http://dx.doi.org/10.1007/978-3-642-41610-1_176-1

Task and Motion Planning

Neil T. Dantam

Synonyms

TMP, TAMP, Integrated Task and Motion Planning (ITMP), Combined Task and Motion Plannning (CTMP), Hybrid Planning

Definition

Task and Motion Planning operates in a combined discrete and continuous space to find a sequence of high-level, discrete actions and corresponding low-level, continuous paths to go from an initial state to a goal state.

Overview

Many activities for robots couple discrete and continuous reasoning. Discrete decisions, called *Task Planning*, include selecting objects to manipulate and the ordering of high-level actions. Continuous decisions, called *Motion Planning*, involve finding valid (i.e., collision-free) trajectories to execute such high-level actions. Separating discrete and continuous decisions is problematic; for example, the next discrete action may specify picking up an object, but there may be no continuous motion for the robot to bring its hand to a configuration that can actually grasp the object to pick it up. Task and Motion Planning (TMP) tightly couples task planning and motion planning, producing a sequence of steps that can actually be executed by a real robot to bring the world from an initial to a goal state.

Neil T. Dantam
Colorado School of Mines, Golden, CO, USA, e-mail: ndantam@mines.edu

Task Planning

Task planning identifies a sequence of discrete actions that change an initial state into a desired goal state. The leading approaches for efficient task planning are heuristic search and constraint-based planning. Heuristic search planners, such as Fast Forward (FF) (Hoffmann and Nebel 2001) and Fast Downward (Helmert 2006) construct a forward search tree from the start state using a heuristic to guide the search towards more promising states or actions. Constraint-based planners, such as Blackbox (Kautz and Selman 1999) and Madagascar (Rintanen 2014), convert the planning problem to a set of constraints–i.e., a Boolean formula—and use a constraint solver—typically, a solver for Boolean satisfiability (SAT)—to find a solution that represents the plan. Logic programming languages, such as Prolog and Answer Set Programming (ASP), have also been applied to task planning (Lifschitz 1999; Tenorth and Beetz 2015).

Motion Planning

Motion planning identifies a continuous path of valid configurations—e.g., joint positions—from an initial state to a desired goal state. Motion planning algorithms find plans within an abstract *configuration space*. Heuristic search methods are effective for motion planning in lower-dimensional applications such as navigation (Hart et al 1968; Stentz et al 1995; Koenig and Likhachev 2002). For high-dimensional robots such as manipulators, it is often difficult to represent the configuration space explicitly, so motion planners call specialized collision checking procedures to test whether an individual configuration is valid based on the positions of each object in the scene and each link of the robot. The leading approaches for efficient, high-dimensional motion planning are sampling-based (Kavraki et al 1996; Kuffner and LaValle 2000; Şucan and Kavraki 2009) and optimization-based (Khatib 1986; Kalakrishnan et al 2011; Schulman et al 2013; Zucker et al 2013) methods. Heuristic search planners are sometimes used for manipulation (Cohen et al 2014), but it is challenging to find general heuristics that work for different robots and scenarios.

Task-Motion Interaction

The key issue in TMP is interaction between task decisions and motion decisions. First, a robot moving an object changes the free configuration space through which the robot can move. If one movable object is occluding motion (i.e., blocking all paths the robot could take) to reach another object, the robot must move the occluding object out of the way to reach the other. Second, if two objects are attached, such as with a fastener or one contained in the other, then moving one object will also move the other. TMP must reason about interactions between task and motion decisions, for example by planning to remove occluding objects to reach a desired target or by moving attached objects together to reduce the number of plan steps.

Integrating Task Planning and Motion Planning

Though planning algorithms are defined over abstract state spaces, efficiency requires using an algorithm that exploits the properties of the space. Thus, while it may be possible to perform motion planning by creating a grid-based discretization of the configuration space and applying a discrete search algorithm, such an approach would not scale computationally beyond a few dimensions. Figure 1 outlines commonly used methods for task planning and motion planning. For these computational reasons, TMP approaches combine a specialized task planning algorithm with a specialized motion planning algorithm to explore each part of the combined discrete-continuous space. Broadly, there are three styles of integration (Erdem et al 2016) between a task planner and motion planner (see Figure 2):

(a) Motion planning is performed *before* task planning. The planner precomputes motion-level feasibility for all considered task actions and uses this information during task planning.
(b) Motion planning is performed *during* task planning. The task planner calls the motion planner as a subroutine (sometimes called a *semantic attachment*) on an as-needed basis during task planning.

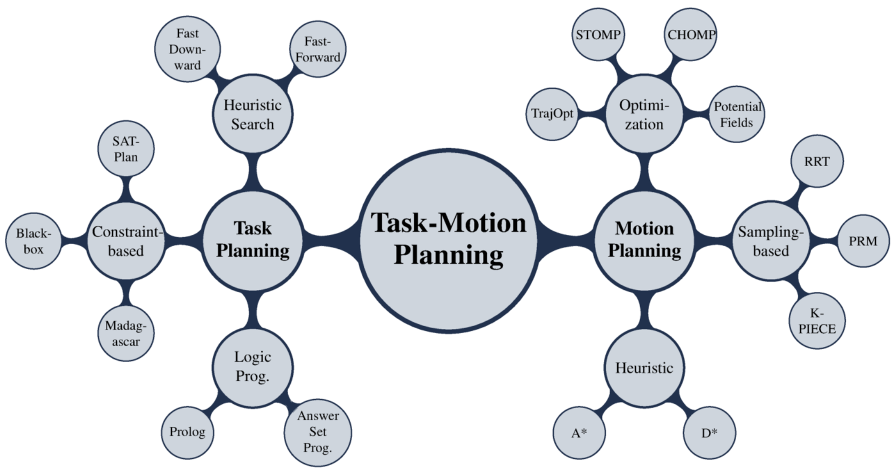


Fig. 1: A map of common methods for task planning and for motion planning.

http://dx.doi.org/10.1007/978-3-642-41610-1_176-1

(c) Motion planning is performed *after and in alternation* with task planning. The task planner first generates a candidate task plan. Then, the motion planner checks feasibility of this candidate. If motion planning fails, the the task planner produces an alternate plan, potentially using additional information or constraints obtained during motion planning.

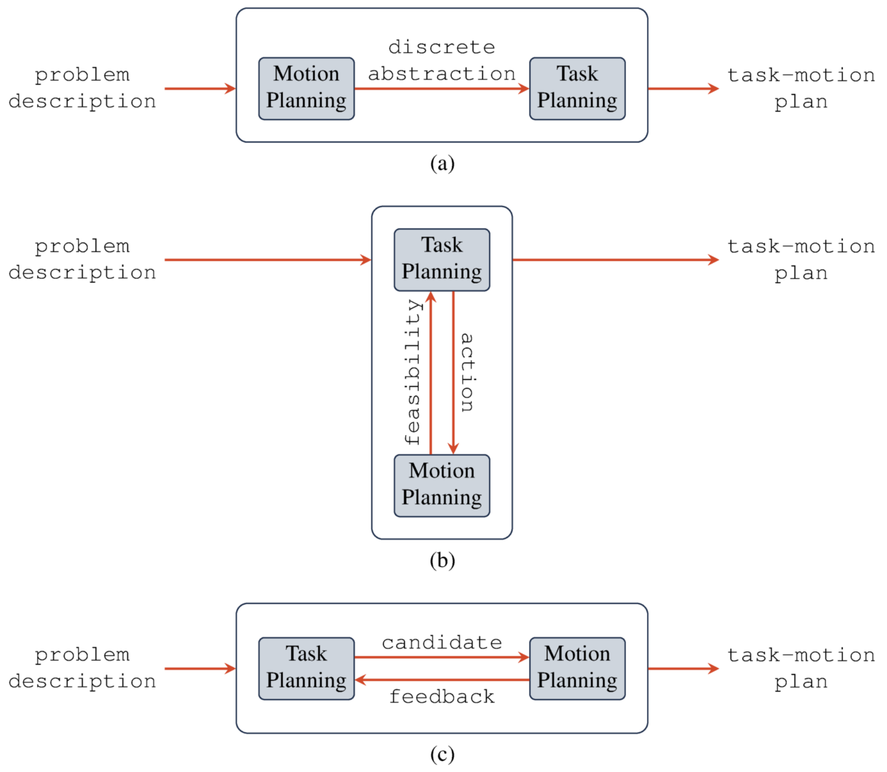


Fig. 2: Possible styles to integrate task planning and motion planning. **(a)** Motion planning performed before task planning. **(b)** Motion planning called as a subroutine (semantic attachment) from within task planning. **(c)** Motion planning and task planning performed in alternation.

Completeness

Completeness is an important challenge in TMP. A complete planner finds a plan when one exists or terminates with failure when no plan exists. Currently, the state-of-the-art in high-dimensional motion planning offers only *probabilistic complete-*

ness, meaning that the planner will eventually find a solution if one exists but will terminate only after a timeout. Thus, if motion planning fails due to a timeout, we cannot be sure whether (1) no plan exists or (2) a plan does indeed exist, and the motion planner just needs more time to find it. Consequently a motion planning failure in TMP may occur in the same two possible cases: (1) the motion is *infeasible* and we must explore alternate actions (e.g., removing an occluding obstacle first) or (2) the motion is *feasible* and we need more time to find the motion plan (e.g., the robot must move through a narrow passage which requires additional planning time to compute).

Addressing completeness in TMP presents a set of trade-offs. If we eliminate actions where motion planning fails—which are likely to be infeasible—the planner can quickly move on to explore other parts of the space. However, since failure of motion planning does not prove infeasibility, eliminating such actions in TMP is incomplete. Alternately, we may later reattempt motion planning for such failed actions, which can provide probabalistic completeness but potentially consumes more time attempting actions that are infeasible. Even the timeout value for motion planning presents a trade-off. A large timeout value may waste time attempting infeasible actions, while a small timeout value may miss feasible actions, thus wasting time exploring infeasible alternate actions or missing the solution altogether. The specifics of addressing these trade-offs are a differentiator in performance and capabilities among TMP approaches. Typically, approaches that progressively expand motion planning, as in Figure 2c, are able to achieve probabilistic completeness (Dantam et al 2018b; Garrett et al 2018), while approaches that eliminate failed (timed-out) motion planning attempts are more limited in completeness (Saribatur et al 2019).

Key Research Findings

Representing Task and Motion Planning Problems

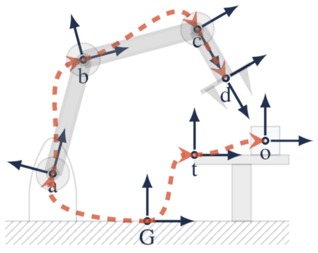
Formulating a TMP problem requires descriptions of the constituent parts: the task domain (state and actions), the motion domain (configuration space), and their interaction. The planner then outputs the sequence of task actions and their corresponding motion plans. To execute the plan, the robot must track the trajectories for each motion plan.

Input

The input to TMP includes the discrete task domain, the continuous motion domain, and the coupling of these two sides.

```
(:action pick-up
   :parameters (?object)
   :precondition (and (clear ?object)
                      (ontable ?object)
                      (handempty))
   :effect (and (not (ontable ?object))
                (not (clear ?object))
                (not (handempty))
                (holding ?object)))
```

(a)



(b)

Fig. 3: Example inputs for task planning and motion planning. **(a)** A task action in the Planning Domain Definition Language (PDDL). **(b)** A kinematic tree used to represent the configuration space for motion planning.

Task Domain

The task domain defines the discrete actions the robot can take. Actions have preconditions, indicating the states in which they may be taken, and effects, indicating how the action changes the state. For example, the action `pick-up` may have a precondition that the object is on the table and the effect that the object is in the robot's hand. There are numerous planning langauges (Fikes and Nilsson 1972; Pednault 1989; Barrett et al 1995) to define actions, states, and goals, with the most widely-used being the Planning Domain Definition Language (PDDL) (McDermott et al 1998). Figure 3a provides a example in PDDL of the `pick-up` action with preconditions and effects.

Motion Domain

The motion domain defines the configuration space for motion planning based on the three-dimensional positions of objects in the environment, the kinematic structure of robot joints, and the geometry—i.e., meshes—of robot links and objects. For a serial manipulator, we represent the robot and objects as a kinematic tree whose nodes are the local coordinate frames for each joint of the manipulator or object in the scene, and we use this kinematic tree to compute the poses of links and objects for a given configuration (see Figure 3b). Historically, Denavit-Hartenberg (DH) parameters (Hartenberg and Denavit 1964) were commonly used to specify robot kinematics, and more recently the XML-based *Universal Robot Definition Format* (URDF) (Willow Garage 2013) has gained widespread use. Generally, we need the joint types (e.g. revolute, prismatic), joint axes, and any fixed transformations between joints to specify the robot kinematics. Using the kinematics and the geometry (meshes) associated with each link or object, we are able to check for collisions to test whether a configuration is valid.

Defining Task-Motion Interaction

TMP also requires definitions for the interaction between the discrete task domain and the continuous motion domain. These definitions relate (1) motion state to task state and (2) task actions to motion planning problems. For example, in the first case, if robot has grasped an object in its hand, we may need a task-level predicate to indicate the robot is holding the object. In the second case, to perform a task-level, `pick-up` action, we need to identify and solve the corresponding motion planning problem to move the robot's hand to a grasping position for the object. Currently, there are no standard formats to define task-motion interaction. Instead, individual planners define such interactions based on their style of task-motion integration (see Figure 2) and the implementation details of the planner.

Output

The Task–Motion Planner finds the sequence of discrete task actions and their corresponding motion plans that will take the system from some initial state or scene to a desired state or scene. The task actions are a valid task plan for the input task domain. The motion plans are typically computed as a sequence of waypoints that the robot must move through.

Execution

A robot must execute the plan in real-time. To track the motion plans, the robot must compute a reference position, velocity, etc. at each timestep by interpolating between the waypoints of the plan. To correct tracking error, feedback control is generally effective. For manipulation domains, the robot must operate its gripper to grasp and release any objects as specified by the actions in the plan.

Planning Approaches

There are a variety of TMP approaches, differentiated by (1) the style of integrating task planning and motion planning (see Figure 2), (2) the specific task planning and motion planning algorithms employed (see Figure 1), and (3) any considerations of completeness.

Motion Planning before Task Planning

The Robosynth framework (Nedunuri et al 2014) first constructs a graph of base positions and arm motions for a mobile manipulator, then uses a constraint solver for Satisfiability Modulo Theories (SMT) (De Moura and Bjørner 2011) to generate a task and motion plan. Wang et al (2016) extend the approach to synthesize policies.

Motion Planning during Task Planning

Dornhege et al (2012) present an extension to PDDL for semantic attachments to call the motion planner as a feasibility check during task planning, and they address the general integration of semantic attachments into forward chaining planners. Erdem et al (2012) integrate motion planning feasibility checks into an ASP solver. Similarly, Yalciner et al (2017) use ASP to consider actions for both actuation and sensing in order to compute conditional plans.

Motion Planning after and in Alternation with Task Planning

The aSyMov planner (Cambon et al 2009) combines a heuristic-search task planner based on metric-FF (Hoffmann 2003) with lazily-expanded Probabilistic Roadmaps (PRMs) (LaValle et al 1999). ASyMov composes PRMs to handle object interaction, e.g., grasping. The computation of aSyMov's PRMs may be amortized over multiple planning runs, but composition of PRMs for many objects may be costly.

Hierarchical Planning in the Now (HPN) (Kaelbling and Lozano-Pérez 2013) interleaves planning and execution for TMP to reduce search depth for long plans. The shorter search depths improve scalability. However, interleaving planning and execution may result in physical backtracking to undo earlier actions that were eagerly executed, requiring reversibility of such actions and time to backtrack.

Srivastava et al (2014) combine off-the-shelf task planners and motion planners. If motion planning fails, the planner reattempts motion planning neglecting all movable obstacles. If the second motion planning attempt succeeds, the planner identifies all movable obstacles intersecting with the path and solves a recursive subproblem to remove those obstacles.

Dantam et al (2018b) combine constraint-based task planning and single-query, sampling-based motion planning (e.g., RRT-connect Kuffner and LaValle (2000)). The planner uses an incremental constraint solver—which is able to quickly solve related constraint problems—and the single-query motion planning handles object coupling (e.g., stacking, attachment). To achieve probabilistic completeness, the planner will later reattempt with increased timeout any actions where motion planning has failed (i.e., timed-out). However, such reattempts may lead to extra computation by attempting to find motion plans for infeasible actions.

Garrett et al (2018) present a sample-based TMP framework based on the observation that many TMP problems are *factored* and *sparse*, i.e., state and action are defined as a set of variables and any particular transition affects only a small number of variables. The planner takes as input conditional samplers to produce grasp poses, motion plans, etc., which it uses to incrementally construct discrete abstractions of the problem, alternating between growing the abstraction and performing a discrete search for a solution. An extension uses a lazy placeholder to stand for specific samples, reducing the necessary abstraction size in cases where many samples would be valid.

Examples of Applications

TMP is applicable in scenarios involving discrete and continuous decisions which interact in some way. Manipulation is a key application area, where a robot may make discrete choices about objects to move, and the moved objects change the continuous configuration space. A benchmark set for TMP presented by Lagriffoul et al (2018) contains several manipulation problems, including block stacking, a Towers of Hanoi variation, and a mobile manipulator rearranging objects.

Future Direction for Research

Ongoing research in TMP aims to both further generalize planning and strengthen guarantees. Scalability, e.g., to larger environments or more agents, is a perpetual issue in such computationally hard areas. Planning for multiple robots presents challenges not just for scalability but also in decentralization and coordination under limited information and communication. Robustness to uncertainty and addressing differential dynamics, e.g., for non-prehensile manipulation or bipedal walking, are important challenges in TMP. Producing optimal plans, e.g., shortest paths or least-cost actions, is another key research goal. It is an open question as to which planning techniques or combinations and extensions thereof will ultimately prove most practical or offer suitable trade-offs for TMP problems, with different approaches building on sampling, optimization, heuristic, and constraint-based methods. Finally, even the direct comparison of different TMP approaches presents challenges due to differing assumptions, e.g., about definitions of actions and goals, so the development of standardized problems and formats is an important task to support and evaluate future TMP research.

References and Cross References

Several related methods address common, special cases of TMP involving robot motion and moveable objects. Navigation among moveable obstacles (NAMO) extends motion planning to consider moving obstacles which occlude the robot's motion (Stilman and Kuffner 2008). NAMO considers two kinds of task actions—moving the robot and transferring an object. Most NAMO work has focused on mobile-robot navigation, though Stilman et al (2007) address manipulation as well.

Minimum constraint removoval (MCR) (Hauser 2014) and minimum constraint displacement (MCD) (Hauser 2013) identify the the minimum number of obstacles or obstacle displacements for the robot to reach a goal, in contrast to NAMO which plans for the robot to perform necessary manipulations to remove obstacles. Rearrangement Planning finds a plan to move a set of objects from initial to goal positions (Krontiris and Bekris 2015, 2016; Han et al 2018)

The field of switching and hybrid systems also considers problems with both discrete and continuous state (Alur et al 1995; Liberzon 2003). Hybrid systems work often focuses on issues of dynamic stability, reachability, and verification, whereas typical challenges in TMP are high-dimensionality, combinatorics, and scalability.

Several approaches combine motion planning, continuous dynamics, and discrete labels or changes (Plaku and Hager 2010; Bhatia et al 2011; Karaman and Frazzoli 2012; Vega-Brown and Roy 2016). These approaches typically focus on handling differential constraints or achieving optimality rather than on task planning combinatorics and scalability.

Finally, implementation and system integration for TMP presents an issue in and of itself due to the quantity and variety of algorithms involved. For motion planning, the Open Motion Planning Library (OMPL) (Şucan et al 2012) has emerged as a de facto standard implementation for sampling-based algorithms. "MoveIt!" (Chitta et al 2012) is a widely used interface for OMPL, which integrates, among many libraries, collision checking using the Flexible Collision Library (FCL) (Pan et al 2012). However, MoveIt! does not itself address TMP. Fast Downward (Helmert 2006) is an efficient task planning system based on heuristic search; extensions of Fast Downward were successful in the 2018 International Planning Competition (Coles et al 2018) and were applied to TMP by Garrett et al (2018). Constraint solvers for Satisfiability Modulo Theories (SMT) (De Moura and Bjørner 2011), such as Z3 (De Moura and Bjørner 2008), offer efficient and flexible symbolic reasoning. The Task-Motion Kit integrates OMPL and Z3 to provide a general TMP framework (Dantam et al 2018a).

References

Alur R, Courcoubetis C, Halbwachs N, Henzinger TA, Ho PH, Nicollin X, Olivero A, Sifakis J, Yovine S (1995) The algorithmic analysis of hybrid systems. Theoretical computer science 138(1):3–34

Barrett A, Christianson D, Friedman M, Kwok C, Golden K, Penberthy S, Sun Y, Weld D (1995) UCPOP user's manual

Bhatia A, Maly MR, Kavraki LE, Vardi MY (2011) Motion planning with complex goals. Robotics & Automation Magazine 18(3):55–64

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

Chitta S, Sucan I, Cousins S (2012) Moveit! [ros topics]. IEEE Robotics & Automation Magazine 19(1):18–19

Cohen B, Chitta S, Likhachev M (2014) Single and dual-arm motion planning with heuristic search. Int Journal of Robotics Research (IJRR) 3(2):305–320

Coles A, Coles A, Martinez M, Sidiropoulos P (2018) International planning competition. https://ipc2018.bitbucket.io

Şucan I, Moll M, Kavraki LE (2012) The open motion planning library. Robotics & Automation Magazine 19(4):72–82

Dantam NT, Chaudhuri S, Kavraki LE (2018a) The task motion kit. Robotics and Automation Magazine 25(3):61–70

Dantam NT, Kingston ZK, Chaudhuri S, Kavraki LE (2018b) An incremental constraint-based framework for task and motion planning. International Journal of Robotics Research 37(10):1134–1151

De Moura L, Bjørner N (2008) Z3: An efficient SMT solver. In: Tools and Algorithms for the Construction and Analysis of Systems, Springer, pp 337–340

De Moura L, Bjørner N (2011) Satisfiability modulo theories: introduction and applications. Communications of the ACM 54(9):69–77

Dornhege C, Eyerich P, Keller T, Trüg S, Brenner M, Nebel B (2012) Semantic attachments for domain-independent planning systems. In: Towards Service Robots for Everyday Environments, Springer, pp 99–115

Erdem E, Aker E, Patoglu V (2012) Answer set programming for collaborative housekeeping robotics: representation, reasoning, and execution. Intelligent Service Robotics 5(4):275–291

Erdem E, Patoglu V, Schüller P (2016) A systematic analysis of levels of integration between high-level task planning and low-level feasibility checks. AI Communications 29(2):319–349

Fikes RE, Nilsson NJ (1972) STRIPS: A new approach to the application of theorem proving to problem solving. Artificial Intelligence 2(3):189–208

Garrett CR, Lozano-Pérez T, Kaelbling LP (2018) Sampling-based methods for factored task and motion planning. The International Journal of Robotics Research 37(13-14):1796–1825

Han SD, Stiffler NM, Krontiris A, Bekris KE, Yu J (2018) Complexity results and fast methods for optimal tabletop rearrangement with overhand grasps. The International Journal of Robotics Research p 0278364918780999

Hart PE, Nilsson NJ, Raphael B (1968) A formal basis for the heuristic determination of minimum cost paths. Trans Systems Science and Cybernetics 4(2):100–107

Hartenberg RS, Denavit J (1964) Kinematic synthesis of linkages. McGraw-Hill

Hauser K (2013) Minimum constraint displacement motion planning. In: Robotics: Science and Systems

Hauser K (2014) The minimum constraint removal problem with three robotics applications. The International Journal of Robotics Research 33(1):5–17

Helmert M (2006) The fast downward planning system. Journal Artificial Intelligence Resesearch 26:191–246

Hoffmann J (2003) The Metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. Journal of Artificial Intelligence Research (JAIR) pp 291–341

Hoffmann J, Nebel B (2001) The FF planning system: Fast plan generation through heuristic search. Journal of Artificial Intelligence Research pp 253–302

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

Kalakrishnan M, Chitta S, Theodorou E, Pastor P, Schaal S (2011) STOMP: Stochastic trajectory optimization for motion planning. In: 2011 IEEE international conference on robotics and automation, IEEE, pp 4569–4574

Karaman S, Frazzoli E (2012) Sampling-based algorithms for optimal motion planning with deterministic μ-calculus specifications. In: American Control Conference, IEEE, pp 735–742

Kautz H, Selman B (1999) Unifying SAT-based and graph-based planning. In: International Joint Conference on Artifial Intelligence, vol 99, pp 318–325

Kavraki LE, Švestka P, Latombe JC, Overmars MH (1996) Probabilistic roadmaps for path planning in high-dimensional configuration spaces. Transactions on Robotics and Automation 12(4):566–580

Khatib O (1986) Real-time obstacle avoidance for manipulators and mobile robots. In: Autonomous robot vehicles, Springer, pp 396–404

Koenig S, Likhachev M (2002) Dˆ* lite. Aaai/iaai 15
Krontiris A, Bekris KE (2015) Dealing with difficult instances of object rearrangement. In: Robotics: Science and Systems
Krontiris A, Bekris KE (2016) Efficiently solving general rearrangement tasks: A fast extension primitive for an incremental sampling-based planner. In: International Conference on Robotics and Automation, IEEE, pp 3924–3931
Kuffner JJ, LaValle SM (2000) RRT-connect: An efficient approach to single-query path planning. In: International Conference on Robotics and Automation, IEEE, vol 2, pp 995–1001
Lagriffoul F, Dantam NT, Garrett C, Akbari A, Srivastava S, Kavraki LE (2018) Platform-independent benchmarks for task and motion planning. IEEE Robotics and Automation Letters 3,(4,):3765–3772,
LaValle SM, Yakey JH, Kavraki LE (1999) A probabilistic roadmap approach for systems with closed kinematic chains. In: Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C), IEEE, vol 3, pp 1671–1676
Liberzon D (2003) Switching in systems and control. Springer Science & Business Media
Lifschitz V (1999) Answer set planning. In: Logic Programming and Nonmonotonic Reasoning, Springer, pp 373–374
McDermott D, Ghallab M, Howe A, Knoblock C, Ram A, Veloso M, Weld D, Wilkins D (1998) PDDL – the planning domain definition language. AIPS-98 Planning Competition Committee
Nedunuri S, Prabhu S, Moll M, Chaudhuri S, Kavraki LE (2014) SMT-based synthesis of integrated task and motion plans from plan outlines. In: International Conference on Robotics and Automation, IEEE, pp 655–662
Pan J, Chitta S, Manocha D (2012) FCL: A general purpose library for collision and proximity queries. In: International Conference on Robotics and Automation, IEEE, pp 3859–3866
Pednault EP (1989) ADL: Exploring the middle ground between STRIPS and the situation calculus. In: Brachman RJ, Levesque HJ, Reiter R (eds) International Conference on Principles of Knowledge Representation and Reasoning, Morgan Kaufman, pp 324–332
Plaku E, Hager GD (2010) Sampling-based motion and symbolic action planning with geometric and differential constraints. In: International Conference on Robotics and Automation, IEEE, pp 5002–5008
Rintanen J (2014) Madagascar: Scalable planning with SAT. In: 8th International Planning Competition (IPC-2014), pp 66–70
Saribatur ZG, Patoglu V, Erdem E (2019) Finding optimal feasible global plans for multiple teams of heterogeneous robots using hybrid reasoning: an application to cognitive factories. Autonomous Robots 43(1):213–238, DOI 10.1007/s10514-018-9721-x, URL https://doi.org/10.1007/s10514-018-9721-x
Schulman J, Ho J, Lee AX, Awwal I, Bradlow H, Abbeel P (2013) Finding locally optimal, collision-free trajectories with sequential convex optimization. In: Robotics: science and systems
Srivastava S, Fang E, Riano L, Chitnis R, Russell S, Abbeel P (2014) Combined task and motion planning through an extensible planner-independent interface layer. In: International Conference on Robotics and Automation, IEEE, pp 639–646
Stentz A, et al (1995) The focussed dˆ* algorithm for real-time replanning. In: IJCAI, vol 95, pp 1652–1659
Stilman M, Kuffner J (2008) Planning among movable obstacles with artificial constraints. The International Journal of Robotics Research 27(11-12):1295–1307
Stilman M, Schamburek JU, Kuffner J, Asfour T (2007) Manipulation planning among movable obstacles. In: Proceedings 2007 IEEE international conference on robotics and automation, IEEE, pp 3327–3332
Şucan IA, Kavraki LE (2009) Kinodynamic motion planning by interior-exterior cell exploration. In: Algorithmic Foundation of Robotics VIII, Springer, pp 449–464
Tenorth M, Beetz M (2015) Representations for robot knowledge in the KnowRob framework. Artificial Intelligence

Vega-Brown W, Roy N (2016) Asymptotically optimal planning under piecewise-analytic constraints. In: Workshop on the Algorithmic Foundations of Robotics

Wang Y, Dantam NT, Chaudhuri S, Kavraki LE (2016) Task and motion policy synthesis as liveness games. In: International Conference on Automated Planning and Scheduling, AAAI, pp 536–540, URL http://www.aaai.org/ocs/index.php/ICAPS/ICAPS16/paper/view/13146

Willow Garage (2013) URDF XML. http://wiki.ros.org/urdf/XML

Yalciner IF, Nouman A, Patoglu V, Erdem E (2017) Hybrid conditional planning using answer set programming. Theory and Practice of Logic Programming 17(5-6):1027–1047

Zucker M, Ratliff N, Dragan AD, Pivtoraiko M, Klingensmith M, Dellin CM, Bagnell JA, Srinivasa SS (2013) CHOMP: Covariant hamiltonian optimization for motion planning. The International Journal of Robotics Research 32(9-10):1164–1193

Cross-References

Kinematics, Kinematics; Planning, Path/Motion Planning; Task and Robot Programming, Robot Task Modeling;

Acknowledgments

This material is based upon work supported in part by the National Science Foundation under Grant No. IIS-1849348.