Robotics and Autonomous Systems 145 (2021) 103866

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Adaptive heterogeneous multi-robot collaboration from formal task specifications

Philipp Schillinger [a,*], Sergio García [b], Alexandros Makris [c], Konstantinos Roditakis [c], Michalis Logothetis [d], Konstantinos Alevizos [d], Wei Ren [e], Pouria Tajvar [e], Patrizio Pelliccione [b,f], Antonis Argyros [c], Kostas J. Kyriakopoulos [d], Dimos V. Dimarogonas [e]

[a] Bosch Center for Artificial Intelligence, Renningen, Germany
[b] Chalmers | University of Gothenburg, Gothenburg, Sweden
[c] Institute of Computer Science, FORTH, Heraklion, Greece
[d] Control Systems Lab, Department of Mechanical Engineering, National Technical University of Athens, Greece
[e] School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden
[f] Gran Sasso Science Institute (GSSI), Italy

ARTICLE INFO

Article history:
Available online 17 August 2021

Keywords:
Robotics
Multi-robot
Temporal logic
HRI
Heterogeneous robots
Task decomposition
Task allocation
Abstraction

ABSTRACT

Efficiently coordinating different types of robots is an important enabler for many commercial and industrial automation tasks. Here, we present a distributed framework that enables a team of heterogeneous robots to dynamically generate actions from a common, user-defined goal specification. In particular, we discuss the integration of various robotic capabilities into a common task allocation and planning formalism, as well as the specification of expressive, temporally-extended goals by non-expert users. Models for task allocation and execution both consider non-deterministic outcomes of actions and thus, are suitable for a wide range of real-world tasks including formally specified reactions to online observations. One main focus of our paper is to evaluate the framework and its integration of software modules through a number of experiments. These experiments comprise industry-inspired scenarios as motivated by future real-world applications. Finally, we discuss the results and learnings for motivating practically relevant, future research questions.

1. Introduction

In the area of industrial automation, effectively coordinating a team of autonomous robots poses significant challenges in various aspects of the system [1]. This ranges from infrastructure topics, e.g., the definition of a communication protocol or controlling the individual robots, up to conceptual research topics, e.g., task decomposition and allocation.

A multi-robot system becomes particularly challenging if it involves a flexible number of heterogeneous robots, such as mobile platforms or manipulators by different manufacturers [2]. Consequently, suitable abstraction layers need to be introduced for establishing communication between the robots [3]. This includes to find the right level of abstraction for enabling meaningful collaboration, but requiring only sparse interactions.

At the same time, a human operator of the system needs to be able to specify the behavior of the robots [4,5]. With the increasing complexity of the underlying system, especially considering the variable number of heterogeneous agents, it becomes less practically feasible to explicitly program individual robots. Thus, it is required that the system accepts high-level goals in a format general enough to be processed by all robots. Based on such goals, the system needs to act autonomously.

Finally, when given a symbolic specification of the expected behavior, it remains computationally challenging to coordinate a team of robots efficiently. This task is even more difficult when there is significant uncertainty regarding the actions required by the robots, for example, due to uncertainty in action execution or uncertainty regarding future observations [6,7].

In this work, we build upon recent advances in multi-robot task planning under uncertainty and integrate diverse state-of-the-art robot capabilities into a comprehensive framework for automated task decomposition, allocation, and execution. As a basis, we formulate so-called *skill models* to encode the capabilities in one consistent formalism and present how to formulate them specifically for the considered example systems, see Fig. 1. In addition, we present the user-friendly formulation of task specifications in order to conveniently instruct the system and generate the desired behavior from the defined skills.

* Corresponding author.
E-mail address: philipp.schillinger@de.bosch.com (P. Schillinger).

https://doi.org/10.1016/j.robot.2021.103866
0921-8890/

Fig. 1. Our framework enables to combine different types of robots for automatically executing industry-motivated tasks.

The presented work is a result of the research project *Co4Robots*.[1] Motivated by the integration efforts for transferring research results to industrial use cases, the main focus of this paper is the presentation of a conceptual framework, both in terms of theoretical foundation and software implementation, that allows for automatically composing and adapting heterogeneous multi-robot behavior from a set of individual robot capabilities. Thus, this system focus of the paper aims particularly at providing a "bigger picture" for the employed methods and is followed by a number of case study experiments in application-inspired scenarios to discuss major findings and learnings.

1.1. Problem statement

As a motivating example for the sort of applications we consider, assume that a fleet of robots in a factory needs to repeatedly monitor a set of machines and supply them with resources whenever a need is determined. Some robots may only be able to determine a supply need, others can only do transportation. In general, each of these capabilities needs to be expressed in a common formalism and the robots need to allocate required tasks according to a specification given to the whole team. In this example, a particular challenge for task allocation results from the non-determinism of required supply transportation, of which the need only gets known during execution.

Motivated by such applications, we consider the following building blocks of our proposed framework. First, robot engineers implement basic, re-usable capabilities of the robots to abstract their functionality.

Problem 1 (*Skill Formulation*)**.** Define a layer of abstraction, called *skills*, that is general enough to describe various types of robotic capabilities (e.g., perception, navigation, manipulation) and that is specific enough to enable task planning as well as feedback during operation for task adaptation.

Second, industrial workers without robotics expertise need to define and parametrize application-specific tasks, irrespective from the available robots.

Problem 2 (*Task Specification*)**.** Define a set of parametrizable task templates that is expressive enough to specify common industrial automation tasks (e.g., transportation, assembly) in a formal way which enables the automatic synthesis of reactive multi-agent policies.

Based on this input, the core problem addressed in this paper is to construct a task model that uses the skills of all available robots to dynamically decompose, allocate, and execute tasks according to the goal specification. Particularly, we consider tasks that require a significant amount of online adaptation.

[1] EU H2020 Research and Innovation Programme, GA No. 731869. Further information can be found at www.co4robots.eu.

Problem 3 (*Cooperative Task Execution*)**.** Given a set of skills and a task specification, dynamically determine and execute suitable actions for all available heterogeneous robots in a distributed fashion so that the system cooperatively achieves the specified task goal.

Finally, the transfer of theoretical results to the implementation of a practical system is important. Consequently, we exemplarily use the framework to coordinate different robots and discuss the major findings during the realization of case study lab experiments which follow the motivated application scenarios.

1.2. Framework overview

Our main contribution is a complete framework that enables the use of existing robot capabilities for planning and reactively executing tasks by multiple robots as given by a single temporal logic goal specification under consideration of a stochastic environment. A conceptual overview of this proposed framework is illustrated in Fig. 2 and outlined in the following.

On the left side of Fig. 2, different skills can be defined. We illustrate the practical use of the framework by discussing the integration of several robot capabilities, summarized in Section 3. However, these are only examples and the skill formulation described in Section 4 enables the abstraction of such capabilities in our framework.

On the right side of Fig. 2, a task specification can be selected from a set of task-specific patterns and parametrized as intended, e.g., selecting a target for inspection or delivery. Details on the task specification are presented in Section 5. The specification is then given to all robots in the system.

Both the skills and the specification are combined to a task model as indicated in the middle of Fig. 2. While this task model includes all robots in the system, we propose a distributed formulation of this model where each robot only models a local subset of the complete system. This is described in Section 6. During execution, the robots can operate mostly decoupled from each other based on this model formulation.

2. Related work

There exists a wide range of frameworks for different robotic applications and specific use cases. To list some examples, the work of Perico et al. [8] proposes a multi-robot coordination framework for a robot soccer application. Sheng et al. [9] and Rozo et al. [10] consider human–robot collaborative manipulation and propose frameworks for learning motion from human demonstration, e.g., in the context of industrial assembly. Further examples are the frameworks described by Charalampous et al. [11] and by Hawes et al. [12] for social mobile robots, including perception and navigation capabilities. However, we consider here no specific type of robots as many use cases in industrial applications of robots require multiple different robotic capabilities including perception, manipulation, and navigation.

Another relevant concern is the practical implementation of a theoretical framework. While ROS [13] is widely used for realizing various robot systems, the implementation of a complex multi-agent system is still challenging. Malavolta et al. [14] analyzed a multitude of existing ROS packages to derive general guidelines on the implementation of robotic systems in ROS. In order to establish communication between multiple individual robot systems, approaches such as the ROS multi-master framework [15] or, for larger applications, a cloud infrastructure [16] can be used.

In the following, we specifically discuss work related to the problems formulated in Section 1.1 for coordinating among individual heterogeneous robotic systems to collaboratively achieve user-defined goals.

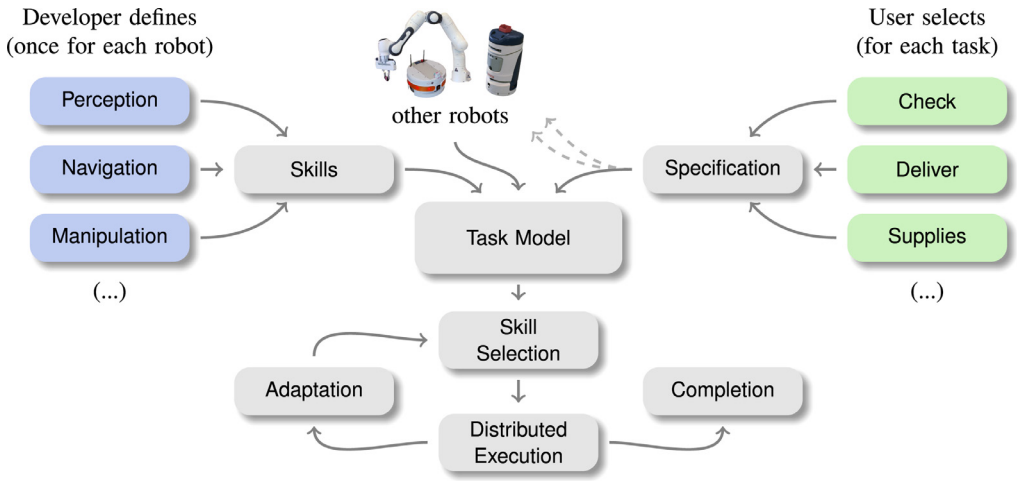


Fig. 2. Overview of the proposed structure. During development, skills are defined for all robots based on their capabilities. The term "developer" refers to robot engineers who implement basic capabilities for a specific hardware system. For executing a task, the user may compose a specification from combining suitable specification patterns. The term "user" refers to an application expert without deep robotics knowledge, but using the robots to achieve a task goal. Based on this input, the system is able to construct a task model and to autonomously execute required skills by repeatedly following an adaptive skill selection and execution loop. The skill selection includes a communication phase between the robots to adjust task allocation.

2.1. Skill formulation

The motivation behind formulating some sort of skill is to obtain a certain formalism for representing actions that can be executed by an agent. For example, finite state machines are classical high-level descriptions of a system's behavior and can be considered as abstract skills. However, there is a sensible trade-off between the generality of a skill and the mathematical operations that can be performed.

In the area of robotic manipulation, commonly used skill formalisms include TP-GMMs [17] and ProMPs [18], both to represent variations of trajectories to perform some sort of abstract actions. A skill formalism that focuses on the state representation, in particular on states corresponding to sensory inputs, has been proposed by [19] with the goal of enabling task planning.

In a broader context, Discrete Event Systems (DES) [20] provide a general formalism to express reactions to observed events as often required for robotic skills. MDP Options [21] are a useful formalism to denote skills in the context of discrete Markov Decision Processes (MDP), a specific type of DES to reason about future costs and system evolution [22]. In this direction, Amato et al. [23] use options as a layer of abstraction to reduce the complexity of policy planning. Similarly, Liu et al. [24] use options for denoting sub-tasks that can be composed by a robot for achieving temporal logic goals. In our paper, we follow this line of research and implicitly formulate skills as MDP Options for a high-level task model.

2.2. Task specification

Many mechanisms have been studied for specifying tasks to robotic applications, that is, specifying the sequence of actions a robotic team must perform. Among those mechanisms are Petri Nets [25] and Statecharts [26], solutions based on them provide formalism while being expressive. However, those solutions often require a step-by-step description of the robots' task. On the contrary, formal temporal logics, as Linear Temporal Logic (LTL) or Computation Tree Logic (CTL) allow a declarative specification of the tasks robots must perform [27,28]. The research community has studied these languages for robotic task specification as they can be automatically processed by planners, i.e., software components able to synthesize a set of actions or *a plan* to be achieved by the robots. An attempt to standardize a language for planning tasks is the Planning Domain Definition Language (PDDL) [29], which has been applied in the robotics domain [30,31].

Domain-Specific Languages (DSLs) [32]) are a type of language that is tailored to a domain and therefore allows the easy description of the user's concerns. In robotics, DSLs have been developed to cover different aspects of robotic applications development, as detailed in the survey of Nordmann et al. [33]. Among those aspects is mission and task specification, a field to which the community has expended considerable effort to conceive solutions that include features that support roboticists. Examples of such features are mechanisms to easily allow the robotic team to handle fault recovery or react to events [34,35].

2.3. Cooperative task execution

Various methods exist to generate the behavior of multiple robots from a given task goal. For example, Ulusoy et al. [36,37] study LTL-based synthesis for surveillance problems, but assume fully deterministic actions and optimize the repetitive behavior. In this line of work, the formalism of trace-closed languages [38] is used to coordinate the agents [39]. Similarly, in our previous work [4], we decompose a specification into independent parts to reduce the coordination effort between the agents. However, all above methods consider deterministic execution and centralized planning.

To incorporate uncertain outcomes of actions, MDPs [40] can be used to model the agents, e.g., for obtaining probabilistic guarantees [41–43] or to design robust policies [44–46]. For instance, Ding et al. [47] use constrained MDPs [48] to incorporate constraints from an LTL specification in an MDP. As investigated by Mosca et al. [49] and Hawes et al. [50], the topic of scheduling and congestion requires special attention when dealing with probabilistic execution. Finally, game theoretic analysis as proposed by Fu et al. [51] can be used to reason about suitable concurrent actions by multiple robots for achieving a specification. In summary, these methods all address some of the aspects we are considering here. However, they do not enable autonomous operation of a heterogeneous multi-robot team from a single temporal logic specification under consideration of stochastic execution, in particular together with integration of state-of-the-art robot capabilities.

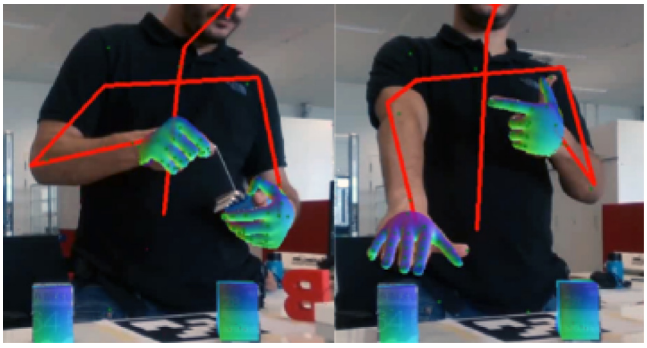


Fig. 3. Example result for the perception methods considered in our application. (left) Human, hands, and objects tracked during work procedure, but no reaction required. (right) Activation gesture detected and pointing towards one of the objects is recognized.

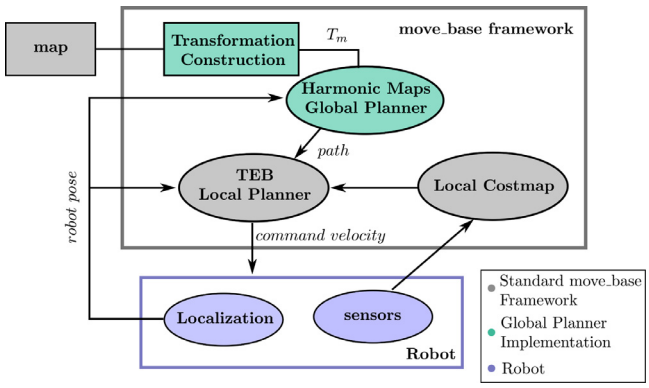


Fig. 4. move_base architecture overview as considered in this paper.

3. Preliminaries: Robot capabilities

A robotic system typically consists of a number of different capabilities, e.g., navigation or manipulation, that each contribute some functionality to the system. This is even more diverse when considering heterogeneous multi-robot systems. Consequently, it is an important first step to assess these capabilities when considering how to automatically compose them.

In the following, we describe the capabilities of a specific example system in the context of an industrial application scenario that is used throughout this paper. While the proposed framework is not limited to these capabilities, they are used to discuss in detail how different sorts of capabilities can be integrated into the framework. Similarly, when using the framework for any other robot or system, the typical development workflow would start with an assessment of existing capabilities, e.g., relevant ROS packages or implemented methods, and afterwards, abstract them to skills as covered in Section 4.

3.1. Perception capabilities

The role of the perception module is considered here to provide a number of different capabilities to the robots. Detection and tracking of multiple rigid objects is required to observe the state of the environment and identify objects of interest. Furthermore, human detection and tracking is considered for enabling the interaction of the system with a human worker. Beyond tracking the human as a whole, specifically hand gestures need to be recognized.

3.1.1. Multiple rigid objects detection and tracking

We adopt a system for multiple rigid object detection and tracking that operates using either RGB or RGB-D input. The considered method is able to handle multiple objects and performs efficiently under occlusions. More specifically, the algorithm employs a 3D model for each object. First, it learns the object's appearance by detecting local features [52] in a training image set and registering them onto the surface of the 3D object model.

To detect the object in an RGB image, the following steps are performed: (i) Local feature detection and matching with the features of the learned object template. (ii) Estimation of the pose using the 3D model to 2D image correspondences and a Random Sample Consensus (RANSAC) algorithm to deal with outliers [53]. (iii) Use of depth information (when available) to calculate a quality metric for the solution. This quality metric is particularly useful when dealing with situations with occlusions that might result in erroneous detections.

3.1.2. Human body pose estimation

For the human body pose estimation problem, we consider a method that is robust against occlusions, which are common in cluttered environments with multiple interacting agents. More precisely, we rely on a recent hybrid human 3D body pose estimation method that uses RGB-D input [54]. The method relies on a deep neural network to get an initial 2D body pose. Using depth information from the sensor, a set of 2D landmarks on the body are transformed in 3D. Then, a multiple hypothesis tracker uses the obtained 2D and 3D body landmarks to estimate the 3D body pose.

In order to safeguard from observation errors, each human pose hypothesis considered by the tracker is constructed using a gradient descent optimization scheme that is applied to a subset of the body landmarks. Landmark selection is driven by a set of geometric constraints and temporal continuity criteria. The resulting 3D poses are evaluated by an objective function that calculates densely the discrepancy between the 3D structure of the rendered 3D human body model and the actual depth observed by the sensor. See [54] for details on the method.

3.1.3. Gestures recognition

The gesture recognition module builds upon the detected pose of each human (which includes the detailed hand pose) to detect static body–hand gestures. A static gesture is detected by measuring the Euclidean distance between the template gesture pose and the detected pose of each frame. Temporal filtering ensures that the posture is detected in several consecutive frames before accepting it as valid, thus avoiding spurious detections.

For example, for the industrial application considered in this paper, we define a pointing gesture used to select appropriate objects for the performed task. When the gesture is detected, the pointing direction is used to determine the selected object. Example detection results are shown in Fig. 3.

3.2. Navigation capabilities

The navigation module provides capabilities for the mobile robots to navigate from one location in the floorplan to a different one, while avoiding any obstacle or restricted area. The navigation methodology consists of two main algorithms that are integrated with the ROS Navigation Stack (`move_base`).[2] Its architecture is depicted in Fig. 4.

[2] See https://wiki.ros.org/move_base.

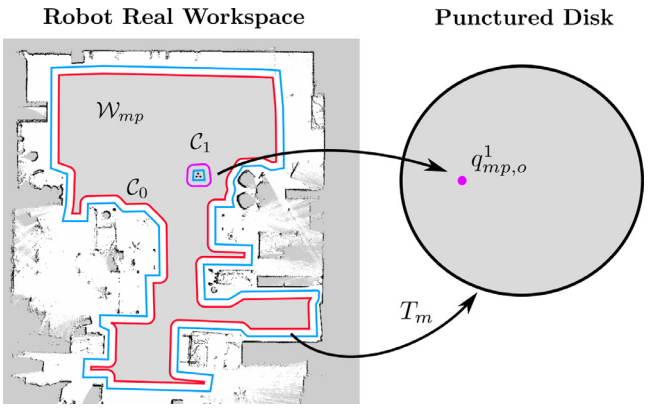


Fig. 5. Transformation of a real workspace to a punctured disk.

3.2.1. Global path planner

The first method is based on Harmonic Potential Fields [55] and used as a global planner. This technique is selected due to its reduced computational requirements and the ability to handle large and complex workspaces. Additionally, it guarantees collision-free navigation with the goal configuration being the sole stable equilibrium.

In short, the method constructs a transformation T_m which maps: (i) the robot's workspace $\mathcal{W}_{mp} \subset \mathbb{R}^2$ to the punctured Euclidean plane, (ii) the outer boundary $\mathcal{C}_0$ of the workspace to infinity, (iii) all obstacle boundaries $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_{N^{mp}_{obs}}$ to $q^i_{mp,o}, \forall i \in (1, 2, \ldots, N^{mp}_{obs}) \in \mathbb{R}^2$, and (iv) the goal positions $p^d_{mp} \in \mathbb{R}^2$ to distinct points $q^d_{mp} \in \mathbb{R}^2$ as shown in Fig. 5.

Refer to [55] for further details on the notation and the method. In summary, a feasible path can be computed efficiently in the resulting workspace $\mathcal{W}_{mp}$ that connects the current configuration of the robot with the desired one.

3.2.2. Local path planner

The Time Elastic Band approach [56] is adopted for the local planner. The initial path generated by the Harmonic Maps technique is optimized with respect to minimizing the trajectory execution time, obstacle avoidance, and compliance with kinodynamic constraints such as satisfying input and state constraints. Moreover, it complies with non-holonomic kinematic constraints by solving a sparse scalarized multi-objective optimization problem.

Let $p_{mp,k} = [x_k, y_k, \theta_k]^\top$ denote a robot pose at a discrete point in time k, where $x_k, y_k, \theta_k \in \mathbb{R}$ represent the planar position and orientation of the robot. Then, a discretized trajectory can be described as $\mathcal{T} = \{p_{mp,k} | k = 1, 2, \ldots, n\}$. The Time Elastic Band method augments the trajectory representation with time intervals $\tau = \{\Delta T_k \in \mathbb{R}_+ | k = 1, 2, \ldots, n-1\}$ for where each $\Delta T_k > 0$ denotes the time that is required for the robot to reach the $p_{mp,k+1}$ from $p_{mp,k}$. Consequently, we obtain the augmented trajectory representation $B := (\mathcal{T}, \tau)$.

Finally, it is required to solve an optimization problem to find body velocity commands for the mobile robot, minimizing time intervals $\sum_{k=1}^{n-1} \Delta T_k^2$ and considering constraints for kinematics, obstacle avoidance, and maximum velocity. The maximum execution time ΔT_{n-1} is defined by the user, while the initial $p_{mp,1} = p_{mp,c}$ and the final $p_{mp,n} = p_{mp,f}$ robot pose are set by the global path planner. We refer to [56] for further details on this optimization.

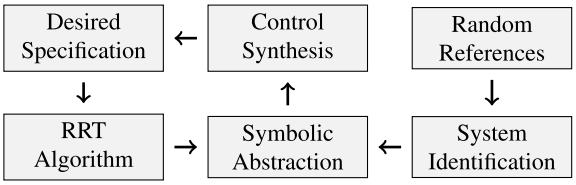


Fig. 6. Configuration of the control strategy for the manipulator.

3.3. Manipulation capabilities

The considered manipulation capabilities enable to move the end-effector of a robotic arm between target configurations of interest, e.g., to realize an assembly sequence or execute pick-and-place tasks. We follow a data-driven dynamic modeling approach to synthesize point-to-point motions and formulate an abstraction-based controller to guarantee safety during each motion. Initial collision-free paths are generated by the Rapidly-exploring Random Tree (RRT) algorithm, specifically RRT-Connect [57] implemented in the Moveit toolbox [58]. This control strategy is summarized in Fig. 6.

3.3.1. Dynamical modeling of the arm

To achieve the safe control synthesis, we first construct a discrete-time model as in [59]. That is, we model the arm dynamics as $x^+ \in Ax + Bu + c \oplus W$ with the state $x \in X$, the input $u \in U$, and the disturbance W. The model is constructed through sampling of command and position data-points along with a number of prescribed trajectories. We then model the arm dynamics as a piecewise-affine system with bounded disturbance by dividing the state space into various regions using hyperplanes and estimate the dynamics in each region by an affine function. See [59] for further details.

In a high-dimensional problem (i.e., 14 in case of a 6-DOF arm plus a 1-DOF gripper), using pre-defined regions is computationally intractable. Therefore, we include the region division as part of the optimization problem that is formulated as a Mixed-integer Linear Program [60].

3.3.2. Abstraction-based control design

Given the reference trajectory for the manipulator, an abstraction-based control design is used, see [61]. First, given an initial state $x_0 \in X_0 \subseteq X$, we choose a bounded region $\mathcal{S}_0 \subset X_0$ such that $x_0 \in \mathcal{S}_0$ and thus, $\mathcal{S}_0$ can be set as the initially chosen region. The intersection between the region $\mathcal{S}_0$ and the reference trajectory is treated as the local specification.

Next, starting from the initial region $\mathcal{S}_0$, we construct a local symbolic abstraction by approximating both state and input sets. Based on the local symbolic abstraction, a standard algorithm can be applied to determine the control inputs such that the manipulator moves to the local specification. Here, we use the common fixed-point algorithm [62].

Third, if the local specification is satisfied, we choose the next bounded region and determine the next local specification. Following the previous mechanism, the local symbolic abstraction is constructed for this chosen region, and then the local control strategy is established similarly. Iterating the above mechanism, the global control strategy is derived and the global specification can be achieved.

4. Skill formulation

A formalism of *skills* allows for representing the capabilities of each individual robotic system in a uniform way and is the basis for the theory behind our framework. In this section, we

define how a skill is expressed in our framework and discuss the specific representation of the different types of robot capabilities from Section 3 in separate sub-sections.

Mathematically, we model a skill as a labeled *Markov Decision Process* (MDP) given by the tuple

$$\mathcal{M} = (S, A, p, R, \lambda)$$

with a discrete state space S and a discrete action space A. Transition probabilities p are defined as the function $p\colon S\times A\times S \to [0, 1]$ where $p(s'|s, a)$ denotes the probability that action a in state s will result in the next state being s'. The reward function $R\colon S\times A \to \mathbb{R}$ denotes for a skill how desirable a specific action a is in a certain state s. Finally, the labeling function $\lambda\colon S \to 2^{\mathcal{AP}}$ assigns a subset of *atomic propositions* $\mathcal{AP}$ to each state $s \in S$ and exactly the propositions in $\lambda(s)$ are considered as being *true* at s.

The notion of these propositions $\mathcal{AP}$ is important for the skill formulation because they create a layer of abstraction. The labeling function λ defines for each skill which propositions hold at which states. Similarly, the task patterns introduced in the next section can be parametrized by these propositions. This creates the required connection between the high-level specification and the low-level skills.

To fully describe the probabilistic dynamics of a skill, a stochastic policy $\pi\colon S \times A \to [0, 1]$ defines the behavior of the robot that executes the skill and $\pi(a|s)$ denotes the probability of performing action a in state s. In particular, π is determined to drive the system dynamics towards a goal region $S_g \subseteq S$ in the state space while staying within a constrained, safe region $S_c \subseteq S$. For the problem of finding such a policy π, we refer to the following sub-sections for specific formulations of this problem in the domains of interest.

Together, $\mathcal{M}$ and π define an absorbing *Markov Chain* where the set of absorbing states is given as $S_g \cup \bar{S}_c$ with $\bar{S}_c = S \setminus S_c$, that is, the set of all goal states and violating states. For some initial state distribution $\hat{s}$, several properties of the system dynamics can be determined [63]. This includes the expected duration $\mathbb{E}[d(\hat{s})]$ until reaching an absorbing state, i.e., the expected *execution time* of the skill. The distribution over absorbing states after termination indicates the probability $p(S_g|\hat{s})$ of reaching a goal state, i.e., the *success probability* of the skill. These measures are particularly useful during task allocation for comparing different skill choices.

The choice of MDPs as skill formalism is motivated by the generality of MDPs to model skills from different domains like robotic navigation or perception, as well as by their ability to describe stochastic dynamics. Still, MDPs have some limitations that need to be considered when defining skills. Most relevant examples are that it may not be trivial to define a state space which fulfills the Markov assumption or to determine all transition probabilities.

Finally, note that skills are modeled for individual robots. On the one hand, this has the limitation that collaboration between multiple robots is only considered on a higher level when coordinating the skills between agents. But on the other hand, this enables to fully abstract away from the robot capabilities to a set of skills and the atomic propositions that these skills influence. Consequently, the robots can have different state and action spaces, including completely separated workspaces.

4.1. Perception skills

As a guideline for how to model perception skills, we describe in the following how to model the specific perception capabilities in Section 3.1 as a labeled MDP $\mathcal{M}$. We define states $s \in S$ with the discretized dimensions $human \in [0, 1]$ denoting the confidence of having detected a human pose, $gesture \in [0, 1]^n$ denoting the detection confidence of the n gestures, $object \in \mathbb{R}^{3m}$ denoting the estimated position of the m objects, $hand \in \mathbb{R}^3$ denoting the estimated hand position. Each action $a \in A$ is given by a set of activations of the perception modules, e.g., one action a can include running human body pose estimation, recognize some gesture i and track poses of objects j and k. The labeling function λ can be understood as an interpretation of the perception results.

The policy π then denotes which modules to activate at each moment and can be defined with respect to a goal region corresponding to a required detection result. For example, object detection is not required if the goal does not specify an object position. Similarly, it can be defined that a computationally expensive hand tracking is only activated after a human body pose has been detected with high confidence. In the remainder of this paper, the particular choice of π is not considered further and may, for example, be defined manually.

In summary, a perception skill defines propositions that abstract the perceived environment. In the experiments, we demonstrate their use for enabling a human co-worker to indicate a supply need at an assembly station, denoted with the proposition $Need_r$. All states $s \in S$ with a high confidence of having detected a pointing gesture and where an object r is close to the hand position are thus labeled with this proposition $Need_r$. Similarly, states with a low detection confidence are labeled with $Need_{unknown}$.

Finally, as a simple example of a different perception skill, consider the detection of objects in the LIDAR scan of a mobile robot. This skill is also used in the experiments further below. To decide whether an object is ready for transportation, the mobile robot repeatedly collects LIDAR scans of a (normally empty) region of interest and we model the state space to count the average number of detections within this region over a certain duration of time. The only action of the robot beyond termination is to obtain another scan. States above a certain threshold are labeled as positive detection, denoted by AS_{ready}.

4.2. Navigation skills

To model navigation capabilities, an intuitive choice for the state space is the location of the robot. For modeling the specific capability described in Section 3.2, we consider the location in a compact workspace $\mathcal{W}_{mp} \subset \mathbb{R}^3$ occupied by a set of inner obstacles $\mathcal{O} \subset \mathcal{W}_{mp}$. Consequently, we denote by $S = \mathcal{W}_{mp} \setminus \mathcal{O}$ the state space and each state $s = [x, y, \theta] \in S$ denotes the grid location and orientation of the robot in the map reference. Similarly, each $R_i \subseteq S$ denotes a region of interest in the workspace and is labeled accordingly, i.e., $i \in \lambda(s)$ for all $s \in R_i$.

Actions to navigate between two states are given by the set of kinematically feasible velocity commands and a deterministic policy $\pi(s)$ is given by the global planner. The local planner, in contrast, is considered transparently as a way of online recovery. Depending on the task, some of the above regions form the goal set $S_g \subseteq S$ of π while others are excluded from the allowed set $S_c \subseteq S$.

The resulting policy for skill execution can be further evaluated during task planning as follows. The modeled probability that the robot in configuration $s \in S$ reaches a desired goal state in $s_g \in S_g$ is $\pi(s_g|s) = 1$ considering that the $\mathcal{W}_{mp} \setminus \mathcal{O}$ is a topologically connected set. Moreover, the expected duration $\mathbb{E}[d(s_0)]$ of the skill with respect to an initial state $s_0 \in S$ is directly obtained from the global planner based on the current configuration $s_0 = [x, y, \theta]$ and the specification of the robot.

In summary, a navigation skill defines propositions that denote regions of interest in a workspace map. In the experiments, we define such regions of interest for example to mark assembly stations i as $atAS_i$.

4.3. Manipulation skills

To integrate the manipulation skills, we consider as example the capability summarized in Section 3.3 and formulate a skill where the states S correspond to joint values discretized by the abstraction-based control design. More specifically, we consider each cell in the abstraction as a state which corresponds to a bounded region in the continuous space. Given robot-specific assumptions on a bounded model error and disturbance, the control synthesis algorithm guarantees that the probability of reaching the subsequent planned state is 1.

In order to denote configurations of interest, e.g., *pick*, *place*, and *home* positions as required for realizing an assembly skill, states in the skill MDP are labeled accordingly. Planning a trajectory to the next required label then provides the initialization for the abstraction-based controller to move between the cells, i.e., states of the MDP.

The expected duration of transitions between two consecutive cells is equivalent to the sampling time of the controller and as such, constant for each cell. As a result, the expected duration of moving between two regions of interest is proportional to the number of cells generated by the symbolic controller.

In summary, a manipulation skill defines propositions that denote certain configurations of the robot, e.g., based on its end-effector pose or joint values. In the experiments, we use it to denote intermediate configurations of an assembly procedure including the proposition $Assembled_i$ to denote for the manipulator at station i that the assembly has been completed.

5. Task specification

Given a set of skills, it needs to be defined what should be achieved with these skills, i.e., the *task* to be executed. In our framework, a single, common task is given to the multi-robot team and is formally expressed as a *Linear Temporal Logic* (LTL) specification. Formally, an LTL specification ϕ classifies temporal sequences $\sigma : \mathbb{N} \to 2^{\mathcal{AP}}$ over atomic propositions $\mathcal{AP}$ as *satisfying* (fulfills the task), denoted by $\sigma \vDash \phi$, or *violating* (does not fulfill the task), denoted by $\sigma \nvDash \phi$.

Together with the notion of atomic propositions in labeled skill MDPs (see Section 4), LTL is used to specify goals and constraints for skill execution, see Fig. 7. This way, a task planner can guarantee that only skills, as well as actions within the skill MDPs, are selected that conform with the given specification. In other words, any temporal sequence of propositions resulting from the actions of the robots and the modeled dynamics of the environment should satisfy the given task formula.

For a complete background on LTL and formal definitions, we refer to [27,28]. In brief, LTL defines logical operators for boolean operations (operations on multiple propositions of a single time step) and for temporal operations (operations on single propositions of multiple time steps). Boolean operators include $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction), and $\Rightarrow$ (implication). Temporal operators include $\Diamond$ (eventually in a subsequent timestep) and $\Box$ (always in subsequent timesteps). These operators can be combined to form expressive tasks, e.g., $\Box\Diamond$ (always eventually) specifies repetitive tasks.

To specify the formula ϕ in a user-friendly way for non-experts, we build upon a catalog of *task patterns* [64,65]. The catalog maps recurrent robotic task specifications identified in the literature to well-known solutions with proven effectiveness expressed in temporal logic. Each of these patterns encodes a formula that is hidden to the user, who only needs to select a pattern and a set of parameters that are application-specific and situation-dependent. In this way we support the user by adding an abstraction layer that simplifies the task specification.

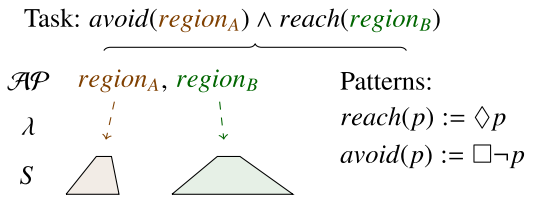


Fig. 7. The labeling function λ of a skill grounds atomic propositions $\mathcal{AP}$ to states $s \in S$ (bottom left). Similarly, task patterns are defined as parametrizable LTL formulas (bottom right). Then, tasks can be specified by selecting multiple patterns and parametrizing them with atomic propositions (top). The result is a complete LTL task specification over atomic propositions.

The patterns catalog is supported by PsALM (Patterns bAsed Mission specifier) [65], a toolchain that allows users to specify complex tasks by composing patterns. In particular, PsALM uses the "and" and "or" logical operators for such a composition of patterns. To provide users with a more flexible and powerful tool for mission specification that also composes the existing catalog of patterns, we developed a Domain-Specific Language (DSL). PROMISE[3] (simPle RObot MIssion SpEcification) [35,66] provides compositional operators that pave the usage of the proposed patterns in practical scenarios. The operators are inspired by Behavior Trees operators [67], a mathematical model of plan execution that allows composing tasks in a modular fashion through a set of nodes representing tasks and connections among them.

As a result, the overall task specification can be composed from an arbitrary number of generic or application-specific task patterns as indicated on the right side of Fig. 2. For illustration, we discuss in the following two example patterns that are used for the experiments in this paper.

5.1. "Check supplies" pattern

The task to check whether one type of supplies is required and to provide this kind of supplies is given by the LTL formula

$$\begin{aligned}\phi_s(i,r) = \Box\Diamond\big(&atAS_i \wedge \neg Need_{unknown} \wedge (\\ &Need_r \Rightarrow \Diamond Delivered_{i,r})\big) \wedge \\ \Box\big(&atAS_i \wedge \neg Need_{unknown} \wedge Need_r \Rightarrow \Box\neg(\\ &atAS_i \wedge \neg Need_r \wedge \neg Need_{unknown})\big)\end{aligned}$$

for some assembly station i and one type of supplies r.

The first part of the first line of this formula states that the need at the given assembly station i should not be unknown, without stating any need explicitly (as this cannot be influenced by the system). However, in the case that the need for resource r is detected, as specified in the second part of the first line of the formula, it is implied that the respective delivery needs to be performed eventually. Finally, the operator $\Box\Diamond$ states that this task should be performed repeatedly.

In addition, the remaining lines state an additional technicality, that is, a supply need cannot be fulfilled by simply checking the need again in the expectation that it would disappear. Otherwise, task planning could consider such a re-checking as a more efficient way of fulfilling the specification. Such a required technical addition to the task specification is likely not obvious to most users and thus, undermines the significant benefit of using expert-defined task patterns.

[3] See https://github.com/SergioGarG/PROMISE_implementation.

5.2. "Check and deliver" pattern

A delivery task of potentially available products is specified as the formula

$$\begin{aligned}\phi_d(i,f) = \Box\Diamond\big(&atAS_i \wedge \neg AS_{unknown} \wedge (\\ &AS_{ready} \Rightarrow \Diamond Delivered_{f,i})\big) \wedge \\ \Box\big(&atAS_i \wedge \neg AS_{unknown} \wedge AS_{ready} \Rightarrow \Box\neg(\\ &atAS_i \wedge \neg AS_{ready} \wedge \neg AS_{unknown})\big) \wedge \\ \Box\big(&atAS_i \wedge \neg AS_{unknown} \wedge \neg AS_{ready} \Rightarrow \Box\neg(\\ &atAS_i \wedge AS_{ready})\big)\end{aligned}$$

for assembly station i and the final delivery location f.

The structure of this formula is similar to the one above. It specifies that the assembly station status in terms of the product availability may not be unknown and in case the product is ready, it should be delivered to the final location. Furthermore, the additional technical statements of the formula in the other lines state that the availability of a product cannot change by simply re-checking the status.

This similarity is possible here because only the goal is specified, not the particular actions done by the robots. Thus, the formula is independent of the available robots. Any team of robots, for which the model defines a way to achieve the specified symbols in the formula, can be used to fulfill the task.

6. Cooperative task execution

The skills described in Section 4 and the task specification defined in Section 5 are used to form a so-called *Task Model* (recall Fig. 2). In order to semantically connect skill MDPs with the LTL task specification, we use the notion of *atomic propositions* $\mathcal{AP}$ as layer of abstraction. Considering a skill, the labeling function λ of the MDP associates states of the respective robot system with such propositions (recall Section 4). During runtime, a sequence of states as resulting from the controlled system dynamics of the respective skills introduces a symbolic sequence σ over propositions. At the same time, the LTL specification ϕ classifies whether the resulting σ is satisfying (recall Section 5).

Since the robots are heterogeneous, their sets of atomic propositions might be different. Following the semantics of the labeling function λ, undefined propositions are considered to be *false*. When propositions are shared by multiple robots, conditions over them can be fulfilled by any of these robots. This enables an abstraction from individual robots and allows for collaboration between the robots by fulfilling different parts of a specification.

In the following, we describe the distributed construction of the task model and how it is used by the robots to cooperatively follow their common task.

6.1. Distributed task model

As a basis for the task model construction, we employ the formalism of a so-called *Büchi Automaton* (BA) for representing a given task specification ϕ in an automaton format derived from the LTL formula, particularly suitable for model construction [27,28]. A BA is defined as the tuple

$$\mathcal{B} = (Q, Q_0, \alpha, \delta, F)$$

with virtual *progress* states Q towards task completion, a set of possible initial states $Q_0 \subseteq Q$, an alphabet of symbols α, a non-deterministic transition relation $\delta \subseteq Q \times \alpha \times Q$, and a set of final *accepting* states $F \subseteq Q$.

In particular, it is well-known [27] how to construct a BA $\mathcal{B}$ based on an LTL formula ϕ such that $\mathcal{B}$ and ϕ are equivalent in the following sense: For any sequence σ it holds that $\sigma \vDash \phi$ if and only if the respective transitions in $\mathcal{B}$ lead from an initial state in Q_0 to an accepting state in F. In the case that σ is infinite, F is visited infinitely often.

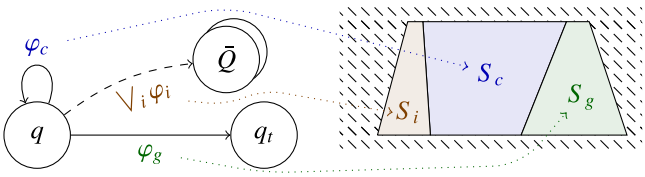


Fig. 8. Proposed hierarchical definition of a skill option. On the upper layer (left), an option is considered as a transition in the task automaton. On the lower layer (right), an option corresponds to a policy planning problem to the goal states of a skill.

Based on this notion, the *Task Model* for some task ϕ is defined as a *Semi-MDP* [21] given by the tuple

$$\mathcal{T} = (Q, O, p, d)$$

with the state space Q, a set of options O, transition probabilities $p\colon Q \times O \times Q \to [0,1]$ and probabilistic option durations $d\colon Q \times O \to \mathbb{R}_{>0}$ as cost function, i.e., rewards are given by the negative expected duration. This follows the definition proposed by Schillinger et al. [63].

In particular, the state space Q of $\mathcal{T}$ is identical to the state space of the BA $\mathcal{B}$ constructed for the task ϕ. Accordingly, each state $q \in Q$ can be considered as denoting some progress towards task completion and some states q are *accepting* states $q \in F$ in $\mathcal{B}$, i.e., denote successful task completion.

Similarly, the options O are chosen such that each $o \in O$ can be associated with a transition in δ of $\mathcal{B}$ as follows. For each transition $(q, \varphi_g, q_t) \in \delta$, we define a set of options $O_q^{q_t}$ and accordingly, $O = \bigcup_{q,\cdot,q_t \in \delta} O_q^{q_t}$. We construct one option $o \in O_q^{q_t}$ for each robot in the case that there exists a feasible policy π. Here, the policy π refers to a policy for the MDP of the robot composed of its defined skills.

A policy π is generally determined by constructing a constrained policy planning problem for the MDP. The above transition condition φ_g denotes the *goal* of π, and the condition φ_c of the self-loop transition $(q, \varphi_c, q) \in \delta$ denotes the *constraints* of π. Note here that each of these conditions φ implies a set of states $\{s \in S\colon \lambda(s) \vDash \varphi\}$ in $\mathcal{M}$. This relation is illustrated in Fig. 8. Consequently, the policy π is planned to reach an MDP state in the goal set while remaining within the constraints set.

Due to the probabilistic nature of skill execution, an option o is not guaranteed to reach its target progress state q_t. Instead, each o may result in one of multiple subsequent progress states q' with some probability $p(q'|q, o)$. In addition, the execution of o in progress state q requires a probabilistic amount of time $d(q, o)$. However, these measures can be estimated from evaluating the policy for a certain initial MDP state distribution [63].

Finally, the task model $\mathcal{T}$ is realized as a *distributed* model in the following sense. Each robot receives the common task ϕ and locally constructs the model under consideration of all options O that represent the robot itself. Consequently, each robot only knows a subset of O and in order to plan a team policy in the task model, the robots need to communicate and exchange the relevant information for selecting between alternative options or for taking transitions for which no own option exists.

6.2. Repetitive skill selection

Given the task model Semi-MDP $\mathcal{T}$ in which each robot can locally construct options for its individual skills, it remains to

determine a task policy $\mu: Q \rightarrow O$ for the team of robots to appropriately select the options $o \in O$ to execute at each progress state $q \in Q$.

We follow the sequential auctioning algorithm proposed in [63] to define such a policy μ online. This is motivated by the distributed nature of the system model, as well as the considerable environment dynamics that require frequent adjustment to online observations. In the following, the most relevant aspects are summarized, but refer to the original work for all details.

The robots follow multiple subsequent auction rounds and in each auction, they share bids of possible next options for them to execute, determined from their locally known skills and current states. Then, the auction determines the preferred allocation choice for the team across all shared bids. In particular, [63] describes a value calculated for each bid such that the auction selects the skill which approximately minimizes the overall task execution time and the approximation is improved over time by an online learning procedure.

The auction rounds are repeated until each robot is assigned at least one option to follow next. Afterwards, each robot executes the local policy of its assigned option either until termination, denoting the completion of the current sub-task, or until another robot requests the next auctioning procedure, e.g., due to the other robot completing its current sub-task.

To illustrate the skill selection procedure, consider the following corner cases. First, assuming that all robots are different and have skills with unique propositions, each robot would individually select among its skills that can be executed next. Second, in the case of a homogeneous team where each robot has the same skills, the first auction round would select the skill of the robot that promises shortest execution time based on the current states of the robots. If the task requires further skills, the same or other robots would be assigned these further skills in subsequent rounds. For repetitive tasks, multiple robots can be assigned to execute the same skill to already prepare future iterations of the task.

Worst-case complexity of a single auction round is $O(|Q|)$ for calculating the bids, occurring in the case of a fully connected task model. This is required in each auction round in parallel for each robot, leading to $O(N \cdot |Q|)$ for N robots because one assignment is decided in each round.

6.3. Adaptive collaboration

The dynamics of task execution are captured by the task model $\mathcal{T}$ and the skill selection procedure μ as follows. The team starts in an initial state $q \in Q_0$ and selects one option $o \in O$ for each robot to follow, according to their skills. Eventually, the option of one robot terminates, causing a transition from q to q' as given by the transition relation δ and new options for each robot are again determined by the sequential auctions. This procedure is repeated until $q' \in F$ and the task is completed or, for repetitive tasks, for any number of iterations.

In this context, collaboration between the robots is achieved by sharing a common task progress $q \in Q$ of $\mathcal{T}$ and each robot is able to cause a new transition in $\mathcal{T}$ based on the atomic propositions $\mathcal{AP}$ in its skills. The role of the auctions is the coordination of transitions that each robot works towards. Also, by repeating the auctions whenever q changes to a new q', the team of robots can adapt their allocated skills to the, potentially unexpected, new progress q'. As given by LTL, such a new progress state can imply different constraints or additional sub-tasks.

The distributed formulation of $\mathcal{T}$ has the benefit that only an incomplete representation of $\mathcal{T}$ needs to be constructed locally by each robot and a centralized planning phase can be avoided.

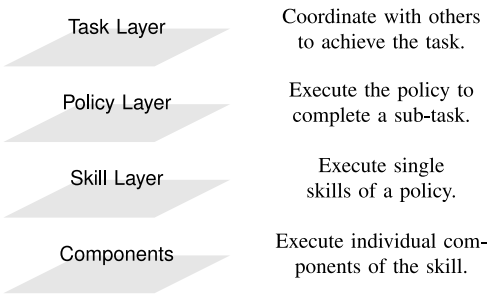


Fig. 9. Illustration of the layered software structure for task execution.

Furthermore, due to the employed auctioning procedure, the information exchange between the robots is reduced both in terms of data amount and frequency of initiation.

Considering the motivation of this paper, the distributed model enables that a robot can contribute regardless of the particular team composition. For example, a task specification might require both navigation and manipulation skills. Without knowing the skills of the rest of the team, a mobile robot can bid for solving the navigation tasks while at the same time a manipulator bids for the manipulation tasks. This form of collaboration is particularly useful when re-using existing skills instead of newly developing each application for a specific team of robots. Similarly, repetitive auctions allow the robots to adapt flexibly. If for example the mobile robot fails in the navigation task and reaches a different progress state in the task model than desired, the same or also a different robot might bid to execute a recovery task in accordance with the specification.

7. Case studies

In order to analyze how effectively our presented framework is suitable to address the motivating problems, we present in the following a number of experiments to resemble relevant use cases. This includes a presentation of the technical details and an illustration of the usage of the framework in practice.

7.1. Software description

For all the experiments, our software implementation of the framework uses ROS and relies on several standard ROS packages. Each robot runs a set of ROS nodes to implement the different capabilities, to control execution, and to coordinate with the rest of the robots. To prevent software conflicts and control the communication between the robots, each robot runs its own ROS master and uses a specific topic namespace for global communication. This essentially isolates each robot in terms of software and only permits the highest software layer to exchange information in a well-defined protocol.

The implementation consists of a layered structure, see Fig. 9, where the highest layer covers task decomposition, planning, and communication during the task auctions. As a result, it commands towards the lower layer a policy over robot skills that needs to be executed locally in order to follow the overall task specification. This policy is tracked on the second layer, which determines and commands the respectively next skill.

Each skill is interfaced as one FlexBE[4] [68] behavior and configured in the planning model. When executing a skill as determined by the policy, FlexBE runs the corresponding behavior and interprets the outcome after the skill terminated to provide

[4] See http://flexbe.github.io.

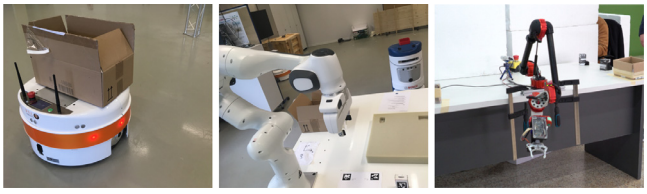


Fig. 10. Different types of robots used in the experiments. From left to right: (1) *TIAGo Base* by *PAL Robotics*, (2) *Panda* by *Franka Emika*, (3) *A-2085-06* by *Hebi Robotics*.

feedback about the result. The skill behavior interfaces with all the low-level components that realize the skill.

Due to the dynamic environment, execution needs frequent adjustments and recovery. This is realized by a feedback loop within each layer and, if recovery fails, it is escalated upwards to the respective next layer. For example, the navigation component can attempt to perform local obstacle avoidance to reach a goal. If this fails, the skill behavior may select a recovery waypoint within the scope of the current skill or adjust the motion parameters. If this fails as well, the policy executor receives an update and may select a different skill according to the planned policy, e.g., a substantially different waypoint or some recovery action. Finally, if this all fails, e.g., because a symbol used in the task specification is affected, communication between the robots can be initiated to re-allocate tasks or to incorporate additional requirements.

7.2. Hardware description

The experiments involve different robots as summarized in the following and shown in Fig. 10. For each robot, the required capabilities are implemented on the *Components* layer as described in Section 3 to integrate them with each other.

The *TIAGo Base*[5] by PAL Robotics is a mobile platform for automating indoor transportation. Here, we use two of these robots to execute delivery tasks and use their onboard sensors to detect objects in front of them. The *Panda*[6] robot by *Franka Emika* is a 7-DoF static manipulator that can be used at collaborative industrial assembly stations. Here, we use the robot to follow an assembly routine of an existing Bosch product and show its integration into the overall coordination system. To integrate the collaborative assembly station into the system for our experiments, we added an RGBD camera (Intel RealSense D415) to perform gesture recognition as required for human–robot interaction. The *A-2085-06* robotic arm[7] from Hebi Robotics is another static manipulator with 6 DoFs, here endowed with a custom end-effector including a gripper and an ATI Mini 45 force/torque sensor. Here, we use the robot to illustrate the integration of manipulation skills into the system.

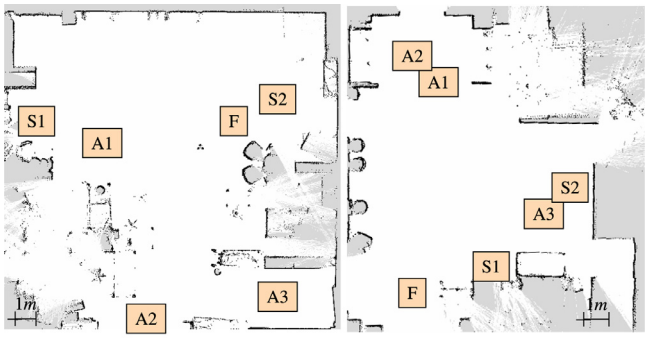
7.3. Workspace description

As initially motivated by the factory application use case (recall Section 1.1), we consider an industrial setting for conducting our experiments as given by the Robotics Lab at the Bosch Corporate Research Campus in Renningen, Germany. This location is suitable to reflect a factory-inspired environment including assembly stations for the transportation tasks and for human–robot collaboration. In addition, we repeated the experiments in the office environment of PAL Robotics in Barcelona, Spain, to show the transferability of the developed system.

[5] See http://pal-robotics.com/robots/tiago-base.

[6] See https://www.franka.de/panda.

[7] See https://www.hebirobotics.com/a-2085-06.



(a) Bosch Research Lab (b) PAL Robotics Office

Fig. 11. Maps of the experiment workspaces. Annotated for illustration are the relevant locations of interest as explained in the respective experiment descriptions.

In both cases, we used one of the mobile robots to record a SLAM map and annotated it with the respective locations of interest. In addition, virtual obstacles have been added manually to prevent the robots from leaving the workspace area. The resulting maps as used by the mobile robots are shown in Fig. 11, including the annotated locations of interest.

7.4. Experiments

Each of the following experiments is defined by skills modeled for the robots and task patterns to achieve. The experiments are sorted by increasing complexity and each one focuses on one specific aspect of the framework. The final experiment combines all previous aspects and can be considered as focusing on their integration.

Experiment I: Distribution of collaborative tasks

In the first scenario, two *TIAGo Base* mobile platforms operate in a workspace with specific regions of interest as depicted in Fig. 11. Repeatedly, the assembly stations denoted by $A1$, $A2$, and $A3$ should be checked if there are products ready for delivery. If a product is detected, the task includes to deliver it to the final location denoted by F. This task is given by the specification

$$\phi_1 = \phi_d(A1, F) \wedge \phi_d(A2, F) \wedge \phi_d(A3, F)$$

and we refer to Section 5.2 for the formulation of the selected task pattern $\phi_d(\cdot)$. See Fig. 12 for an illustration of this task.

Each of the two mobile robots is independently modeled as follows. The state space is given by a floor plan for navigation S_{map}, see background of Fig. 11, as well as the discrete state variables S_{carry} and S_{status_i}. The set

$$S_{carry} = \{\mathtt{n}, \mathtt{p}_{A1}, \mathtt{p}_{A2}, \mathtt{p}_{A3}\}$$

denotes carrying a product and $\mathtt{n}$ models carrying nothing while $\mathtt{p}_i$ denotes that one of the products from the respective assembly stations is carried. The sets

$$S_{status,i} = \{\mathtt{u}, \mathtt{b}, \mathtt{r}\}$$

for $i \in \{A1, A2, A3\}$ denote the status of the respective assembly station i as known to the robots where $\mathtt{u}$ denotes unknown, $\mathtt{b}$ denotes busy, and $\mathtt{r}$ denotes that a product is ready for pick-up.

The robots have a navigation skill, realized as described in Section 3.2 and abstracted as described in Section 4.2. In addition, each robot has three more skills: Two skills for loading a product at each of the assembly stations and for unloading a product at the delivery location, as well as one skill for determining the

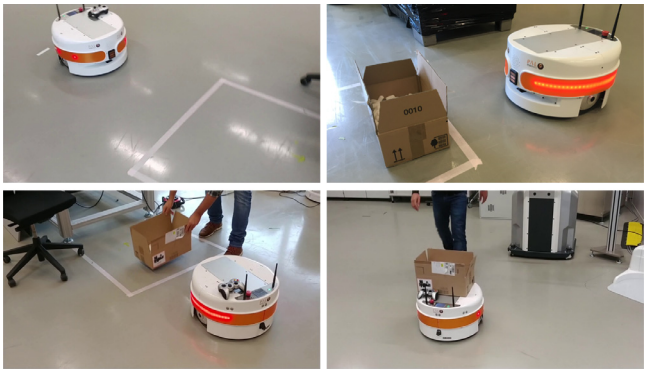


Fig. 12. Example situations during Experiment I. Shown top are cases when the robot did not (left) or did (right) observe a product for delivery. Shown bottom is the transportation process.

Fig. 13. Example situations during Experiment II. The assembly station follows an assembly routine (top left) and the human worker requests supplies when needed (top right). Shown bottom is the delivery of requested supplies.

status of an assembly station. Here, we assume that loading and unloading are done manually by the worker at the respective station. For recognizing whether a product can be delivered and whether loading has been done, we use the LIDAR scanner of the robots and implement the skill as simple perception skill following the example in Section 4.1.

In summary, this experiment indicates the collaboration and adaptability of two homogeneous robots in a simple delivery scenarios. It can be observed that the robots effectively allocate sub-tasks to check the individual assembly stations and, whenever a finished product is observed, consider transportation as another sub-task that needs to be allocated eventually. More details are discussed in Section 8.

Experiment II: Specification of human–robot interaction

For the second experiment, we consider heterogeneity and model the assembly station $A1$ in more detail. That is, we consider a human worker and a *Panda* static manipulator working collaboratively at the station. While the manipulator follows an external assembly routine, as would be the case when retro-fitting our framework to an existing factory, the human worker monitors the process and needs to request some supplies s_1 and s_2 whenever required. This is expressed by the task

$$\phi_2 = \phi_s(A1, s_1) \wedge \phi_s(A1, s_2)$$

and we refer to Section 5.1 for the formulation of the selected task pattern $\phi_s(\cdot)$. See Fig. 13 for an illustration of this task.

We extend the model of a mobile robot by two pick-up locations for supplies, marked $S1$ and $S2$ in Fig. 11. Also, the state variable S_{carry} is extended by the values $\{\mathtt{s}_1, \mathtt{s}_2\}$ to model the two types of supplies being carried. For loading and unloading of supplies, we re-use the previously defined skills.

In addition, we partially model the assembly station $A1$ as a third agent to demonstrate the integration of human–robot collaboration. The assembly station has one state variable

$$S_{need} = \{\mathtt{n}, \mathtt{s}_1, \mathtt{s}_2, \mathtt{u}\}$$

that denotes whether no supplies are required ($\mathtt{n}$), one of them is needed ($\mathtt{s}_1$ or $\mathtt{s}_2$), or the need is unknown ($\mathtt{u}$).

One perception skill is considered that enables the assembly station to determine the need if unknown. Such a need is indicated by the worker with a pointing gesture towards one of the two supply indicator objects or no gesture when no supplies are required. The skill is implemented as described in Section 3.1 and abstracted as discussed in Section 4.1.

This experiment illustrates the ability of the framework to consider a completely different sort of agent and use the combination of agents to realize the task. Since the mobile robots themselves cannot observe a supply need, the perception skill of the assembly station is used to transition in the shared task model to a state where the need is known, enabling in turn the mobile robots to bid for the respectively needed transportation.

Experiment III: Integration of assembly routines

The third experiment considers the explicit modeling of the assembly task, leading to a more fine-grained integration of the manipulator. We use the *A-2085-06* manipulator at station $A1$ and model an exemplary assembly routine for it. That is, we define a simple pick-and-place sequence and require that the sequence is completed before the supply need can be checked.

To let a user define such an additional task, we introduce a new task pattern together with implementing the new robot:

$$\phi_a(i) = \Box\Diamond \mathit{Assembled}_i \wedge \Box(\neg \mathit{Assembled}_i \Rightarrow \mathit{Need}_{\mathit{unknown}})$$

to denote that the supply need cannot be known before the assembly sequence at station i has been completed. If such a constraint would not be desired, the pattern $\phi'_a(i) = \Box\Diamond \mathit{Assembled}_i$ would be sufficient to only denote the repetitive completion of the sequence. Consequently, the task specification is given as

$$\phi_3 = \phi_s(A1, s_1) \wedge \phi_s(A1, s_2) \wedge \phi_a(A1)$$

which extends ϕ_2 by the new assembly sequence. See Fig. 14 for an illustration of this extension.

The assembly sequence is implemented by manipulation skills as described in Section 3.3 and integrated as described in Section 4.3. For determining the need for supplies, we still use the previous perception skill and can leave it unchanged.

Building upon the previous experiment, this one demonstrates how a new task pattern can be added to capture a newly defined skill. Based on the constraint specified in the new task, the observation of a supply need requires to first complete the assembly procedure and thus, demonstrates both manipulation and navigation tasks coordinated in the framework. This integration can be useful when no external assembly routine is required and enables a closer coordination of the agents.

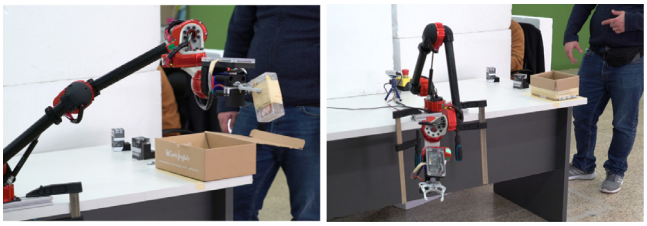


Fig. 14. Example situations during Experiment III. The assembly station is explicitly modeled by a pick-and-place routine (left) and only when done, the human worker may request supplies (right).

Experiment IV: Dynamic production assistance

In the final experiment, we combine the previous experiments and demonstrate how the proposed multi-robot framework enables the implementation of a robotic assistance system in production scenarios.

Intuitively, combining the previous tasks should be simple. Since the required skill implementations and individual robot models are already defined, formulation of a task that requires all of the previous capabilities is indeed just a concatenation of the respective task patterns:

$$\begin{aligned}\phi_4 = {} & \phi_d(A1, F) \wedge \phi_d(A2, F) \wedge \phi_d(A3, F) \\ & \wedge \phi_s(A1, s_1) \wedge \phi_s(A1, s_2) \\ & \wedge \phi_a(A1)\end{aligned}$$

Still, while minimal user effort is required to specify the task, the complexity of the problem increases significantly, in particular the planning complexity. We discuss our experience of running this integration experiment in detail in Section 8.

8. Results & learnings

A video that includes experiments at both locations is available online.[8] In the following, we discuss the results of the experiments and evaluate the suitability of the proposed framework to address the motivating problems.

8.1. Engineering effort

A core motivation for the proposed framework is to provide an engineering tool to represent re-usable skills developed for robots (c.f. Problem 1) and parametrizable task patterns from which an application engineer can choose (c.f. Problem 2).

Indeed, when realizing the experiments, only minimal effort specific to the experiments was required to define a new scenario. Overall, the same workflow was possible across all experiments, despite their different application focus:

(1) Implement and model the skills of the robot, independent of the specific task, as well as possible task patterns.
(2) Compose and parametrize task patterns for the specific task, independent of the particular robot skills.

This clear separation alone already simplified the implementation significantly and indicates solving Problems 1 and 2 for the considered multi-robot coordination.

Then, using the presented framework, the selection and execution of skills according to the specification could be achieved completely autonomously (c.f. Problem 3). When using the framework with a fixed system as opposed to a number of different single experiments like in this paper, only the above Step (2) is needed when changing the tasks. This requires significantly less expert knowledge when composing a new task from a set of application-specific task patterns.

Finally, the autonomous skill selection procedure and the adaptive collaboration resulting from it effectively removed the need for programming specific tasks for the robots and, instead, enabled to specify on a high level the goal that should be achieved by the team. Based on this experience, we conclude that the framework proved to be a tremendous help as engineering tool in realizing the described use cases and, in terms of engineering, scales well to defining larger applications.

Table 1
Variations of non-deterministic events (table columns) during the iterations (table rows) of the mission shown in Fig. 15. For each event, "C" denotes the robot which checked the status and "D" denotes the robot which performed the delivery if required. Times denote respective completion.

Product at A1	Product at A2	Product at A3	Supplies
station busy C: M1 (2:04) D: –	*station busy* C: M1 (1:20) D: –	*station busy* C: M2 (1:31) D: –	*need* s_2 C: AS (1:09) D: M2 (3:09)
station busy C: M1 (3:19) D: –	*station busy* C: M1 (3:43) D: –	*station busy* C: M2 (4:22) D: –	*need* s_2 C: AS (3:27) D: M2 (5:32)
product ready C: M2 (6:07) D: M1 (10:08)	*product ready* C: M1 (5:36) D: M1 (8:21)	*station busy* C: M2 (7:27) D: –	*need* s_1 C: AS (5:47) D: M2 (9:39)
product ready C: M2 (10:42) D: M2 (12:03)	*station busy* C: M2 (10:27) D: –	*station busy* C: M1 (11:01) D: –	*no demand* C: AS (10:20) D: –
station busy C: M1 (12:49) D: –	*station busy* C: M2 (13:10) D: –	*station busy* C: M1 (12:06) D: –	*no demand* C: AS (12:18) D: –
station busy C: M1 (13:48) D: –	*station busy* C: M2 (13:45) D: –	*station busy* C: M1 (14:16) D: –	*need* s_2 C: AS (13:41) D: M1 (15:20)

8.2. Experiment discussion

In the following, we take a detailed look at the resulting behavior of the system in order to better understand the quality of solutions. This indicates the sort of use cases that can be achieved well with the proposed framework and the difficulties that may become apparent in large-scale applications.

To evaluate performance in detail, Fig. 15 visualizes the skills that have been executed by the two mobile robots (*Mobile 1* and *Mobile 2*) and the assembly station robot (*Assembly*) during a run of Experiment IV of around 15 min of autonomous operation. In addition, Table 1 shows the observed events and respective task assignments during this run of the experiment. The given task is repetitive and it can be seen that execution indeed varies significantly in the shown six iterations depending on the transportation and supply needs observed during operation. Online adaptation and task allocation results from the auctions between the robots.

We briefly highlight two important aspects that can be observed during the execution of this experiment. First, at marker ① and subsequent iterations, the *Assembly* robot starts again to execute the assembly sequence that we require before observing a supply demand, even though the demand has already been determined for the current iteration. This is due to the skill selection (see Section 6.2) that seamlessly considers the next iterations for task allocation to let the robots prepare future required tasks in parallel as long as no current constraints are violated. Consequently, the robot can then check the supply demand immediately in the next iteration.

[8] See https://www.youtube.com/watch?v=GdMmyrzlP8o.

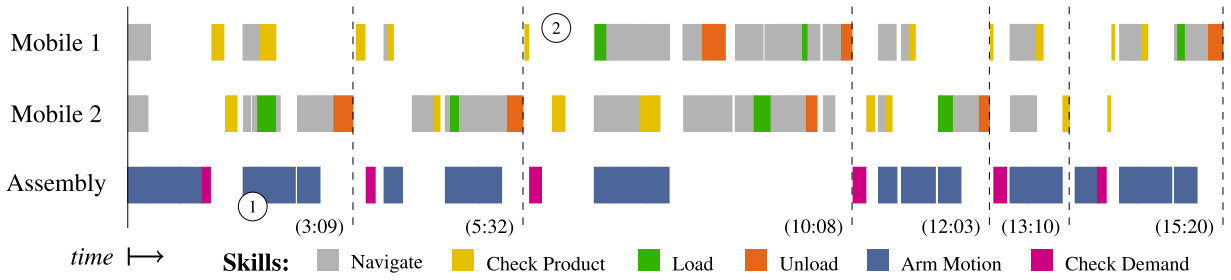


Fig. 15. Visualization of the skills that each of the mobile robots as well as the manipulator at station A1 executed during one real-world run of Experiment IV with a duration of around 15 min of autonomous operation. Time progresses from left to right, dashed lines denote completion of one mission iteration at the stated time. Data has been obtained by monitoring the ROS communication between the *Policy Layer* and the *Skill Layer* for each of the robots.

Second, at marker ②, the mobile robots checked assembly stations A1 and A2 and both detected a transportation request (c.f. Table 1, 3rd row). However, only *Mobile 1* directly carries out the transportation task while *Mobile 2* instead checks the remaining station A3, which is further away from the other two (c.f. Fig. 11(b)). Afterwards, *Mobile 2* can deliver the required supplies and *Mobile 1* also transports the other finished product, resulting in an effective parallelization of the tasks.

Still, there are also some limitations and trade-offs that showed during the experiments. For example, it can be observed in Fig. 15 that several parts of the execution are not fully parallelized. This has multiple reasons. First, needs for transportation that are detected late during an iteration can cause a bottleneck. This is indeed considered by the skill selection procedure, which accounts for probabilistic task outcomes, and it can be observed that in most iterations, observation tasks are assigned before transportation, even if some needs are already known (c.f. marker ②). Second, our implementation of the communication phase for updating the task allocation requires that all robots finished their current policy action. This is no issue if the individual actions are fine-grained, but is visible here in particular for the coarse assembly sequence steps. In a commercial application, this should be implemented to happen in parallel, which is possible with the theoretical framework. Finally, we observed that communication via wifi often added notable latency (c.f. the gap after marker ②). To reduce this effect, we limited communication when one sub-task was close to being finished, i.e., has a very low duration estimate. Consequently, short tasks are often executed sequentially.

In terms of efficiency, we measured that each agent sent a total of 82 bids during auctions in the experiment in Fig. 15. Each bid typically requires to evaluate multiple policies to estimate the expected duration and the resulting state distribution. Thus, to improve the planning efficiency, policies of the agents were cached and re-used whenever suitable, as enabled by the used policy planner. After two days of experiments including the one discussed above, the cache for all robots together contained 245 policies with a total file size of around 200 MB. The number of different policies mainly depends on the application complexity and variations of task-level observations during execution. In practice, for long-term applications, we suggest pruning the cached policies to only store the most frequently used ones.

One consequence of the efficient and decentralized auctioning approach for task allocation, however, is the limited ability to detect task infeasibility. The agents repeatedly coordinate up to the minimally required horizon for assigning at least one task to each agent, which can be significantly shorter than one complete iteration of the task. In contrast, a centralized approach to plan in advance would detect such infeasibility, however, computational complexity would be significantly higher.

8.3. Scalability & future work

Scaling to real-world applications, like factory automation and indoor logistics, with tens of robots and tasks with a significantly larger number of parametrized specification patterns is mainly a challenge for two parts in the system: initial model construction and the repetitive skill selection.

Definition of the skill models is done for each robot independently, thus it is not influenced by the number of robots. The only challenge for model construction is the complexity of the LTL specification and it is well-known that LTL model checking scales worst-case exponentially in the number of propositions. However, the use of task patterns provides the potential for a significantly more efficient model construction because repetitions of the formula structure can be exploited and indeed, a model template can be pre-computed for each pattern once. Combination of these templates is significantly more efficient and suggests a promising scalability.

For each skill selection, the number of auction rounds scales linearly with the number of agents and the number of sub-task choices. While this is much worse for the exponential complexity of a centralized planning approach, we expect it to become noticeable at some scale also for the proposed approach. In other words, there is generally a trade-off between the degree of parallel execution and a less extensive coordination phase for future task assignments. To address this, variations of the skill selection procedure, more specifically, termination conditions for the auction phase, should be considered in practical applications. This may include skipping of full auction rounds in cases when no unexpected observations have been made and only re-allocate in cases where required. This formal knowledge exists in the planning model since expected outcomes can be obtained from each skill in the MDP formalism.

For future work, balancing the described repetitive auctions with some centralized initial planning can be promising, e.g., by performing a number of formal checks before passing a task to the robots. This is encouraged by the computational benefits obtained from caching of policies and models for task patterns. In particular, the repetitive nature of a task suggests the importance of efficient long-term behavior. Still, in our practical experience, the unpredictable evolution of the environment limits the transfer of theoretical guarantees to practice, so it remains subject to future work to find a reasonable balance.

Finally, an important practical direction for future work concerns increased robustness against communication failure. As indicated by the observed wifi issues, communication between the different robots is a reliability bottleneck in practical applications. While this can be mitigated to some degree by using a robust fleet management instead of a ROS multi-master network, also the theoretical framework should be improved in this regard, e.g., by considering lost agents.

9. Conclusions

The paper presents and discusses a framework for multi-robot collaboration in applications inspired by typical industrial automation tasks. As enabled by the pattern-based task specification, non-expert users are able to instruct a heterogeneous team of robots with a set of application-specific, temporally-extended tasks. The use of Linear Temporal Logic as the formalism to express tasks and the combination with online auctions for dynamic task allocation allows for an automated decomposition and meaningful collaboration between the different robots. Whenever new online observations are obtained, the re-allocation of tasks ensures that the team always follows the specified reactions and incorporates possible additional tasks. The basis for this automated execution from a goal specification is given by a skill formalism using Markov Decision Processes. This allows for a common abstraction of various robot capabilities and we present the integration of different state-of-the-art methods in perception, navigation, and manipulation.

To summarize our learnings from the conducted case study experiments, a few remaining challenges have been identified. This includes (1) improved robustness to communication issues and runtime failures of agents, (2) a careful and principled trade-off between extensive pre-planning and reactivity during execution, as well as (3) satisfaction guarantees and completeness of the distributed, receding horizon task allocation. Despite these challenges, our framework proved to be useful in practice and may be transferred to different robot capabilities, task patterns, and allocation algorithms in the future.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the EU H2020 Research and Innovation Programme under GA No. 731869 (*Co4Robots*). The authors would like to thank all members of the project consortium for their support in realizing the presented demonstrations. In particular, thanks goes to *PAL Robotics* for providing the two *TIAGo Bases* as well as to *Bosch Corporate Research* and *PAL Robotics* for providing the infrastructure for preparation and execution of the experiments.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.robot.2021.103866.

References

[1] A. Farinelli, E. Zanotto, E. Pagello, et al., Advanced approaches for multi-robot coordination in logistic scenarios, Robot. Auton. Syst. 90 (2017) 34–44.
[2] H. Wang, W. Chen, J. Wang, Coupled task scheduling for heterogeneous multi-robot system of two robot types performing complex-schedule order fulfillment tasks, Robot. Auton. Syst. (2020) 103560.
[3] P. García, P. Caamaño, R. Duro, F. Bellas, Scalable task assignment for heterogeneous multi-robot teams, Int. J. Adv. Robot. Syst. 10 (2) (2013) 105.
[4] P. Schillinger, M. Bürger, D. Dimarogonas, Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems, Int. J. Robot. Res. 37 (7) (2018) 818–838.
[5] Y. Carreno, R. Petrick, Y. Petillot, Towards long-term autonomy based on temporal planning, in: Annual Conference Towards Autonomous Robotic Systems, Springer, 2019, pp. 143–154.
[6] C. Menghi, S. García, P. Pelliccione, J. Tumova, Multi-robot LTL planning under uncertainty, in: International Symposium on Formal Methods, Springer, 2018, pp. 399–417.
[7] P. Schillinger, M. Bürger, D. Dimarogonas, Improving multi-robot behavior using learning-based receding horizon task allocation, in: Robotics: Science and Systems, RSS, 2018.
[8] D. Perico, T. Homem, A. Almeida, I. Silva, C. Vilão, V. Ferreira, R. Bianchi, Humanoid robot framework for research on cognitive robotics, J. Control Autom. Electr. Syst. 29 (4) (2018) 470–479.
[9] W. Sheng, A. Thobbi, Y. Gu, An integrated framework for human–robot collaborative manipulation, IEEE Trans. Cybern. 45 (10) (2014) 2030–2041.
[10] L. Rozo, M. Guo, A. Kupcsik, M. Todescato, P. Schillinger, M. Giftthaler, M. Ochs, M. Spies, N. Waniek, P. Kesper, M. Bürger, Learning and sequencing of object-centric manipulation skills for industrial tasks, in: IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems, IROS, 2020.
[11] K. Charalampous, I. Kostavelis, A. Gasteratos, Robot navigation in large-scale social maps: An action recognition approach, Expert Syst. Appl. 66 (2016) 261–273.
[12] N. Hawes, C. Burbridge, F. Jovan, L. Kunze, B. Lacerda, L. Mudrova, J. Young, J. Wyatt, D. Hebesberger, T. Kortner, et al., The strands project: Long-term autonomy in everyday environments, IEEE Robot. Autom. Mag. 24 (3) (2017) 146–156.
[13] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Ng, ROS: An open-source robot operating system, in: ICRA Workshop on Open Source Software, Vol. 3, Kobe, Japan, 2009, p. 5.
[14] I. Malavolta, G. Lewis, B. Schmerl, P. Lago, D. Garlan, How do you architect your robots? State of the practice and guidelines for ROS-based systems, in: Proceedings of the 42nd International Conference on Software Engineering: Software Engineering in Practice, 2020.
[15] A. Tiderko, F. Hoeller, T. Röhling, The rOS multimaster extension for simplified deployment of multi-robot systems, in: Robot Operating System (ROS), Springer, 2016, pp. 629–650.
[16] I. Osunmakinde, R. Vikash, Development of a survivable cloud multi-robot framework for heterogeneous environments, Int. J. Adv. Robot. Syst. 11 (10) (2014) 164.
[17] S. Calinon, A tutorial on task-parameterized movement learning and retrieval, Intell. Serv. Robot. 9 (1) (2016) 1–29.
[18] A. Paraschos, C. Daniel, J. Peters, G. Neumann, Probabilistic movement primitives, in: Advances in Neural Information Processing Systems, 2013, pp. 2616–2624.
[19] G. Konidaris, L. Kaelbling, T. Lozano-Perez, From skills to symbols: Learning symbolic representations for abstract high-level planning, J. Artificial Intelligence Res. 61 (2018) 215–289.
[20] C. Cassandras, S. Lafortune, Introduction to Discrete Event Systems, Springer Science & Business Media, 2009.
[21] R. Sutton, D. Precup, S. Singh, Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning, Artif. Intell. 112 (1) (1999) 181–211.
[22] J. Tsitsiklis, On the control of discrete-event dynamical systems, Math. Control Signals Systems 2 (2) (1989) 95–107.
[23] C. Amato, G. Konidaris, L. Kaelbling, Planning with macro-actions in decentralized pOMDPs, in: International Conference on Autonomous Agents and Multi-Agent Systems, International Foundation for Autonomous Agents and Multiagent Systems, 2014, pp. 1273–1280.
[24] X. Liu, J. Fu, Compositional planning in Markov decision processes: Temporal abstraction meets generalized logic composition, in: 2019 American Control Conference, ACC, IEEE, 2019, pp. 559–566.
[25] V. Ziparo, L. Iocchi, D. Nardi, P. Palamara, H. Costelha, Petri net plans: A formal model for representation and execution of multi-robot plans, in: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, Vol. 1, International Foundation for Autonomous Agents and Multiagent Systems, 2008, pp. 79–86.
[26] U. Thomas, G. Hirzinger, B. Rumpe, C. Schulze, A. Wortmann, A new skill based robot programming language using UML/P statecharts, in: 2013 IEEE International Conference on Robotics and Automation, IEEE, 2013, pp. 461–466.
[27] C. Baier, J.-P. Katoen, Principles of Model Checking, MIT press Cambridge, 2008.
[28] C. Belta, B. Yordanov, E. Gol, Formal Methods for Discrete-Time Dynamical Systems, Vol. 89, Springer, 2017.
[29] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, D. Wilkins, PDDL-the planning domain definition language, Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational, 1998.
[30] M. Cashmore, M. Fox, D. Long, D. Magazzeni, B. Ridder, A. Carrera, N. Palomeras, N. Hurtos, M. Carreras, Rosplan: Planning in the robot operating system, in: Twenty-Fifth International Conference on Automated Planning and Scheduling, 2015.
[31] Y.-q. Jiang, S.-q. Zhang, P. Khandelwal, P. Stone, Task planning in robotics: An empirical comparison of PDDL-and ASP-based systems, Front. Inf. Technol. Electron. Eng. 20 (3) (2019) 363–373.

[32] D. Schmidt, Guest editor's introduction: Model-driven engineering, Computer 39 (2) (2006) 25–31.
[33] A. Nordmann, N. Hochgeschwender, S. Wrede, A survey on domain-specific languages in robotics, in: Simulation, Modeling, and Programming for Autonomous Robots, Springer, 2014, pp. 195–206.
[34] P. Doherty, F. Heintz, D. Landén, A distributed task specification language for mixed-initiative delegation, in: International Conference on Principles and Practice of Multi-Agent Systems, Springer, 2010, pp. 42–57.
[35] S. García, P. Pelliccione, C. Menghi, T. Berger, T. Bures, High-level mission specification for multiple robots, in: Proceedings of the 12th ACM SIGPLAN International Conference on Software Language Engineering, 2019, pp. 127–140.
[36] A. Ulusoy, S. Smith, X. Ding, C. Belta, Robust multi-robot optimal path planning with temporal logic constraints, in: International Conference on Robotics and Automation, ICRA, IEEE, 2012, pp. 4693–4698.
[37] A. Ulusoy, S. Smith, X. Ding, C. Belta, D. Rus, Optimality and robustness in multi-robot path planning with temporal logic constraints, Int. J. Robot. Res. 32 (8) (2013) 889–911.
[38] A. Stefanescu, Automatic Synthesis of Distributed Transition Systems (Ph.D. thesis), Universitaet Stuttgart, 2006.
[39] Y. Chen, X. Ding, A. Stefanescu, C. Belta, Formal approach to the deployment of distributed robotic teams, IEEE Trans. Robot. 28 (1) (2012) 158–171.
[40] M. Puterman, Markov Decision Processes: discrete Stochastic Dynamic Programming, John Wiley & Sons, 1994.
[41] X. Ding, S. Smith, C. Belta, D. Rus, LTL control in uncertain environments with probabilistic satisfaction guarantees, IFAC Proc. Vol. 44 (1) (2011) 3515–3520.
[42] M. Lahijanian, S. Andersson, C. Belta, Temporal logic motion planning and control with probabilistic satisfaction guarantees, IEEE Trans. Robot. 28 (2) (2012) 396–409.
[43] J. Fu, U. Topcu, Probably approximately correct MDP learning and control with temporal logic constraints, in: Robotics: Science and Systems, RSS, 2014.
[44] E. Wolff, U. Topcu, R. Murray, Robust control of uncertain Markov decision processes with temporal logic specifications, in: Conference on Decision and Control, CDC, IEEE, 2012, pp. 3372–3379.
[45] E. Wolff, U. Topcu, R. Murray, Efficient reactive controller synthesis for a fragment of linear temporal logic, in: International Conference on Robotics and Automation, ICRA, IEEE, 2013, pp. 5033–5040.
[46] C.-I. Vasile, V. Raman, S. Karaman, Sampling-based synthesis of maximally-satisfying controllers for temporal logic specifications, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2017, pp. 3840–3847.
[47] X. Ding, A. Pinto, A. Surana, Strategic planning under uncertainties via constrained markov decision processes, in: International Conference on Robotics and Automation, ICRA, IEEE, 2013, pp. 4568–4575.
[48] E. Altman, Constrained Markov Decision Processes, Vol. 7, CRC Press, 1999.
[49] A. Mosca, C.-I. Vasile, C. Belta, D. Raimondo, Multi-robot routing and scheduling with temporal logic and synchronization constraints, in: Proceedings of the 2019 2nd International Conference on Control and Robot Technology, 2019, pp. 40–45.
[50] N. Hawes, C. Street, B. Lacerda, M. Mühlig, Multi-robot planning under uncertainty with congestion-aware models, in: Proceedings of AAMAS 2020, International Foundation for Autonomous Agents and Multiagent Systems, 2020.
[51] J. Fu, H. Tanner, J. Heinz, Concurrent multi-agent systems with temporal logic objectives: Game theoretic analysis and planning through negotiation, IET Control Theory Appl. 9 (3) (2015) 465–474.
[52] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, ORB: An efficient alternative to SIFT or SURF, in: International Conference on Computer Vision, IEEE, 2011, pp. 2564–2571.
[53] M. Lourakis, X. Zabulis, Model-based pose estimation for rigid objects, Springer, Berlin, Heidelberg, 2013, pp. 83–92.
[54] A. Makris, A. Argyros, Robust 3d human pose estimation guided by filtered subsets of body keypoints, in: Machine Vision Applications, 2019.
[55] P. Vlantis, C. Vrohidis, C. Bechlioulis, K. Kyriakopoulos, Robot navigation in complex workspaces using harmonic maps, in: International Conference on Robotics and Automation, ICRA, IEEE, 2018, pp. 1726–1731.
[56] C. Rösmann, F. Hoffmann, T. Bertram, Integrated online trajectory planning and optimization in distinctive topologies, Robot. Auton. Syst. 88 (2017) 142–153.
[57] J. Kuffner, S. LaValle, Rrt-connect: An efficient approach to single-query path planning, in: Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065), Vol. 2, IEEE, 2000, pp. 995–1001.
[58] D. Coleman, I. Sucan, S. Chitta, N. Correll, Reducing the barrier to entry of complex robotic software: A moveit! case study, 2014, arXiv preprint arXiv:1404.3785.
[59] P. Tajvar, A. Varava, D. Kragic, J. Tumova, Robust motion planning for non-holonomic robots with planar geometric constraints, in: The International Symposium on Robotics Research October 6–10, 2019, Hanoi, Vietnam, 2019.
[60] C. Papadimitriou, K. Steiglitz, Combinatorial Optimization: aLgorithms and Complexity, Courier Corporation, 1998.
[61] W. Ren, D. Dimarogonas, Dynamic quantization based symbolic abstractions for nonlinear control systems, in: IEEE Conference on Decision and Control, IEEE, 2019, pp. 4343–4348.
[62] P. Tabuada, Verification and Control of Hybrid Systems: A Symbolic Approach, Springer Science & Business Media, 2009.
[63] P. Schillinger, M. Bürger, D. Dimarogonas, Hierarchical LTL-Task MDPs for Multi-Robot Coordination through Auctioning and Learning, 2019.
[64] C. Menghi, C. Tsigkanos, P. Pelliccione, C. Ghezzi, T. Berger, Specification patterns for robotic missions, IEEE Trans. Softw. Eng. (2019) 1.
[65] C. Menghi, C. Tsigkanos, T. Berger, P. Pelliccione, PsALM: Specification of dependable robotic missions, in: Proceedings of the 41st International Conference on Software Engineering: Companion Proceedings, IEEE Press, 2019, pp. 99–102.
[66] S. García, P. Pelliccione, C. Menghi, T. Berger, T. Bures, PROMISE: High-level mission specification for multiple robots, in: Proceedings of the 42nd International Conference on Software Engineering Companion, ICSE'20 Companion, 2020.
[67] M. Colledanchise, Behavior Trees in Robotics (Ph.D. thesis), KTH Royal Institute of Technology, 2017.
[68] P. Schillinger, S. Kohlbrecher, O. von Stryk, Human-Robot Collaborative High-Level Control with Application to Rescue Robotics, in: IEEE International Conference on Robotics and Automation, 2016.

Philipp Schillinger is currently a research scientist at the Bosch Center for Artificial Intelligence with research interests to make autonomous robots more intelligent and cooperative for improved human interaction. He received his B.Sc. (2013) and M.Sc. (2015) degree in Electrical Engineering from TU Darmstadt, Germany, as well as a Ph.D. (2019) degree from KTH Royal Institute of Technology, Sweden. During his studies, he participated at multiple robot competitions including the DARPA Robotics Challenge and Robocup. In 2015, he released the widely used open-source behavior engine FlexBE for ROS.

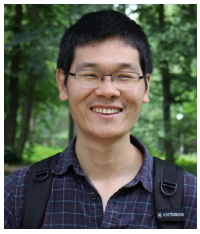
Sergio García is currently a Ph.D. student at the University of Gothenburg (Sweden). His main research interests are in the intersection between robotics and software engineering, striving to conceive well-engineered processes to help developers and users to create robotic applications. He received his B.Sc. (2013) and M.Sc. (2016) degree in Electronics Engineering from the University of Alcalá, Spain. His current research is involved with the Co4Robots H2020 EU project.

Alexandros Makris (m) is currently a post-doctoral researcher at ICS-FORTH. From 2010 to 2014 he was associate researcher at INRIA Grenoble - Rhone-Alpes. He obtained his Ph.D. in Computer Science from the Department of Informatics of the University of Athens in 2010. He holds a Diploma in Electrical and Computer Engineering from the National Technical University of Athens. His main research interests are computer vision, perception for robotics, probabilistic models, intelligent vehicles, and remote sensing.

Konstantinos Roditakis is a Ph.D. student of Computer Science at the Computer Science Department (CSD), University of Crete (UoC) and member of Computational Vision and Robotics Laboratory (CVRL) at the Institute of Computer Science (ICS), Foundation for Research and Technology-Hellas (FORTH) in Heraklion, Crete, Greece. He received a B.Sc. (2014) and M.Sc. degree (2017) in Computer Science from CSD-UoC. His current research interests fall in the areas of computer vision and machine learning, pose estimation and fine-grained analysis of human activities. He is also interested in applications of computer vision in the fields of robotics and virtual reality.

Michalis Logothetis received a Diploma in Electrical and Computer Engineering from National Technical University of Athens (NTUA) in 2016 and is currently pursuing a Ph.D. in Mechanical Engineering at the same university. His main research interests involve motion planning and control for cooperative manipulation and transportation tasks using heterogeneous multi-robotic systems (mobile or static manipulators and humans). He has participated as a researcher in EU-funded and National robotics projects.

Konstantinos Alevizos received a Degree in Physics from National and Kapodistrian University of Athens in 2016 and a MS Degree in Automation Systems from National Technical University of Athens (NTUA) in 2018. He is currently pursuing a Ph.D. in Mechanical Engineering at NTUA. His main research interests focus on motion planning and interaction control for cooperative human–robot systems. He has participated as a researcher in EU-funded and National robotics projects.



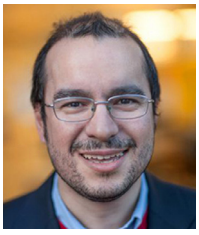
Wei Ren received his B.Sc. degree from Hubei University, China, and his Ph.D. degree from the University of Science and Technology of China, China, in 2011 and 2018. He was a visiting student at the University of Melbourne, Victoria, Australia. Currently, he is a postdoctoral fellow at KTH Royal Institute of Technology, Stockholm, Sweden. His research interests include networked control systems, nonlinear systems, symbolic abstraction, multi-agent systems and hybrid systems.

Pouria Tajvar received the B.Sc. degree in electrical engineering from the Ferdowsi University of Mashhad in 2015 and M.Sc. Degree in automation and control engineering from the Politecnico di Milano in 2017. He is currently pursuing the Ph.D. degree in the robotics, perception, and learning division (RPL) at the KTH royal institute of technology under the supervision of Prof. Jana Tumova. His research interests are in data-driven control synthesis and primitive-based motion planning.

Patrizio Pelliccione (male) is an Associate Professor at DISIM - University of L'Aquila and an Associate Professor at the Department of Computer Science and Engineering at Chalmers | University of Gothenburg. He got his Ph.D. in 2005 at the University of L'Aquila (Italy) and from February 1, 2014 he is Docent in Software Engineering, title given by the University of Gothenburg. His research topics are mainly in software engineering, software architectures modeling and verification, autonomous systems, and formal methods. He has co-authored more than 120 publications in journals and international conferences and workshops in these topics. He has been on the program committees for several top conferences, he is a reviewer for top journals in the software engineering domain, and he organized as program chair international conferences like ICSA2017 and FormaliSE 2018. He is very active in European and National projects. He is the PI for Co4Robots H2020 EU project for the University of Gothenburg. In his research activity he has collaborated with several industries such as Volvo Cars, Volvo AB, Ericsson, Jeppesen, Axis communication, Systemite AB, Thales Italia, Selex Marconi telecommunications, Siemens, Saab, TERMA, etc. More information is available at http://www.patriziopelliccione.com.

Antonis Argyros (m) is a Professor of Computer Science at the Computer Science Department (CSD), University of Crete (UoC) and a researcher at the Institute of Computer Science (ICS), Foundation for Research and Technology-Hellas (FORTH) in Heraklion, Crete, Greece. He received a B.Sc. degree in Computer Science (1989) and a M.Sc. degree in Computer Science (1992), both from the Computer Science Department, University of Crete. On July 1996, he completed his Ph.D. on visual motion analysis at the same Department. His current research interests fall in the areas of computer vision and pattern recognition.

Kostas J. Kyriakopoulos received a Diploma in Mechanical Engineering from NTUA, in 1985 and the MS & Ph.D. in Electrical, Computer & Systems Engineering (ECSE) from Rensselaer Polytechnic Institute (RPI), Troy, NY in 1987 and 1991, respectively. He did research at the NASA Center for Intelligent Robotic Systems and he was an Assistant Professor at ECSE-RPI and the New York State Center for Advanced Technology in Automation and Robotics. Since 1994 he has been with the Control Systems Laboratory of the Mechanical Engineering Department at NTUA, Greece, where he currently serves as a Professor and Director for the Control Systems Lab. His current interests are in the area of Nonlinear Control Systems applications in Sensor Based Motion Planning & Control of multi-Robotic Systems: Manipulators & Vehicles (Mobile, Underwater and Aerial) and Micro-& Bio-Mechatronics.



Dimos V. Dimarogonas was born in Athens, Greece, in 1978. He received the Diploma in Electrical and Computer Engineering in 2001 and the Ph.D. in Mechanical Engineering in 2007, both from the National Technical University of Athens (NTUA), Greece. Between May 2007 and March 2010, he held postdoctoral positions at KTH Royal Institute of Technology, Stockholm, Sweden and at LIDS, MIT, Boston, USA. He is currently a Professor at the Division of Decision and Control Systems, School of EECS, at the KTH Royal Institute of Technology. His current research interests include Multi-Agent Systems, Hybrid Systems and Control, Robot Navigation and Networked Control. He serves in the Editorial Board of Automatica and the IEEE Transactions on Control of Network Systems and is a Senior Member of the IEEE. He received an ERC Starting Grant from the European Commission for the proposal BUCOPHSYS in 2014 and was awarded a Wallenberg Academy Fellow grant in 2015.