# A review of classical and learning based approaches in task and motion planning

Kai Zhang, Eric Lucet, Julien Alexandre Dit Sandretto, Selma Kchir, David Filliat

# A review of classical and learning based approaches in task and motion planning

Kai ZHANG[1,2][0000−0003−1129−9944], Eric LUCET[1][0000−0002−9702−3473], Julien ALEXANDRE DIT SANDRETTO[2][0000−0002−6185−2480], Selma KCHIR[1][0000−0003−3047−6846], and David FILLIAT[2,3][0000−0002−5739−1618]

[1] Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France
[2] U2IS, ENSTA Paris, Institut Polytechnique de Paris, 91120 Palaiseau, France
[3] FLOWERS, INRIA, ENSTA Paris, Institut Polytechnique de Paris, 91120 Palaiseau, France
kai.zhang@cea.fr

**Abstract.** Robots are widely used in many tedious and simple works. But, with the advance of technology, they are expected to work in more complex environments and participate in more challenging tasks. Correspondingly, more intelligent and robust algorithms are required. As a domain having been explored for decades, task and motion planning (TAMP) methods have been applied in various applications and have achieved important results, while still being developed, particularly through the integration of more machine learning approaches. This paper summarizes the development of TAMP, presenting its background, popular methods, application environment, and limitations. In particularly, it compares different simulation environments and points out their advantages and disadvantages. Besides, the existing methods are categorized by their contribution and applications, intending to draw a clear picture for beginners.

**Keywords:** Task and motion planning, · simulation environment, · review.

## 1 Introduction

Robotics has been explored for decades to assist, complement or replace humans. Since robots never feel tired and make less mistakes in repetitive works, they have the potential to replace humans in certain jobs. In industrial production, robots have been widely used for repetitive and well-defined tasks such as assembling cars and moving goods in assembly lines, clearly increasing productivity. To bring the benefit of robots to daily lives, many works explore the adaptation of robots from industrial environments to daily living environments. However, unlike the industrial environment, which is mostly structured and static, the daily living environment is full of change, which poses new challenges for robot control algorithms. Besides, robots take on more complex tasks, such as household chores or delivery tasks. Therefore, more efficient algorithms need to be proposed to enable robots to understand the environment and behave accordingly.

To complete a complicated task, also named long-horizon task, a popular solution is to decompose it into simple subtasks and solve them one by one. This type of methods can be summarized as two steps, task planning and motion planning.

From the view of task planning, the complicate tasks could be divided into several fundamental abstract tasks, like navigation, grasp, push, pull, etc. This task-decomposing process is independent of the robot types but based on the definition of abstract short-horizon tasks. For example, the "open door" task could be considered as a sequence of abstract actions, like grasping the door handle, then opening the door. Therefore, the difficulties of task planning are the definition of abstract actions and the decomposing strategy.

With a subtask, motion planning converts it into executable control parameters for the robots to accomplish the task. For example, in the subtask of grasping the door handle, by taking the handle position and robot position, the inverse kinematic method calculates the configuration of each joint of the arm so that the gripper could reach the handle successfully. Since the robot is not alone in the environment, calculating a feasible moving trajectory is challenging.

In summary, task planning and motion planning share some similarities but also can be distinguished by their difference. They play similar roles in the long-term task-solving part, which means both of them convert the task from a higher and harder level to a lower and easier level. As for the difference, task planning plays in the discrete space and is often independent of the robots while the motion planning lives in either discrete space or continuous space and varies among robots. For example, the planning of navigation trajectory can be continuous but planning of start and stop position is discrete.

This paper is an extension of a previous conference paper [59] where we added several recent advancement on task and motion planning (TAMP) methods. Besides, a detailed explanation about the comparison criteria of simulation environments is provided to help with choosing the most suitable one. We also enriched the challenge section, especially on situational mapping since it is crucial for future TAMP methods to take more context into account.

In a first step, task planning and motion planning are introduced separately along with explanation of some popular tasks in section 2. Then, the TAMP methods are introduced and compared in three different categories in section 3. Section 4 describes some public simulation environments and tools used for TAMP. Before the summary in section 6, the current challenges are detailed based on current TAMP advances in section 5.

## 2 Background

### 2.1 Task planning

Task planning aims to divide a long-horizon task into several short-horizon subtasks, which makes the original difficult task easier to solve. For example, in a complex washing task [19], given the initial and final states, task planning

generates a sequence of intermediate states and corresponding abstract actions (move, wash, store...). By executing the abstract actions, the robot could transit from the initial state to the final state.

A more mathematical definition can be expressed as: given as input the initial state $S_0$ and final state $S_g$, task planning produces several intermediate states $S_i, i = 1, ..., g-1$, where each $S_i$ could be achieved from $S_{i-1}$ through the transition actions $A_i, i = 1, ..., g$.

In most cases, the transition actions are abstract and independent of the robot. A popular description format is the STRIPS-style form, which contains at least the name, precondition and effect of the action. For example, the action *move* from place *P1* to place *P2* can be described as:

Move(P1,P2)
Pre: (atPos(P1)) (not(atPos(P2)))
Eff: (atPos(P2)) (not(atPos(P1)))

In general, action $A_i$ could be either discrete or continuous. The discrete action space is made of a finite number of actions, such as turning left and turning right. In contrast, the continuous action space is infinite and each action is set by a parameter. For example, turning left 30 degrees is different from turning left 30.1 degrees. Comparing to discrete action space, it takes more time to find the action sequence in a continuous action space.

To solve the task planning problem, several planning methods have been proposed, such as behavior tree methods [19], heuristic search methods [10], operator planning methods, etc. A more detailed overview and discussion can be found in the introduction book [31]. Thanks to their simplicity and efficiency, they are widely used in the decision making games like chess, Tower of Hanoi, etc.

Instead of a hand-crafted approach, reinforcement learning (RL) methods learn strategies to map observations from the current situation to the next subgoals by maximizing rewards. Popular RL methods include DQN [38] for discrete actions and DDPG [35] or Soft Actor-Critic(SAC) [16] for continuous actions. By default, these methods learn the solution of a single task, hence they couldn't be used as a general task planner. But in planning problems, the RL approach can be applied to learn goal-conditioned policies, which takes in the observation and goal information and outputs appropriate next subgoals as actions. For example, to solve a 2D navigation task, [6] uses a goal-conditioned RL to generate a sequence of intermediate waypoints that the robot could follow to reach the goal. Such policies can also be trained using imitation learning, and their generalization to large problems is considered in [51]. They propose Action Schema Networks to solve probabilistic and classical planning problems by mimicking the policy generated by traditional planners.

### 2.2 Motion planning

Motion planning can be thought of as a bridge between low-level control parameters and high-level tasks. Given a goal position corresponding to a high-level

task, the motion planning algorithm generates a trajectory or a number of control parameters, such as the robot's angular and linear speed, to guide it to its destination in the navigation subtask.

For motion planning, a number of algorithms have been developed, such as inverse-kinematic approaches for manipulation tasks or shortest path planning methods for navigation tasks. Ghallab's planning book [15] provides a more thorough introduction and overview of the existing techniques.

In addition to the classical approaches, learning techniques are also employed. For example, [61] proposes a reinforcement learning technique for visually guided navigation in enclosed spaces. The motion parameters are generated by a network to govern the movement of the robots based on the visual observations. More approaches based on reinforcement learning can be found in [48].

The cost function to be optimized is another dimension of the motion planning systems. Different from most approaches that have a single objective such as the distance to obstacles, motion planners can also develop plans based on the specific context of the local environment. For instance, a context-aware costmap [36] can be created by combining numerous semantic layers, each of which defines a different sort of constraint, resulting from mobile or static obstacles and from areas with specific danger or expected behavior. Then, a practical and secure trajectory could be produced by planning using the context-aware costmap. For instance, a proactive obstacle avoidance technique is presented in [42], where the robot tends to steer clear of people from their back area. Other examples could include situation where the robot must go on the correct side of the road or, when faced with stairs, it should choose the wheelchair accessible ramp.

### 2.3 Task and motion planning objectives

Although there are many complex global objectives for TAMP in the human environment, the majority of them can be viewed as a mixture of more simple prototypical tasks. We think that if the TAMP approaches could successfully handle these fundamental problems at scale, they could be easily applied to address more challenging tasks. We present three of these fundamental tasks below:

- Rearrangement (Re). Figure 1(a) illustrates how the robot must maneuver a number of obstacles in order to avoid colliding with these obstacles while reaching for the desired target object. Collaboration can be necessary for the rearrangement task for numerous robots, which commonly happens when a robot's arm is physically unable to access certain areas of the environment [4]. In rearrangement tasks, we typically concentrate on planning arm actions.
- Navigation among movable obstacles (NAMO). Compared to pure navigation tasks, a robot might interact with the surroundings while navigating to the desired point. For instance, it can actively remove movable obstacles in order to make a blocked trajectory feasible. Figure 1(b) provides an illustration where the robot should clear the obstructions in the hallway before

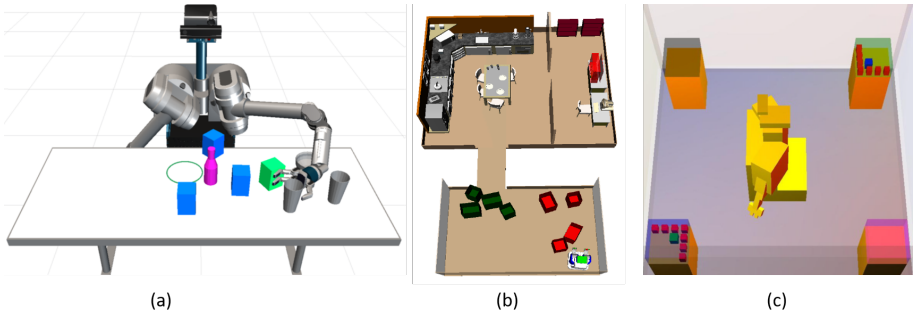


**Fig. 1.** Demonstration of tasks (Figure is from [59]): (a) Rearrangement task. The robot needs to move the green box from its start pose to the goal region indicated by the green circle [25]. (b) Navigation among movable obstacles. The robot needs to remove the green obstacles before pushing the red boxes to the kitchen region [24]. (c) Pick-Place-Move task. The robot should pick the blue cube and place it in the bottom left container [9].

entering the kitchen. Compared to rearrangement, we typically intertwine arm and mobile base actions.

– Pick-Place-Move (PPM) task. The robot's basic actions are to pick up an object, move it, and put it in a container, as seen in Figure 1(c). The PPM can also be used for assembly and/or disassembly job, which should take the order of the manipulated objects into account. Here again, plans typically intertwine arm and mobile base actions.

In all these three fundamental tasks, task planning usually entails choosing which object to grasp or manipulate, and where to put it, while motion planning will generate arms and mobile base motions to execute these decisions.

## 3 Task and motion planning methods

After introducing task planning and motion planning separately, we now focus on methods that perform both simultaneously. This is challenging because these two modules should be designed carefully to guarantee the effectiveness of the system. For example, the task planner should generate feasible plans for the motion planner and adjust the next output according to the results of motion planner. The connection between these two planners contributes to the intelligent behaviours of the robot.

We describe the main TAMP methods into three categories, namely classical methods that are historically developed first using various algorithmic approaches, learning methods that try to tackle the task using purely data based algorithms and hybrid methods that combine the previous two techniques in various ways.

### 3.1 Classical methods

The classical methods are mainly divided into two categories, sampling-based and optimization-based methods. Some examples can be found in Figure 2. The widely-used sampling-based methods usually include symbolic operators, which specify the prerequisites of applying an action and its effect. A comprehensive review on sampling methods and optimization methods to solve TAMP problems can be found in [8].

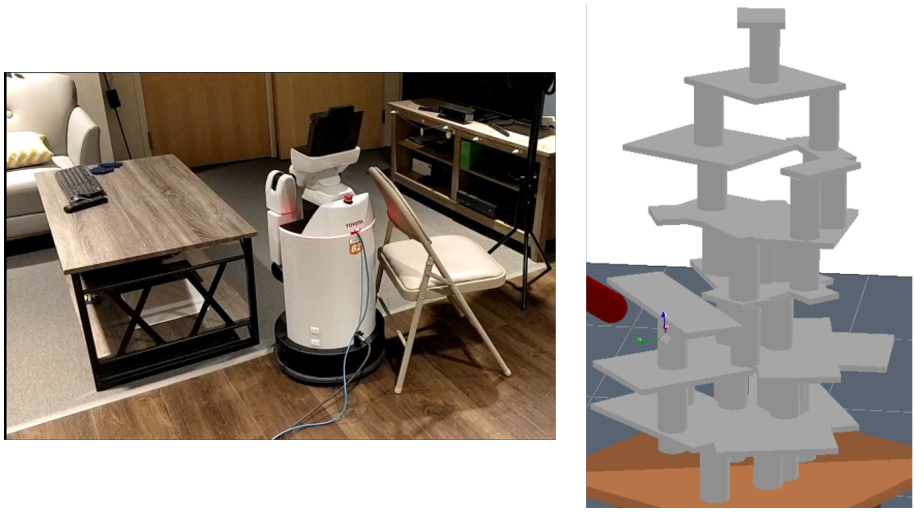

**Fig. 2.** TAMP applications of using classical methods. The left figure is from [52] where robot navigates based on sampling methods. The right figure is from [50] where robot builds the highest tower based on optimization methods.

Given a long-horizon task with description of initial and final state, sampling-based methods randomly sample states from the continuous state space in order to find valuable intermediate states, like key frames of a video. Afterward, searching approaches are used to identify a series of workable transition operators between the sampled intermediate states, and the sampler can produce new samples if an existing one doesn't result in a feasible plan. The frequently adopted searching methods include heuristic search [52], forward search [19], or backward search (a more complete overview on searching methods can be found in [15]). From the resulting sequence of intermediate states and operation between them, classical motion planning methods, such as A* [17], RRT Connect [29] for the robot base or inverse kinematics for the robot arm [13,47], are applied to control the robot to follow the plan.

However, sampling-based methods do not perform well on some particular problems. First, they cannot generally identify infeasible problem instances and will therefore not terminate in such cases. Second, the sampling procedure can

only be used on the space that has already been investigated, therefore they are unable to provide solutions for problems that require identifying values from unknown space [11]. For example, in a partial observation case, the robot can only plan a path to a waypoint within the range of observation. Finally, sampling techniques often fail when the task description is not specific enough. For example, in a pouring task, if we ask the robot to pour as much milk as possible into the cup without indicating the amount to reach, the sampling planner will fail since it cannot find an ultimate state.

Optimization based approaches present characteristics that solve these shortcomings of the sampling methods. Different from terminating on a specific state, the objective is represented as a cost function along the temporal axis. Then, an optimization strategy is utilized to minimize the cost with respect to constraints and finally outputs a workable solution. Benefiting from the integrated time axis in the objective function, the optimization method is appropriate to solve the problems in continuous action space. For example, in a manipulation problem where a robot should assemble cylinders and plates on a table to build the highest possible stable tower [50], the action sequences are generated by a straightforward symbolic planning approach but the optimal final and intermediate placements of each component are discovered through optimization.

In addition, there are also some TAMP methods based on specific handcrafted strategy. For example, [37] presents an active path cleaning algorithm for the NAMO task, which integrates obstacle classification, collision detection, local environment reconstruction and interaction with obstacles. To solve the situation where the obstacle is unknown, an affordance-based method [52] is developed to help the robot identify if the obstacle is movable through interaction.

### 3.2 Learning based methods

In learning based methods, robot acquires skills from numerous trials including success and failure cases, instead of relying on a manually designed model associated to sampling or optimization. As one of the most popular frameworks, RL learns to produce the plans sequentially through a policy mapping the current observation of the environment to an action [4]. To solve a TAMP problem, a reward function is defined, usually giving a positive reward when the robot finishes the task, and a negative reward when it fails. Then, the learning mechanism behind RL finds the policy that maximizes the sum of rewards. One of the important difficulty of RL for long TAMP problems is to explore efficiently the environment, as taking random actions requires a prohibitive number of trials until a solution can be found [34]. Therefore, hierarchical RL (HRL) has been proposed to solve this sparse reward problem by decomposing the complicated tasks into simpler subtasks so that the robot could get positive reward easily and accomplish the difficult tasks step by step [1]. Some examples can be found in Figure 3.

An intuitive idea of HRL is to design and train two networks, one for high level task generation and another one for primitive motion control. In [18], learning sensorimotor primitives is accomplished by training a low-level network with

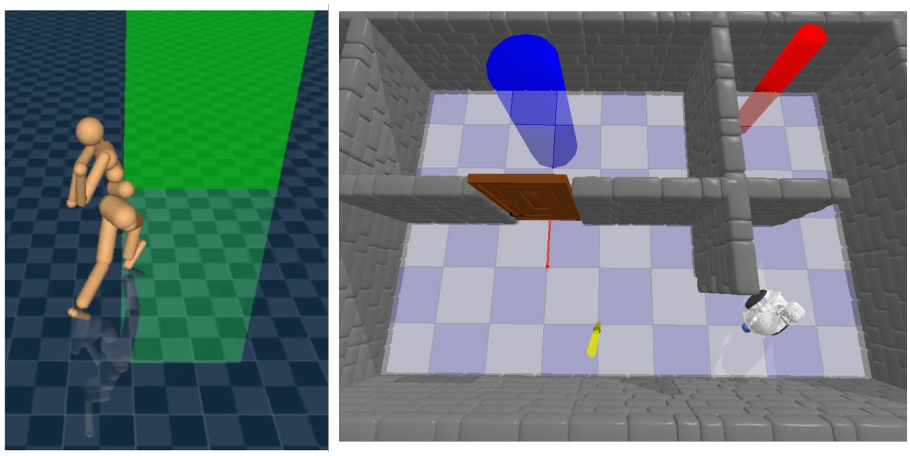

**Fig. 3.** TAMP applications of using learning-based methods. The left figure is from [18] where the humanoid robot passes a virtual gate. Two policies including subgoal generation and balance control are optimized separately. The right figure is from [34] where robot navigates to blue goal point. The subgoal generation and motion control modules are optimized jointly.

access to proprioceptive sensors on simple tasks. Then, this pre-trained module is fixed and connected to a high-level network, with access to all sensors, which drives behavior by modulating the inputs to the low-level network. Similarly, in [30], a high-level module learns the strategy of generating subgoals and a low-level module learns the actions to complete each subgoal. Due to the limitations of previous approaches in generalization capabilities, a task-independent approach is designed by reformulating the task description [40]: absolute observations are replaced with relative observations, such as position and distance to reduce the dependence on the task environment. Additionally, they use an off-policy RL technique, which requires less interactive training data, to simplify training in a real-world setting.

Training high-level strategies and low-level actions separately would lose the opportunity for joint optimization, leading to the possible failure of converging at the optimal strategy. Therefore, in [32], the authors describe a joint training strategy that learns the policy in three levels for a navigation task. The top level generates subtasks by the current state and the goal state, while the middle level decomposes the subtasks into directly feasible goals and the lowest level generates action control parameters to reach the goal. By maximizing the success rate, the three levels are optimized jointly. However, in a NAMO task, given a final location, the higher-level subgoal creation network must generate goals not only for the robot base but also for the arm to interact with. Therefore, a HRL method is proposed in [34] to generate heterogeneous subgoals for the robot to interact with obstacles during navigation. To find appropriate interactions with various obstacles, a neural interaction engine [58] that predicts the effects of

actions is integrated into a policy-generating RL network. With this network, the actions with the greatest success rate are picked by comparing the effects of different actions on the success rate of completing the task in the future.

Although these learning methods achieve satisfactory results in simulated environments, transferring from simulated environments to real applications is difficult because the trained models usually cannot be used directly in real scenarios. In most cases, they need to be retrained in the application environment. For example, in the solution proposed in [34], the trained models map sensor data to actions. When migrating this model to the real environment, the mapping between sensor data and actions needs to be re-established due to the huge differences between the virtual and real environments. In addition, real sensor data is noisy. Even at the same observation location and observation environment, the sensor data observed twice may be different, and this variation may lead to erroneous actions of the model. It is also important to note that training data in real environments is expensive and the effectiveness of learning-based methods is affected by the required size of the training dataset. Therefore there are very few real world applications that rely only on learning methods.

### 3.3 Hybrid methods

Although both classical and learning-based methods can solve some TAMP tasks, they both have limitations. For example, the symbolic operators used in sampling-based methods are usually designed manually, which requires expertise and is time-consuming. In addition, the sampling approach is inefficient because of environmental constraints, where only a small fraction of a huge space may meet the requirements, but random sampling requires verification of thousands of sample points to find a suitable location. Learning methods avoid manual operations, but they provide less freedom to add additional constraints such as no collision tolerance. The obtained models have poor migration capabilities due to the mappings from observations to behaviors are closely related to the experimental environment. Moreover, the transfer of learning methods from simulated to real environments proves to be difficult due to the high cost of constructing training datasets and the unrealistic representation of the environment, which can be caused by sensor noise, varying illumination conditions, noisy action execution, etc.

Therefore, some researchers adopt the hybrid strategy, using a classical TAMP solver structure that include some learning components. This includes learning to generate symbolic operators from dataset [46,41,28], learning to guide the operator search [24,22], and learning to generate feasible subgoals [55]. Some instances can be found in Figure 4.

**Learn to generate symbolic operators** Learning symbolic operators from a dataset provides the basic elements, i.e., primitive actions, for the task planning. With the learned operators, a conventional tool such as PDDL [14] or its

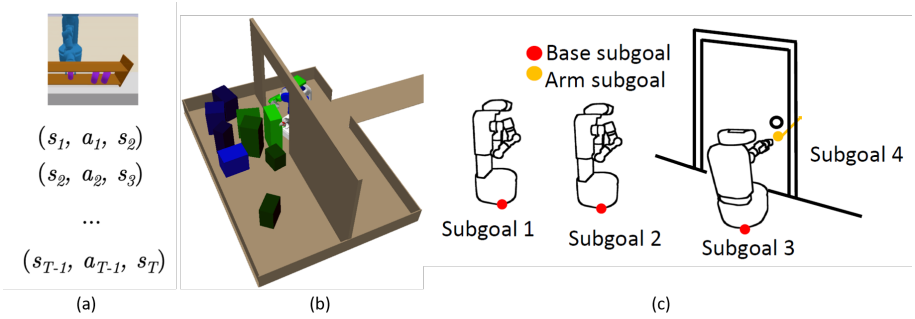


**Fig. 4.** TAMP applications of using hybrid methods. Figure (a) is from [46] where the symbolic operators are learned from a dataset before applying TAMP methods. (b) is from [24] where the RL is leveraged to order the priority of the manipulated objects. (c)[55] integrates the RL method for subgoal generation with classical motion control algorithms to solve a NAMO task.

extensions [57,12] is applied to search feasible plans, which consists of primitive actions. Then, a motion planning algorithm is used to directly convert the primitive actions into executable control parameters.

A supervised learning strategy is introduced in [41] to learn symbolic operators from a training dataset. Each training instance contains the current state, an action, and the effect after applying the action. The action model is searched by maximizing the likelihood of the action effect, but taking into account a complexity penalty. To reduce the requirement of expensive training datasets, a learning-from-experience approach [28] first applies the action to the agent and records state through experience. Then, it converts continuous states into decision trees and finally into symbolic operators. Similarly, in [46], the original dataset is created from demonstrations and the abstract symbolic operators are generated by statistical clustering methods and used for further task planning.

**Learn to search** In a problem containing a large number of actions and states, classical search methods are less efficient because the search space is too large. Reinforcement learning methods provide a way to learn the search strategy from experience thus avoiding traversing the entire space to find a solution [2]. In [24], a graph is used as a representation of the search space due to its scalability. Nodes are abstract actions, while edges are transition priorities. A Q-value function is learned from the training dataset using a neural network to rank the abstract actions and calculate the priorities to guide efficient search. In addition to searching in the discrete space, a generative model provides multiple feasible candidates in the continuous action space to avoid being blocked by an infeasible solution [23]. Similarly, a model is applied to the dataset to learn the probability of success [53,54]. Then, in the same domain but new scenario, it predicts the success rate of each action. By selecting the actions with higher success rates, the search space is greatly reduced. In addition, the success rate is also used to plan

subtasks. In [60], a model is trained to generate a success rate map with respect to navigation subgoals. A threshold is then set to constrain the search space of subgoals in the environment, thus speeding up the task planning process.

**Learn to decompose** In addition to operator-based methods, some direct subtask generation methods based on reinforcement learning have been proposed. These methods propose directly feasible subtasks for which classical motion planning methods are employed in order to control the robot. In a NAMO task, [55] use the SAC algorithm [16] to generate subtasks for the robot's arm and base. The algorithm use a distance reduction reward and success reward to train the network to generate subtasks, in conjunction with traditional motion planning methods such as RRT connect [29] and inverse kinematic methods to verify the feasibility of subtasks.

In summary, hybrid approaches typically apply a learning-based approach to task planning, or part of the task planning process, and then employ classical motion planning algorithms to generate control parameters. The use of the classical motion planning in these strategies offers better transferability to the real world than pure learning algorithms, while the learning-based part provides more efficient solutions than classical task planning methods.

Finally, Table 1 provides an overview of the application of the presented classical, learning based and hybrid methods on the fundamental tasks proposed in section 2.3.

**Table 1.** Applications of TAMP methods on the three fundamental tasks. Re: Rearrangement, NAMO: Navigation Among Movable Obstacle, PPM: Pick Place Move.

| | Classical methods | Learning based methods | Hybrid methods |
|---|---|---|---|
| Re | [47,50,13] | [4] | [2,53,54] |
| NAMO | [37,52] | [34,39,58] | [22,21,55] |
| PPM | [19,20,9] | | [24,28,11,60] |

## 4 Benchmarks and tools

The validation of algorithm performance is an important but difficult step when developing robotics algorithms. Testing directly the effects of the approaches in a real environment is a straightforward approach, but it is extremely time-consuming, expensive, unstable, and potentially unsafe. Therefore, it is common to first validate in a virtual environment and then migrate to real conditions if expectations are met. To facilitate experimental validation, a number of interactive simulation environments have been proposed for a wide variety of tasks.

In this section, we compare several simulation environments that are designed for navigation and manipulation tasks, which include iGibson2 [33], AI2THOR

[27], TDW [7], Sapien [56], Habitat2 [49] and VirtualHome [43]. The generic simulators like Gazebo[26], Bullet[3], are not considered since they are not designed for task and motion planning but for more general usage. Instead of focusing on the low-level view of which type of rendering they use, we focus on their usability for TAMP tasks and their extensibility to real environments. We describe below our selected comparison criteria and the results of our evaluation for each environment is presented in Table 2.

**Table 2.** Comparison among different interactive simulation environments. Table is from [59].

| | | iGibson2 | AI2THOR | TDW | Sapien | Habitat2 | VirtualHome |
|---|---|---|---|---|---|---|---|
| Provided environment | | 15 homes (108 rooms) | 120 rooms | - | - | - | build from 8 rooms |
| Interactive objects | | 1217 | 609 | 200 | 2346 | - | 308 |
| ROS support | | ✓ | × | × | ✓ | ✓ | × |
| Uncertainty support | | ✓ | × | × | × | ✓ | × |
| Supported tasks | Re | + | +++++ | ++++ | ++++ | ++++ | ++++ |
| | NAMO | +++++ | ++ | +++ | +++ | ++ | ++++ |
| | PPM | +++++ | +++++ | ++++ | ++++ | +++++ | +++++ |
| Speed | | ++ (GPU) | ++ | ++ | +++ | ++++ | ++ |
| Sensors | | RGBD, Lidar | RGBD | RGBD | RGBD | RGBD | RGBD |

- **Provided environment and interactive objects.** A simulation environment with embedded scenes and interactive objects is easier to use since constructing the scenes is difficult for a beginner. A large number of interactive objects provide the users more freedom to adapt the environment to the tasks.
- **ROS support**. Robot Operating System (ROS) [44] is a generic and widely-used framework for building robot applications. It offers a standard software platform to developers across industries that will carry them from research and prototyping all the way through to deployment and production. With ROS support, we could easily transfer our algorithm from simulation environment to real applications. Hence, an environment supporting ROS has better transferability.
- **Uncertainty support.** In the real environment, the sensor data always contains noise and uncertainty. Introducing uncertainty in simulation environment provides more realistic results for the algorithms.
- **Support tasks.** Here we evaluate the usability of each environment to the three fundamental tasks described in 2.3. The score is given based on the

provided environment, types of interaction, control interfaces, etc. A higher score means that it is easier to apply the environment to the task.
- **Speed.** Rendering speed is important in simulation experiments. All the environments can use only CPU for rendering except the iGibson2.
- **Sensors.** We list which type of sensors the simulation environment supports (RGBD: Color and depth camera).

## 5 Challenges

Although TAMP methods have been explored for decades, they still lack robustness and generalization capabilities and face limitations in practical applications. In this section, we present several potential areas for improvement.

### 5.1 Observation uncertainty

Uncertainty in observations is usually caused by sensor noise, which is unavoidable in practical applications. While this is a very general concern, there are two solutions that target specifically this problem by (a) modeling the noise and reducing it by multiple observations [20], and (b) using learning methods to map the noisy data directly to actions [4]. An operator-based TAMP approach is proposed in [20] to address the observation uncertainty in the PPM task. This uncertainty arises in the localization of the robot and the target object. They eliminate the noise by modeling the noise of localization using a Gaussian model after multiple observations of the object from the robot. A noteworthy point is that they model the relative difference between the two objects rather than focusing on a single object because it may have systematic errors. In [4], raw sensor data is fed directly to a neural network, aiming to map raw observations into action sequences through reward optimization. This approach is simple because it does not require a complex modeling process, but a large number of training scenarios, 30,000 in their experiments, is required for the neural network to withstand the effects of random errors on the observations.

In summary, although these approaches are effective in reducing the effect of observation errors on actions, the corresponding experimental setting remain quite simple and the robot has a large free space to operate. Therefore, this raises the question of whether the method is practical in limited and complex environments where robots need to perform household tasks. Making multiple observations from multiple angles in a small environment is not easily achievable and the learning-based methods are too inefficient. There are therefore still many challenges in the integration of uncertainty reduction actions in the TAMP problems.

### 5.2 Action uncertainty

Using a symbolic operator, an action may produce different action effects given the same preconditions and actions. For example, the *pick* action may be performed by grabbing from the top of an object or from its side. This ambiguity

may lead to failure when trying to place the object steadily. In a PPM task described in [46], the robot needs to pick an object and place it on a shelf. Due to the limited space under the ceiling of the shelf, the robot needs to choose an appropriate pick action. They collected a dataset from which several picking actions were obtained, such as picking from the side and picking from the top. By backtracking the target state, a solution is found so that the robot could choose actions based on the target state.

Such backtracking method can certainly solve the uncertainty of the action, but it requires sufficient observation of the environment, which is usually not satisfied in practical applications. Another option is to replan based on observations. For example, initially the robot grasps an object from the top and observes that the current grasping setup is not appropriate for placing. It can replan and change the grasping configuration by temporarily putting the object on a table and then placing the object on the shelf. Nevertheless, replanning is usually a costly operation, and integrating efficient replanning strategies in TAMP is an interesting research direction.

### 5.3 Context-aware decision making

Real tasks are often more complex than the tasks artificially designed for research, and the robot needs to derive solutions by considering the relevant situational information for each task. For example, imagine a block building task where various shapes of blocks are provided and the goal is to assemble a car model. Solving the task is not possible without considering the shape of each block and of the car. Contextual analysis and mapping should therefore be adapted to as many tasks as possible and can benefit a variety of domains, including safe navigation, action verification, and understanding ambiguous tasks.

To enable safe navigation, situational mapping allows robots to create their own danger zones based on the type of obstacles. For example, for stationary obstacles, such as walls and tables, the danger zones are relatively small, while for mobile obstacles, such as humans and vehicles, the danger zones are larger. Specifically, the shape of the danger zone can be related to the direction and speed of movement of the mobile obstacle. As an example, in [45], the real-time human behavior is analyzed to generate danger zones for safe navigation in crowded scenes.

Due to the noise of sensors and action uncertainty, a robot might apply an action to an object but fail to change its state. For example, it can fail to grasp a bottle even after the grasp action. If there is no action verification step, all the following actions are vain. Utilizing the semantic information as a validation, like detecting whether there is still a bottle at the grasping point, is important to help the robot correct such mistakes in time.

Sometimes, the description of a task allocated to a robot is ambiguous. For example, in PPM task, instead of asking the robot to pick an object at location A and put it at location B, the task could be described as picking an object in the living room and putting it in the kitchen. Without the precise location instruction, the robot should recognize and understand the environment, like

judging whether the current room is the kitchen. Many research on semantic environment modelling and navigation can be exploited, such as the work proposed in [5] where the robot navigates with consideration of semantic information of environment and chooses suitable route to destination.

In summary, TAMP has a long way to go to provide the capability to solve diverse kind of tasks in a unified approach, and a complete and coherent environment and context modelling is a key element in this direction.

### 5.4 Balance between optimum and feasibility

This last challenge is on the definition of the TAMP final objective, which can be either to find an optimal solution or limited to finding a feasible solution. For example, in NAMO task, it might happen that the robot can either bypass an obstacle through moving a longer distance or clear the obstacle and pass it, which may correspond (depending on the particular situation) to a feasible solution and to the optimal solution respectively. The choice between these two solutions is of higher level than the TAMP task, and could be made by considering the semantic background, such as whether the task is urgent or whether one wants to secure its completion by avoiding more uncertain actions (such as manipulation). But it is therefore interesting to develop TAMP approaches that can provide sets of solutions with different characteristics, such as different optimality/risk tradeoffs.

## 6 Conclusion

This paper, as an extension of [59], reviews recent advances in TAMP, including tasks, useful simulation environments, approaches and existing challenges. Three fundamental tasks, namely rearrangement, navigation among movable obstacles and pick-place-move task, are detailed. To facilitate the experiments, this paper also illustrates some simulation environments so the readers may readily select the appropriate experiment environment based on their needs. Besides, certain TAMP approaches are categorized into three groups based on whether they employ machine learning methodologies. Furthermore, their experiment tasks are also listed, which helps readers to easily choose a baseline according to their objectives and prior knowledge. Finally, we highlight the current difficulties with the goal of identifying potential investigation directions, in particular the need of algorithms that are more resilient to perception and action uncertainty and that can leverage environment semantics.

## ACKNOWLEDGEMENTS

This work was carried out in the scope of OTPaaS project. This project has received funding from the French government as part of the “Cloud Acceleration Strategy” call for manifestation of interest.

## References


1. Barto, A.G., Mahadevan, S.: Recent advances in hierarchical reinforcement learning. Discrete event dynamic systems **13**(1), 41–77 (2003)
2. Chitnis, R., Hadfield-Menell, D., Gupta, A., Srivastava, S., Groshev, E., Lin, C., Abbeel, P.: Guided search for task and motion plans using learned heuristics. In: 2016 IEEE International Conference on Robotics and Automation (ICRA). pp. 447–454. IEEE (2016)
3. Coumans, E., Bai, Y.: Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org (2016–2021)
4. Driess, D., Ha, J.S., Toussaint, M.: Deep visual reasoning: Learning to predict action sequences for task and motion planning from an initial scene image. In: Robotics: Science and Systems 2020 (RSS 2020). RSS Foundation (2020)
5. Drouilly, R., Rives, P., Morisset, B.: Semantic representation for navigation in large-scale environments. In: 2015 IEEE International Conference on Robotics and Automation (ICRA). pp. 1106–1111. IEEE (2015)
6. Eysenbach, B., Salakhutdinov, R.R., Levine, S.: Search on the replay buffer: Bridging planning and reinforcement learning. Advances in Neural Information Processing Systems **32** (2019)
7. Gan, C., Schwartz, J., Alter, S., Mrowca, D., Schrimpf, M., Traer, J., De Freitas, J., Kubilius, J., Bhandwaldar, A., Haber, N., et al.: Threedworld: A platform for interactive multi-modal physical simulation. In: Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1) (2021)
8. Garrett, C.R., Chitnis, R., Holladay, R., Kim, B., Silver, T., Kaelbling, L.P., Lozano-Pérez, T.: Integrated task and motion planning. Annual review of control, robotics, and autonomous systems **4**, 265–293 (2021)
9. Garrett, C.R., Lozano-Pérez, T., Kaelbling, L.P.: Ffrob: An efficient heuristic for task and motion planning. In: Algorithmic Foundations of Robotics XI, pp. 179–195. Springer (2015)
10. Garrett, C.R., Lozano-Perez, T., Kaelbling, L.P.: Ffrob: Leveraging symbolic planning for efficient task and motion planning. The International Journal of Robotics Research **37**(1), 104–136 (2018)
11. Garrett, C.R., Lozano-Pérez, T., Kaelbling, L.P.: Sampling-based methods for factored task and motion planning. The International Journal of Robotics Research **37**(13-14), 1796–1825 (2018)
12. Garrett, C.R., Lozano-Pérez, T., Kaelbling, L.P.: Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. In: Proceedings of the International Conference on Automated Planning and Scheduling. vol. 30, pp. 440–448 (2020)
13. Garrett, C.R., Paxton, C., Lozano-Pérez, T., Kaelbling, L.P., Fox, D.: Online replanning in belief space for partially observable task and motion problems. In: 2020 IEEE International Conference on Robotics and Automation (ICRA). pp. 5678–5684. IEEE (2020)
14. Ghallab, M., Howe, A., Knoblock, C., Mcdermott, D., Ram, A., Veloso, M., Weld, D., Wilkins, D.: PDDL—The Planning Domain Definition Language (1998), http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.37.212
15. Ghallab, M., Nau, D., Traverso, P.: Automated planning and acting. Cambridge University Press (2016)
16. Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al.: Soft actor-critic algorithms and applications. arXiv preprint arXiv:1812.05905 (2018)

17. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE transactions on Systems Science and Cybernetics **4**(2), 100–107 (1968)
18. Heess, N., Wayne, G., Tassa, Y., Lillicrap, T., Riedmiller, M., Silver, D.: Learning and transfer of modulated locomotor controllers. arXiv preprint arXiv:1610.05182 (2016)
19. Kaelbling, L., Lozano-Perez, T.: Hierarchical task and motion planning in the now. In: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA (2010)
20. Kaelbling, L.P., Lozano-Pérez, T.: Integrated task and motion planning in belief space. The International Journal of Robotics Research **32**(9-10), 1194–1227 (2013)
21. Kim, B., Kaelbling, L.P., Lozano-Pérez, T.: Adversarial actor-critic method for task and motion planning problems using planning experience. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 8017–8024 (2019)
22. Kim, B., Shimanuki, L.: Learning value functions with relational state representations for guiding task-and-motion planning. In: Conference on Robot Learning. pp. 955–968. PMLR (2020)
23. Kim, B., Shimanuki, L., Kaelbling, L.P., Lozano-Pérez, T.: Representation, learning, and planning algorithms for geometric task and motion planning. The International Journal of Robotics Research **41**(2), 210–231 (2022)
24. Kim, B., Wang, Z., Kaelbling, L.P., Lozano-Pérez, T.: Learning to guide task and motion planning using score-space representation. The International Journal of Robotics Research **38**(7), 793–812 (2019)
25. King, J.E., Cognetti, M., Srinivasa, S.S.: Rearrangement planning using object-centric and robot-centric action spaces. In: 2016 IEEE International Conference on Robotics and Automation (ICRA). pp. 3940–3947. IEEE (2016)
26. Koenig, N., Howard, A.: Design and use paradigms for gazebo, an open-source multi-robot simulator. In: 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566). vol. 3, pp. 2149–2154. IEEE
27. Kolve, E., Mottaghi, R., Han, W., VanderBilt, E., Weihs, L., Herrasti, A., Gordon, D., Zhu, Y., Gupta, A., Farhadi, A.: Ai2-thor: An interactive 3d environment for visual ai. arXiv preprint arXiv:1712.05474 (2017)
28. Konidaris, G., Kaelbling, L.P., Lozano-Perez, T.: From skills to symbols: Learning symbolic representations for abstract high-level planning. Journal of Artificial Intelligence Research **61**, 215–289 (2018)
29. Kuffner, J.J., LaValle, S.M.: 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, pp. 995–1001. IEEE (2000)
30. Kulkarni, T.D., Narasimhan, K., Saeedi, A., Tenenbaum, J.: Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. Advances in neural information processing systems **29** (2016)
31. LaValle, S.M.: Planning algorithms. Cambridge university press (2006)
32. Levy, A., Konidaris, G., Platt, R., Saenko, K.: Learning multi-level hierarchies with hindsight. In: International Conference on Learning Representations (2018)
33. Li, C., Xia, F., Martín-Martín, R., Lingelbach, M., Srivastava, S., Shen, B., Vainio, K.E., Gokmen, C., Dharan, G., Jain, T., et al.: igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. In: 5th Annual Conference on Robot Learning (2021)

34. Li, C., Xia, F., Martin-Martin, R., Savarese, S.: Hrl4in: Hierarchical reinforcement learning for interactive navigation with mobile manipulators. In: Conference on Robot Learning. pp. 603–616. PMLR (2020)
35. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. In: ICLR (Poster) (2016)
36. Lu, D.V., Hershberger, D., Smart, W.D.: Layered costmaps for context-sensitive navigation. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 709–715. IEEE (2014)
37. Meng, Z., Sun, H., Teo, K.B., Ang, M.H.: Active path clearing navigation through environment reconfiguration in presence of movable obstacles. In: 2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM). pp. 156–163. IEEE (2018)
38. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. nature **518**(7540), 529–533 (2015)
39. Nachum, O., Gu, S., Lee, H., Levine, S.: Near-optimal representation learning for hierarchical reinforcement learning. In: International Conference on Learning Representations (2018)
40. Nachum, O., Gu, S.S., Lee, H., Levine, S.: Data-efficient hierarchical reinforcement learning. Advances in neural information processing systems **31** (2018)
41. Pasula, H.M., Zettlemoyer, L.S., Kaelbling, L.P.: Learning symbolic models of stochastic domains. Journal of Artificial Intelligence Research **29**, 309–352 (2007)
42. Patel, U., Kumar, N.K.S., Sathyamoorthy, A.J., Manocha, D.: Dwa-rl: Dynamically feasible deep reinforcement learning policy for robot navigation among mobile obstacles. In: 2021 IEEE International Conference on Robotics and Automation (ICRA). pp. 6057–6063. IEEE (2021)
43. Puig, X., Ra, K., Boben, M., Li, J., Wang, T., Fidler, S., Torralba, A.: Virtualhome: Simulating household activities via programs. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8494–8502 (2018)
44. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y., et al.: Ros: an open-source robot operating system. In: ICRA workshop on open source software. vol. 3, p. 5. Kobe, Japan (2009)
45. Samsani, S.S., Muhammad, M.S.: Socially compliant robot navigation in crowded environment by human behavior resemblance using deep reinforcement learning. IEEE Robotics and Automation Letters **6**(3), 5223–5230 (2021)
46. Silver, T., Chitnis, R., Tenenbaum, J., Kaelbling, L.P., Lozano-Pérez, T.: Learning symbolic operators for task and motion planning. In: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 3182–3189. IEEE (2021)
47. Srivastava, S., Fang, E., Riano, L., Chitnis, R., Russell, S., Abbeel, P.: Combined task and motion planning through an extensible planner-independent interface layer. In: 2014 IEEE international conference on robotics and automation (ICRA). pp. 639–646. IEEE (2014)
48. Sun, H., Zhang, W., Yu, R., Zhang, Y.: Motion planning for mobile robots—focusing on deep reinforcement learning: A systematic review. IEEE Access **9**, 69061–69081 (2021)
49. Szot, A., Clegg, A., Undersander, E., Wijmans, E., Zhao, Y., Turner, J., Maestre, N., Mukadam, M., Chaplot, D., Maksymets, O., Gokaslan, A., Vondrus, V., Dharur, S., Meier, F., Galuba, W., Chang, A., Kira, Z., Koltun, V., Malik, J., Savva, M., Batra, D.: Habitat 2.0: Training home assistants to rearrange their habitat. In: Advances in Neural Information Processing Systems (NeurIPS) (2021)

50. Toussaint, M.: Logic-geometric programming: An optimization-based approach to combined task and motion planning. In: Twenty-Fourth International Joint Conference on Artificial Intelligence (2015)
51. Toyer, S., Thiébaux, S., Trevizan, F., Xie, L.: Asnets: Deep learning for generalised planning. Journal of Artificial Intelligence Research **68**, 1–68 (2020)
52. Wang, M., Luo, R., Önol, A.Ö., Padir, T.: Affordance-based mobile robot navigation among movable obstacles. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 2734–2740. IEEE (2020)
53. Wang, Z., Garrett, C.R., Kaelbling, L.P., Lozano-Pérez, T.: Active model learning and diverse action sampling for task and motion planning. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 4107–4114. IEEE (2018)
54. Wang, Z., Garrett, C.R., Kaelbling, L.P., Lozano-Pérez, T.: Learning compositional models of robot skills for task and motion planning. The International Journal of Robotics Research **40**(6-7), 866–894 (2021)
55. Xia, F., Li, C., Martín-Martín, R., Litany, O., Toshev, A., Savarese, S.: Relmogen: Integrating motion generation in reinforcement learning for mobile manipulation. In: 2021 IEEE International Conference on Robotics and Automation (ICRA). pp. 4583–4590. IEEE (2021)
56. Xiang, F., Qin, Y., Mo, K., Xia, Y., Zhu, H., Liu, F., Liu, M., Jiang, H., Yuan, Y., Wang, H., et al.: Sapien: A simulated part-based interactive environment. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11097–11107 (2020)
57. Younes, H.L., Littman, M.L.: Ppddl1. 0: An extension to pddl for expressing planning domains with probabilistic effects. Techn. Rep. CMU-CS-04-162 **2**, 99 (2004)
58. Zeng, K.H., Weihs, L., Farhadi, A., Mottaghi, R.: Pushing it out of the way: Interactive visual navigation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9868–9877 (2021)
59. Zhang, K., Lucet, E., Sandretto, J.A.D., Kchir, S., Filliat, D.: Task and motion planning methods: applications and limitations. In: 19th International Conference on Informatics in Control, Automation and Robotics ICINCO 2022). pp. 476–483. SCITEPRESS-Science and Technology Publications (2022)
60. Zhang, X., Zhu, Y., Ding, Y., Zhu, Y., Stone, P., Zhang, S.: Visually grounded task and motion planning for mobile manipulation. arXiv preprint arXiv:2202.10667 (2022)
61. Zhu, Y., Mottaghi, R., Kolve, E., Lim, J.J., Gupta, A., Fei-Fei, L., Farhadi, A.: Target-driven visual navigation in indoor scenes using deep reinforcement learning. In: 2017 IEEE international conference on robotics and automation (ICRA). pp. 3357–3364. IEEE (2017)