in the future and $\mathcal{G}\phi = \neg\mathcal{F}\neg\phi$ *always* holding from now on. Similar to their unbounded variants, $\mathcal{F}_{[a,b]}\phi = \top U_{[a,b]}\phi$ and $\mathcal{G}_{[a,b]}\phi = \neg\mathcal{F}_{[a,b]}\neg\phi$.

The interpretation over finite traces and the quantitative evaluation of spatial relations allows us to define quantitative semantics for temporal relations. We interpret a temporal formula ϕ over finite sequences of states. We denote a sequence of states from time t to $t+k$ as $s_{t:t+k} = s_t s_{t+1} \dots s_{t+k}$.

Definition 14 (Quantitative Semantics of Temporal Relations)
We define $\rho(s_{t:t+k}, \phi)$ as:

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

assuming $0 \le a \le b$ and $a \le k$.

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

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

By defining $\rho(s_{t:t+k}, \phi) \ge 0$ implies $s_{t:t+k} \models \phi$, we can map the quantitative semantics to qualitative semantics of classical approaches [67]. We denote the set of all traces where ϕ holds true as $L(\phi)$.

Remark 1 (Comparison to other temporal logics)
The syntax and semantics of temporal relations are in line with LTL_f [30] and TLTL [34], which express temporal properties over finite intervals of time. As spatial relations are real-valued functions, our language expresses a fragment of STL with robust semantics [68] over discrete intervals of time.

V. Monitoring and Planning with SpaTL

Using the quantitative semantics of SpaTiaL, we can directly monitor the satisfaction of a given specification on a sequence of states. This can be applied to video data with tracked objects, as we demonstrate in Sec. VI-B and Sec. VI-C. 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.

A. Online Monitoring

The spatial relations can be evaluated online with respect to the information about objects. The quantitative semantics allow one to obtain fine-grained information about how much a spatial relation is satisfied or violated. Let us demonstrate the monitoring on a simple example:

Example 1 (Monitoring the Motion of an Object)
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 $\phi_{\text{bp}} = (\mathcal{O}_{\text{cube}}\ \text{enclIn}\ \mathcal{O}_{\text{goal}})$.

Figure 5a illustrates the example and the motion of the object $\mathcal{O}^t_{\text{cube}}$ over time t. The computed quantitative semantics for the specification ϕ_{bp} are shown in Fig. 5b (blue graph). Values below zero correspond to violating while values greater or equal zero correspond to satisfying the specification. As soon as the object $\mathcal{O}_{\text{cube}}$ is not enclosed in the goal region any more, the specification is violated again. In Fig. 5b, we also show the satisfaction values when replacing the enclIn relation with other spatial relations. Since our spatial relations are based on distances, their values are in the same value range. This property allows us to evaluate the satisfaction of spatial relations when moving objects uniformly within a grid. Otherwise, one would need to create a non-uniform grid, which makes computationally more difficult to find the right direction, e.g., to push an object in.

B. Automaton-Based Monitoring and Online Planning

To facilitate SpaTiaL planning, we apply automaton-based methods from LTL. We translate a SpaTiaL formula into an LTL_f formula by abstracting spatial subformulae into atomic propositions. From there, we construct a *Deterministic Finite Automaton* (DFA) to represent the temporal structure. Every LTL_f formula can be represented by a DFA [30], [67] We

Fig. 5. 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.

Fig. 6. DFA for Ex. 2 where state 1 and state 2 are the initial and accepting states, respectively.

use DFAs to repeatedly plan the next action. Through the quantitative semantics, we translate the abstract high-level action into a formulation more suited for low-level control. We demonstrate how to solve the low-level problem through *gradient maps*, using the quantitative semantics of SpaTiaL. The overall planning procedure is demonstrated in Subsec. VI-D.

Definition 15 (Deterministic Finite Automaton)
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. A finite run *of a DFA on a trace $\tau = \tau_0\tau_1 \ldots \tau_n$, where $\tau_i \in \Sigma$, is the sequence of states $q_0 q_1 \ldots q_{n+1}$ 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 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

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

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 $m : AP \to, \mathcal{R}$, $m(goal) = r_1, m(collide) = r_2$.

We then 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 [69]. The resulting automaton for (1) is depicted in Figure 6.

1) Automaton-Based Monitoring: We use the automaton $A_{\phi_t} = (Q, \Sigma = 2^{AP}, \delta, q_0, F)$ to monitor the satisfaction of ϕ qualitatively. The process is summarized in Alg. 1. In each step i, we evaluate each spatial relation represented through AP and m, creating a symbol $\sigma_i = \{x \in AP \mid \rho(m(x), i) \geq 0\}$ (line 5). Then, we update the current state using the transition function δ (line 6). Since the automaton is deterministic and complete, there always exists exactly one matching transition for every possible evaluation. At any point, we determine the current acceptance of the property by checking if the current state is an accepting state.

Algorithm 1: Automaton-Based Monitoring

Data: SpaTiaL formula ϕ, $k \in \mathbb{N}$

1 create ϕ_t and m;
2 construct DFA $A_{\phi_t}(Q, \Sigma, \delta, q_0, F)$;
3 $q_c = q_0$; `// set current state to initial state`
4 **for** $i \in \{0, \dots, k\} \subseteq \mathbb{N}$ **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 **return** $q_c \in F$;

2) Automaton-Based Planning: For planning, we additionally require no usage of the next-operator $\mathcal{X}$. Excluding $\mathcal{X}$ does not impair our ability to express interesting properties. This is common in relevant works, see [70] and the references therein. By omitting usage of bounded temporal operators and $\mathcal{X}$, we obtain a *stutter-insensitive* temporal property.

Lemma 1 (Stutter-Insensitive Properties)
[67], [71]. Every stutter-insensitive LTL$_f$ formula can be translated into a DFA $A = (Q, \Sigma = 2^{AP}, \delta, q_i, F)$ where for every state $q \in Q$, there exists a self-loop *transition such that $q = \delta(q, \sigma)$ for some $\sigma \in \Sigma$.*

Utilizing the structure of A_{ϕ_t}, 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. The procedure is outlined in Alg. 2. First, we construct A_{ϕ_t} and keep track of the current state exactly as in Alg. 1 (up to line 6). By finding the shortest path to an accepting state from the current state, 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

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

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

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

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 [72] without the domain- and problem-specific constraints and pruning rules.

Remark 2 (Pruning the DFA)
Some symbols σ and therefore transitions might be infeasible, 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 pruning of infeasible transitions might be done through specific domain knowledge, depending on the application. In the following experiments, we employ online pruning of edges that turn out to be infeasible through Lemma 2.

Algorithm 2: 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 (see Rem. 2);
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 $q_1 q_2 \dots q_n \in Q^*$ 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 $\quad \Sigma_p = \{\sigma \in \Sigma \mid \delta(q_c) = q_{next}\}$;
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 to satisfy (see Lemma 2);

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

Figuratively, Lemma 2 aims at satisfying any symbol that triggers a planned transition while avoiding triggering any unwanted transitions. In general, solving Lemma 2 depends on the robot's dynamics. However, by abstracting between high-level planning and low-level control in SpaTiaL, we allow users to apply their own methods to execute the planned next action. We demonstrate a solutions 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 16 (Gradient Maps)
Given Σ_p and Σ_c, an object $\mathcal{O}$, and a point $p \in \mathbb{R}^2$, we write $\tilde{\mathcal{O}}$ 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 $\tilde{\mathcal{O}}$.

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

Gradient maps are a uniform sampling approach in the task-space of a selected object to determine the gradient of

Fig. 7. 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.

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 [73].

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 Figure 7. 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 Alg. 2 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 Subsec. VI-D.

VI. Experiments

To demonstrate the capabilities of SpaTiaL, we designed three different experiments that are based on real recorded data and simulations. Each experiment focuses on different aspects of SpaTiaL (see contributions in Sec. I-B):

1) in Sec. VI-B, we demonstrate how SpaTiaL can be used to monitor object-centric robotics tasks such as pushing and pick-and-place tasks. This experiment highlights the creation of specifications, the use of quantitative semantics, and dealing with uncertainties;
2) in Sec. VI-C, we use SpaTiaL to monitor social distancing of agents in the Stanford drone dataset. This experiment highlights the usage of temporal properties with time bounds;
3) in Sec. VI-D, we show how 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.

A. Implementation

We implemented a SpaTiaL parser and interpreter in Python 3, using Lark [74] to parse the logic, Shapely [75] for geometric computations and MONA [69] with LTLf2DFA [76] 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 methods to parse SpaTiaL specifications, define objects through sets of convex polygons and then automatically interpret the formula to obtain satisfaction values, as well as DFA planning functionalities. An overview over the functionality is available at https://kth-rpl-planiacs.github.io/SpaTiaL/. The full code for all experiments, Including gradient map computation and simulation setup is also available. The prototypical simulations of Subsec. VI-D are implemented in PyBullet [77].

B. Monitoring Robotics Tasks

The evaluation of spatial relations is computationally efficient and enables the use of SpaTiaL in real time 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 [78]. The computation times per frame of both experiments can be found in Tab. I 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.

a) 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. 8). We placed 2 red, 2 green, and 2 blue blocks in the scene and specify additional phantom objects that serve as goal areas. The considered specification requires the human to push the red and green blocks to their corresponding goal regions *redGoal* and *greenGoal*, respectively. At the same time, the human needs to push the blocks through a passage that is formed

Fig. 8. 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. 9. Tree representation of the parsed specification ϕ_{blocks}. The recursive computation of the quantitative semantics allows us to also monitor individual subtrees.

by the two blue blocks. This specification ϕ_{blocks} is encoded using the eventually and always operators as shown in Tab. II.

Figure 8 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 scene, the human didn't move the red block that is further away from its goal region. As described in Sec. V, one can also monitor subparts of the specification for more fine granular information. Figure 9 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 graph in Fig. 8b).

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 remaining red block is succcessfully pushed into its goal region (see satisfaction graph of the red goal in Fig. 8b). Finally, the specification ϕ_{blocks} is satisfied at $t = 50.1s$ with a value of 15.35.

TABLE I
MEAN AND STANDARD DEVIATION OF THE COMPUTATION TIME PER FRAME IN OUR MONITORING EXPERIMENTS OVER FIVE RUNS.

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

b) 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 in the plate. This requirement is encoded in the specification ϕ_{pnp} using the until operator in Tab. II. 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 ϵ to enlarge an object due to measurement 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.

Figures 10a to 10e 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. 10f 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 graph in Fig. 10f).

C. Monitoring agents in the environment

SpaTiaL can also be used to monitor agents in the environment. Bounded temporal operators allow one to specify that

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

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

Fig. 10. Results of our experiment in which we monitor a pick and place task. (a) to (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.

(a) (b) (c) (d)

Fig. 11. Excerpts of our experiment where we monitor which agents in the environment respect social distancing to other objects. The images show the frame 9000 (5min) 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.

Fig. 12. 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.

certain spatial relations need to hold or not hold within a given time interval. To demonstrate the bounded time operators and the nesting of temporal operators, we choose a surveillance application in which a drone is recording an environment. In our case, we focus on detecting violations of social distancing using the Stanford Drone dataset [79]. We select the bookstore data where a drone is hovering and recording the bounding boxes of pedestrians, cyclists and other objects.

We look at 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:

(ϕ_1) $(\mathcal{G}\ \text{close} \rightarrow (\mathcal{G}_{[30,60]} \neg \text{close}),$

(ϕ_2) $(\mathcal{G}\ \text{close} \rightarrow (\mathcal{G}_{[90,180]} \neg \text{close}),$

(ϕ_3) $\big(\mathcal{G}\ \text{close} \rightarrow (\mathcal{F}_{[0,60]}\neg\text{close})\big)$,
(ϕ_4) $\big(\mathcal{G}\ \text{close} \rightarrow (\mathcal{F}_{[0,150]}\neg\text{close})\big)$,
where $\text{close} \in \{\text{close}_1, \text{close}_2\}$. 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 seconds or 3 to 6 seconds 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 1 second or 5 seconds after contact.

Figure 11 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 III provides a more detailed analysis of the number of satisfying or violating objects. Finally, Fig. 12 shows the satisfaction value for the object with ID 224 over time.

D. Task and Motion Planning

In the following two experiments, we demonstrate the online planning and monitoring technique outlined in Subsec. V-B. The first experiment features a robot arm pushing colored blocks and emphasizes the individual steps of DFA planning and using the gradient maps to find continuous paths to push objects along. The second experiment features a robot arm with a more complex pick-and-place specification, requiring the robot to arrange a table.

a) Pushing Task and Motion Planning: Consider a robot arm mounted on a plane surface as depicted in Figure 13. Exactly as in Ex. 3, three colored blocks are situated in a rectangular workspace. Blocks are defined through their shape, color, position and orientation. The robot is required to push the blocks around to fulfill the SpaTiaL specification:

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

where r, g and b stand for red, green and blue, 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 Subsec. V-B. The resulting automaton is depicted in Figure 14, with the following mapping from atomic propositions to spatial subformulae:

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

Fig. 13. Simulator environment for the pushing TAMP experiment.

Given the initial configuration of objects depicted in Figure 13 and Figure 15, 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 unfeasible and pruned. The next (and only) path to an accepting state is $3 \rightarrow 4 \rightarrow 5$.

Figure 15(a) 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 [80]. 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. 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 Figure 15(b). After execution, the specification is satisfied and the algorithm terminates.

b) 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 II. 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.

Fig. 14. 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.

(a)

(b)

Fig. 15. In the pushing task experiment, the robot has to satisfy a given specification by pushing the blocks around the workspace.

Since we monitor online, our planning method computes an alternative next step in case of an execution failure like tipping over neighbouring objects or losing grasp of the current object. An exemplary execution progression is depicted in Figure 16. 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.

VII. Conclusions

We presented SpaTiaL, a spatio-temporal framework that unifies the specification, monitoring, and planning for various robotic tasks. SpaTiaL provides various spatial relations between objects, such as an object is left of or overlaps with another object. These relation resemble nature language which increases the usability by non-expert users [59], [60]. The spatial relations operate in continuous space and provide quantitative semantics based on signed distances. Signed distances have the advantage that violation or satisfaction values are comparable in their value range, enabling uniform sampling of gradient maps. Moreover, objects can be defined as arbitrary sets of convex polygons and by enlarging them, we can also consider measurement uncertainties.

Spatial relations can be composed with temporal operators to specify various temporal patterns, e.g., constraints or sequences of configurations. Temporal specifications can also be nested and bounded with time intervals. In various experiments, we showed that SpaTiaL can be used to monitor tasks in real-time and for online planning. For the latter, we can decompose a specification into temporal and spatial abstractions. This decomposition has the advantage that high-level and low-level control can be treated separately. SpaTiaL is open-source and can be modified and extended by anyone, e.g., by composing new spatial relations out of existing ones.

In the future, we would like to add more spatial relations and support for computations in 3D spaces. We will expand the planning algorithm by providing general pruning rules for infeasible DFA transitions. Lastly, we will improve our current solution for finding a motion plan to fulfill the desired next DFA transition (see Lemma 2).

Acknowledgments

This research has been carried out as part of the Vinnova Competence Center for Trustworthy Edge Computing Systems and Applications at KTH Royal Institute of Technology and partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. We thank Ioanna Mitsioni, Fernando S. Barbosa, and Alexis Linard for their feedback.

References

[1] C. Menghi, C. Tsigkanos, P. Pelliccione, C. Ghezzi, and T. Berger, "Specification patterns for robotic missions," *IEEE Transactions on Software Engineering*, 2019.

[2] A. Billard and D. Kragic, "Trends and challenges in robot manipulation," *Science*, vol. 364, no. 6446, 2019.

[3] R. E. Fikes, P. E. Hart, and N. J. Nilsson, "Learning and executing generalized robot plans," *Artificial intelligence*, vol. 3, pp. 251–288, 1972.

[4] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "PDDL-the planning domain definition language," Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational, Tech. Rep., 1998.

[5] M. Fox and D. Long, "PDDL2.1: An extension to PDDL for expressing temporal planning domains," *Journal of artificial intelligence research*, vol. 20, pp. 61–124, 2003.

[6] H. Kress-Gazit, M. Lahijanian, and V. Raman, "Synthesis for robots: Guarantees and feedback for robot behavior," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 211–236, 2018.

Fig. 16. 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.

TABLE II
CONSIDERED SPECIFICATIONS IN THE EXPERIMENTS OF SEC. VI-B AND SEC. VI-D.

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

TABLE III
EVALUATION RESULTS OF THE FOUR SOCIAL DISTANCING SPECIFICATIONS ϕ_1 TO ϕ_4 FOR THE TWO TOO CLOSE DEFINITIONS.

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

[7] E. Karpas and D. Magazzeni, "Automated planning for robotics," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 417–439, 2020.

[8] T. Bylander, "The computational complexity of propositional strips planning," *Artificial Intelligence*, vol. 69, no. 1-2, pp. 165–204, 1994.

[9] S. Jiménez, T. De La Rosa, S. Fernández, F. Fernández, and D. Borrajo, "A review of machine learning for automated planning," *The Knowledge Engineering Review*, vol. 27, no. 4, pp. 433–467, 2012.

[10] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "STRIPS planning in infinite domains," *arXiv preprint arXiv:1701.00287*, 2017.

[11] D. Aineto, S. Jiménez, and E. Onaindia, "Learning STRIPS action models with classical planning," in *Proc. of the Int. Conf. on Automated Planning and Scheduling*, vol. 28, no. 1, 2018.

[12] E. P. Pednault, "ADL: Exploring the middle ground between STRIPS and the situation calculus." in *Proc. of the Int. Conf. on Principles of Knowledge Representaion and Reasoning*, 1989, pp. 324–332.

[13] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "PDDLStream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning," in *Proc. of the Int. Conf. on Automated Planning and Scheduling*, vol. 30, no. 1, 2020, pp. 440–448.

[14] M. Cashmore, M. Fox, D. Long, D. Magazzeni, B. Ridder, A. Carrera, N. Palomeras, N. Hurtos, and M. Carreras, "ROSplan: Planning in

the robot operating system," in *Proc. of the Int. Conf. on Automated Planning and Scheduling*, vol. 25, no. 1, 2015.

[15] B. Finkemeyer, T. Kröger, and F. M. Wahl, "Executing assembly tasks specified by manipulation primitive nets," *Advanced Robotics*, vol. 19, no. 5, pp. 591–611, 2005.

[16] T. Kröger, B. Finkemeyer, and F. M. Wahl, "Manipulation primitives—a universal interface between sensor-based motion control and robot programming," in *Robotic systems for handling and assembly*, 2010, pp. 293–313.

[17] C. Pek, A. Muxfeldt, and D. Kubus, "Simplifying synchronization in cooperative robot tasks-an enhancement of the manipulation primitive paradigm," in *Proc. of the IEEE Int. Conf. on Emerging Technologies and Factory Automation*, 2016, pp. 1–8.

[18] R. Ghzouli, T. Berger, E. B. Johnsen, S. Dragule, and A. Wasowski, "Behavior trees in action: a study of robotics applications," in *Proc. of the ACM SIGPLAN Int. Conf, on Software Language Engineering*, 2020, pp. 196–209.

[19] P. Ögren, "Convergence analysis of hybrid control systems in the form of backward chained behavior trees," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6073–6080, 2020.

[20] C. I. Sprague, Ö. Özkahraman, A. Munafo, R. Marlow, A. Phillips, and P. Ögren, "Improving the modularity of AUV control systems using behaviour trees," in *Proc. of the IEEE/OES Autonomous Underwater Vehicle Workshop*, 2018, pp. 1–6.

[21] Z. Zhao, Z. Zhou, M. Park, and Y. Zhao, "SyDeBO: Symbolic-decision-embedded bilevel optimization for long-horizon manipulation in dynamic environments," *arXiv preprint arXiv:2010.11078*, 2020.

[22] T. Silver, R. Chitnis, J. Tenenbaum, L. P. Kaelbling, and T. Lozano-Pérez, "Learning symbolic operators for task and motion planning," *arXiv preprint arXiv:2103.00589*, 2021.

[23] J. Loula, K. Allen, T. Silver, and J. Tenenbaum, "Learning constraint-based planning models from demonstrations," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2020, pp. 5410–5416.

[24] M. Toussaint, "Logic-geometric programming: An optimization-based approach to combined task and motion planning," in *Proc. of the Int. Joint Conf. on Artificial Intelligence*, 2015.

[25] M. A. Toussaint, K. R. Allen, K. A. Smith, and J. B. Tenenbaum, "Differentiable physics and stable modes for tool-use and manipulation planning," in *Robotics: Science and Systems Foundation*, 2018.

[26] T. Migimatsu and J. Bohg, "Object-centric task and motion planning in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 844–851, 2020.

[27] A. Pnueli, "The temporal logic of programs," in *Proc. of the Annual Symposium on Foundations of Computer Science*. IEEE, 1977, pp. 46–57.

[28] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE transactions on robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.

[29] C. Finucane, G. Jing, and H. Kress-Gazit, "LTLMoP: Experimenting with language, temporal logic and robot control," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*. IEEE, 2010, pp. 1988–1993.

[30] G. De Giacomo and M. Y. Vardi, "Linear temporal logic and linear dynamic logic on finite traces," in *Proc. of the Int. Conf. on Artificial Intelligence*. Association for Computing Machinery, 2013, pp. 854–860.

[31] O. Kupferman and M. Y. Vardi, "Model checking of safety properties," *Formal Methods in System Design*, vol. 19, no. 3, pp. 291–314, 2001.

[32] B. Lacerda, D. Parker, and N. Hawes, "Optimal and dynamic planning for markov decision processes with co-safe LTL specifications," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*. IEEE, 2014, pp. 1511–1516.

[33] P. Schillinger, M. Bürger, and D. V. Dimarogonas, "Decomposition of finite LTL specifications for efficient multi-agent planning," in *Distributed Autonomous Robotic Systems*, 2018, pp. 253–267.

[34] X. Li, C.-I. Vasile, and C. Belta, "Reinforcement learning with temporal logic rewards," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2017, pp. 3834–3839.

[35] C.-I. Vasile, J. Tumova, S. Karaman, C. Belta, and D. Rus, "Minimum-violation scLTL motion planning for mobility-on-demand," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*. IEEE, 2017, pp. 1481–1488.

[36] J. Tumova, G. C. Hall, S. Karaman, E. Frazzoli, and D. Rus, "Least-violating control strategy synthesis with safety rules," in *Proc. of the Int. Conf. on Hybrid systems: computation and control*, 2013, pp. 1–10.

[37] R. Koymans, "Specifying real-time properties with metric temporal logic," *Real-time systems*, vol. 2, no. 4, pp. 255–299, 1990.

[38] R. Alur, T. Feder, and T. A. Henzinger, "The benefits of relaxing punctuality," *Journal of the ACM*, vol. 43, no. 1, pp. 116–146, 1996.

[39] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *Proc. of the Int. Conf. on Formal Modeling and Analysis of Timed Systems*, 2010, pp. 92–106.

[40] L. Lindemann and D. V. Dimarogonas, "Control barrier functions for signal temporal logic tasks," *IEEE control systems letters*, vol. 3, no. 1, pp. 96–101, 2018.

[41] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications," in *Proc. of the IEEE Conf. on Decision and Control*, 2014, pp. 81–87.

[42] F. S. Barbosa, D. Duberg, P. Jensfelt, and J. Tumova, "Guiding autonomous exploration with signal temporal logic," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3332–3339, 2019.

[43] L. Lindemann and D. V. Dimarogonas, "Robust motion planning employing signal temporal logic," in *Proc. of the American Control Conf.*, 2017, pp. 2950–2955.

[44] C. Menghi, C. Tsigkanos, T. Berger, and P. Pelliccione, "PsALM: specification of dependable robotic missions," in *Proc. of the IEEE/ACM Int. Conf. on Software Engineering*. IEEE, 2019, pp. 99–102.

[45] N. Correll, K. E. Bekris, D. Berenson, O. Brock, A. Causo, K. Hauser, K. Okada, A. Rodriguez, J. M. Romano, and P. R. Wurman, "Analysis and observations from the first amazon picking challenge," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 1, pp. 172–188, 2016.

[46] K. Ramirez-Amaro, E.-S. Kim, J. Kim, B.-T. Zhang, M. Beetz, and G. Cheng, "Enhancing human action recognition through spatio-temporal feature learning and semantic rules," in *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2013, pp. 456–461.

[47] K. Ramirez-Amaro, M. Beetz, and G. Cheng, "Transferring skills to humanoid robots by extracting semantic representations from observations of human activities," *Artificial Intelligence*, vol. 247, pp. 95–118, 2017.

[48] M. Diehl, C. Paxton, and K. Ramirez-Amaro, "Automated generation of robotic planning domains from observations," *arXiv preprint arXiv:2105.13604*, 2021.

[49] G. Izatt and R. Tedrake, "Generative modeling of environments with scene grammars and variational inference," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2020, pp. 6891–6897.

[50] F. Paus, T. Huang, and T. Asfour, "Predicting pushing action effects on spatial object relations by learning internal prediction models," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2020, pp. 10 584–10 590.

[51] L. Manuelli, W. Gao, P. Florence, and R. Tedrake, "kpam: Keypoint affordances for category-level robotic manipulation," *arXiv preprint arXiv:1903.06684*, 2019.

[52] P. Jund, A. Eitel, N. Abdo, and W. Burgard, "Optimization beyond the convolution: Generalizing spatial relations with end-to-end metric learning," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2018, pp. 4510–4516.

[53] S. Guadarrama, L. Riano, D. Golland, D. Go, Y. Jia, D. Klein, P. Abbeel, T. Darrell, *et al.*, "Grounding spatial relations for human-robot interaction," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2013, pp. 1640–1647.

[54] R. Kartmann, F. Paus, M. Grotz, and T. Asfour, "Extraction of physically plausible support relations to predict and validate manipulation action effects," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3991–3998, 2018.

[55] R. Mojtahedzadeh, A. Bouguerra, E. Schaffernicht, and A. J. Lilienthal, "Support relation analysis and decision making for safe robotic manipulation tasks," *Robotics and Autonomous Systems*, vol. 71, pp. 99–117, 2015.

[56] S. Panda, A. A. Hafez, and C. Jawahar, "Single and multiple view support order prediction in clutter for manipulation," *Journal of Intelligent & Robotic Systems*, vol. 83, no. 2, pp. 179–203, 2016.

[57] B. Rosman and S. Ramamoorthy, "Learning spatial relationships between objects," *The International Journal of Robotics Research*, vol. 30, no. 11, pp. 1328–1342, 2011.

[58] D. Driess, J.-S. Ha, R. Tedrake, and M. Toussaint, "Learning geometric reasoning and control for long-horizon tasks from visual input," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2021.

[59] M. Skubic, D. Perzanowski, S. Blisard, A. Schultz, W. Adams, M. Bugajska, and D. Brock, "Spatial language for human-robot dialogs," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 34, no. 2, pp. 154–167, 2004.

[60] M. Nicolescu, N. Arnold, J. Blankenburg, D. Feil-Seifer, M. Banisetty, Santosh, A. Palmer, and T. Monteverde, "Learning of complex-structured

tasks from verbal instruction," in *Proc. of the IEEE-RAS Int. Conf. on Humanoid Robots*, 2019, pp. 747–754.
[61] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.
[62] H. Oleynikova, A. Millane, Z. Taylor, E. Galceran, J. Nieto, and R. Siegwart, "Signed distance fields: A natural representation for both mapping and planning," in *RSS Workshop: Geometry and Beyond-Representations, Physics, and Scene Understanding for Robotics*, 2016.
[63] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "CHOMP: Covariant hamiltonian optimization for motion planning," *The Int. Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.
[64] C. Ericson, *Real-time collision detection*. CRC Press, 2004.
[65] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The Int. Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.
[66] J. F. Allen, "Maintaining knowledge about temporal intervals," *Communications of the ACM*, vol. 26, no. 11, pp. 832–843, 1983.
[67] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.
[68] A. Donzé, T. Ferrere, and O. Maler, "Efficient robust monitoring for stl," in *Int. Conf. on Computer Aided Verification*. Springer, 2013, pp. 264–279.
[69] J. G. Henriksen, J. Jensen, M. Jørgensen, N. Klarlund, R. Paige, T. Rauhe, and A. Sandholm, "Mona: Monadic second-order logic in practice," in *Int. Workshop on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 1995, pp. 89–110.
[70] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 53, no. 1, pp. 287–297, 2008.
[71] A. E. B. Salem, "Improving the model checking of stutter-invariant LTL properties," Ph.D. dissertation, Université Pierre et Marie Curie-Paris VI, 2014.
[72] Y. Kantaros, M. Malencia, V. Kumar, and G. J. Pappas, "Reactive temporal logic planning for multiple robots in unknown environments," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*. IEEE, 2020, pp. 11 479–11 485.
[73] T. Lozano-Perez, "Spatial planning: A configuration space approach," in *Autonomous robot vehicles*, 1990, pp. 259–271.
[74] "Lark Github repository," https://github.com/lark-parser/lark, accessed: 2021-07-20.
[75] "Shapely Github repository," https://github.com/Toblerity/Shapely, accessed: 2021-07-20.
[76] "LTLf2DFA Github repository," https://github.com/whitemech/LTLf2DFA, accessed: 2021-07-20.
[77] E. Coumans and Y. Bai, "PyBullet, a python module for physics simulation for games, robotics and machine learning," https://pybullet.org.
[78] J. Wang and E. Olson, "Apriltag 2: Efficient and robust fiducial detection," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2016, pp. 4193–4198.
[79] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese, "Learning social etiquette: Human trajectory understanding in crowded scenes," in *Proc. of the European conference on computer vision*. Springer, 2016, pp. 549–565.
[80] D. Harabor and A. Grastien, "Online graph pruning for pathfinding on grid maps," in *Proc. of the AAAI Conf. on Artificial Intelligence*, vol. 25, 2011.

Christian Pek is a postdoctoral researcher in the Division of Robotics, Perception and Learning at KTH Royal Institute of Technology. He is working on advanced robotic systems with a high degree of autonomy, robustness, and safety. Chris was a PhD student at the Technical University of Munich and a research assistant in the motion planning group at BMW Group. He graduated with the Master of Science degree in computer science and robotics from the Technical University of Braunschweig, Germany, and the University of Auckland, New Zealand.

Georg Friedrich Schuppe is 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 agents (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. Since 2020, he has been a Ph.D. student in the Division of Robotics, Perception and Learning at KTH Royal Institute of Technology. His current 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 the areas of 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.