Reasoning and Learning in Real-World Systems for Long-Term Autonomy (LTA)
Papers from the AAAI 2018 Fall Symposium

Towards Perception Aware Task-Motion Planning

Antony Thomas, Sunny Amatya, Fulvio Mastrogiovanni, Marco Baglietto
Department of Informatics, Bioengineering, Robotics, and Systems Engineering
University of Genoa, Via All'Opera Pia, 13
{antony.thomas@dibris.unige.it, sunnyamatya@gmail.com, fulvio.mastrogiovanni@unige.it, marco.baglietto@unige.it}

Abstract

We present an integrated Task-Motion Planning (TMP) framework in belief space. Autonomous robots operating in real world complex scenarios require planning in the discrete (task) space and the continuous (motion) space. We develop a framework for integrating belief space reasoning within a hybrid task planner. The expressive power of PDDL+ combined with heuristic driven semantic attachments performs the propagated and posterior belief estimates while planning. The underlying methodology for the development of the hybrid planner is discussed, providing suggestions for further improvements and future work. Furthermore, our approach is unified with ROSPlan and we validate key aspects of our approach using a realistic synthetic simulation.

1 Introduction

Autonomous robots operating in complex real world scenarios require different levels of planning to execute their tasks. High-level (task) planning helps to break down a given set of tasks into a sequence of sub-tasks, depending on the required level of abstraction. Actual execution of each of these subtasks would require low-level control actions (i.e., motions). Hence, planning should be performed in the task-motion or the discrete-continuous space.

Traditionally, task planning and motion planning have evolved as two independent fields. Since the seminal work of Fikes and Nilsson (Fikes and Nilsson 1971), planning has emerged as a specific field within AI. Techniques like, planning graphs, planning as satisfiability and heuristic-search planning have led to dominant task planning approaches like Fast Downward (Helmert 2006) and FF (Hoffmann and Nebel 2001) planning systems, that are capable of highly efficient planning in larger domains, handling mathematical expressions and temporal aspects. Similarly, the field of motion planning has evolved profusely. Probabilistic Roadmaps (PRMs) (Kavraki et al. 1996; Kavraki, Kolountzakis, and Latombe 1998) and Rapidly-exploring Random Trees (RRTs) (Kuffner and LaValle 2000) are the two notable approaches for high-dimensional motion planning. These are *Sampling based approaches* providing faster execution and better efficiency.

In recent years, combining high-level task planning with the low-level motion planning has been a subject of great interest among the Robotics and Artificial Intelligence (AI) community. This is inevitable as one of the ultimate goals in Robotics is to create autonomous agents accepting high-level task descriptions and executing them without further human intervention. Planning frameworks such as the Planning Domain Definition Language (PDDL) (McDermott et al. 1998) mainly focus on high-level task planning supposing that the geometric preconditions (e.g., grasping poses for a pick-up task (Srivastava et al. 2014)) for the robot motion to carry out these tasks are achievable. However, in reality, such an assumption can be catastrophic as an action or sequence of actions generated by the task planning algorithm might turn out to be unfeasible at the controller execution level.

Let us consider a simple scenario where a robot is given the task of picking up an object. In terms of task planning, a *pick_up* action would suffice, subject to satisfying the action preconditions, i.e., the robot hand being free and the object being graspable. However, it is possible that the robot is too close to the object and the *pick_up* action cannot be performed due to robot's end-effector's reachability workspace. This would require the robot to assume a different grasping pose, invoking a motion command that leads to a suitable position. Though a simple scenario, it clearly illustrates the need for a combined TMP strategy. Several recent works (Cambon, Alami, and Gravot 2009; Kaelbling and Lozano-Pérez 2012; Dornhege et al. 2012; Srivastava et al. 2014; Toussaint 2015; Dantam et al. 2018) have motivated the need for a combined Task-Motion Planning (TMP) approach pointing out the drawbacks of treating them as separate.

Given an initial state and a suitable goal state, Task-Motion Planning (TMP) synthesizes a plan interleaving discrete high-level tasks with continuous low-level motion. In robotics planning, reaching a goal pose x_g from an initial pose x_0 requires continuous collision-free motion planning. However, reaching a goal alone is not sufficient as most often we require the goal condition to be satisfied subject to application dependent costs. As a result, certain decisions are to be made with regards to visiting specific landmarks, the order and type of actions to be performed. Yet, the most important challenge for TMP problems is finding the right correspondence between the task planner and the motion planner. Given a discrete action, a TMP should be able to recognize the corresponding geometry requirements to trig-

ger the action. Similarly, once the action is triggered, the corresponding motion planning problems are to be identified.

Furthermore, real-world scenarios often induce uncertainties. Such uncertainties arise due to insufficient knowledge about the environment, inexact robot motion or imperfect sensing. In such scenarios, the robot poses or other variables of interest can only be dealt with in terms of probabilities. Planning is therefore done in the *belief* space, which corresponds to the probability distributions over possible robot states. Consequently, for efficient planning and decision making, it is required to reason about future belief distributions due to candidate actions and the corresponding expected observations. Such a problem falls under the category of partially observable Markov decision processes (POMDPs). Hence, the task planner should be capable of reasoning in the belief space while synthesizing a plan. Besides, the task planner also requires some amount of geometrical information about the environment and the robot itself, the lack of which leads to undesirable plans. At the execution level, the motion planner might encounter unexpected scenarios notwithstanding the plan provided. This calls for a re-plan, updating the task planner with the new belief, resulting in a cyclic interdependency. Consequently, both task and motion planning are interdependent and should not be considered as separate processes.

Our contributions in this paper are as follows: (1) developing an integrated TMP algorithm for planning in the belief space. (2) The expressive power of PDDL+ is exploited to simulate robot motion and the belief updates within PDDL+. Our domain description can hence be employed for any mobile robot planning problem in general. (3) We sample landmark observable viewpoints, thereby facilitating perception aware TMP. Our sampling strategy is also directed towards parsimoniously selecting connected waypoints thereby reducing the state space explosion (4) The developed framework is also unified with ROSPlan (Cashmore et al. 2015), which is a framework for embedding task planning in ROS and hence can be easily adapted by the planning community. Furthermore, we provide an interface for PDDL+ planning in ROSPlan.

2 Related Work

The genesis of TMP can be credited to Fikes and Nilsson for their work on STRIPS (Fikes and Nilsson 1971) which further led to the Shakey project (Nilsson 1984). Shakey's planner had access to basic geometric knowledge like the objects in a room, connectivity between rooms. However, it performed a logical search first, assuming that the resulting robot motion plans can be formulated. This assumption limits the capability of the agent as the high-level actions may turn out to be non executable due to geometric limitations. Later works either carried out the generated plans, validating them using a robot motion planner (Dornhege et al. 2009) or performed a combined search in the logical and geometric spaces using a state composed of the both the symbolic and geometric paths (Cambon, Alami, and Gravot 2009). The aSyMov planner used in (Cambon, Alami, and Gravot 2009) used a combination of Metric-FF (Hoffmann 2003) and a sampling based motion planner. In contrast, we use a hybrid temporal task planner (Piotrowski et al. 2016) incorporating robot state uncertainty. Srivastava et al. implicitly incorporate geometric variables in a PDDL-based planning model (Srivastava et al. 2014). An interface layer then converts PDDL plans to numeric values of the symbols to check the validity of each action in the configuration space.

Many of the planning problems need to plan well ahead of time which also results in increased dimensionality, as more and more objects and constraints get added. Longer planning horizons and higher dimensionality directly affects the computational time for these planners. Kaelbling and Lozano-Péres (2012) propose a hierarchical approach that tightly integrates the logical and geometric planning. The complexities arising out of long horizon planning are tackled to the extent that planning is done at different levels of abstraction, thereby reducing the long horizons to a number of feasible sub-plans of shorter horizon. This regression-based planner assumes that the actions are reversible while backtracking. In contrast to their earlier work the serializability assumption of the subgoals is relaxed. Kaelbling and Lozano-Péres (2013) further extended their work to consider the current state uncertainty, modeling the planning problem in the belief space. Uncertain outcomes are modeled by converting a Markov decision processes (MDP) into a weighted graph, thereby modifying their earlier approach of *hierarchical planning in the now*. Belief update is then performed when observations are obtained. Phiquepal and Toussaint (2017) discuss an ongoing work for TMP under partial observability, computing long-horizon policies that are arborescent in nature.

The above discussed approaches focus on finding feasible plans sacrificing optimality and hence emphasizes on performance. Toussaint (2015) performs optimization over an objective function based on the final geometric configuration (and the cost thereby), finding approximately locally optimal solutions by minimizing the objective function. The planning problem is modeled as a constraint satisfaction problem with symbolic states used to define the constraints in the optimization. Lozano-Péres and Kaelbling (2014) model the motion planning as a constraint satisfaction problem over a subset of the configuration space. Iteratively Deepened Task and Motion Planning (IDTMP) is a constraint based task planning approach that incorporates geometric information (motion feasibility) at the task planning level (Dantam et al. 2018). In our approach, the waypoints fed into the task planner are generated using the motion planner, similar to the motion planner information that guides the IDTMP task planner.

3 Preliminaries

Let x_k denote the robot pose at any time k defined by $x_k \doteq (x, y, \theta)$, x and y are the robot Cartesian coordinates and θ is the heading. We use z_k to denote the measurement acquired at time k and u_k for the control action applied at time k. Extended Kalman Filter (EKF) is used to represent the robot belief at a given time, using the robot state mean μ_k and covariance Σ_k. The robot dynamics is modeled using the standard odometry based motion model

$$
\begin{aligned}
x' &= x + \delta_{trans} \cdot \cos(\theta + \delta_{rot1}) \\
y' &= y + \delta_{trans} \cdot \sin(\theta + \delta_{rot1}) \\
\theta' &= \theta + \delta_{rot1} + \delta_{rot2}
\end{aligned} \tag{1}
$$

where $u_k \doteq (\delta_{rot1}, \delta_{trans}, \delta_{rot2})$ is the control applied. For brevity we write Eq. 1 as $x_{k+1} = f(x_k, u_k)$. We assume Gaussian noise and the process covariance is given by

$$
W_k = \begin{bmatrix} \alpha_1 \cdot \delta_{rot1}^2 + \alpha_2 \cdot \delta_{trans}^2 & 0 & 0 \\ 0 & \alpha_3 \cdot \delta_{trans}^2 + \alpha_4 \cdot (\delta_{rot1}^2 + \delta_{rot2}^2) & 0 \\ 0 & 0 & \alpha_2 \cdot \delta_{trans}^2 + \alpha_1 \cdot \delta_{rot2}^2 \end{bmatrix} \tag{2}
$$

where α_1-α_4 are robot-specific error parameters (Thrun, Burgard, and Fox 2005) modeling the accuracy of the robot motion. To process the landmarks in the environment we measure the range and the bearing of the landmark relative to the robot's local coordinate frame. It is to be noted that we assume the data association problem is solved and hence given a measurement we know the corresponding landmark that generated it. Such a model can be represented by

$$
z_k = \begin{bmatrix} r \\ \phi \end{bmatrix} + v_k \ , \ v_k \sim \mathcal{N}(0, Q_k) \tag{3}
$$

where r, ϕ is the range and bearing respectively and v_k the zero-mean Gaussian noise. For brevity, Eq. 3 will be written as $z_k = h(x_k, l^i)$

The motion (Eq. 1) and observation (Eq. 3) models can be written probabilistically as $p(x_{k+1}|x_k, u_k)$ and $p(z_k|x_k)$ respectively. Given an initial distribution $p(x_0)$, and the motion and observation models, the posterior probability distribution at time k can be written as

$$
p(X_k|Z_k, U_{k-1}) = p(x_0) \prod_{i=1}^{k} p(x_k|x_{k-1}, u_{k-1}) p(z_k|x_k) \tag{4}
$$

where $X_k = \{x_0, ..., x_k\}$, $Z_k = \{z_0, ..., z_k\}$ and $U_k = \{u_0, ..., u_{k-1}\}$. This posterior probability distribution is the *belief* at time k, denoted by $b[X_k] \sim \mathcal{N}(\mu_k, \Sigma_k)$. Similarly, given an action u_k, the propagated belief can be written as

$$
b[\bar{X_{k+1}}] = p(X_k|Z_k, U_{k-1}) p(x_{k+1}|x_k, u_k) \tag{5}
$$

Given the current belief $b[X_k]$, the control u_k, the propagated belief parameters can be computed using the standard EKF prediction as

$$
\begin{aligned}
\bar{\mu}_{k+1} &= f(\mu_k, u_k) \\
\bar{\Sigma}_{k+1} &= F_k \Sigma_k F_k^T + V_k W_k V_k^T
\end{aligned} \tag{6}
$$

where F_k and V_k are the Jacobians of $f(\cdot)$ with respect to x_k and u_k respectively. For ease of representation, we denote $R_k \doteq V_k W_k V_k^T$. Upon receiving a measurement z_k, the posterior belief $b[X_{k+1}]$ is computed using the EKF update equations

$$
\begin{aligned}
K_k &= \bar{\Sigma}_{k+1} H_k^T (H_k \bar{\Sigma}_{k+1} H_k^T + Q_k)^{-1} \\
\mu_{k+1} &= \bar{\mu}_{k+1} + K_k (z_{k+1} - h(\bar{\mu}_{k+1}, l^i)) \\
\Sigma_{k+1} &= (I - K_k H_k) \bar{\Sigma}_{k+1}
\end{aligned} \tag{7}
$$

where H_k is the Jacobian of $h(\cdot)$ with respect to x, K_k is the Kalman gain and $\mathbb{I} \in \mathbb{R}^{3\times3}$.

4 TMP Design and Implementation

In this Section we detail our TMP planner concept and approach. We begin by making the following observation. Planning in the belief space to obtain an optimal control policy essentially requires synthesizing a sequence of actions that minimize an application dependent objective function. Finding such an action sequence inherently involves searching in the motion space. Consequently, we employ task planning to perform this search.

4.1 Rationale and Scenario

PDDL based planning frameworks are restricted, as they are incapable of handling rigorous numerical calculations. Most approaches perform such calculations via an external module or *semantic attachments*, e.g. (Dornhege et al. 2012). The term semantic attachment was coined by Weyhrauch (1980) to describe attaching algorithms to function and predicate symbols via external procedure. Yet, the effects returned by these semantic attachments are not exploited in identifying *helpful actions* and hence do not provide any heuristic guidance, deeming the task unsolvable most often. An action is considered *helpful* if it achieves at least one of the lowest level goals in the relaxed plan to the state at hand (Hoffmann 2003). Recently Bernardini et al. (2017) developed a PDDL based POPF-TIF planner to implicitly trigger such external calls via a specialized semantic attachments called *external advisors*. They classify variables into direct, indirect and free variables. Direct (free) variables are the normal PDDL function variables whose values are changed in the action effects, in accordance with the PDDL semantics. The indirect variables are affected by the changes in the direct variables. A change in a direct variable triggers the external advisor which in turn updates the indirect variables. POPF-TIF is based on the temporal extension of the metric-FF planner (Hoffmann 2003). An intriguing feature of the planner is that it uses approximate values of the indirect variables at the Temporal Relaxed Plan Graph (TRPG) construction stage resulting in an efficient goal-directed search. During the forward state space search, the external advisor is called, updating the indirect variables with the exact values.

Leveraging the ROSPlan framework (Cashmore et al. 2015) and using semantic attachments that incorporate heuristic evaluation during the Relaxed Plan Graph (RPG) construction, we develop a hybrid planning framework capable of reasoning in the robot belief space while synthesizing a plan. We use PDDL+ (Fox and Long 2006) to model the planning task, providing the robot with a sequence of actions that can be passed on to the low-level controller for execution. PDDL+ provides the ability to model continuous temporal change via *processes* and discrete exogenous activities in the environment via *events*. The processes are similar to durative actions and the events are akin to instantaneous actions. However, processes and events are distinct from actions since a process or an event is triggered as soon as its precondition is satisfied whereas an action trigger depends on the planner search strategy. State uncertainty is incorporated in our model and synthesizing an efficient plan

requires performing the belief updates within the task planner. PDDL+ *processes* enable the simulation of robot motion with time and the *events* are leveraged to perform the corresponding belief updates. In our case, we use the DiNo planner (Piotrowski et al. 2016) since it enables heuristic search for linear and non-linear systems using the entire set of PDDL+ features.

In this paper, we consider a mobile robot in a known environment (i.e., map is given) with uncertainty in its initial pose. The set of landmarks in the environment are given by $l = \{l^1, l^2, ..., l^n\}$. The landmarks are features in the environment and are not to be confused with the landmarks in heuristic planning where they are intended as a set of operators such that each plan must contain some element of this set. The goal is to reach a certain final state x_g with the localization uncertainty not greater than a given bound. Starting from an initial pose, the corresponding goal leading plan is to be synthesized, minimizing the makespan. We use the trace of the state covariance to quantify the uncertainty and the uncertainty condition is mathematically written as $\mathrm{Tr}(\Sigma_k) < \eta$. To incorporate belief evolution while planning, the DiNo planner is extended to support external calls evaluating the belief at each planning stage. The belief, the process and the measurement noise are assumed to be Gaussian.

Our TMP approach depends directly on the temporal planning horizon and the PDDL+ process discretization used. A longer horizon and shorter discretization increases planning complexity and directly affects the plan time. In Section 5.2, we evaluate the performance based on these factors and analyze how our approach cope with the changes in these elements.

4.2 Planner Workflow

An overview of our TMP planner framework is shown in Figure 1. We assume that the environment map, robot initial belief $\mathcal{N}(\mu_0, \Sigma_0)$, the goal pose to be reached x_g and the minimum pose certainty η required at goal are known. As discussed in the beginning of the Section, we utilize task planning to synthesize a plan, performing search in the motion space. Standard RRT based approaches sample waypoints that connects the start and end locations. However, to reduce pose uncertainty it is to be ensured that such connected paths have ample number of waypoints from which landmarks can be observed. To facilitate this perception-aware search we implement an RRT based potential field approach to sample such relevant waypoints in the environment. Waypoints near the potential field of the landmarks are pulled closer towards it and once a sufficient number (currently user defined) of such waypoints are generated, the further nodes are pushed away from the potential field. This sampled set of waypoints will be denoted as $wp = \{wp^1, ..., wp^m\}$.

The PDDL+ based event *belief update* triggers the semantic attachment call to the external library. The external library performs the belief updates (Eqs. 6, 7) attaching to the event effects the updated belief. Our semantic attachment effects guides the staged RPG (SRPG) construction in DiNo. The belief estimates returned by the semantic attachments guides the SRPG in identifying the *helpful actions* (e.g., the landmarks that can be visited in the next state), besides providing a heuristic function to select the next best landmark to visit. Based on the heuristics, a breadth first algorithm is performed to determine the forward state space leading to a state graph. The plan synthesized from the state graph such that the makespan is minimized by choosing least cost path that also satisfies all the constraints.

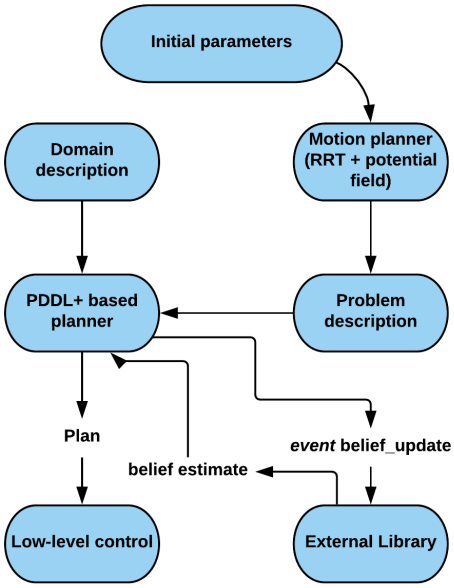


Figure 1: The TMP planner workflow.

4.3 External Calls in PDDL+

The PDDL+ description for our mobile robot scenario is shown in Figure 2. We extend the DiNo planner to incorporate semantic attachments, computing the propagated belief $b[\bar{X_{k+1}}]$ upon executing a control u_k at state x_k, as well as the posterior belief $b[X_{k+1}]$ upon obtaining a measurement. Since we are in the planning phase and yet to obtain observation, we simulate future observations z_{k+1} given the propagated belief $b[\bar{X_{k+1}}]$, the set of landmarks l and the measurement model given in Eq. (3. Given a pose $x \in b[\bar{X_{k+1}}]$, the nominal observation $\hat{z} = h(x, l^i)$ is corrupted with noise to obtain z_{k+1}.

Now we describe the formal algorithm for performing the belief updates. This procedure is summarized in Algorithm 1. The focal elements in the planning domain are: the action *goto_waypoint*, the event *belief_update* that triggers the external call to evaluate to perform the belief updates and the process *odometry* that simulates the robot motion between each planner discretization Δ. Starting from a given waypoint (line 2) the *goto_waypoint* action (line 3) initiates the robot motion towards a connected waypoint. The immediate effect of this action is to initialize the distance between the two waypoints (line 4) which starts the process *odometry* as seen in line 6. The process effect simulates the translational motion at each Δ and decreasing the distance between the waypoints by $\delta_{trans_k} = \Delta \times dFactor$ (line 7).

Event *belief_update* is immediately initiated (line 9) which triggers the semantic attachment call to perform belief update, returning the propagated belief $b[X_{k+1}]$ to the event effect (line 9). If a landmark $l^i \in l$ is within the sensor range, the posterior belief $b[X_{k+1}]$ is evaluated returning its trace (line 11) to the event effect. To ensure the *process-event-process* ordering we employ a variable *counter* as shown in Figure 2.

Algorithm 1 Belief update implementation in PDDL+

Input: Set of waypoints wp, set of landmarks l, trace of initial pose covariance $\mathrm{Tr}(\Sigma_0)$, upper bound on the goal trace η, motion discretization factor $dFactor$, PDDL+ process discretization Δ

1: **while** $\mathrm{Tr}(\Sigma_k) > \eta$ *and* $\neg$(robot_at *goalpose*)) **do**
2: (robot_at wp_from)
3: :action $goto\_waypoint$
4: $d(from,to)$ = distance(wp_from, wp_to)
5: **while** $d(from,to) > -\delta_{trans_k}$ **do**
6: :process $odometry$
7: $d(from,to) \leftarrow d(from,to) - \delta_{trans\_k}$
8: :event $belief\_update$
9: $\mathrm{Tr}(\bar{\Sigma}_{k+1}) \leftarrow \mathrm{Tr}(\Sigma_k)$ ▷ Eq. 6
10: **if** landmark within sensor range **then**
11: $\mathrm{Tr}(\Sigma_{k+1}) \leftarrow \mathrm{Tr}(\bar{\Sigma}_{k+1})$ ▷ Eq. 7
12: **end if**
13: **end while**
14: :action *reached*
15: (robot_at wp_from) $\leftarrow$ (robot_at wp_to)
16: **end while**
17: **return** Plan

5 Experimental Results

We evaluate our approach in a simple yet realistic experiment in the Gazebo simulator. Consider the corridor environment as seen in Figure 3 where the turtlebot robot starting from the initial waypoint (s in figure) needs to reach near the only plug point or the goal waypoint (g in figure) to recharge the batteries. The turtlebot is initially oriented towards g. Due to the short length of the charging cable, given the mean goal pose, there is a bound on the maximum pose uncertainty the robot can afford. The cubes marked 1-4 are the landmarks in environment. The *slam_gmapping* ROS package is used to build the environment map. The resulting map of the environment is shown in Figure 4a. The turtlebot with its laser scanner can be seen facing the goal waypoint marked in blue.

In the remainder of this section we discusses a number of test cases performed with the same starting and goal waypoints as shown in Figure 3. It is to be noted that for each given landmark we consider a potential field originating at its center. With the inclusion of the occupancy grid obtained during the mapping and the potential field, the RRT is constructed maintaining a closer proximity to the landmarks. The computational complexity depends exponentially on the number of waypoints m generated. The heuris-

```
(define (domain landmark)

(:requirements :typing :durative-actions :fluents :time
:strips
:disjunctive-preconditions :durative-actions
:negative-preconditions :timed-initial-literals)

(:types
                waypoint
                robot
                covariance
)

(:predicates
                (robot_at ?r - robot ?wp - waypoint)
                (visited ?wp - waypoint)
                (observe)
                (moving ?r - robot ?to - waypoint)
                (connected ?from ?to - waypoint)
                (lessthan ?c ?f - covariance)
)

(:functions
        (distance ?wp1 ?wp2 - waypoint) (cov) (counter)
        (update_covariance)     (predict_covariance)
        (relativeD) (dFactor) (finalTrace)
)
;; relativeD- a variable to store the distance
between state and wp as robot moves
;; dFactor \times #t - distance factor, to see
how many times kalman prediction to be done
;; cov is the initial covarianc trace,
finalTrace- the required trace upon reaching the goal state
;; action - Move between any two waypoints,
along the straight line between the two waypoints

(:action goto_waypoint
         :parameters (?r - robot ?from ?to - waypoint)
         :precondition (and (robot_at ?r ?from)
      (observe) (not(robot_at ?r ?to))
(not (visited ?to)) (connected ?from ?to))
         :effect (and
(not (robot_at ?r ?from))
      (assign (relativeD) (distance ?from ?to))
          (moving ?r ?to) (not (observe))
      (assign (counter) 0)
      (increase (update_covariance) 0)
      (increase (predict_covariance) 0))
)

(:action reached
                :parameters (?r - robot ?to - waypoint)
                :precondition (and (moving ?r ?to)
        (<= (relativeD) 0))
                :effect (and
(robot_at ?r ?to) (visited ?to)
        (not (moving ?r ?to) (observe))
)

(:event belief_update
                :parameters ()
                :precondition (and (> (counter) 0) )
                :effect (and
(assign (cov)
        (update_covariance)) (assign (counter) 0))
)

;; to calculate the number of update steps needed
(:process odometry
                :parameters ()
                :precondition (and
(> (relativeD) (-dFactor)) )
                :effect (and
(decrease (relativeD) (* #t (dFactor)))
                (increase (counter) (* #t 1)))
)
)
```

Figure 2: Domain description for the mobile robot scenario. The *process* odometry is used to simulate the robot translation and the *event* belief_update performs the belief propagation and posterior computation using semantic attachments.

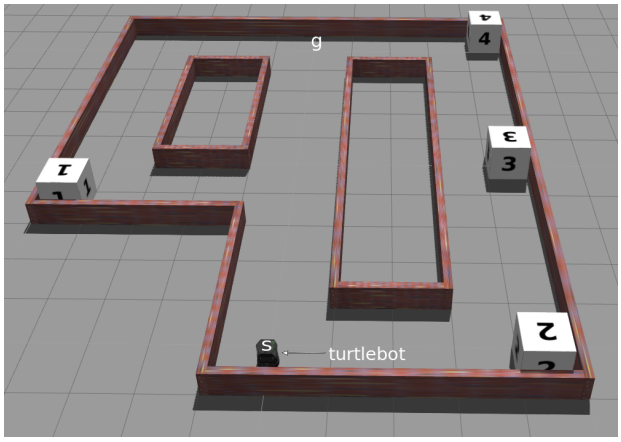


Figure 3: Corridor environment in Gazebo.

tic based search of the DiNo planner, reduces this state space explosion significantly. Furthermore, due to our potential field based RRT sampling, we are able to prune unwanted state expansions by generate parsimoniously connected waypoints which are sufficient for synthesizing satisficing plans.

5.1 ROSPlan Simulation

The initial variance in x, y, θ are 0.6m^2, 0.6m^2, 0.02rad respectively giving $\text{Tr}(\Sigma_0) = 1.22$. PDDL+ process discretization of $\Delta = 1$ and a temporal horizon of $T = 20$ is used but with different values of η. Motion discretization factor $dFactor = 1$, giving $\delta_{trans_k} = 1$ (see Section 4.3). Performing the belief updates at each δ_{trans_k} helps in pruning the nearby waypoints and thereby reducing the state space explosion. However in the *difficult* regions where one cannot afford to prune close-by waypoints, the updates are performed upon reaching each of these waypoints. Unless otherwise mentioned, the number of waypoints for each case $m = 40$. The different test cases are detailed below.

Case 1: For this case $\eta = 0.3$. The plan generated is shown via the red path in Figure 4b, starting form initial state to the goal state. This is plan is quite expected due to the landmark rich nature of the path and the tight bound on the final trace. *Case 2*: For this case $\eta = 0.6$. The plan generated is the one following the red path shown in Figure 4c. *Case 3* has the same $\eta = 0.6$, however the path followed is along the one with 3 landmarks (Figure 4d). This is due to the randomness of the potential field based RRT motion planner and the bound on the trace being less tight. Since the objective function requires the planning time to be minimized, the planner returns the path which minimizes the makespan which in turn is determined by the waypoint locations in the environment. As seen in Table 1, the states expanded and the planning time is much larger than in *Case 2* (around 4.8 times). This is due to the large number of connected waypoints. *Case 4* has $\eta = 0.6$ and the path followed (see Figure 4e) is similar to the one in *Case 3*. However, as evident from Figures 4e, in this case the number of connected waypoint is much less, resulting in a much lesser planning time (see Table 1). *case 5* (see Figure 4f) is similar to the *case 2* but with differences in the number of connected waypoints. This is reflected in the states expanded and the planning times as seen in Table 1.

Cases 6,7 and *8* have $\eta = 0.9$. However as discussed previously the sampling of waypoints affects the plan as can be seen in Figures 4g- 4i. In Figure 4g the plan generated is such that the trajectory upon reaching a waypoint near the goal, is extended to a waypoint near landmark 4, from which it can be observed. *Case 8* has the highest number of connected waypoints and this is reflected in the corresponding row in Table 1.

Case 9 is similar to *Case 1* but with higher number of connected waypoints. In *Case 10*, $m = 20$ and $\eta = 0.6$. The reduction in waypoint significantly reduces the state space explosion and the planning time, as can be seen in the last row of Table 1. However it can be seen in Figure 4k that the upper path has no waypoints sampled. Though it is true that for our all the cases discussed so far this doesn't cause any alarm, in general a lesser number of waypoints might resulting in no plans being produced, even though there exists one. For example, changing the starting or the gaol location in our experiments can lead to a path via landmark 1. Finding such a minimum number of waypoints is another planning problem by itself.

Test cases	States expanded	Plan time (s)
Case 1	6184	116.62
Case 2	6934	156.62
Case 3	33769	744.36
Case 4	2710	60.64
Case 5	4847	99.74
Case 6	7646	190.44
Case 7	7289	143.04
Case 8	56544	1119.00
Case 9	28595	684.60
Case 10	388	4.60

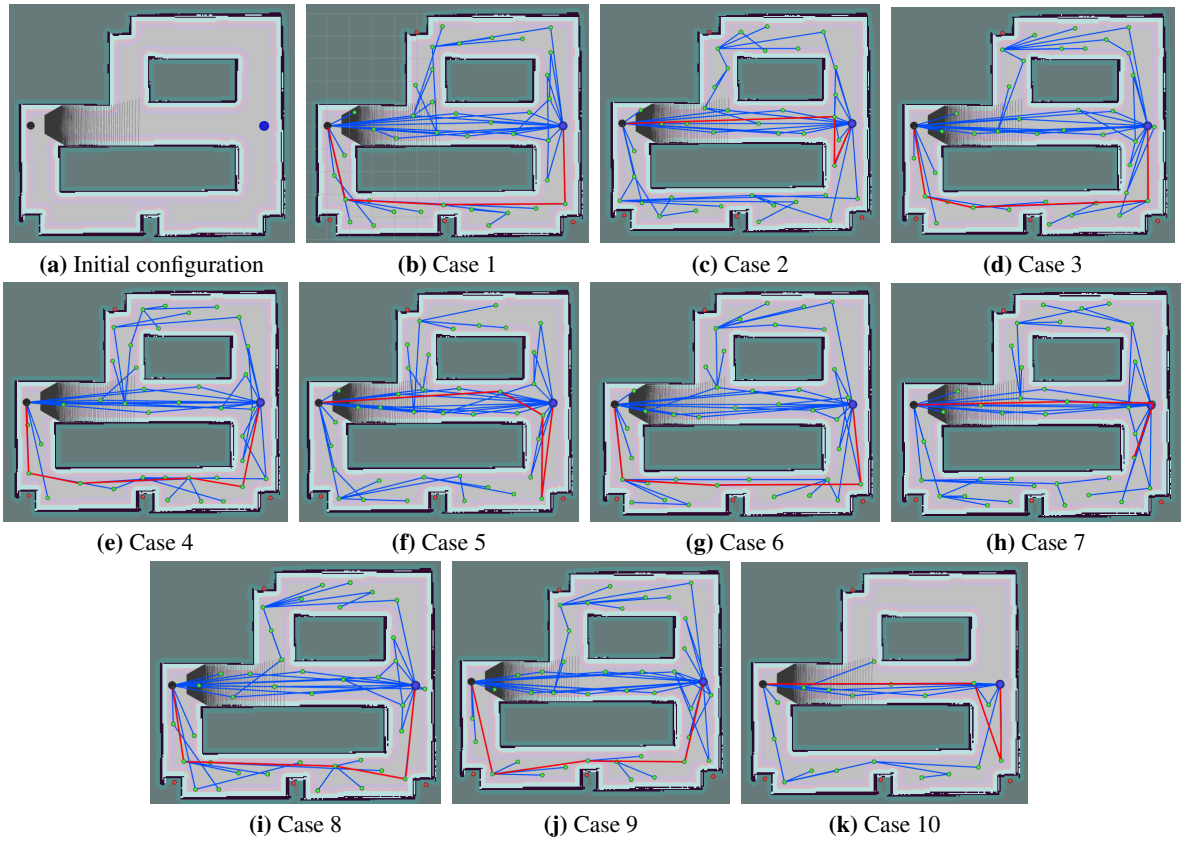
Table 1: Different test case with number of states expanded in each case and the corresponding planning time in seconds.

It is to be noted that in *Cases 2, 5, 7 and 10*, the robot might collide with the walls due to the initial covariance increase. This shows that a motion plan alone fails and logical reasoning of action effects is required in conjunction with motion planning. For example, rather than just the heuristic search, an A* algorithm incorporating the $\text{Tr}(\Sigma_g)$ in $g(\cdot)$ would prevent the generation of such flawed plans.

5.2 Performance Analysis

To analyze the performance based on the temporal planning horizon and the PDDL+ process discretization we perform different simulations using *Case1* in Section 5.1 as our base case. We use different discretization Δ and time horizon T and analyze the states explored, computation/plan time t and the goal covariance trace $\text{Tr}(\Sigma_g)$.

From table 2 we can observe that the planning time is inversely proportional to the discretization Δ. This is because we perform belief updates based on $\delta_{trans_k} = \Delta \times dFactor$.



(a) Initial configuration **(b)** Case 1 **(c)** Case 2 **(d)** Case 3

(e) Case 4 **(f)** Case 5 **(g)** Case 6 **(h)** Case 7

(i) Case 8 **(j)** Case 9 **(k)** Case 10

Figure 4: Initial configuration and the trajectories for the different test cases.

Δ	T	States expanded	t (s)	Tr(Σ_g)
0.5	20	47500	1231.60	0.076
0.5	25	112601	N/A	N/A
0.5	30	112601	N/A	N/A
0.5	35	112601	N/A	N/A
1.0	20	6184	121.88	0.153
1.0	25	12788	249.14	0.156
1.0	30	23533	566.44	0.143
1.0	35	41682	1092.96	0.181
2.0	20	750	17.56	0.267
2.0	25	1290	39.66	0.273
2.0	30	995	37.98	0.290
2.0	35	967	47.78	0.255

Table 2: Performance parameters for different temporal planning horizons and PDDL+ process discretization. N/A-planner ran out of memory.

Since $dFactor = 1$, the number of updates is directly proportional to Δ, thereby increasing the states expanded and hence the plan time. It is seen that the temporal horizon and total states searched are correlated, directly affecting the total plan time. It can be observed from the Table that for a longer temporal horizon T and a larger Δ, the goal condition is satisfied by expanding fewer states. Larger the value of T, the more deeper SRPG is built resulting in better heuristic. However, for 2 given waypoints, lower value of Δ implies additional belief updates, leading to more states being expanded. Conversely, lower discretization and a larger time horizon will not result in successful plans. Longer T can lead to more states being expanded and a large Δ can lead to unvalidated plans. Hence an efficient plan requires a trade-off between the two.

6 Conclusion

We have discussed an ongoing research for mobile robot localization in the area of TMP, equipping a hybrid task planner with the capability of reasoning in the belief space of the robot. Expressive power of PDDL+ combined with heuristic base semantic attachments simulate the belief evolutions given an action sequence and the corresponding expected future observations. The underlying methodology of the hybrid planner has been discussed, validating the approach using a realistic synthetic simulation in Gazebo. We also provide the first step towards perception aware TMP, by sampling information rich waypoints and reasoning about landmark observations from these waypoints in the planning phase. Our sampling strategy combined with motion discretization help reduce the state space explosion while planning.The TMP planner is also unified with the ROSPlan framework.

While the scalability to larger domains still remains a

challenge, exploiting the planning-as-model-checking nature of the of the DiNo planner along withing efficient caching of plans might help in tackling this issue to some extent. Effective sampling strategies can help in pruning the unwanted state expansion. The extant of such pruning needs to be studied in detail. At present, DiNo supports only the minimize *total-time* metric and hence the plan synthesized is the one with minimum makespan. In our experiments the goal condition required only a bound on the trace of the final covariance. Yet, most often we require different costs to be minimized throughout the plan and hence we are working towards incorporating the same. We agree that in this paper we discuss in detail the planning algorithm and not the low-level motion control for execution and leave it for future work. However, we have successfully tested the execution for all our experiments in ROSPlan.

References

Bernardini, S.; Fox, M.; Long, D.; and Piacentini, C. 2017. Boosting Search Guidance in Problems with Semantic Attachments. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 29–37.

Cambon, S.; Alami, R.; and Gravot, F. 2009. A hybrid approach to intricate motion, manipulation and task planning. *The International Journal of Robotics Research* 28(1):104–126.

Cashmore, M.; Fox, M.; Long, D.; Magazzeni, D.; Ridder, B.; Carrera, A.; Palomeras, N.; Hurtos, N.; and Carreras, M. 2015. ROSPlan: Planning in the Robot Operating System. In *International Conference on Automated Planning and Scheduling*, 333–341.

Dantam, N. T.; Kingston, Z. K.; Chaudhuri, S.; and Kavraki, L. E. 2018. An incremental constraint-based framework for task and motion planning. *The International Journal of Robotics Research* 0(0):0278364918761570.

Dornhege, C.; Gissler, M.; Teschner, M.; and Nebel, B. 2009. Integrating symbolic and geometric planning for mobile manipulation. In *Safety, Security & Rescue Robotics (SSRR), IEEE International Workshop on*, 1–6. IEEE.

Dornhege, C.; Eyerich, P.; Keller, T.; Trüg, S.; Brenner, M.; and Nebel, B. 2012. Semantic Attachments for Domain-Independent Planning Systems. In *Towards Service Robots for Everyday Environments*. Springer Berlin Heidelberg. 99–115.

Fikes, R. E., and Nilsson, N. J. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial intelligence* 2(3-4):189–208.

Fox, M., and Long, D. 2006. Modelling Mixed Discrete-Continuous Domains for Planning. *Journal of Artificial Intelligence Research* 27(1):235–297.

Fox, D.; Burgard, W.; and Thrun, S. 1997. The Dynamic Window Approach to Collision Avoidance. *IEEE Robotics & Automation Magazine* 4(1):23–33.

Helmert, M. 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research* 26:191–246.

Hoffmann, J., and Nebel, B. 2001. The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research* 14:253–302.

Hoffmann, J. 2003. The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables. *Journal of Artificial Intelligence Research* 20:291–341.

Kaelbling, L. P., and Lozano-Pérez, T. 2012. Integrated robot task and motion planning in the now. Technical Report 2012-018, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Kaelbling, L. P., and Lozano-Pérez, T. 2013. Integrated task and motion planning in belief space. *The International Journal of Robotics Research* 32(9-10):1194–1227.

Kavraki, L. E.; Svestka, P.; Latombe, J.-C.; and Overmars, M. H. 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation* 12(4):566–580.

Kavraki, L. E.; Kolountzakis, M. N.; and Latombe, J.-C. 1998. Analysis of probabilistic roadmaps for path planning. *IEEE Transactions on Robotics and Automation* 14(1):166–171.

Kuffner, J. J., and LaValle, S. M. 2000. Rrt-connect: An efficient approach to single-query path planning. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, 995–1001. IEEE.

Lozano-Pérez, T., and Kaelbling, L. P. 2014. A constraint-based method for solving sequential manipulation planning problems. In *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, 3684–3691. IEEE.

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. PDDL- The Planning Domain Definition Language. In *AIPS-98 Planning Competition Committee*.

Nilsson, N. J. 1984. Shakey the robot. Technical Report 323, Airtificial Intellignece Center, SRI International, Menlo Park, California.

Phiquepal, C., and Toussaint, M. 2017. Combined task and motion planning under partial observability: An optimization-based approach. *RSS Workshop on Integrated Task and Motion Planning*.

Piotrowski, W. M.; Fox, M.; Long, D.; Magazzeni, D.; and Mercorio, F. 2016. Heuristic Planning for PDDL+ Domains. In *AAAI Workshop: Planning for Hybrid Systems*.

Srivastava, S.; Fang, E.; Riano, L.; Chitnis, R.; Russell, S.; and Abbeel, P. 2014. Combined task and motion planning through an extensible planner-independent interface layer. In *Robotics and Automation (ICRA), IEEE International Conference on*, 639–646. IEEE.

Thrun, S.; Burgard, W.; and Fox, D. 2005. *Probabilistic Robotics*. MIT press.

Toussaint, M. 2015. Logic-geometric programming: An optimization-based approach to combined task and motion planning. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Weyhrauch, R. W. 1980. Prolegomena to a theory of mechanized formal reasoning. *Artificial Intelligence* 13.