Bachelor Project

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Robotic Motion Planning Guided by Demonstration

David Kovář

Supervisor: Ing. Vladimír Petrík, Ph.D.
January 2023

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Kovář David** Personal ID number: **483491**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Robotic Motion Planning Guided by Demonstration

Bachelor’s thesis title in Czech:

Plánování robotického pohybu naváděného pomocí demonstrace

Guidelines:

1. Implement several robot object manipulation path planning environments for benchmarking in the Humanoid Path Planning Planner framework [1, 2].
2. Design and implement a random sampler for biasing the randomized motion planning [3] towards the demonstrations.
3. Prepare a set of demonstrations for benchmarking.
4. Compare the designed planning from demonstrations with the Humanoid Path Planning planner [1, 2].

Bibliography / sources:

1] Lamiraux, Florent, and Joseph Mirabel. "Prehensile Manipulation Planning: Modeling, Algorithms and Implementation." IEEE Transactions on Robotics (2021).
[2] Mirabel, Joseph, and Florent Lamiraux. "Manipulation planning: building paths on constrained manifolds." (2016).
[3] Kuffner, James J., and Steven M. LaValle. "RRT-connect: An efficient approach to single-query path planning." In Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065), vol. 2, pp. 995-1001. IEEE, 2000.

Name and workplace of bachelor’s thesis supervisor:

Ing. Vladimír Petrík, Ph.D. Applied Algebra and Geometry CIIRC

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **07.06.2022** Deadline for bachelor thesis submission: **10.01.2023**

Assignment valid until: **19.02.2024**

Ing. Vladimír Petrík, Ph.D.	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

. Date of assignment receipt

Student’s signature

Acknowledgements

I cannot express enough thanks to my supervisor, Ing. Vladimir Petrik, Ph.D. as he gave me the opportunity to work on this topic while being generously helpful and patient with me. I also want to express my gratitude to CIIRC CTU Junior Researcher, Kateryna Zorina, MSc., as she allowed me to participate on her paper regarding this thesis topic and helped me a lot with this thesis as well.

I am very grateful for the opportunity to work with all of them.

Additionally, I would like to thank Florent Lamiraux, Ph.D. of LAAS-CNRS, for the consultations and help he has provided me on the topic of Humanoid Path Planner.

In the end I would like to thank prof. Ing. Vaclav Hlavac, CSc., and Dr. Ing. Josef Sivic, as they have recommended me to my supervisor, even though the recommendation was based on misunderstanding. Sometimes luck is an important ingredient as any other.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, date 10. 1. 2023

Abstract

This thesis aims to solve task and motion planning problems by leveraging a video demonstration of a human-executed solution of the task at hand. We introduce a new sampling-based task and motion planner guided by demonstration that extend the planner that we proposed in [Zorina et al., 2023]. We benchmark the proposed solution on real life task and motion planning problems against the state-of-the-art task planners. Our results show, that utilizing the whole human demonstration can lead to the performance improvement especially in a number of iterations required to solve the planning problem.

Keywords: task and motion planning, demonstration guided planner, sampling based planner, benchmarking

Supervisor: Ing. Vladimír Petrík, Ph.D.

Abstrakt

Tato bakalářská práce si klade za cíl řešit problém plánování úkolů a pohybu s využitím video demonstrace, která poskytuje ukázku člověkem vykonaného řešení daného problému. Představujeme nový, demonstrací navádění plánovač, založený na vzorkování, který je rozšířením plánovače představeného v [Zorina et al., 2023]. Navržené řešení ohodnotíme pomocí benchmarku na reálných problémech typu plánování úkolů a pohybu proti nejmodernějším plánovačům. Naše výsledky ukazují, že využití celé člověkem vykonané demonstrace může vést ke zlepšení výkonnosti, zejména co se počtů iterací k vyřešení problému týče.

Klíčová slova: plánování úkolů a pohybu, demonstrací naváděný plánovač, vzorkovací plánovače pohybu, benchmarking

Překlad názvu: Plánování robotického pohybu naváděného pomocí demonstrace

Contents

Figures

Tables

Chapter 1

Introduction

The goal of this thesis is to solve task and motion planning problem, that is defined in combined task and robot joint space. These problems are usually solved by sampling-based motion planners or by learning the control policy, where learning is often guided by the demonstration. However, sampling-based motion planner needs a lot of time to explore the combined configuration space and policy learning requires resource-consuming offline training phase. In our case, we aim to use demonstration to guide the sampling-based motion planner and therefore combine the benefits of demonstration with planning.

In previous work [Zorina et al., 2023] we have proposed a new method of path planning guided by video demonstration. This method used demonstration to acquire pick order of known objects in pick and place tasks and used the poses of the known objects before they were picked up to create a grasp. Both of these additions provided effective aid to the method as knowing the pick and place order reduces the complexity of task configuration space. Not to mention generating a discrete grasp in infinite configuration space is a complex task, and this way we could aid the algorithm in generating it.

The method was benchmarked against state of the art sampling based task and motion planners. The benchmarking was done on real-life problems which are difficult to solve even for state-of-the-art solvers. The method proved to be successful and held well against the state-of-the-art planners, as can be seen in Fig. 1.1.

In this thesis, our goal is to propose a new extension of this method. The full potential of the demonstration has not yet been used in the [Zorina et al., 2023] as the 6D pose of the known objects when being carried by the demonstrator is not used. Thus, we aim to use these 6D object poses to further aid the planner and guide it along the demonstration.

The contributions of this thesis are:

1. Implementation of the benchmarking environment for profiling the planners. These environments were used in our ICRA 2023 submission [Zorina et al., 2023] for demonstrating benefits of the human demonstration.

2. Proposal and implementation of an extension to the work [Zorina et al., 2023]. The proposed extension utilizes the whole demonstration instead of the key points used in [Zorina et al., 2023].

3. Benchmark of the proposed extension with [Zorina et al., 2023] that shows improved performance of the proposed method.

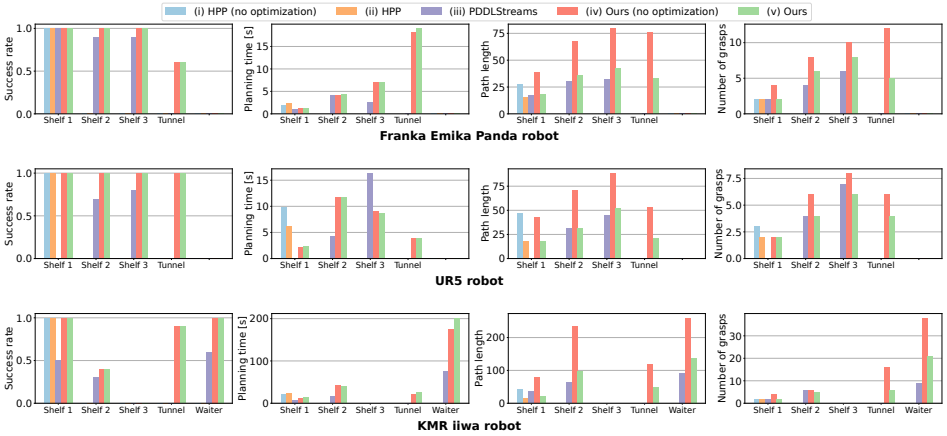

Figure 1.1: Previous approach results reported for different robots (rows) and different metrics (columns). We report (from left): the success rate, the planning time [s], the path length, and the number of grasps. For the success rate, the higher number the better; for the other metrics lower numbers are better. For more detailed explanation please see benchmarking section. This figure was taken from [Zorina et al., 2023].

Chapter 2

Related Works

2.1 Path Planning in Robotics

For this work, the path planning problem can be defined as the search for a collision-free path from the given start configuration to the given goal configuration. This thesis studies task-and-motion planning for which the configuration composes of robot configuration (*i.e.* joint angles) and objects poses. The output of the path planning algorithm is a sequence of collision free configurations.

2.1.1 Grid-Based Planning

Numerous path planning algorithms have been proposed in the past for the path planning in robotics such as Djikstras algorithm [Dijkstra, 1959] or A*[Hart et al., 1968a]. For example, algorithm A* [Hart et al., 1968b], a gird-based search algorithm, is used on the discretized configuration space. This approach is effective for low dimensional problems *e.g.* for mobile robot path planning on 2D plane. However, for high dimensional problems, these grid-based search planners suffer from a course of dimensionality.

For example, in the case of a 2D mobile robot, the configuration space can have up to 3 DoF (two for translation and one for rotation of the robot). If we made grid with n cells per DoF than the grid would be of size n^3, where 3 is number of DoF. But in case of path planning for robotic manipulator in 3D space with 7 joints of either revolute or prismatic type, the problem has 7 DoF. The grid size would be n^7, which is already difficult to represent with sufficient accuracy (*i.e.* cell size). The memory complexity grows furthermore as we include manipulated objects into the configuration space, as every manipulated object in 3D space adds additional 6 DoF. With additional objects or more complex manipulators, the complexity grows exponentially, and such grids can no longer be stored in memory. Because of this course of dimensionality issue, we use sampling-based path planning algorithm in this work as it aims to solve path planning problems with higher number of DoF.

2.1.2 Sampling-Based Planning

Essential path planning algorithms for this work are the ones based on Rapidly-exploring Random Tree [LaValle et al., 1998]. Rapidly-exploring Random Tree (RRT) is an algorithm designed for path planning in high-dimensional spaces. It is a general algorithm that can be applied also to problems described by a set of nonlinear differential constraints or to problems that require kinodynamic planning, i.e. driving a robot from initial state to goal state while avoiding obstacles and obeying both the kinematics and dynamics constraints. RRT builds a graph (tree) in the configuration space from the given starting configuration (root of the tree). For RRT, each configuration is a node in the graph and an edge connecting two nodes is the transition between the two configurations. RRT will start the graph with starting configuration as an only node $\boldsymbol{q}_{\text{init}}$, it will then generate random configuration $\boldsymbol{q}_{\text{rand}}$ from configuration space, look for the nearest node $\boldsymbol{q}_{\text{near}}$ of the graph and then attempts to create a collision free node $\boldsymbol{q}_{\text{new}}$ which is in direction from $\boldsymbol{q}_{\text{near}}$ to $\boldsymbol{q}_{\text{rand}}$ but within boundaries of certain step size from $\boldsymbol{q}_{\text{near}}$. Then $\boldsymbol{q}_{\text{new}}$ is added to the tree under the condition that it is collision free and a path between $\boldsymbol{q}_{\text{near}}$ and $\boldsymbol{q}_{\text{new}}$ exists. Example of growing RRT tree is shown in Fig. 2.1.

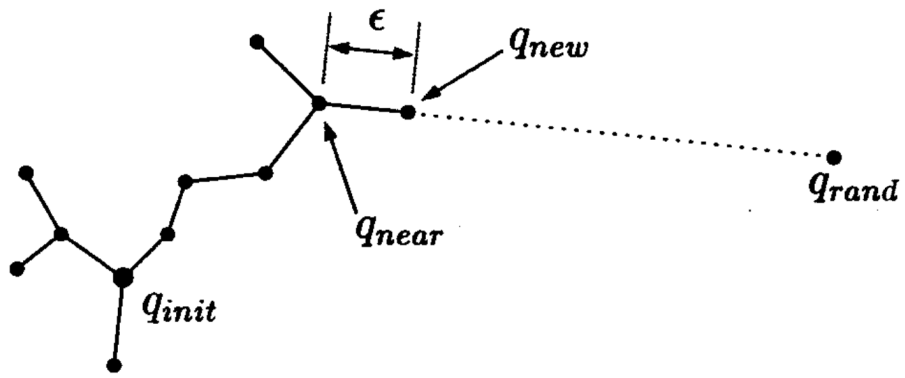

Figure 2.1: Example of the tree growing in RRT. Here $\boldsymbol{q}$ is $\boldsymbol{q}_{\text{rand}}$. A new node $\boldsymbol{q}_{\text{new}}$ is constructed in the direction of $\boldsymbol{q}_{\text{rand}}$ and added to the tree if it is collision-free. The process is repeated iteratively until the goal is find. This picture was taken from [Kuffner and LaValle, 2000].

Disadvantage of RRT is that it can take a long time to find connection between two desired configurations. An extention of RRT called RRT-connect [Kuffner and LaValle, 2000] address this issue by growing two trees from the goal configuration and start configuration towards each other. RRT-connect achieves this by swapping in-between the growing of both trees. Whenever a new node $\boldsymbol{q}_{\text{new}_a}$ is added to tree $\boldsymbol{\tau}_a$, RRT-connect attempts to find the connection between $\boldsymbol{q}_{\text{new}_a}$ and the second tree $\boldsymbol{\tau}_b$. RRT-connect will treat $\boldsymbol{q}_{\text{new}_a}$ as $\boldsymbol{q}_{\text{rand}}$ for the tree $\boldsymbol{\tau}_b$ and will connect it with a node from $\boldsymbol{\tau}_b$ and thus form a connection between $\boldsymbol{\tau}_a$ and $\boldsymbol{\tau}_b$, or it will generate $\boldsymbol{q}_{\text{new}_b}$ in a similar

way as RRT would and connects it to τ_b. This way, for each new node of the tree a second node is added to the other tree that is heading in the direction of the tree with the new node. An example of two trees growing is shown in Fig. 2.2.

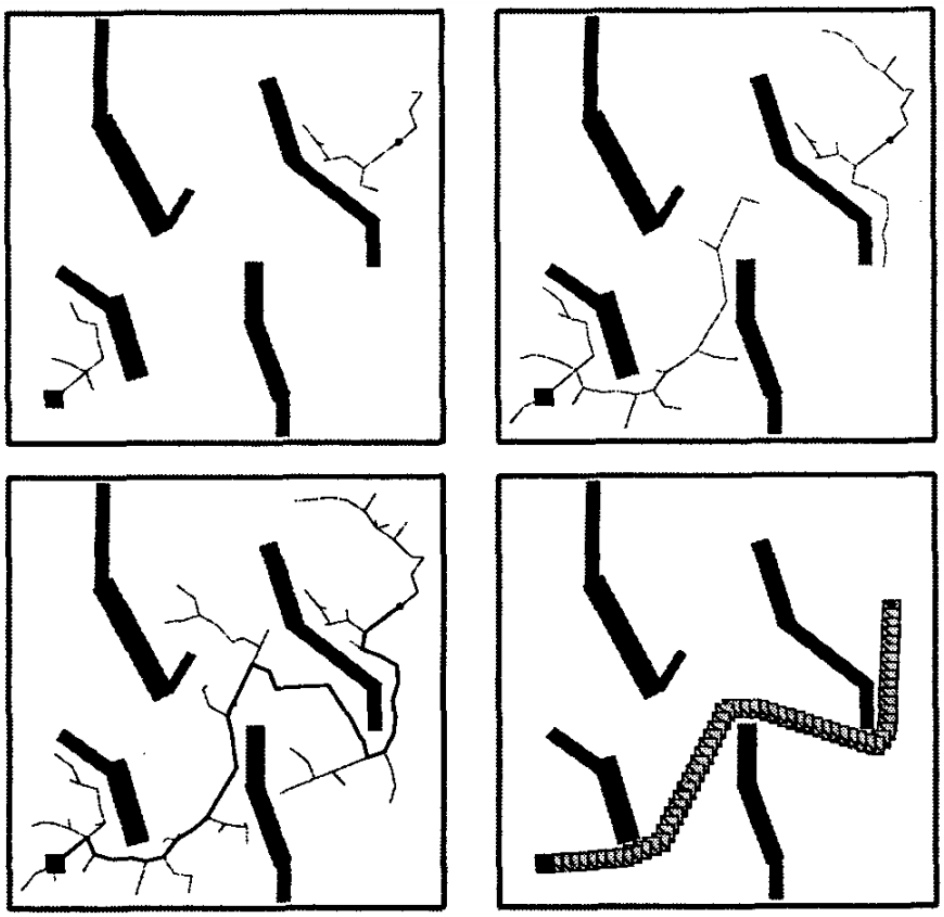

Figure 2.2: Example of two RRT trees growing from start and goal configurations. It is shown that both trees are approaching each other quickly without unnecessary exploration of the space. This picture was taken from [Kuffner and LaValle, 2000].

RRT-connect is also powered by advanced "extend" function. Where RRT would just move to the $\boldsymbol{q}_{\text{rand}}$ from $\boldsymbol{q}_{\text{near}}$ in bounds of given step, RRT-connect iteratively calls the step function until it either reaches $\boldsymbol{q}_{\text{rand}}$ or finds an obstacle and returns the last correct configuration. RRT-connect and building multiple trees simultaneously have been shown to be powerful for path planning problem in high dimensions, and we utilize the same idea in this thesis.

There are other RRT based algorithms such as RRT * [Karaman and Frazzoli, 2010], which utilizes cost functions to grow the tree and optimize the path with more samples, or RRT +, which aims to solve the planning problem in real time.

Probabilistic Roadmaps [Kavraki et al., 1996] is another algorithm that aims to solve path planning tasks with high DoF. Probabilistic Roadmaps (PRM) unlike RRT or RRT-connect has a learning phase. In this learning phase PRM will generate given amount of random configurations, check them for validity, and attempts to connect each node to k nearest neighbours or to the neighbours in some given distance. A local planer is used to make connections between these nodes and to verify their validity. In the planning phase, the learned roadmap is used to find the path between the start and the goal configurations. If goal and start configurations are not part of already learned graph, PRM will attempt to connect these configurations to the learned graph nodes in a similar manner as in learning phase. Finally, a simple graph path planning algorithm can be called such as A * [Hart et al., 1968b] or Djikstra's algorithm [Dijkstra, 1959] to find the path from start to goal configuration.

This work uses a RRT-based algorithm similar to RRT-connect. Multiple trees are created based on the information extracted from the video demonstration. These trees grow towards each other to find the path efficiently.

2.2 Learning from Demonstration

This work aims to use video demonstration to aid RRT-connect based planner in solving task and motion planning. There are other works using demonstrations in order to solve task and motion planning and other robotics problems. These works, unlike this one, use learning from demonstration.

Learning from demonstration is one of many approaches to policy learning where policy is a mapping between world state and actions. The policy can be trained by general reinforcement learning [Sutton and Barto, 2018] or by behavior cloning, which is a supervised learning technique, where labeled training data are provided to the agent. These training data are demonstrations of various solutions of the given problem [Argall et al., 2009]. Demonstration can be provided for example by manually guiding the robot through the task or by providing examples of human executing the task [Argall et al., 2009].

Works that utilize learning through demonstration differ in multiple aspects. One of the aspects is demonstration type. For example, work [Ye and Alterovitz, 2017] uses Kinesthetic demonstration, method in which robot is physically guided by human operators. In [Wen et al., 2022], human demonstration is recorded by camera depicting the operator manipulation of a known object. The object pose is acquired for the demonstration through 6D pose determining neural network. Work [Evrard et al., 2009] use demonstrations in which robots are being teleoperated by humans. This thesis utilizes recorded human demonstration, where the required information is the pose of the manipulated objects, similar to [Wen et al., 2022].

Difference can be also found in the number of demonstrations that are necessary for the learning to succesfully find a good policy. Works [Evrard et al., 2009] and [Wen et al., 2022] aim to use only one human demonstration

while work [Wu et al., 2020] try to limit the number of demonstrations to only a few expert demonstrations. In similar manner to [Evrard et al., 2009] and [Wen et al., 2022], we aim to use only one demonstration to solve the given task-and-motion planning problem.

Important distinction between works in learning from demonstration is their method of deriving policy. According to [Argall et al., 2009], there are three main approaches to policy derivation. Use of (i) mapping function, where policy is learned by fiding an approximation function to the state-action mapping, an approach chosen for example in [Jenkins et al., 2000], (ii) system model, where model of world dynamics is learned and policy is derived from this information, we can see this approach *e.g.* in [Ollis et al., 2007], or (iii) plans method, where a sequence of actions is produced by planner that utilizes trained model [Kuniyoshi et al., 1994]. This thesis will not be using any of the a aforementioned methods as our approach does not require learning phase as we will use information acquired from demonstration directly in path planning algorithm.

2.3 Task and Motion Planning

The goal of task and motion planning (TAMP) algorithms is to find a path that connects the given start and goal configurations that are defined in the task space. The one example of the task space was presented in [Mansouri et al., 2021] and it consists of a set of holes in the ground that multiple robots are supposed to drill. Whenever a hole is driller, no robot can longer move through it as excess material from the drilling form an obstacle. The goal of the planner is to find a feasible path, where all the holes are drilled and robots can safely return to the home position. In our case, the task space consists of a set of objects that robot needs to manipulate and rearrange.

PDDLStream

This **manipulation TAMP** was studied in several works. The notable one is PDDLStream [Garrett et al., 2018] and its predecessor STRIPStream [Garrett et al., 2017]. The first work STRIPStream uses Stanford Research Institute Problem Solver (STRIPS) [Fikes and Nilsson, 1971] to solve a planning problem on a symbolic level. The symbolic plan is then used by motion planner to find a robot motion which execution results in the given symbolic plan. Two algorithms were designed in [Garrett et al., 2017] to ground the symbolic planner into the robot motion: (i) incremental and (ii) focused algorithm. Both algorithms use *streams* from which the continuous quantities are sampled for the purpose of discrete symbolic planning. In the follow-up work, called PDDLStream [Garrett et al., 2018], authors used Planning Domain Definition Language (PDDL) [Aeronautiques et al., 1998] for solving symbolic planning problems. Additional two algorithms, (iii) binding and (iv) adaptive, were designed to solve the TAMP problems and increased the performance of the planning in one robot mobile manipulation task.

Combining discrete symbolic planning with continuous path planning allow to solve complex rearrangement tasks as shown in Fig. 2.3.

PDDLStream and STRIPStream both utilizes (i) symbolic planning provided by PDDL or STRIPS, and (ii) conditional sampling of continuous variables provided by streams. While these works aim to solve the task planning at symbolic level, we aim to guide the planner sampler via providing way points extracted from the video demonstration.

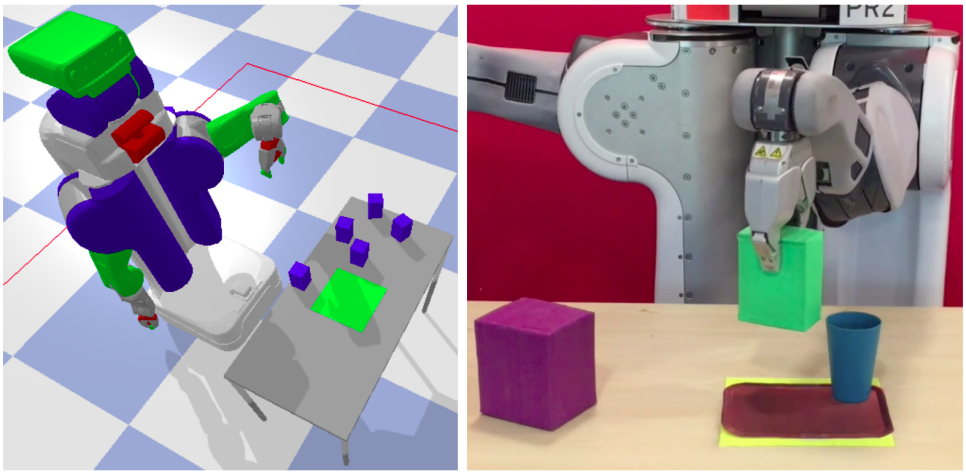

Figure 2.3: Example of rearrangement tasks solved by PDDLStream algorithm. On the left, simulation of task is shown where the robot needs to pack all blue cubes into the green regions. On the right a real-world alternative is shown where robot is supposed to "serve a meal" on the brown tray. This picture was taken from [Garrett et al., 2018].

Humanoid Path Planner

Humanoid Path Planner [Lamiraux and Mirabel, 2022] (HPP) is another work that is solving manipulation TAMP and is a crucial one for this work. HPP aims to solve prehensile manipulation planning, and it does so through use of constraint graph, a way of representing the numerical constraints of the manipulation problem used to model the manipulation planning as can be seen in Fig. 2.4.

These constrains are represented as submanifolds of the configuration space representing grasp and/or contact constraints. By modeling the problem through constraint graph a Manipulation-RRT, a proposed extension of RRT algorithm, is used to find a path between given configurations. It does so by using transitions between submanifolds to move in between them. By modeling through constraint graph and use of Manipulation-RRT, HPP can find grasp configurations in otherwise infinite space and solve problems if the following quantities are provided:

1. Description of movable objects, collision objects and robots in the scene;
2. Description of the contact surfaces;

3. Description of the handles of the movable objects;
4. Description of the grippers of the robots;
5. Scene configurations as a vectors of poses of the robots and movable objects present in the scene.

This work builds on HPP software and uses its libraries to implement our own planning algorithm which utilizes object poses extracted from video demonstration to create waypoints in the roadmap.

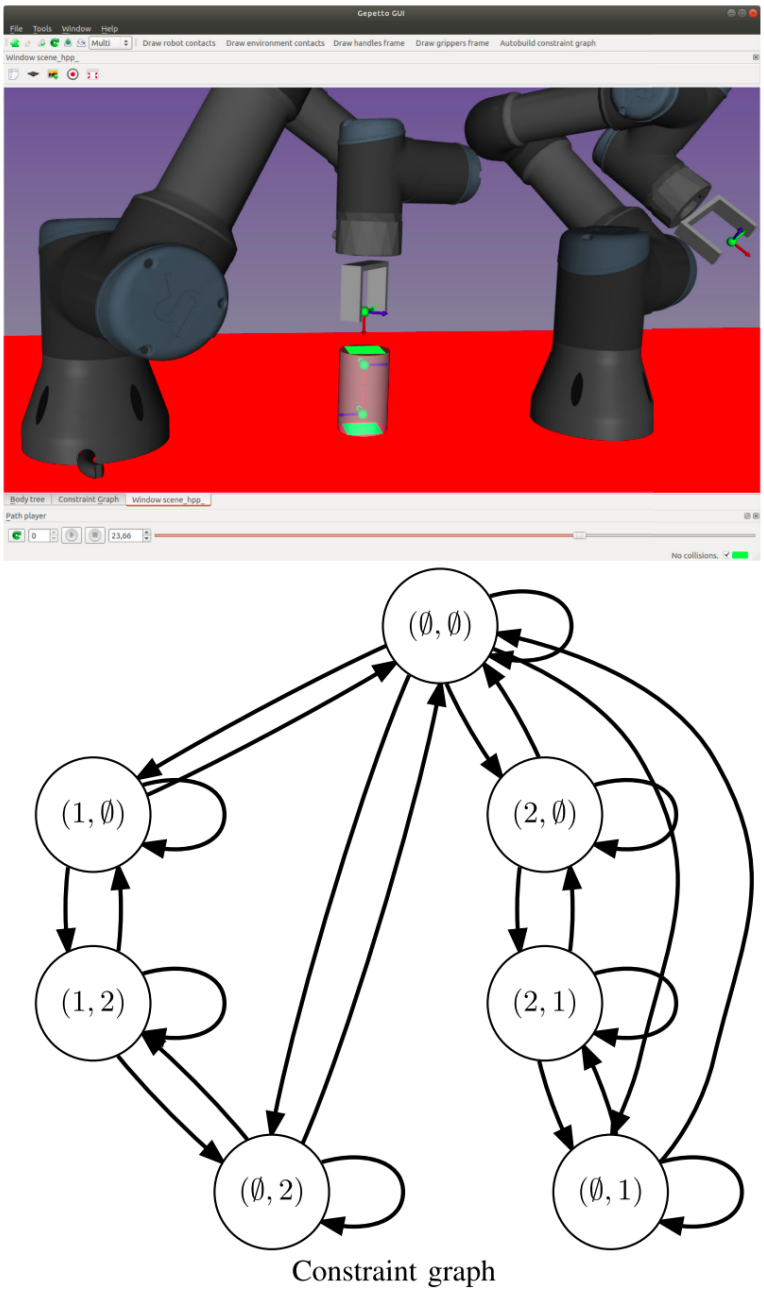

Figure 2.4: Example of manipulation planning problem. Top figure shows two UR3 robots with one gripper each manipulating a cylinder with two manually defined handles (shown as RGB frame inside the cylinder). The environment contains one rectangular contact surface visualized in red color. The cylinder has two rectangular contact surfaces (green). Bottom figure shows the corresponding constraint graph for two robots (*i.e.* grippers) and one object. Each node is represented by the state of each gripper, either holding no handle, handle one or handle two. For example, (∅, 1) means that the first robot's gripper is empty and the second gripper grasps handle 1 of the cylinder. In this state, there is no placement constraint, *i.e.* cylinder is not forced to lie on the ground. The placement constraint is enforced only for state $(\emptyset, \emptyset)$.

Chapter 3

Approach

This chapter formulates the addressed problem and describes the state-of-the-art solution that we proposed in [Zorina et al., 2023]. The first two sections of this chapter are based on results presented in [Zorina et al., 2023] while the extension, proposed in this thesis, is discussed in the last section.

3.1 Problem Formulation

The proposed method uses video demonstration to guide task and motion planning algorithm for the given task. Visualization of the overall pipeline of the method is shown in Fig. 3.1. The input to the proposed approach is the video demonstration, where a human solves the given task, and geometrical description of the scene. The output is a collision-free robot path in which the robot solves the task.

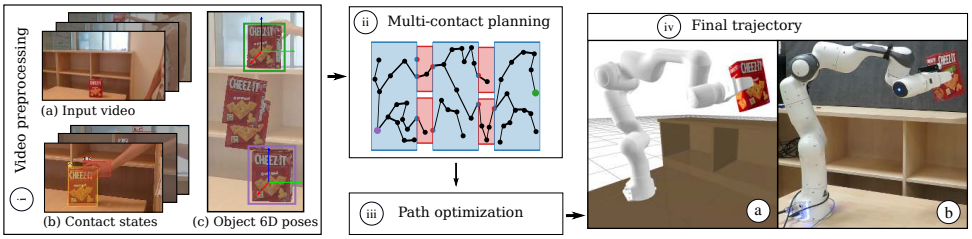

Figure 3.1: Approach overview. (i) First, we extract contact states and 6D object poses from the input instructional video. (ii) Next, we grow multiple trees in the admissible configuration space until we find a path between the start and goal configurations. (iii) This path is then further shortened by an optimization module, and (iv) executed either in simulation (iv-a) or on a real-world robot (iv-b). This figure was taken from [Zorina et al., 2023]

3.1.1 Video Demonstration

For each task, a video demonstration is necessary in order to acquire objects poses and human/object contacts from a human execution of the task. Portion of this demonstration can be seen in 3.4 The video demonstration depicts known manipulated objects, human solving the task and various furniture on

which the objects can be placed. In the demonstration, human solves the task by moving each object from their start state to their goal state always one by one. As this approach aims to solve task and motion planning problem with only one arm manipulators, just one object can be carried by the human in demonstration at a single time.

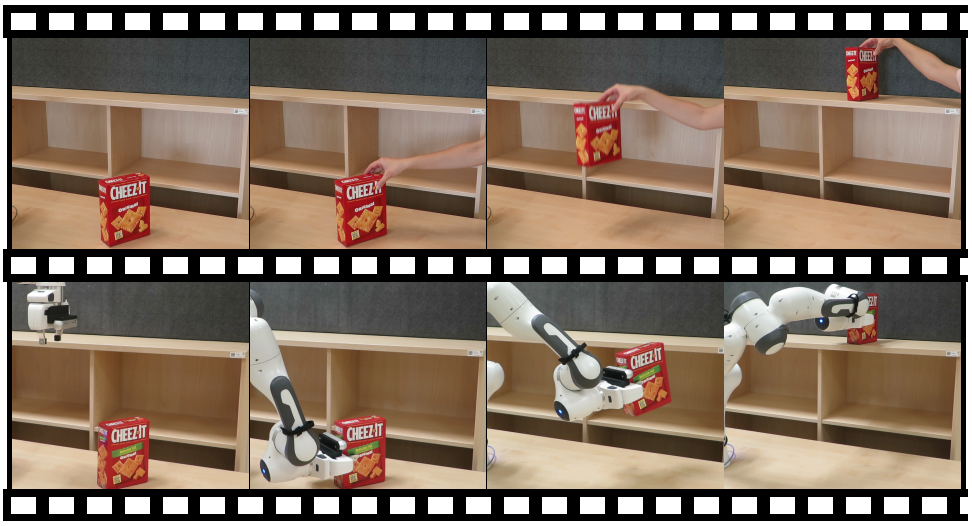

Figure 3.2: Previous results example. Example of the solution by the previous approach of one object pick and place task. Video demonstration at the top and real life robot demonstration at the bottom.

In previous approach, work [Shan et al., 2020] is used to obtain sequence of contact states $c_k \in \{\text{grasp, release, none}\}^N$, where N is the number of movable objects and the contact changes are indexed by $k \in \{1, \ldots, T\}$ where T is the number of changes in contact states. By using this convention we now have variable c_k that contains the contact state for each objects in the scene at the time of contact state change k. If object is not manipulated its contact state is set to 'none'. However, in the proposed approach, the number of c_k is insufficient as we aim to use the whole demonstration and not only the changes of contact states. We denote by c_t the contact state at time t, where $t \in \{1, \ldots, S\}$ where S is the number of frames of the input video.

Additionally to the contacts, multiple 6D poses are estimated from the demonstration. As we assume the objects in the demonstration are known, as it is in many cases of practical robotics set-ups, we can use a pose estimator of known objects. In our case, render and compare poses estimator CosyPose [Labbé et al., 2020] is used to estimate 6D poses of all the objects in all frames of the video. The 6D poses were previously sampled only at time-steps k, however in the propoesd approach we estimate 6D poses for each time-step t. Thus we can denote the objects poses as $A_t \in SE(3)$, where for each of the objects we estimate the 4×4 homogeneous matrix that describes the pose of the object in the common frame of reference. The contact states and object poses are used to constrain the configuration space in which the planning is performed.

3.1.2 Scene Description

In order to fully constrain the task , the tasks scene needs to be described geometrically. Thus 3D models of the furniture and manipulated objects are necessary as well as description of the robotic manipulator that is to be used to solve the task as well as its 3D model.

Each (i) manipulated objects has to have description of its handles and contact surfaces, (ii) furniture object of its contact surfaces and (iii) robot of its grippers and optionally contact surfaces. Contact surface is a set of points describing a surface on which the manipulated object can be placed. Handle is a pose in the reference frame of the object describing a specific grasp on the object by a gripper. Gripper definition is necessary in order to construct grasp between robot gripper and object handle. For the definition of handles, grippers and contact surfaces we followed [Lamiraux and Mirabel, 2022].

3.1.3 Task Configuration Space

In order to find path in the tasks configuration space, we will need to constrain it to create an admissible configuration space. These constraints will prevent configurations where the object is free in the air or moving without robots assistance. In [Lamiraux and Mirabel, 2022], which is used internally in our method, the admissible configuration space is represented by states. The configurations in these states are represented by object poses and robot configuration. Each of these states and its configurations are subjected to placement and/or grasp constraints. These constraint enforces that object is either (i) grasped by the robot and static in the reference frame of the gripper or (ii) placed on a contact surface and static in the reference frame of the contact surface.

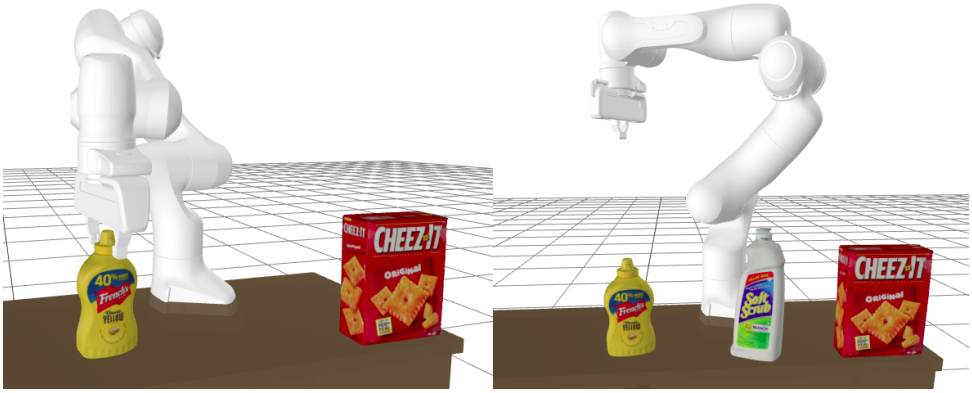

Figure 3.3: Admissible configurations example. On the left, example configuration from the intersection of placement and grasp states and on the right example configuration from the placement state.

We define placement states $\mathcal{P}_i$, as states in which all of the objects are subjected to the placement constraint. We define Grasp states $\mathcal{G}_i$, as states in which one, or multiple objects, are subjected to the grasped constraint

and the rest of the objects are subjected to the placement constraint. For example, in [Lamiraux and Mirabel, 2022] state 'free' is a placement state of admissible configurations in which all of the objects are placed on contact a surfaces and a robot gripper is free of any objects i.e. Fig 3.3. Placement and grasp state can be transitioned through the intersection of neighbouring states, *i.e.* a state in which both the placement and grasp constraints are satisfied for one object. This means that the object is on a contact surface but grasped by a robot, i.e. Fig 3.3

3.2 Existing Approach

In [Zorina et al., 2023] the aim was to simplify the planning by simplifying the admissible configuration space with the use of recognized contact states and detected 6D object poses from the demonstration video. By constructing placement states from unique sets of object poses from the instructional video, the pick and place order of movable objects in the demo can be deduced. This is due to the fact that consecutive placement states are connected by grasp states as the object needs to be grasped and moved in order to change the placement state, i.e. to change the pose of one object.

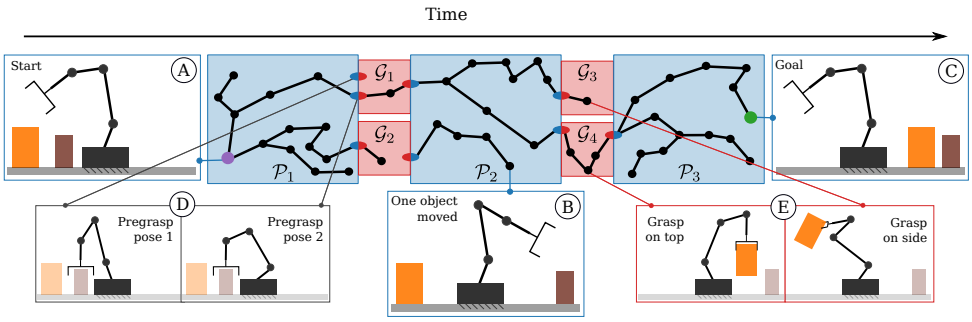

Figure 3.4: The admissible configuration space. The following configurations A, B and C lie in placements states $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$. Configurations in D lies in the neighboring state. And configurations E lie in grasp states $\mathcal{G}_3$ and $\mathcal{G}_4$.

In example Fig. 3.4 the simplified admissible space for two movable objects and one robot can be seen. As we can see the consecutive placement states are connected by grasp states as the object needs to be grasped and moved in order to change the placement state, i.e. to change the pose of one object. We have discussed in Subsec. 3.1.2 that there can be multiple handles for each object. This means that there are multiple grasp states, meaning multiple ways to grasp a specific object. In the Fig. 3.4 each object has two defined handles, (i) grasp on top and (ii) grasp on the side. Three placement states have been created, (i) placement state $\mathcal{P}_1$ is constrained (given) by the initial poses of objects, the next (ii) placement state $\mathcal{P}_2$ is constrained by the starting pose of the orange object and the goal pose of the brown object, and (iii) the last placement state $\mathcal{P}_3$ is constrained by the goal poses of both objects.

For the path to be feasible, it must contain only the configurations from the admissible space and satisfy additional constraints of the environment, which means that the robot configuration must be within the joint limits, and the configuration must be collision-free. We refer to the space in which the configurations satisfy all restrictions as $\mathcal{C}_{\text{free}}$.

Transitioning between the placement and grasp states, which is necessary to move the objects, remains a challenge for RRT sampling-based planners as it needs to sample configurations that satisfy constraints from both neighboring states. To address that issue, work [Zorina et al., 2023] designed an extension of RRT-connect [Kuffner and LaValle, 2000] that grows multiple trees simultaneously with tree roots sampled at the transitions between placement and grasp states. The overview of the algorithm [Zorina et al., 2023] is shown in Alg. 1. The algorithm is split into two main routines: (i) sampling of a new tree at the transition between the placement and grasp states, and (ii) growing an existing tree at a randomly selected state. routines are randomly selected to determinated which will be used in each iteration with a Bernoulli distribution controlled by a parameter $\eta_{\text{sample\_tree}}$. The algorithm stops when both start and goal configurations are connected into a single tree or if the algorithm runs out of resources, e.g. maximum number of iterations or maximum planning time.

3.2.1 Sampling Configuration From Given State

One of the main capabilities required by the algorithm is sampling from the given state. Similarly to [Lamiraux and Mirabel, 2022], a random configuration is sampled from Euclidean space and a numerical solver [Nocedal and Wright, 2006] is called iteratively to compute the configuration that satisfies the numerical constraints of the state. In order to sample from the transition connecting two states, the constraints from both states are merged. However, to assign a unique state to each configuration sampled from the transition, two identical configurations $\boldsymbol{q}_{\text{from}}$ and $\boldsymbol{q}_{\text{to}}$ are constructed and states are assigned to them.

3.2.2 New Tree Sampling at Transition

A sampling of a new tree at transition between the placement and grasp states is performed as follows. First, a pair of configurations $(\boldsymbol{q}_{\text{from}}, \boldsymbol{q}_{\text{to}})$ is sampled so that they would satisfy the constraints of the randomly selected transition. If both configurations also satisfy the constraints of the environment (*i.e.* respect the joint limits and are collision-free), a new tree containing a root $(\boldsymbol{q}_{\text{from}})$ is created with a leaf $(\boldsymbol{q}_{\text{to}})$. Attempt to link both the created configurations to the existing trees in their corresponding states is made.

3.2.3 Tree Growing

Growing the tree in a randomly selected state is performed as follows. A new configuration is sampled in the admissible configuration space and the tree is extended in its direction. A state $\mathcal{S}$ is chosen at random and

Algorithm 1 Multi-contact RRT guided by demonstration

Require: Contact states c_k, object 6D poses A_k, $\boldsymbol{q}_{\text{start}}$, $\boldsymbol{q}_{\text{goal}}$, $\eta_{\text{sample\_tree}}$, step size δ

$\mathcal{T} \leftarrow \{\text{init\_tree}(\boldsymbol{q}_{\text{start}}), \text{init\_tree}(\boldsymbol{q}_{\text{goal}})\}$ ▷ Existing trees
repeat
  $p \sim \mathcal{U}(0, 1)$
  if $p < \eta_{\text{sample\_tree}}$ **then** ▷ Sampling a new tree
    $k \sim \{1, \ldots, T\}$
    $\boldsymbol{q}_{\text{from}}, \boldsymbol{q}_{\text{to}} \leftarrow \text{sample\_on\_transition}(c_k, A_k)$
    if $\boldsymbol{q}_{\text{from}} \notin C_{\text{free}}$ **or** $\boldsymbol{q}_{\text{to}} \notin C_{\text{free}}$ **then**
      continue
    end if
    $t \leftarrow \text{init\_tree}(\boldsymbol{q}_{\text{from}})$
    $t.\text{add\_edge}(\boldsymbol{q}_{\text{from}}, \boldsymbol{q}_{\text{to}})$
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{t\}$
    $\text{attempt\_link}(\boldsymbol{q}_{\text{from}}, \mathcal{T}, \delta)$ ▷ Alg. 2
    $\text{attempt\_link}(\boldsymbol{q}_{\text{to}}, \mathcal{T}, \delta)$ ▷ Alg. 2
  else ▷ Tree growing
    $\mathcal{S} \leftarrow \text{sample\_state}(\mathcal{T})$
    $\boldsymbol{q}_{\text{rand}} \leftarrow \text{sample\_configuration}(\mathcal{S})$
    $\boldsymbol{q}_{\text{nn}} \leftarrow \text{nearest\_neighbor}(\boldsymbol{q}_{\text{rand}}, \mathcal{S}, \mathcal{T})$
    $\boldsymbol{q}_{\text{step}} \leftarrow \boldsymbol{q}_{nn} + \delta(\boldsymbol{q}_{\text{rand}} - \boldsymbol{q}_{\text{nn}})$
    if $\boldsymbol{q}_{\text{step}} \notin C_{\text{free}}$ **then**
      continue
    end if
    $t \leftarrow \text{get\_tree}(\boldsymbol{q}_{\text{nn}})$
    $t.\text{add\_edge}(\boldsymbol{q}_{\text{nn}}, \boldsymbol{q}_{\text{step}})$
    $\text{attempt\_link}(\boldsymbol{q}_{\text{step}}, \mathcal{T}, \delta)$ ▷ Alg. 2
  end if
until $\text{get\_tree}(\boldsymbol{q}_{\text{start}}) = \text{get\_tree}(\boldsymbol{q}_{\text{goal}})$ or out of resources

configuration $\boldsymbol{q}_{\text{rand}}$ is sampled from it. The nearest neighbor of the sampled configuration $\boldsymbol{q}_{\text{nn}}$ is found so that $\boldsymbol{q}_{\text{nn}}$ also lies in the state $\mathcal{S}$. A new configuration $\boldsymbol{q}_{\text{step}}$ is computed along the segment between $\boldsymbol{q}_{\text{nn}}$ and $\boldsymbol{q}_{\text{rand}}$ in the manually defined step-size distance δ from $\boldsymbol{q}_{\text{nn}}$. If $\boldsymbol{q}_{\text{step}}$ is collision-free, it is added to the tree. Finally, a link is attempted with the new configuration to the existing trees that have nodes in the same state.

3.2.4 Attempt to Link

This function, described in Alg. 2, is used by Alg. 1 to connect given configuration $\boldsymbol{q}$ to the existing trees. Its goal is to search for a linear path between the given configuration $\boldsymbol{q}$ and the trees with nodes in the same state as the configuration. For each tree, the nearest neighbor $\boldsymbol{q}_{\text{nn}}$ is found. Then, configurations along the segment from $\boldsymbol{q}_{\text{nn}}$ to $\boldsymbol{q}$ are added to the tree unless a collision is detected. If the entire path is collision-free, we connect the trees

Algorithm 2 Attempt to link function connects a given configuration to other trees

Require: configuration $\boldsymbol{q}$, set of trees $\mathcal{T}$, step size δ,

```
1: S ← get_state(q)
2: for t ∈ T \ get_tree(q) do
3:     if t has nodes in S then
4:         q_nn ← nearest_neighbor(q, S, {t})
5:         q_parent ← q_nn
6:         for q_step ∈ {q_nn, q_nn + δ, ..., q_nn + nδ, q} do
7:             if q_step ∉ C_free then
8:                 break
9:             end if
10:            t.add_edge(q_step, q_parent)
11:            q_parent ← q_step
12:        end for
13:        if q_step = q then
14:            merge t and tree containing q
15:        end if
16:    end if
17: end for
```

that contain the configurations $\boldsymbol{q}_{\text{nn}}$ and $\boldsymbol{q}$.

3.3 Proposed Approach

The previous approach [Zorina et al., 2023] have successfully solve the task-and-motion planning as can be seen in Fig. 1.1. The existing method is robust and efficient, but the demonstration can still be used furthermore as we propose in this section. Cosypose [Labbé et al., 2020], the method we have used to capture known object poses from the demonstration video, can be used to estimate object poses not only in the neighboring and placement states but also in the grasp state.

We propose, to use the estimated object poses to further aid and guide the method in constructing the tree in the admissible configuration space. Instead of sampling configurations randomly in the whole state space, we sample configurations in the neighborhood of the estimated poses. This way, we can create grasp state configurations by solving inverse kinematics. This new acquired configurations are used instead of randomly generated configuration for the given state, thus guiding the planner by providing it with configurations along the demonstration.

We propose a modification of the Alg. 1 in which we replace the line 17 with calling of the proposed method shown in Alg. 3. The proposed modification acquires the random configuration $\boldsymbol{q}_{\text{rand}}$ by utilizing the given demonstration. Individual subroutines of the algorithm are described next.

Algorithm 3 Generate demo based configuration function generates grasp state configuration based on demonstration

Require: Contact states c_t, poses A_t, state $\mathcal{S}$, length noise parameter Δ_l, angle noise parameter Δ_α,

if $\mathcal{S}$ is $\mathcal{G}_i$ **then**
  $\boldsymbol{t}, R \leftarrow$ random_object_pose(c_t, A_t, $\mathcal{S}$)
  $\delta\boldsymbol{t} \sim \mathcal{N}(\mathbf{0}, \Delta_l^2 I)$ ▷ I is 3x3 identical matrix
  $\delta\boldsymbol{r} \sim \mathcal{N}(\mathbf{0}, \Delta_\alpha^2 I)$
  $t \leftarrow t + \delta\boldsymbol{t}$
  $R \leftarrow R\exp(\delta\boldsymbol{r})$
  $\boldsymbol{q}_{rand} \leftarrow$ inverse_kinematics($\boldsymbol{t}, R$, $\mathcal{S}$)
else
  $\boldsymbol{q}_{\text{rand}} \leftarrow$ sample_configuration($\mathcal{S}$)
end if

3.3.1 Random Object Pose

This function randomly select an estimated object pose that correspond to the given state $\mathcal{S}$. For example, if the state correspond to *grasping the first object* it will randomly select the pose from the demonstration such that the first object is grasped by human operator. Note, that we are selecting only the poses of manipulated objects (*i.e.* in grasp state) as they can be used to guide the robot. If object is not grasped, we select the robot configuration randomly.

3.3.2 Adding Noise

In this function we add random noise to the pose of the grasped manipulated object in order to boost the exploration capability of the planner and to mitigate the influence of limited reachability space of the robot. The noise is applied only to the manipulated object, not to the other objects in the scene. The noise parameters Δ_l and Δ_α represents the standard deviations of Gaussian distributions $\mathcal{N}(0, {\Delta_l}^2)$ and $\mathcal{N}(0, {\Delta_\alpha}^2)$ with three dimensional zero mean. The translation and rotation noise is sampled from the distributions and added to the randomly selected object pose. For the rotation, we sample a rotation vector that is transform to rotation matrix via the exponential mapping. This noisy object pose is used to solve inverse kinematics to obtain configuration that is used to guide the planner.

3.3.3 Inverse kinematics

In order to solve inverse kinematics task for the given object pose, we use HPP [Lamiraux and Mirabel, 2022]. HPP utilizes constraints for defining the grasp and placement states. These constraints create an interconnected state spaces which they define as constraint graph, for example see HPP constraint graph shown in Fig. 2.4. Additionally, HPP allows to define new constraints

in the constraint graph. By creating a constraint in which we set the pose of the grasped object, we create a state in which the object can be suspended in the air without the aid of the robot. Next, by creating a transition from this state to the desired grasp state, we now have a neighboring state, in which we can sample grasp configuration for a fixed object pose.

In practice, we internally use two HPP constraint graphs. One is used only for the creation of the new constraints in order to create the state, from which we can sample the grasp pose while the other one is used for planning. By giving the inverse kinematics function the noised manipulated object pose and the grasp state, we can compute configuration of the robot that grasps the object at the given noisy pose. If inverse kinematics fails to find a solution, it return *None* and algorithm continues with the next iteration.

Chapter 4

Benchmarking

In order to test newly proposed extension to [Zorina et al., 2023], the benchmarking of the extension will be done across a few types of task and TAMP problems, similarly as done in [Zorina et al., 2023]. All of the following tasks are parameterizable and for the algorithm to solve them a demonstration is required. The tasks, demonstrations and the metrics used to evaluate the approach are described in this section.

4.1 Tasks

Three challenging tasks were defined to benchmark taks-and-motion planning algorithms: (i) shelf task, where goal is to rearrange objects into the shelf; (ii) waiter task, where goal is to use mobile robot to rearrange objects; and (iii) tunnel task, where goal is to move object through the narrow passage.

4.1.1 Shelf Task

The scene of this task is composed of (i) two furniture objects, (ii) n amount of various manipulatable objects and (iii) a robotic manipulator. Value of n varies, we use one, two and three objects in this thesis. The furniture consists of one table and one shelf and tables height, length and depth are parameterizable. The furniture poses can vary, but are constrained as the task must be solvable by non-mobile robot. The robot pose can also vary. Rendered example of the shelf task is shown in Fig. 4.1.

The goal of this task is to rearrange given objects from their starting pose to their goal pose by using the robotic manipulator. For example, moving the objects from the table to the shelf or vice versa. Three robots were configured for this task: UR5, Franka Emika Panda, and KUKA IIWA. This task challenges state-of-the-art planners as multiple objects needs to be rearranged in a single task. Complexity of this task can be controlled through the amount of the objects needing rearrangement. The start and goal poses of the objects are extracted from the demonstration and the pose of the robot is sampled multiple times for each demonstration.

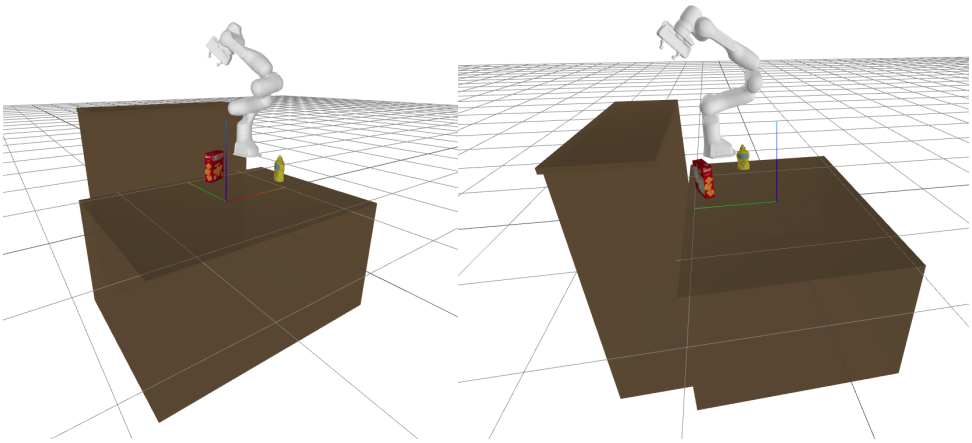

Figure 4.1: Rendering of the shelf task from two different viewpoints. There is (i) one table, (ii) one shelf, (iii) robotic manipulator Franka Emika Panda and (iv) two objects from YCBV dataset shown in their starting configuration. The goal is to move the objects onto the shelf.

4.1.2 Waiter Task

The scene of this task is composed of (i) mobile robotic manipulator, (ii) n manipulated objects, (iii) and furniture that consists of tables and shelf as in shelf task. The main difference to the shelf task is the use of mobile robotic platform. We use KUKA IIWA mounted on KUKA KMR platform for this task. Unlike in the shelf task there is no varying robot pose as the robot is of mobile type. Rendering of waiter task can be seen in Fig 4.2.

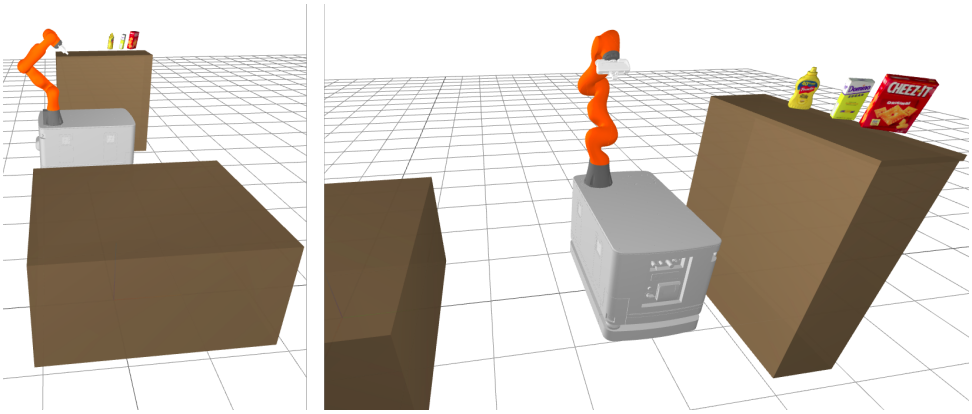

Figure 4.2: Rendering of the waiter task from two different viewpoints. There is (i) one table, (ii) one shelf, (iii) mobile robotic manipulator Kuka IIWA KMR and (iv) three objects from YCBV dataset.

Waiter task simulates the job of a waiter. Waiters, in order to minimize the traveled distance, use trays to transfer multiple objects at once. In this task a mobile robotic manipulator with a tray-like placement surface is tasked to move n amount of various objects across two locations at distance (*i.e.* shelf

and table). The tray-like placement surface in front of the robot can be used to move the objects around the scene.

The challenge for state-of-the-art planners is discovering the tray surface as a mean of minimizing the distance traveled between the two locations. Complexity can be controlled through the amount of objects that needs to be transferred.

4.1.3 Tunnel Task

Tunnel tasks scene is composed of (i) one manipulatable object, (ii) fixed robotic manipulator and (iii) parameterizable tunnel. Rendering of this task is shown in Fig 4.3. Goal of this task is to move the object with the robotic manipulator through the tunnel. This can be achieved by inserting the object into the tunnel from one side and than picking it on the other side. Robot cannot move the object around the tunnel as this is forbidden by invisible wall.

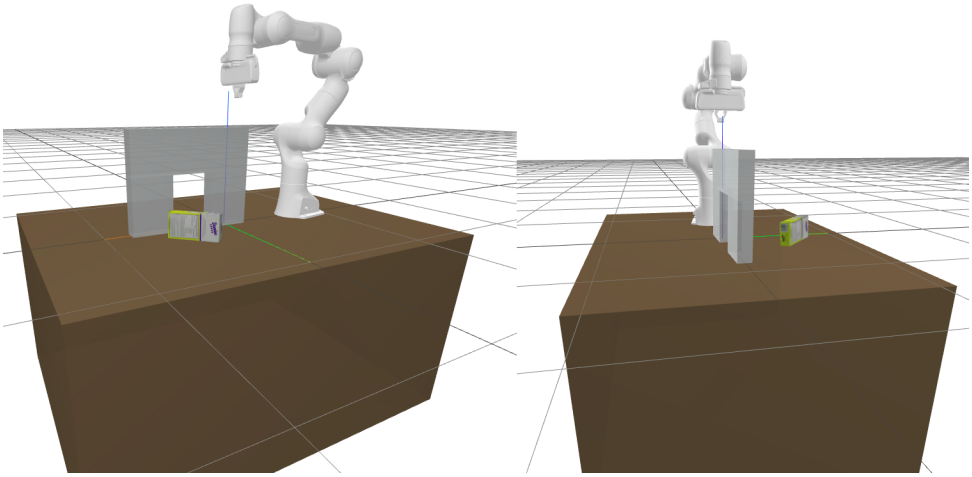

Figure 4.3: Rendering of the tunnel task from two different viewpoints. There is (i) one table, (ii) one tunnel, (iii) robotic manipulator Franka Emika Panda and (iv) one object from YCBV dataset. The goal is to move the object through the narrow passage inside the tunnel.

The tunnel has following parameters, (i) tunnel width, (ii) tunnel length, (iii) tunnel height and (iv) thickness of tunnel walls. Its pose may vary. Similar to the shelf task, robot pose is sampled based on the demonstration.

The challenge of this task for planners is the narrow passage of the tunnel and the necessity to regrasp the object inside the tunnel. The planner needs to discover this passage in the configuration space in order to solve this task.

4.2 Demonstration

Each demonstration is tied to specific task. We record human demonstrations by camera and extract the manipulated object poses. The post-processed

demonstration includes: (i) robot, furniture and object types, (ii) poses of robot and furniture, (iii) object poses acquired from the human demonstration, (iv) contacts (*i.e.* grasped/released) information and (v) furniture parameters *e.g.* dimensions. Rendering of post-processed demonstration is shown in Fig 4.4.

The start and goal pose of the objects are extracted from the demonstration and used to specify the task instance. Several robot poses are sampled randomly in such a way that both start and goal poses are reachable by the robot. These poses are then stored alongside the human demonstration to achieve repeatability of the benchmark. This description carried in the demonstration guaranties that all the objects are reachable in their start and goal pose, by the given robot, from the the given robot pose. Rendering of different robot poses is shown in Fig 4.5.

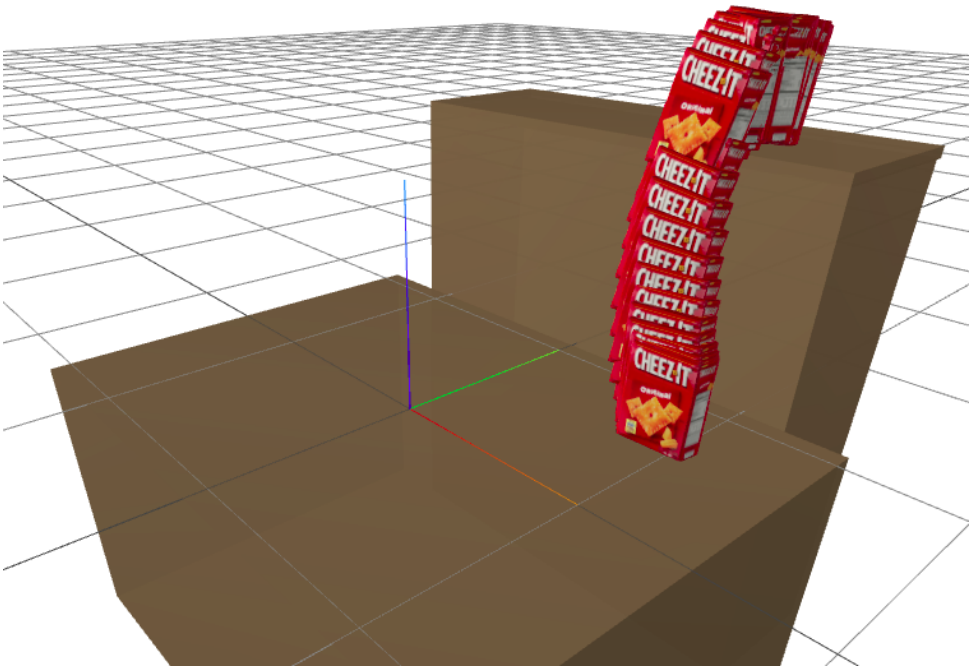

Figure 4.4: Rendering of the demonstration for the shelf task with one object. The figure shows the temporal evolution of the cheez-it box object poses.

4.3 Metrics

In order to compare the proposed approach with the previous one suggested in [Zorina et al., 2023] and other works attempting to solve TAMP tasks the following metrics will be evaluated:

1. The ability of the algorithm to solve the task. For each task there is a time limit in which the solver needs to solve the task. For shelf and tunnel tasks, the limit is 180 seconds.

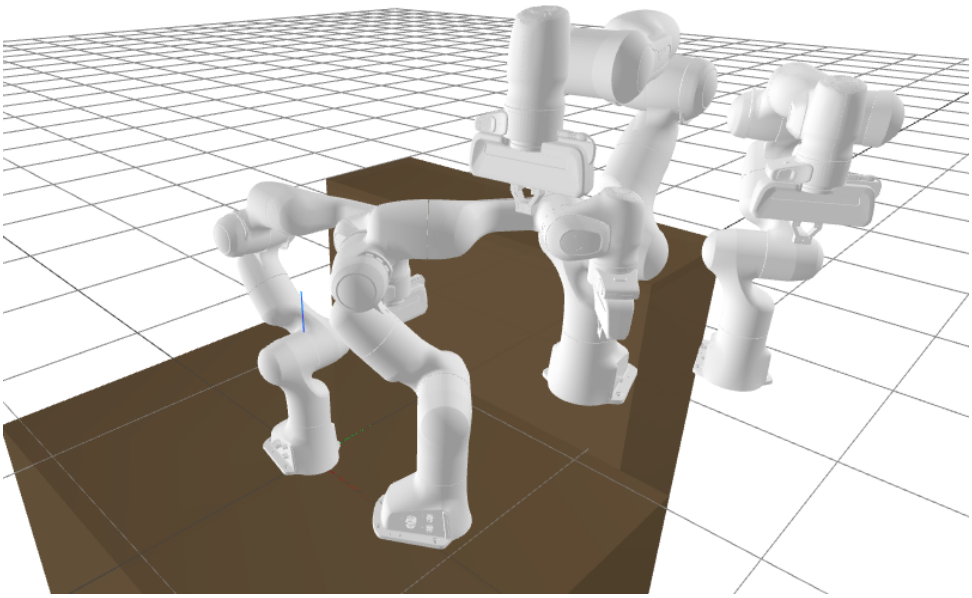

Figure 4.5: Rendering of few of the robot poses from the demonstration for shelf task. Note, that the robot can levitate for the purpose of benchmark. In real robot experiment the robot pose would be attached to the table desk.

2. The time required for the algorithm to solve the task. This will only apply for all the results that were able to solve the task in the given time limit.
3. Number of iterations required to solve the task.
4. The path length of the planned path. Specifically, the sum of euclidean distances in between the consecutive robot configurations of the planned path.

Chapter 5

Experiments and Results Discussion

In this chapter we provide results of series of experiments conducted with the newly proposed method. We also provide benchmarks of the newly proposed method with the previous method [Zorina et al., 2023] and state-of-the-art planner [Lamiraux and Mirabel, 2022]. However, in the chapter 4 we have introduced over all three tasks on which to benchmark but due to insufficient time and its complexity we did not menage to benchmark on waiter task. Nevertheless, the proposed waiter task was used in [Zorina et al., 2023].

5.1 Finding Noise Parameters

Our method utilizes a noise function as explained in Subsec. 3.3.2. The noise function utilizes two parameters, Δ_l and Δ_α, which are used to sample the noise vectors. In this experiment we attempt to find the value of the noise parameters that yields the best results.

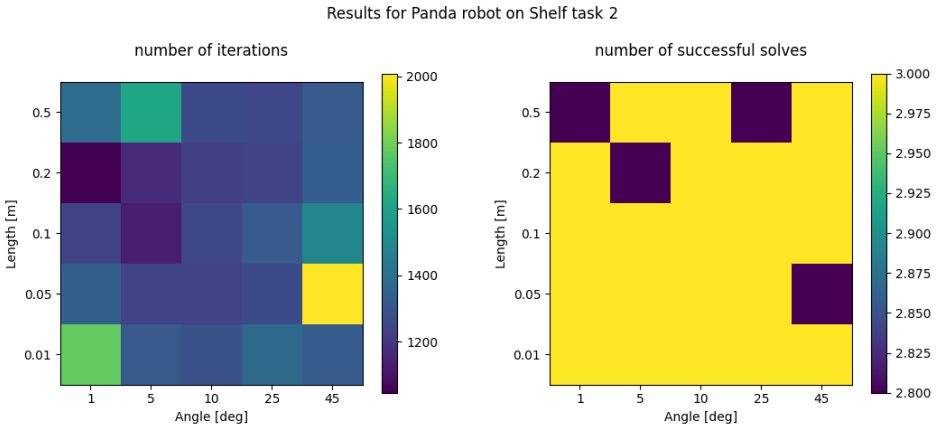

Figure 5.1: Plot of the results for different noise parameters for Franka Emika Panda robot solving shelf task 2. Averaged number of iterations is shown on the left and averaged number of successful solves is shown on the right.

For the first attempt we have selected the model of the Franka Emika Panda robot and the shelf task 2 (*i.e.* the shelf task environment with two objects), described in Subsec. 4.1.1, to test upon. We solve the task for each

set of parameters on five robot poses and each of the robot poses on three different random seeds. Thus for each set of parameters we attempt to solve the task fifteen times. The results of this benchmark can be seen in Fig. 5.1.

The interesting part about the noise parameter results is the fact that lover angle noise yields better results. This might have been caused by the fact that the object handles can be now rotated in a way that they are hard to reach for the robot and so the inverse kinematics fail in creating the configuration. Other possible explanation is that with higher angular noise, the configurations are now more distorted and further away from the ones in the tree for the direct planner to connect. Based on the results presented in Fig. 5.1, we selected noise parameters $\Delta_l = 0.2$ m and $\Delta_\alpha = 1°$ to be used in the rest of the experiments.

5.1.1 Results for Found Noise Parameters

With the newly acquired results for noise parameters $\Delta_l = 0.2$ m and $\Delta_\alpha = 1°$ we can now compare the results of the proposed method against [Zorina et al., 2023]. We benchmark across shelf task 1 - 3 (*i.e.* from one to three manipulated objects in the scene) and tunnel task for robots: Franka Emika Panda, UR5, and KUKA IIWA. For every task, we run the benchmarking for ten different robot poses and for every pose we run the benchmark for 10 random seeds. Therefore, we solve each task 100 times for a single robot.

Results for Panda robot

For Franka Emika Panda robot the newly proposed method outperfrom the state of the planner in terms of iterations as can be seen in Fig. 5.3 for the shelf task and in Fig. 5.2 for the tunnel task. In terms of success rate, the methods yield similar results. However, the new methods results were poor in solving time. Even though it took less iterations to find the solutions it took longer time to find it. More on this topic is given in Section 5.1.2.

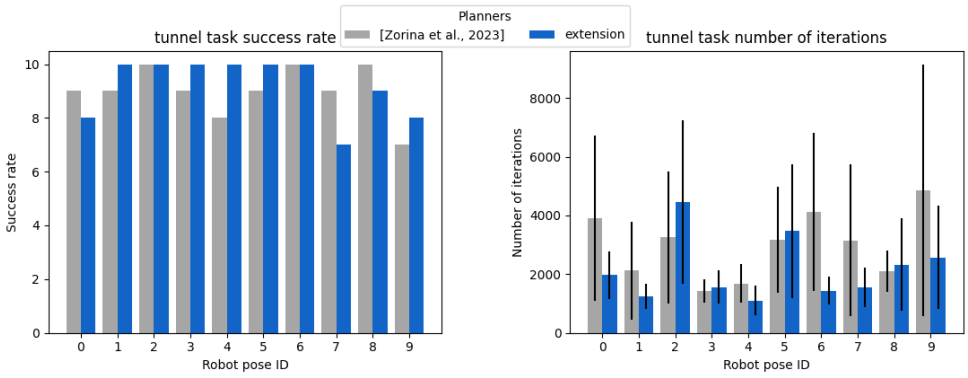

Figure 5.2: Results of benchmarking for Franka Emika Panda robot on tunnel task with the following noise parameters $\Delta_l = 0.2m$ and $\Delta_\alpha = 1°$. In grey are the results of previous method and in blue the results of the new one. On the left, number of successful solutions and on the right number of iterations.

Interesting part about the results are the poses in which [Zorina et al., 2023] yields better results in number of iterations. It seems that in these poses the problems can get solved in low amount of iterations by both of these methods. For example the poses 2, 9 and 10 in Fig. 5.3. It seems like a trend that the proposed approach usually takes more iterations to solve in these cases. Perhaps the problem in these poses for the given robot is that they are simpler than in the other cases. Thus when it comes to solving it only few random configurations in grasped plane are perhaps necessary. So when the proposed method tries to construct grasp configuration in a direction of the demonstration, the [Zorina et al., 2023] just shoots a random configuration in that space. In these cases the random configuration might have a higher chance of being successfully created than configuration for which inverse kinematics needs to be solved. And since it gained the configuration faster, it gets the solution faster too.

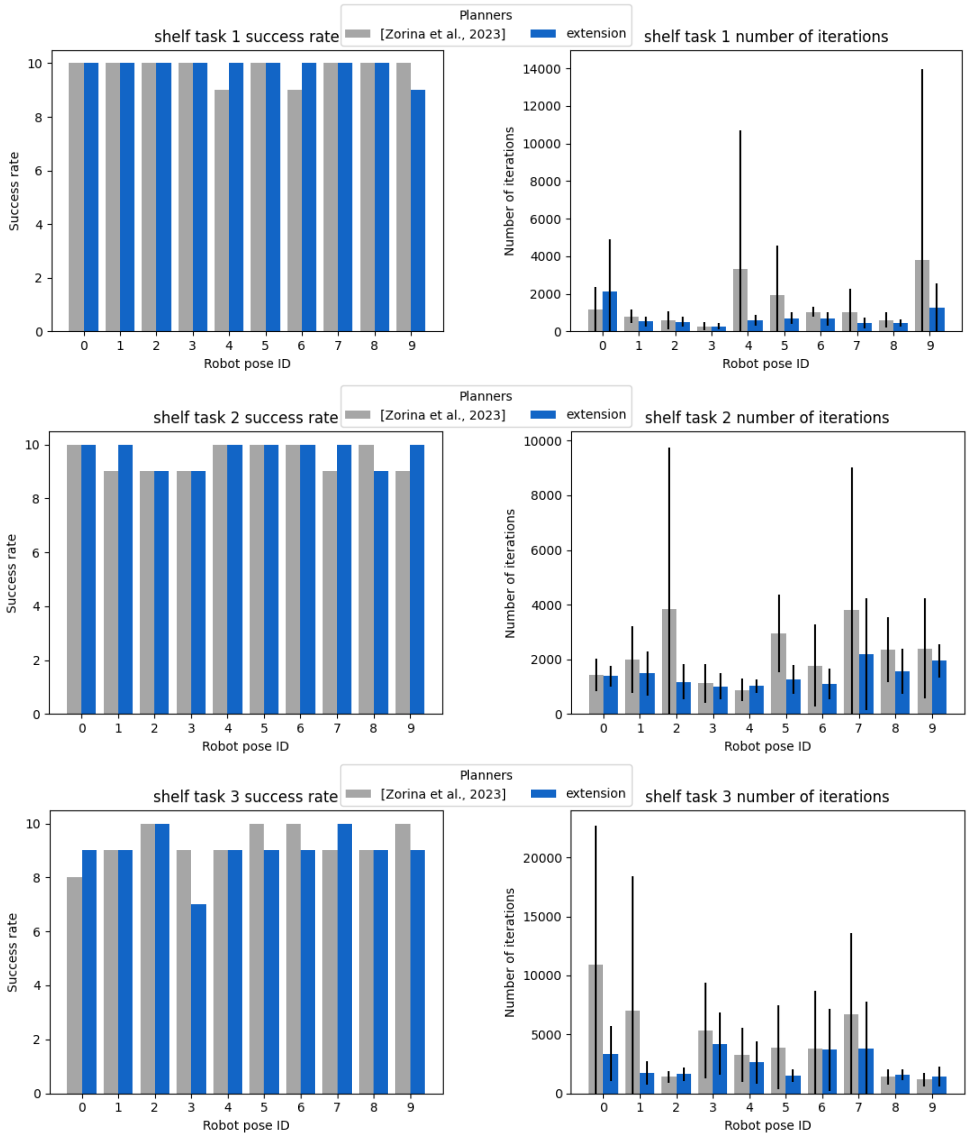

Figure 5.3: Results of benchmarking for Franka Emika Panda robot on shelf task 1 - 3 with the following noise parameters $\Delta_l = 0.2$ m and $\Delta_\alpha = 1°$. In grey are the results of previous method and in blue the results of the new one. On the left, number of successful solutions and on the right number of iterations.

Results for UR5 robot

Result for UR5 robot yielded similar results as for the Panda robot in the case of shelf tasks. The number of iterations has been reduced. The number of successful solutions stayed overall the same. And the solution time was still worse than the one of [Zorina et al., 2023]. The results can be seen in Fig. 5.5. However in the case of tunnel task, there was no overall improvement in the iterations from previous method, maybe even slight deterioration, as can be seen in Fig. 5.4. A slight deterioration can be also seen in the success rate.

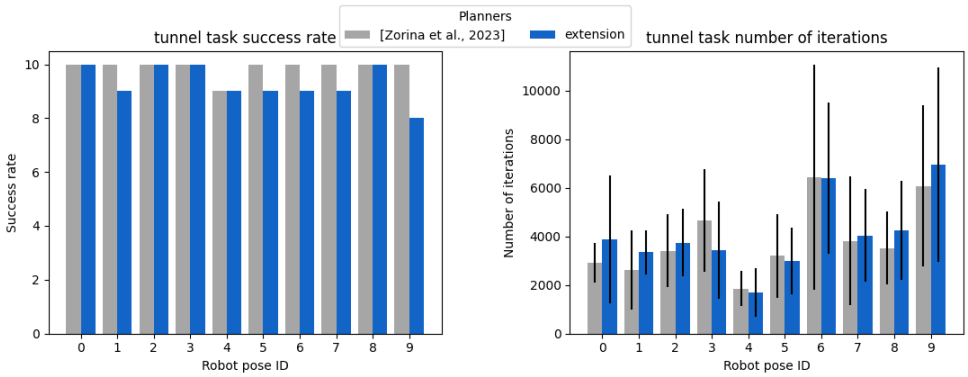

Figure 5.4: Results of benchmarking for UR5 robot on tunnel task. In grey are the results of previous method and in blue the results of the new one. On the left, number of successful solutions and on the right number of iterations.

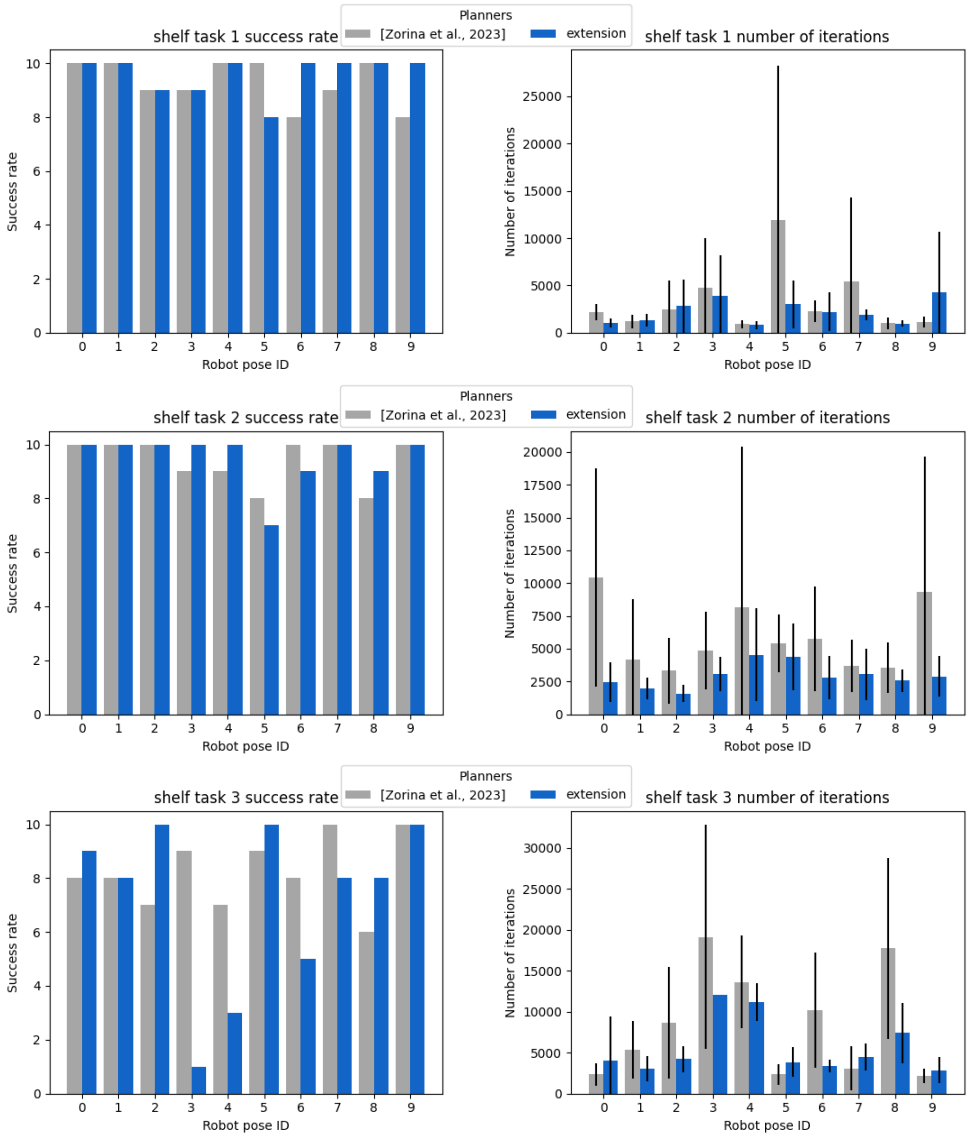

Figure 5.5: Results of benchmarking for UR5 robot on shelf task 1-3 with the following noise parameters $\Delta_l = 0.2$ m and $\Delta_\alpha = 1°$. In grey are the results of previous method and in blue the results of the new one. On the left, number of successful solutions and on the right number of iterations.

Results for Kuka IIWA robot

Result for Kuka IIWA robot had brought some different results. The extension no longer yields the smallest amount of iterations against [Zorina et al., 2023]. However, when we look into the success rate we will see that [Zorina et al., 2023] has failed more times in solving then the proposed extension. This indicates that the reason for lower amount of iterations on the previous approach side is caused by not being able to solve the task as the new approach solves it but at the cost of more iterations. The results can be seen in Fig. 5.7. However in the case of tunnel task, there was no overall improvement in the iterations from previous method, maybe even slight deterioration, as can be seen in Fig. 5.6. Though unlike in the case of UR5 robot the success rate state overall similar.

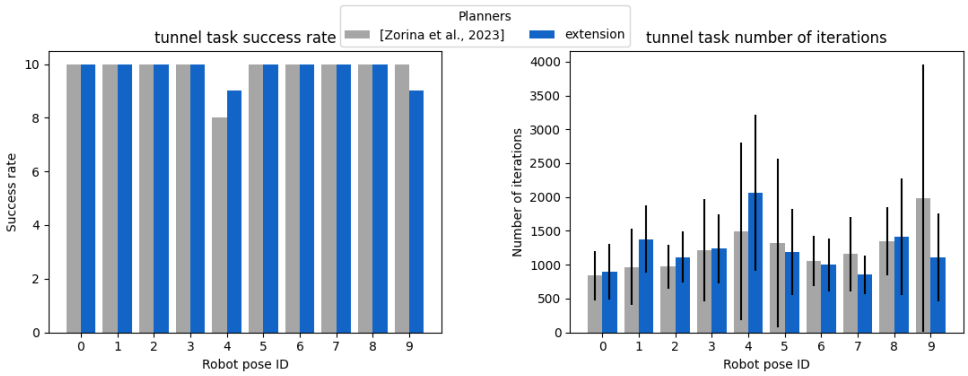

Figure 5.6: Results of benchmarking for Kuka IIWA robot on tunnel task with the following noise parameters $\Delta_l = 0.2m$ and $\Delta_\alpha = 1°$. In grey are the results of previous method and in blue the results of the new one. On the left, number of successful solutions and on the right number of iterations.

5.1.2 Time Complexity

In the Fig. 5.8 we can see the longer solving time for the Franka Emika Panda robot on shelf task 1 - 3 and tunnel task. Since the solving time is longer for the new method but the number of iterations is lower compared to [Zorina et al., 2023] we must ponder how it came to be. It seems like part of the newly implemented method takes up significant amount of computational time in each iteration. Perhaps since this method was programmed in python programming language, the problem could lie there. It might be enough to rewrite the new method code in a programming language, such as c++, that is faster and where the code could be further optimized.

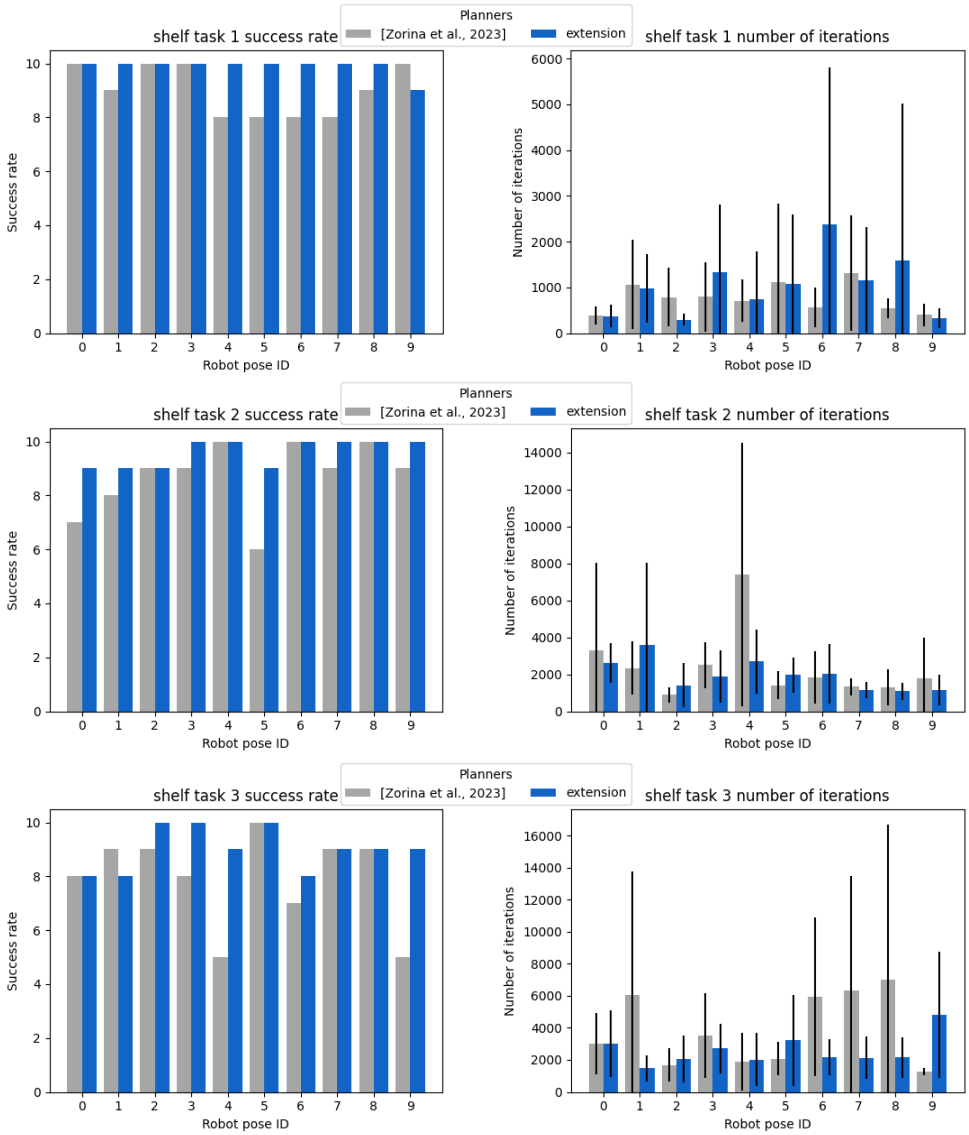

Figure 5.7: Results of benchmarking for Kuka IIWA robot on shelf task 1-3 with the following noise parameters $\Delta_l = 0.2$ m and $\Delta_\alpha = 1°$. In grey are the results of previous method and in blue the results of the new one. On the left, number of successful solutions and on the right number of iterations.

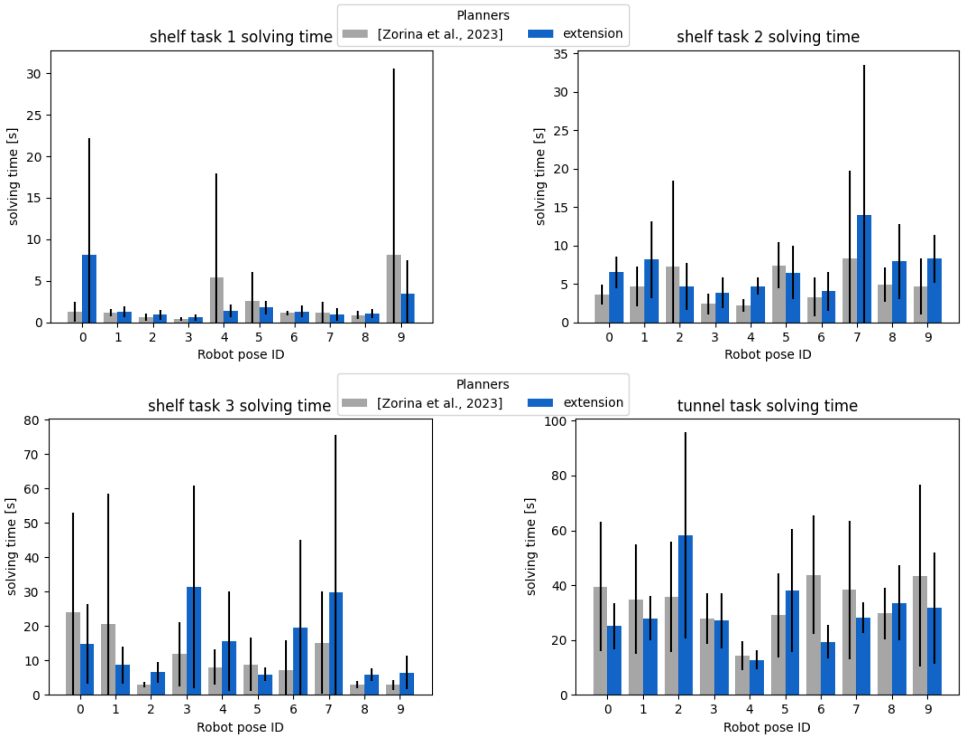

Figure 5.8: Results of benchmarking of solving time for shelf task 1 - 3 and tunnel task. In blue are the results of our extension and in grey we have the previous approach.

5.2 Overall Results of Benchmarking

After acquiring the result for all the robots, we have set out to fully benchmark our results with [Zorina et al., 2023] and [Lamiraux and Mirabel, 2022]. We have benchmarked all of the aforementioned planners on shelf task 1 - 3 and tunnel task. We have benchmarked them on 10 different robot poses and each robot pose has been benchmarked on 10 different random seeds. We will be measuring (i) success rate, (ii) solving time, (iii) number of iterations and (iv) path length. However, we did not manage to get iterations from HPP, thus we left them empty. The results can be seen in Figs. 5.9, 5.10 and 5.11.

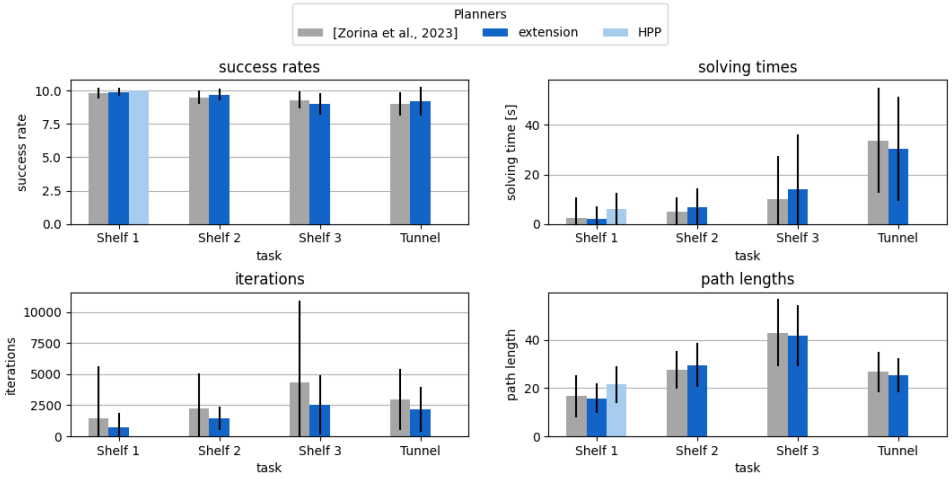

Figure 5.9: Results of overall benchmarking for Franka Emika Panda Robot with [Zorina et al., 2023] in grey and [Lamiraux and Mirabel, 2022] in light blue. In upper left corner we have success rate, in upper right corner solving time, lower right corner number of iterations and lower left corner the path length.

In the figures containing the results one thing is apparent: HPP was not able to solve shelf task 2 - 3 and tunnel task for any of the robots. The new approach has shorter solving time and path length compared to HPP for Panda and UR5 robots. However for Kuka IIWA robot, the opposite is true.

In Fig. 5.9 which contains results for Franka Emika Panda Robot, we can see an overall improvement in the number of iterations of the solver from the previous method. There was no overall improvement in the success rate and in path length. However, it seems that for shelf task 1 - 2 the planing time has risen.

In Fig. 5.10 that contains results for UR5 Robot, we can see an improvement from the previous approach in the number of iterations of the solver for shelf tasks 1 - 3, but no longer for tunnel task. The success rate has slightly degraded as well as path length. It seems that for shelf task 1 - 2 the planing time has risen.

In Fig. 5.11 which contains results for Kuka IIWA Robot, we can see an improvement from [Zorina et al., 2023] in number of iterations of the solver

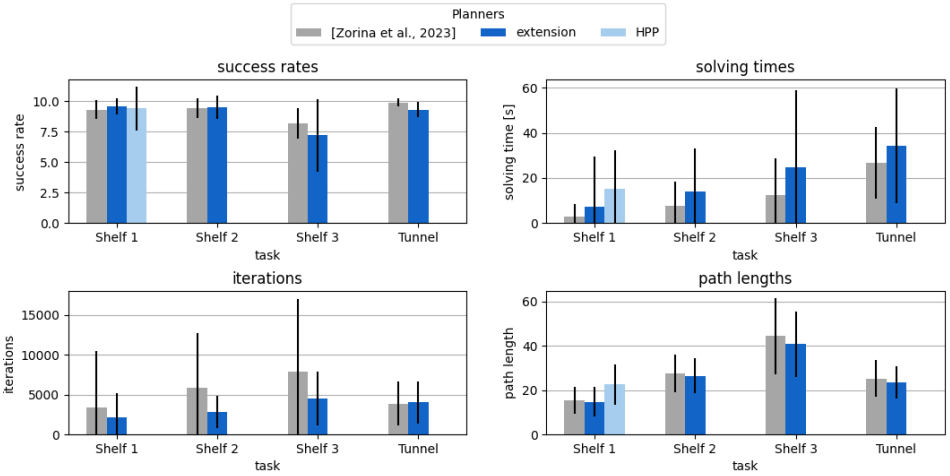

Figure 5.10: Results of overall benchmarking for UR5 Robot with [Zorina et al., 2023] in grey and [Lamiraux and Mirabel, 2022] in light blue. In upper left corner we have success rate, in upper right corner solving time, lower right corner number of iterations and lower left corner the path length.

for shelf task 2 - 3 but not for tunnel task and shelf task 1. However, the success rate was improved for shelf tasks 1 - 3. The planning time has risen for shelf task 1 - 3 but went down for tunnel task. Path length has went up for shelf task 1 - 3 but went down for tunnel task.

To conclude, the proposed method is able to solve the task in less amount of iterations, however, the time complexity is increased due to the additional complexity of solving inverse kinematics. Nevertheless, we believe that the lower number of iterations will lead to time improvement if implemented in C++ programming language.

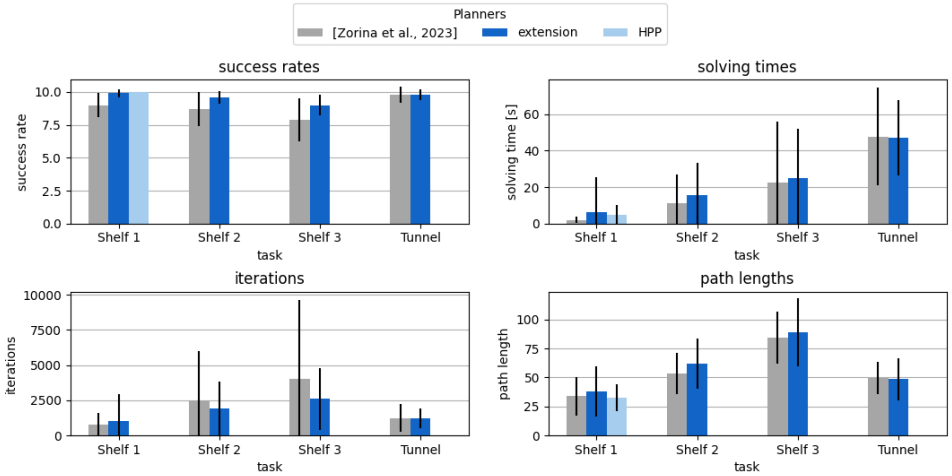

Figure 5.11: Results of overall benchmarking for Kuka IIWA robot with [Zorina et al., 2023] in grey and [Lamiraux and Mirabel, 2022] in light blue. In upper left corner we have success rate, in upper right corner solving time, lower right corner number of iterations and lower left corner the path length.

Chapter 6

Conclusion

The focus of this thesis was to present a new method of using video demonstration to guide path planners. We have extended upon previous work presented in [Zorina et al., 2023] and had further improved the video demonstration guided algorithm by utilizing more information from the video.

We had prepared environments in which we benchmark the proposed extension on real life task and motion planning problems that proved to be challenging even for state-of-the-art planners. We have than compared the extension with [Zorina et al., 2023] and [Lamiraux and Mirabel, 2022] in the proposed benchmark.

We have found the best result yielding parameters for the planner and have shown that this new approach is an improvement. The results show that the new method can significantly reduce the number of iterations needed to solve the given problems on different robots or even yield higher success rate. However, we have also found that the method is slower than [Zorina et al., 2023]. This might have been caused due to implementation through python programming language and should be addressed in future development.

Additional plan for our future work includes: (i) an analysis of the code to potentially find which section is slowing down the method and optimizing it for further development; (ii) additional benchmarking with other state-of-the-art planners such as [Garrett et al., 2018]; and (iii) benchmarking of waiter task with mobile robot.

Bibliography

[Aeronautiques et al., 1998] Aeronautiques, C., Howe, A., Knoblock, C., McDermott, I. D., Ram, A., Veloso, M., Weld, D., SRI, D. W., Barrett, A., Christianson, D., et al. (1998). Pddl| the planning domain definition language. *Technical Report, Tech. Rep.*

[Argall et al., 2009] Argall, B. D., Chernova, S., Veloso, M., and Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483.

[Dijkstra, 1959] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271.

[Evrard et al., 2009] Evrard, P., Gribovskaya, E., Calinon, S., Billard, A., and Kheddar, A. (2009). Teaching physical collaborative tasks: object-lifting case study with a humanoid. In *2009 9th IEEE-RAS International Conference on Humanoid Robots*, pages 399–404.

[Fikes and Nilsson, 1971] 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):189–208.

[Garrett et al., 2017] Garrett, C. R., Lozano-Pérez, T., and Kaelbling, L. P. (2017). STRIPS planning in infinite domains. *CoRR*, abs/1701.00287.

[Garrett et al., 2018] Garrett, C. R., Lozano-Pérez, T., and Kaelbling, L. P. (2018). Stripstream: Integrating symbolic planners and blackbox samplers. *CoRR*, abs/1802.08705.

[Hart et al., 1968a] Hart, P., Nilsson, N., and Raphael, B. (1968a). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107.

[Hart et al., 1968b] Hart, P., Nilsson, N., and Raphael, B. (1968b). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107.

[Jenkins et al., 2000] Jenkins, O. C., Mataric, M. J., Weber, S., et al. (2000). Primitive-based movement classification for humanoid imitation. In *Proceedings, First IEEE-RAS International Conference on Humanoid Robotics (Humanoids-2000)*, pages 1–18.

[Karaman and Frazzoli, 2010] Karaman, S. and Frazzoli, E. (2010). Incremental sampling-based algorithms for optimal motion planning. *CoRR*, abs/1005.0416.

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

[Kuffner and LaValle, 2000] Kuffner, J. and LaValle, S. (2000). Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 2, pages 995–1001 vol.2.

[Kuniyoshi et al., 1994] Kuniyoshi, Y., Inaba, M., and Inoue, H. (1994). Learning by watching: extracting reusable task knowledge from visual observation of human performance. *IEEE Transactions on Robotics and Automation*, 10(6):799–822.

[Labbé et al., 2020] Labbé, Y., Carpentier, J., Aubry, M., and Sivic, J. (2020). Cosypose: Consistent multi-view multi-object 6d pose estimation. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J.-M., editors, *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 574–591, Cham. Springer International Publishing.

[Lamiraux and Mirabel, 2022] Lamiraux, F. and Mirabel, J. (2022). Prehensile manipulation planning: Modeling, algorithms and implementation. *IEEE Transactions on Robotics*, 38(4):2370–2388.

[LaValle et al., 1998] LaValle, S. M. et al. (1998). Rapidly-exploring random trees: A new tool for path planning.

[Mansouri et al., 2021] Mansouri, M., Pecora, F., and Schüller, P. (2021). Combining task and motion planning: Challenges and guidelines. *Frontiers in Robotics and AI*, page 133.

[Nocedal and Wright, 2006] Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer, New York, NY, USA, 2e edition.

[Ollis et al., 2007] Ollis, M., Huang, W. H., and Happold, M. (2007). A bayesian approach to imitation learning for robot navigation. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 709–714.

[Shan et al., 2020] Shan, D., Geng, J., Shu, M., and Fouhey, D. F. (2020). Understanding human hands in contact at internet scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Sutton and Barto, 2018] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

[Wen et al., 2022] Wen, B., Lian, W., Bekris, K., and Schaal, S. (2022). You only demonstrate once: Category-level manipulation from single visual demonstration. *Robotics: Science and Systems (RSS) 2022*.

[Wu et al., 2020] Wu, B., Xu, F., He, Z., Gupta, A., and Allen, P. K. (2020). Squirl: Robust and efficient learning from video demonstration of long-horizon robotic manipulation tasks. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9720–9727.

[Ye and Alterovitz, 2017] Ye, G. and Alterovitz, R. (2017). Guided motion planning. In *Robotics research*, pages 291–307. Springer.

[Zorina et al., 2023] Zorina, K., Kovar, D., Lamiraux, F., Mansard, N., Carpentier, J., Sivic, J., and Petrik, V. (2023). Multi-contact task and motion planning guided by video demonstration. *[In review] IEEE International Conference on Robotics and Automation*.