RBE 550 Final Project: A Brief Introduction to Robotic Task and Motion Planning with PDDL and PDDLStream

Haotian Liu, Lehong Wang

***Abstract*—This report explores the integration of Task and Motion Planning (TAMP) within robotics through the lens of the Planning Domain Definition Language (PDDL), focusing on the advanced framework of PDDLStream. It commences with an introduction to task planning methodologies and progresses to practical illustrations of TAMP problem-solving using PDDLStream. The culmination of the study demonstrates the application of PDDLStream in resolving the classical Sussman Anomaly problem, accompanied by a visualization of the robotic movements in the PyBullet physics simulation environment. Our report examination not only underscores the efficacy of PDDLStream in complex TAMP scenarios but also highlights its potential to enhance robotic planning and execution.**

I. Introduction

Researchers argue that we humans have a good intuition for how a robot can achieve a task [1]. In an example task planning scenario: a robot can move between a finite set of locations, pick and place objects at those locations, and describe what it should do by breaking the solution down into individual actions [2]. In the **Figure** 1, a robot needs to pick an apple from the shelf and place it on the table.

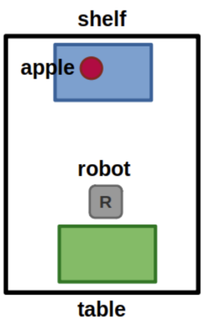

Figure 1. Caption

Example movement sequential:

```
Move to the shelf
Pick up the apple
Move back to the table
Place the apple
```

However, when encountering scenarios where the task environment grows in complexity, such as the addition of multiple target objects, diverse obstacles, and varied target locations, coupled with intricate goals like "*ensuring all books are appropriately shelved while other items are placed on the table, or simply preparing a tuna sandwich*"? The robot must effectively rationalize the relationships between the target objects and their designated locations [3, 4]. Additionally, manipulation tasks in robotics often involve navigating a high-dimensional search space, replete with numerous constraints that must be meticulously managed [5]. Furthermore, the robot manipulation often has a high dimensional search space with many constraints. Besides that, the quest for optimization in these tasks—be it minimizing time or reducing the total number of actions—adds another layer of complexity [6]. As the scale of the problem amplifies, human reasoning may reach its limitations, underscoring the potential necessity of deploying an automated planner [7–9].

Inspired by this, our curiosity was piqued regarding the capabilities of state-of-the-art task and motion planners, particularly PDDLStream [10], in executing classic task motion problems. We first study how to write PDDL files [11, 12] to set up the *domain* and *problem*, which gives the execution environment, start, and goal state of a task for robots. Moreover, our study emphasizes understanding the translation of high-level task descriptions into executable robotic actions. This translation is key to enabling robots to perform tasks with precision and efficiency autonomously. We present a detailed explanation of planning strategy with robot actions.

II. Related Work

A. Task and Motion Planning

1) Task Planning: Task planning is a machine that can autonomously reason about the state of the environment using a *self-model* and devise a sequence of *actions* or a *plan* to achieve a *goal* [13]. The key aspects of task planning are the following. ***a)*** Goal specification: The Planner defines what the robot needs to achieve. This could be as simple as moving an object from one place to another or as complex as assembling parts in a factory [14]. ***b)*** Action sequencing: The planner determines the order in which actions must be performed. This involves breaking down the goal into subtasks and figuring out the best sequence to complete these tasks efficiently and effectively [15]. ***c)*** Dealing with uncertainty: Robots often operate in dynamic environments where they may encounter unexpected obstacles or changes. The task planner must account for these uncertainties, potentially adapting as new information becomes available [16]. ***d)*** Integration with perception: Task planning relies on the robot's understanding of its environment, which comes from sensors and perception. The robot needs to interpret this sensory information to make decisions about its actions

[17]. ***e)*** Integration with motion planning: While task planning decides '*what*' actions to take, motion planning handles '*how*' to execute those actions physically. This includes navigating paths, manipulating objects, or interacting with environments and is more about the robot's movements and trajectories [13].

2) Motion Planning: Robot motion planning is a fundamental aspect of robotics that focuses on computing a collision-free path from start to goal for a robot within an environment populated with various obstacles [14, 18]. This trajectory is often executable by the robot without any physical interference from the surrounding elements. Some key aspects of motion planning are the following. ***a)*** Plan a path for a robot from an initial configuration to a goal configuration that avoids obstacles. ***b)*** Set up a sequence of continuous configurations. ***c)*** Configurations often are high-dimensional; for example, a 7 DOF robot arm grasps a block (**Figure** 2). ***d)*** Having high-level approaches: Geometric decomposition, Sampling-based, and Optimization-based.

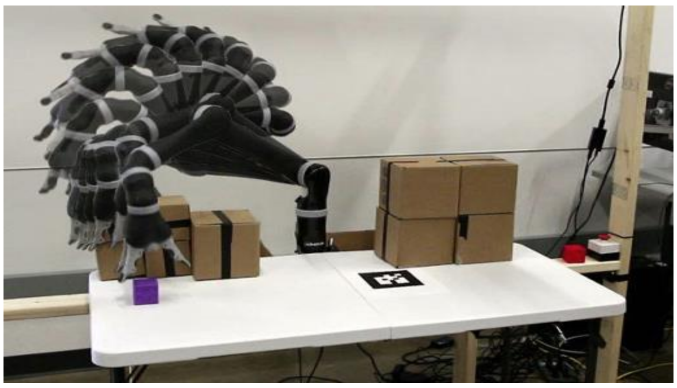

Figure 2. A high-dimensional manipulation example

B. PDDL: Planning Domain Definition Language

Planning Domain Definition Language (PDDL) is a family of languages that allows us to define a planning problem. As planning has evolved, so has the language used to describe it, and as such, there are now many versions of PDDL available with different levels of expressivity. Six main evolutions of PDDL, starting with the syntax defined for the 1998 International Conference on Autonomous Planning and Scheduling's AIPS competition [12]. It then further evolves into PDDL2.1 [11], PDDL2.2 [19], PDDL3.0 [20], and PDDL+ [21].

Here, we present the **Figure** 3 to illustrate the fundamental structure of PDDL and the meaning of each part in a file.

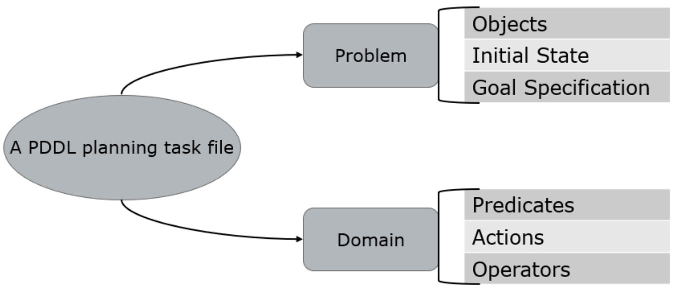

Figure 3. PDDL Structure

In problem:

- **Objects**: Things in the world that interest us.
- **Initial state**: The state of the world that we start in.
- **Goal specification**: Things that we want to be true.

In domain:

- **Predicates**: Properties of objects we are interested in; can be true or false.
- **Actions/Operators**: Ways of changing the state of the world.

III. PDDLStream examples

The key idea behind PDDLStream is that it extends PDDL with a notion of streams, which are generic, user-defined Python functions that sample continuous parameters such as a valid sample certifies that stream and provides any necessary predicates that act as preconditions for actions. In a fully observable space, so that the planning of tasks and motion levels is no longer a hierarchical call relationship, tasks can still be completed while satisfying various constraints in the continuous space (collision detection, maintaining posture). Also, PDDLStream has an adaptive technique that balances exploration (searching for discrete task plans) vs. exploitation (sampling to fill in continuous parameters).

In the context of PDDLStream, consider a robotic cooking task as a case study. The task begins with the robot retrieving an item of food, represented as a green object, from its initial position on the table (Step 1 in Figure 4). Once the food is securely grasped (Step 2 in Figure 4), the robot then transports it to the sink (Step 3 in Figure 4), depicted as a blue platform, where the washing process is undertaken. Following the cleaning phase, the next step involves the robot re-grasping the object (Step 4 in Figure 4) and transporting the object to the target area (Step 5 in Figure 4). In the final step, the robot places the food onto a designated heating area, represented by the red platform, where the cooking process takes place (Step 6 in Figure 4). This sequence of actions showcases the robot's ability to perform a multi-step task, integrating object handling, movement coordination, and task sequencing within the framework of PDDLStream.

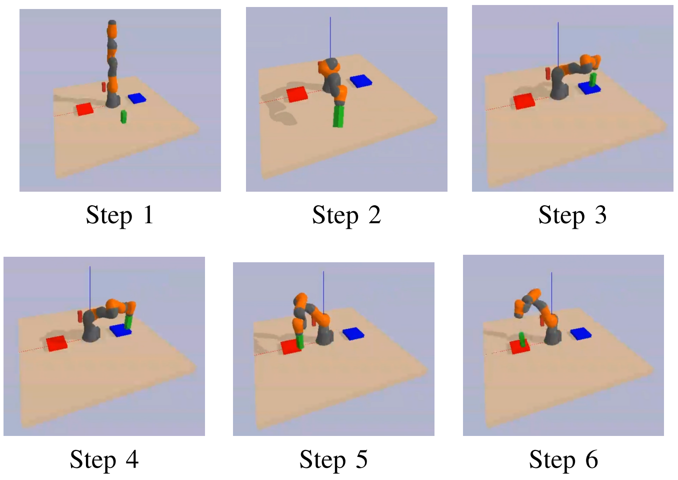

Figure 4.

Here, we present the plan output of this robot cooking task from PDDLStream.

```
Solved: True
Cost: 0.000
Length: 10
Deferred: 0
Evaluations: 2
 1) move_free q0 q4 c4
 2) pick 4 p0 g0 q4 c0
 3) move_holding q4 q13 4 g0 c5
 4) place 4 p4 g0 q13 c2
 5) clean 4 2
 6) pick 4 p4 g0 q13 c2
 7) move_holding q13 q7 4 g0 c6
 8) place 4 p2 g0 q7 c1
 9) move_free q7 q0 c3
10) cook 4 3
```

IV. Implementation of PDDLStream to Sussman Anomaly

A. Sussman Anomaly

In the exploration of artificial intelligence planning methodologies, the Sussman Anomaly presents a classical problem that underscores the limitations of non-interleaved planning algorithms, as initially explicated by Gerald Sussman [22]. Most modern planning systems have evolved beyond the non-interleaved paradigm, enabling them to navigate such complexities adeptly. While the significance of the problem is now a historical one, it is still useful for explaining why planning is non-trivial [23–25]. The problem posits a scenario involving three blocks, designated **A**, **B**, and **C**, situated on a table. The objective is to rearrange these blocks such that block **A** is placed atop block **B**, which in turn is positioned atop block **C** (**Figure** 5), with the constraint that only one block can be moved at a time.

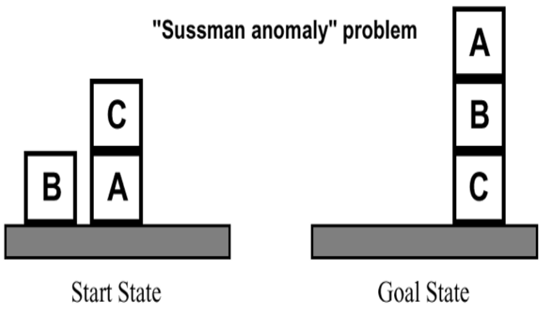

Figure 5. Demonstrate the start and goal state of Sussman Anomaly

The Anomaly will show that naively pursuing one subgoal **X** after you satisfy the other subgoal **Y** may not work because steps required to accomplish **X** might undo things subgoal **Y**. For example, in **Figure** 6, the top diagram tries to focus on subgoal: **B** atop of **C**, but putting **A** atop of **B** can't be done without undoing **B** atop of **C**; the bottom diagram tries to focus on subgoal: **A** atop of **B** first, but trying to put **B** atop of **C** would cause **A** atop of **B** undone.

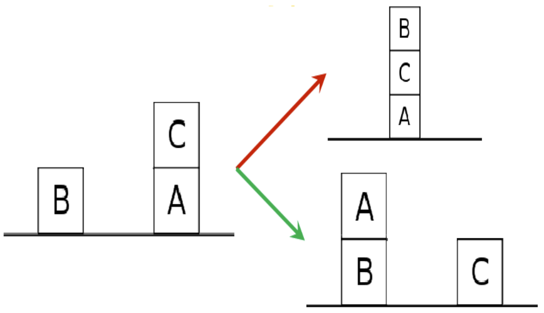

Figure 6. A demonstration about why we can not apply all operations to achieve a single goal at a time

B. The plan output and explanation of Sussman Anomaly task solved from PDDLStream frame

1) Plan output: Output from PDDLStream

```
Solved: True
Cost: 0.000
Length: 6
Deferred: 0
Evaluations: 3

1) pick c p1 g1 q1 t1
2) move_free q1 q2 t2
3) place c p2 g1 q2 t3
4) pick b p3 g2 q2 t4
5) move_holding q2 q3 b g2 t5
6) place b p4 g2 q3 t6
7) pick a p5 g3 q3 t7
8) move_holding q3 q4 c g3 t8
9) place a p6 g3 q4 t9
```

2) Plan explanation:

1) **pick c p1 g1 q1 t1**: The robot picks block **C** from pose **p1** using grasp **g1**, starting from configuration **q1**, following trajectory **t1**.
   - **Action**: The robot executes a **pick** action.

- **Block**: It picks up the block **C**.
- **Pose (p1)**: The initial pose of block **C** before being picked up.
- **Grasp (g1)**: The grasp configuration used to pick up the block **C**.
- **Configuration (q1)**: The robot's configuration when it starts the action.
- **Trajectory (t1)**: The trajectory followed by the robot to execute the pick action.

2) **move-free q1 q2 t2**: The robot moves freely (without holding a block) from configuration **q1** to **q2** along trajectory **t2**.
   - **Action**: The robot moves freely without holding any block.
   - **From configuration (q1)**: The starting configuration of the robot.
   - **To configuration (q2)**: The destination configuration of the robot.
   - **Trajectory (t2)**: The trajectory followed by the robot to move from **q1** to **q2**.
3) **place c p2 g1 q2 t3**: The robot places block **C** into pose **p2** still using grasp **g1** and ends in configuration **q2**, following trajectory **t3**.
   - **Action**: The robot performs a **place** action.
   - **Block**: It places block **C**.
   - **Pose (p2)**: The target pose where block **C** is to be placed.
   - **Grasp (g1)**: The grasp configuration used while placing block **C**.
   - **Configuration (q2)**: The robot's configuration when it places the block.
   - **Trajectory (t3)**: The trajectory followed by the robot during the placement of the block.
4) **pick b p3 g2 q2 t4**: The robot picks up block **B** from pose **p3** using grasp **g2**, starting from configuration **q2**, following trajectory **t4**.
   - **Action**: The robot picks up block **B**.
   - **Pose (p3)**: The initial pose of block **B** before being picked up.
   - **Grasp (g2)**: The grasp configuration for block **B**.
   - **Configuration (q2)**: The robot's configuration at the start of the pick action.
   - **Trajectory (t4)**: The trajectory followed to pick up block **B**.
5) **move-holding q2 q3 b g2 t5**: The robot moves from configuration **q2** to configuration **q3** while holding block **B** with grasp **g2**, following trajectory **t5**.
   - **Action**: The robot moves while holding block **B**.
   - **From Configuration (q2)**: The starting configuration while holding the block.
   - **To Configuration (q3)**: The destination configuration for the robot.
   - **Block**: The block being moved, **B**.
   - **Grasp (g2)**: The grasp configuration used to hold block **B**.
   - **Trajectory (t5)**: The trajectory followed while moving block **B**.
6) **place b p4 g2 q3 t6**: The robot places block **B** onto pose **p4** using grasp **g2**, changing from configuration **q3** to the next configuration along trajectory **t6**.
   - **Action**: The robot places block **B**.
   - **Pose (p4)**: The target pose for placing block **B**.
   - **Grasp (g2)**: The grasp configuration used while placing **B**.
   - **Configuration (q3)**: The robot's configuration during the placement.
   - **Trajectory (t6)**: The trajectory followed during the placement action.
7) **pick a p5 g3 q3 t7**: The robot picks up block **A** from pose **p5** using grasp **g3**, starting from configuration **q3**, and proceeds along trajectory **t7**.
   - **Action**: The robot picks up the block **A**.
   - **Pose (p5)**: The initial pose of block **A**.
   - **Grasp (g3)**: The grasp configuration for block **A**.
   - **Configuration (q3)**: The robot's configuration at the start of the pick action.
   - **Trajectory (t7)**: The trajectory followed to pick up block **A**.
8) **move-holding q3 q4 a g3 t8**: The robot, holding block **A** with grasp **g3**, moves from configuration **q3** to configuration **q4** following trajectory **t8**.
   - **Action**: The robot moves while holding block **A**.
   - **From Configuration (q3)**: The starting configuration while holding the block.
   - **To Configuration (q4)**: The destination configuration for the robot.
   - **Block**: The block being moved, **A**.
   - **Grasp (g3)**: The grasp configuration for block **A**.
   - **Trajectory (t8)**: The trajectory followed while moving block **A**.
9) **place a p6 g3 q4 t9**: The robot places block **A** into pose **p6** while holding it with grasp **g3**, transitioning from configuration **q4** via trajectory **t9**.
   - **Action**: The robot places block **A**.
   - **Block**: The block **A** being placed.
   - **Pose (p6)**: The target pose for placing block **A**.
   - **Grasp (g3)**: The grasp configuration used by the robot to hold block **A** during placement.
   - **Configuration (q4)**: The robot's configuration when it performs the placement of block **A**.

- **Trajectory (t9)**: The trajectory that the robot's end effector follows to place block **A** onto its target pose.

C. Visual understanding by PyBullet simulation

In **Figure** 7, the robot resolving the Sussman Anomaly adheres closely to the sequential steps outlined in the first part, the output plan, of this subsection. This approach is evident in each stage of the process, for example, from steps 1 to 3. The robot first picks block **C**, then moves it to a free position, and finally places it on the side of block **B** (Red block is A; Green block is B; Blue block is C).

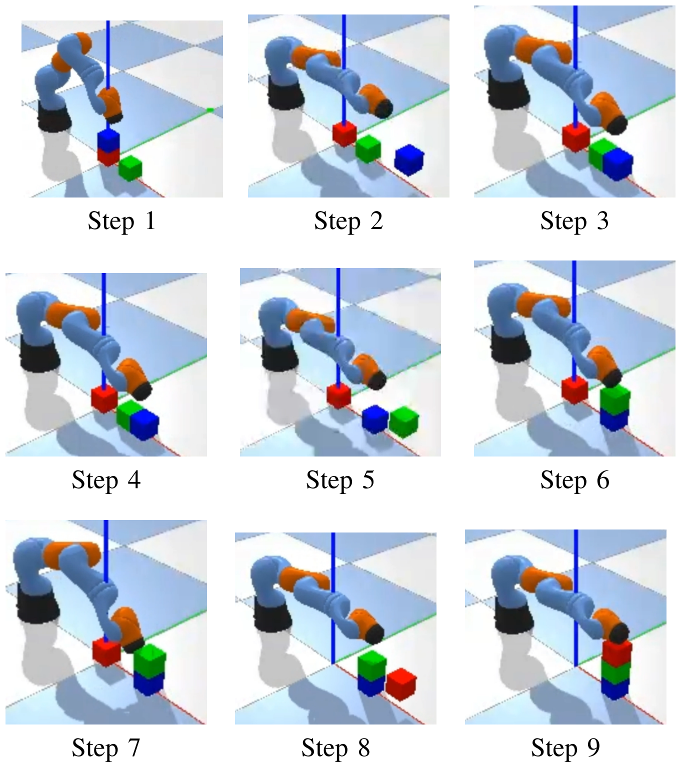

Figure 7.

D. Advantages of using PDDLStream

Previous studies reveal that PDDLStream is a hybrid approach that combines the strengths of PDDL for task planning with the ability to integrate streams that can handle numerical, geometric, and other continuous planning problems. This hybrid approach is particularly beneficial for complex tasks like the Sussman Anomaly, where both discrete decision-making and continuous planning are required. Furthermore, PDDLStream automates the creation sequence actions that not only solve the high-level task planning problem but also ensure that the solution is executable, taking into account the environment constraints.

V. Conclusion

From our study, the capabilities of PDDLStream are exemplified through its adeptness in implementing an interleaved planning approach to address task motion planning challenges. In our demonstration, we highlighted two distinct scenarios: the robotic cooking task, derived directly from the PDDLStream's source code, and the Sussman Anomaly, a task we defined independently. For each case, we provided detailed illustrations of the planned actions, offering a clear, step-by-step breakdown of the tasks involved. Additionally, to enhance comprehension, we utilized the PyBullet simulator to show visualizations. Allowing us to vividly depict each scenario in a simulated physical environment, thereby enabling a more engaging and visually rich understanding of how PDDLStream navigates and solves complex task motion planning problems.

VI. Limitations

PDDLStream assumes that the task space is fully observable and does not consider the uncertainty of the observed objects and the robot's own state. In the future, it would be possible to try planning in belief states, using a Bayesian filter for state estimation of the manipulated object. In addition, it is possible to develop some search heuristics to speed up the task planning process (since sampling at the motion level involves high dimensional task spaces, sometimes it can be challenging to find a valid plan) and deep learning methods can be employed.

VII. Acknowledgement

We extend our heartfelt gratitude to Professor Chamazas for his enriching and insightful lectures, which significantly enhanced our understanding of state-of-the-art motion planning algorithms. His detailed and thoughtful responses to each of our queries greatly contributed to our learning journey. Additionally, we would like to express our sincere appreciation for the dedicated efforts and timely feedback provided by our teaching assistant, Mr. Zhong. Their combined guidance has been instrumental in deepening our knowledge and skills in this complex field.

References

[1] S. Alatartsev, S. Stellmacher, and F. Ortmeier, "Robotic task sequencing problem: A survey," *Journal of intelligent & robotic systems*, vol. 80, pp. 279–298, 2015.

[2] K. He, M. Lahijanian, L. E. Kavraki, and M. Y. Vardi, "Towards manipulation planning with temporal logic specifications," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 346–352.

[3] J. Mao, C. Gan, P. Kohli, J. B. Tenenbaum, and J. Wu, "The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision," *arXiv preprint arXiv:1904.12584*, 2019.

[4] H. Wu, J. Mao, Y. Zhang, Y. Jiang, L. Li, W. Sun, and W.-Y. Ma, "Unified visual-semantic embeddings: Bridging vision and language with structured meaning representations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6609–6618.

[5] Z. Kingston, M. Moll, and L. E. Kavraki, "Sampling-based methods for motion planning with constraints," *Annual review of control, robotics, and autonomous systems*, vol. 1, pp. 159–185, 2018.

[6] Y.-H. Kim and B. K. Kim, "A multi-robot task planning system minimizing the total execution time for hospital service," in *ICCAS 2010*. IEEE, 2010, pp. 379–384.

[7] J. S. Penberthy, D. S. Weld *et al.*, "Ucpop: A sound, complete, partial order planner for adl." *Kr*, vol. 92, pp. 103–114, 1992.

[8] R. E. Fikes and N. J. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving," *Artificial intelligence*, vol. 2, no. 3-4, pp. 189–208, 1971.

[9] J. Mao, T. Lozano-Pérez, J. B. Tenenbaum, and L. P. Kaelbling, "Pdsketch: Integrated planning domain programming and learning," *arXiv preprint arXiv:2303.05501*, 2023.

[10] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 30, 2020, pp. 440–448.

[11] M. Fox and D. Long, "Pddl2. 1: An extension to pddl for expressing temporal planning domains," *Journal of artificial intelligence research*, vol. 20, pp. 61–124, 2003.

[12] H. L. Younes and M. L. Littman, "Ppddl1. 0: An extension to pddl for expressing planning domains with probabilistic effects," *Techn. Rep. CMU-CS-04-162*, vol. 2, p. 99, 2004.

[13] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual review of control, robotics, and autonomous systems*, vol. 4, pp. 265–293, 2021.

[14] J.-C. Latombe, *Robot motion planning*. Springer Science & Business Media, 2012, vol. 124.

[15] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 1470–1477.

[16] A. Thomas, F. Mastrogiovanni, and M. Baglietto, "Towards multi-robot task-motion planning for navigation in belief space," *arXiv preprint arXiv:2010.00780*, 2020.

[17] J. Mao, C. Gan, P. Kohli, J. B. Tenenbaum, and J. Wu, "The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision," *arXiv preprint arXiv:1904.12584*, 2019.

[18] M. Mohanan and A. Salgoankar, "A survey of robotic motion planning in dynamic environments," *Robotics and Autonomous Systems*, vol. 100, pp. 171–185, 2018.

[19] S. Edelkamp and J. Hoffmann, "Pddl2. 2: The language for the classical part of the 4th international planning competition," Technical Report 195, University of Freiburg, Tech. Rep., 2004.

[20] A. Gerevini and D. Long, "Plan constraints and preferences in pddl3," Technical Report 2005-08-07, Department of Electronics for Automation ..., Tech. Rep., 2005.

[21] S. Thiébaux, J. Hoffmann, and B. Nebel, "In defense of pddl axioms," *Artificial Intelligence*, vol. 168, no. 1-2, pp. 38–69, 2005.

[22] G. J. Sussman, "A computational model of skill acquisition," 1973.

[23] S. Russell and P. Norvig, "Artificial intelligence: a modern approach, 4th us ed," *University of California, Berkeley*, 2021.

[24] N. Gupta and D. S. Nau, "On the complexity of blocks-world planning," *Artificial intelligence*, vol. 56, no. 2-3, pp. 223–254, 1992.

[25] D. E. Wilkins, "Can ai planners solve practical problems?" *Computational intelligence*, vol. 6, no. 4, pp. 232–246, 1990.