

# Integrated Task and Motion Planning of Multi-Robot Manipulators in Industrial and Service Automation

by

Ilknur Umay

A thesis  
presented to the University of Waterloo  
in fulfillment of the  
thesis requirement for the degree of  
Doctor of Philosophy  
in  
Mechanical and Mechatronics Engineering

Waterloo, Ontario, Canada, 2020

© Ilknur Umay 2020

## Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner: Kamal Kant Gupta  
Professor, School of Engineering Science,  
Simon Fraser University

Supervisor(s): William Melek  
Professor, Dept. of Mechanical and Mechatronics Engineering,  
University of Waterloo  
Baris Fidan  
Professor, Dept. of Mechanical and Mechatronics  
Engineering, University of Waterloo

Internal Members: Soo Jeon  
Associate Professor, Dept. of Mechanical and Mechatronics  
Engineering, University of Waterloo  
James Tung  
Associate Professor, Dept. of Mechanical and Mechatronics  
Engineering, University of Waterloo

Internal-External Member: Stephen L. Smith  
Associate Professor, Dept. of Electrical and Computer Engineering,  
University of Waterloo

### **Author's Declaration**

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

## Abstract

Efficient coordination of several robot arms in order to carry out some given independent/cooperative tasks in a common workspace, avoiding collisions, is an appealing research problem that has been studied in different robotic fields, with industrial and service applications. Coordination of several robot arms in a shared environment is challenging because complexity of collision free path planning increases with the number of robots sharing the same workspace. Although, research in different aspects of this problem such as task planning, motion planning and robot control has made great progress, the integration of these components is not well studied in the literature.

This thesis focuses on integrating task and motion planning multi-robot-arm systems by introducing a practical and optimal interface layer for such systems. For a given set of specifications and a sequence of tasks for a multi-arm system, the studied system design aims to automatically construct the necessary waypoints, the sequence of arms to be operated, and the algorithms required for the robots to reliably execute manipulation tasks.

The contributions of the thesis are three-fold. First, an algorithm is introduced to integrate task and motion planning layers in order to achieve optimal and collision free task execution. Representation via shared space graph (SSG) is introduced to check whether two arms share certain parts of the workspace and to quantify cooperation of such arm pairs, which is essential in selection of arm sequence and scheduling of each arm in the sequence to perform a task or a sub-task. The introduced algorithm allows robots to autonomously reason about a structured environment, performs the sequence planning of robots to operate, and provides robots and objects path for each task to succeed a set of goals.

Secondly, an integrated motion and task planning methodology is introduced for systems of multiple mobile and fixed base robot arms performing different tasks simultaneously in a shared workspace. We introduce concept of dynamic shared space graph (D-SSG) to continuously check whether two arms sharing certain parts of the workspace at different time steps and quantify cooperation of such arm pairs, which is essential to the selection of arm sequences and scheduling of each arm in the sequence to perform a task or a sub-task. The introduced algorithm allows robots to autonomously reason about complex human involving environments to plan the high level decisions (sequence planning) of robots to operate, and calculates robots and objects path for each task to succeed a set of goals.

The third contribution is design of an integration algorithm between low-level motion planning and high-level symbolic task planning layers to produce alternate plans in case of kinematic and geometric changes in the environment to prevent failure in the high-level

task plan.

In order to verify the methodological contributions of the thesis with a solid implementation basis, some implementations and tests are presented in the open-source robotics planning environments ROS, Moveit and Gazebo. Detailed analysis of these implementations and test results are provided as well.

## **Acknowledgements**

I want to thank my supervisors Professor William Melek and Professor Baris Fidan, for their guidance, motivations, and support during my research. I would like to thank my colleagues Dylan Gareau, Andrew McCormick, and Zehra Camlica, for their patients in some useful discussions. Also, a special thanks to Brandon J. DeHart and Alexander Werner for their feedbacks' and support on the software-hardware related problems.

A special thanks to my family. It was not possible to complete the thesis without their support.

Finally, I would like to thank all the little people who made this thesis possible.

## **Dedication**

This is dedicated to my family.

# Table of Contents

<b>List of Figures</b>	<b>x</b>
<b>List of Acronyms</b>	<b>xv</b>
<b>Nomenclature</b>	<b>xvi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Scopes and Objectives . . . . .	5
1.2 Contributions . . . . .	6
1.3 Outline of the Thesis . . . . .	8
<b>2 Background, Problem Definition and Literature Review</b>	<b>9</b>
2.1 Problem Definition, Notation, and Foundation . . . . .	9
2.2 Motion Planning . . . . .	14
2.2.1 Deterministic methods . . . . .	15
2.2.2 Sampling-based methods . . . . .	16
2.2.3 Optimization-based methods . . . . .	17
2.2.4 Constrained motion planning . . . . .	17
2.3 Task planning . . . . .	19
2.3.1 The Planning Domain Definition Language . . . . .	21
2.4 Manipulation planning . . . . .	21
2.4.1 Multi-layer manipulation planners . . . . .	22
2.4.2 Single layer planners . . . . .	23



<b>3</b>	<b>Shared Space Graph Based Manipulation Planning</b>	<b>28</b>
3.1	Introduction . . . . .	28
3.2	Algorithm Development . . . . .	29
	3.2.1 Stage 1: Multi Arm Sequence Planning . . . . .	30
	3.2.2 Stage 2: Multi Arm Motion Planning . . . . .	32
3.3	Simulation Results and Testing . . . . .	33
3.4	Summary . . . . .	40
<b>4</b>	<b>Vision-guided Dynamic-Shared Space Graph Based Manipulation Planning</b>	<b>41</b>
4.1	Introduction . . . . .	41
4.2	Algorithm Development . . . . .	43
	4.2.1 Stage 1: Multi Arm Sequence Planning . . . . .	44
	4.2.2 Stage 2: Multi Arm Motion Planning . . . . .	47
4.3	Simulation Results and Testing . . . . .	49
4.4	Summary . . . . .	53
<b>5</b>	<b>Integrated Task and Motion planning with Dynamic Shared Space Graph</b>	<b>54</b>
5.1	Introduction . . . . .	54
5.2	Combining Task and Motion Planning Through Our Interface Layer . . . . .	56
5.3	Implementation Framework . . . . .	57
5.4	Algorithm Analysis . . . . .	59
5.5	Simulation Results and Testing . . . . .	60
5.6	Summary . . . . .	70
<b>6</b>	<b>Proposed System Design for Automated Kitchen Environment</b>	<b>71</b>
6.1	Background and Literature Review . . . . .	71
6.2	Automated MaM Kitchen Environment . . . . .	74
6.3	Task and Motion Planning . . . . .	74
	6.3.1 Action Space for the Kitchen . . . . .	75
	6.3.2 Assembly . . . . .	76
	6.3.3 Plan Execution . . . . .	77
6.4	Summary . . . . .	79

<b>7</b>	<b>Conclusions and Future Work</b>	<b>80</b>
	<b>References</b>	<b>82</b>
<b>A</b>		<b>99</b>
A.1	Robot Specification Parameters . . . . .	99
A.2	ROS . . . . .	102
A.2.1	Meet Points Calculation . . . . .	104

# List of Figures

1.1	Examples of MaM setups. . . . .	2
1.2	An example with $n = 4$ arms, where each arm has a fixed base. In this setup, no arm is able to bring the object from its initial location to the target region. . . . .	3
1.3	A general layout for task-motion and integrated task-motion planning problems in the literature . . . . .	4
1.4	A roadmap of the thesis. . . . .	8
2.1	Transfer, transit and manipulation path representations . . . . .	11
2.2	Multi Robot Planning Approaches: An initial and a goal configuration are given as input for each robot. . . . .	18
3.1	Transfer, transit and manipulation path representations . . . . .	29
3.2	A summary of proposed integrated task and motion planning . . . . .	30
3.3	The object and right gripper's trajectory while object is moved from $MP$ to $o_{goal}$ where gazebo/link_states/pose[24], gazebo/link_states/pose[28] and joint_states/position[12] are movo::right_gripper_finger1's pose, pringles_small::link's pose, and right_gripper_finger1_joint's pose respectively. . . . .	34
3.4	a) SSG representation based on shared workspace. b) The shortest path is found using Dijkstra. . . . .	34
3.5	Scheme 1: a) An example of $MP$ representation for three robots, where $R1$ , $R2$ and $R3$ indicate workspace of the each robot. b) The shared and individual working regions. Scheme 2: Parallel working principle based on shared and independent working areas . . . . .	35
3.6	a)A 3-DOF single arm in 2D. Path planning time is around 70.19 seconds via $A^*$ b) 2 arms, 3-DOF each, case. Planning time is more than 30 minutes . . . . .	35
3.7	A setup consist of two arms with 3 DoF each. The first robot brings the $o$ from its initial to a $MP$ , and the second robot takes the $o$ to $o_{goal}$ . . . . .	36

3.8	Proposed scenario implementation using the Kinova Movo: The first Jaco arm brings the $o$ from its initial to a meet point $MP$ . Left side of the each figure are representing the Gazebo model and right side are showing RViz model created by mapping from gazebo model. Note that initial and goal location of the object detected by a kinect sensor on the movo's head. . . .	37
3.9	Proposed scenario implementation using the Kinova Movo: The second Jaco arm brings the $o$ from its meet point $MP$ to goal. Left side of the each figure are representing the Gazebo model and right side are showing RViz model created by mapping from gazebo model. Note that initial location of the object detected by a kinect sensor on the movo's head. . . . .	39
4.1	An example with $n = 4$ arms, where each arm has a fixed base. In this setup, no single arm is able to bring the object from its initial location to the target region . . . . .	42
4.2	An example of mobile and fixed base robots in a shared environment. In this setup, we have a kinova-movo, mobile based robots, and two Fanuc LRMate 200ic, fixed-base arms. . . . .	43
4.3	The implementation process of the algorithm . . . . .	44
4.4	Flow diagram of overall process. . . . .	45
4.5	Roadmap of the motion planning layer. . . . .	48
4.6	Proposed scenario implementation using the Kinova Movo: The second Jaco arm brings the $o$ from its initial to goal position under clustered environment.	50
4.7	Color and shape features from the objects that were sitting on the table are extracted using object segmentation on 3D point cloud data from gazebo world. . . . .	52
4.8	Proposed scenario implementation using the 2 Fanuc arms and a Kinova Movo	52
5.1	Flow diagram of overall process. . . . .	58
5.2	Gazebo world of Uwaterloo Robohub is considered as grocery setup. . . . .	60
5.3	Objects of the world for the grocery store example(ii). . . . .	61
5.5	Proposed scenario implementation using the Kinova Movo and 2 Fanuc LR-Mate 200ic robot: The left Fanuc arm brings the object from its initial to goal position. . . . .	61
5.6	Output of movo-lrmate grocery setup. . . . .	62
5.12	Proposed scenario experimentation using the Kinova Movo in Robohub environment . . . . .	63

5.13	Planning and execution times for problems solved within different types of environmental changes and different number of collision objects in the cluttered table domains. For each case, 10 different pick and place simulation studies are generated. . . . .	64
5.14	Some of the pick-place tasks executed while solving an instance of the different number of objects cluttered table with the environmental changes. .	65
5.4	Case study(ii): The setup of the proposed grocery store example with Kinova-Movo and Fanuc-LRmate200ic arms. . . . .	66
5.7	Problem and Domain generation for grocery store example(ii). . . . .	67
5.8	Initial reachability graph for case study (i). . . . .	68
5.9	Output of case study(i) before environmental changes. . . . .	68
5.10	Output of case study(i) without integration . . . . .	69
5.11	Proposed scenario implementation using the Kinova Movo: The second Jaco arm brings the $o$ from its initial to goal position under clustered environment.	69
6.1	Detailed concept of the robotic platform for restaurants. . . . .	74
6.2	An example of collaborative pasta mixing task. . . . .	75
6.3	Two examples of durative-action definition and their effect for automated MaM kitchen environment . . . . .	76
6.4	The objects of the proposed robotic kitchen environment . . . . .	77
6.5	Output of task planning layer for two pasta orders . . . . .	78
A.1	DH-parameters and robot's link-length values for $JACO^2$ Spherical 7 – $DOF$	99
A.2	MovoBeta specifications . . . . .	100
A.3	Movo specifications . . . . .	101
A.4	Multi Robot Planning Approaches: An initial and a goal configuration are given as input for each robot. . . . .	104
A.5	Intersection of two robots' workspace for 3D case . . . . .	105
A.6	Meet point representation where, $r_1$ , and $r_2$ , $x$ , $d = d_1 + d_2$ , represent the first and second robot's workspaces, meet point, and distance between the two robots' bases, respectively. If $d < r_1 + r_2$ , then there is a shared workspace between two robots. . . . .	105

A.7	Representation of down sampled version of our reachability database for Jaco arm. (a) Reachable poses represented by colored arrows, the color encodes the manipulability of the corresponding manipulator configuration. (b) The cross sectional view of the voxelized 3d workspace of a jaco arm, the color indicates the number of grasping poses contained in the 3d voxel.	106
A.8	A three structured of the dual-fanuc-lrmate200ic arms with gripper attached	107
A.9	Three fanuc arm environment . . . . .	108
A.10	Each action (primitive task) is introduced as a subtask to motion planning layer . . . . .	108
A.11	Fanuc-LRMate200ic arm specifications . . . . .	109
A.12	Panda arm specifications . . . . .	110
A.13	Panda workspace views . . . . .	111

# List of Acronyms

DaM	Dual-Arm or Multi-Arm Manipulator
DOF	Degrees Of Freedom
NAMO	Navigation Among Movable Obstacles
O	Set of parametrized propositional actions defined by preconditions and effects
PDDL	Planning Domain Definition Language
P	Planning Task
FF	Fast Forward
$CS$	Configuration Space
$MP$	Meet Point
$s_i$	Each robot arm in a set of robot arms $S$
$A_i$	Each robot arm in arm network $A$
$T_a$	A sequence of tasks
$T_m$	A sequence of subtask
$\mathcal{M}$	A set of paths
$o_i$	A single rigid-body object in a set of objects $o$
$C_{A_i}$	Configuration space of arm $A_i$
$SSG$	Shared Space Graph
$W_i$	Workspace of $i$ 'th arm
FPC	Fixed Path Coordination
FRC	Fixed Roadmap Coordination
HTN	Hierarchical Task Network
MP	Meet Point
PRM	Probabilistic Road Map
PrP	Prioritized Planning
ROS	Robotic Operating System
RRT	Rapidly exploring random tree
SBL	Single query Bidirectional Lazy in collision checking
SSG	Shared Space Graph

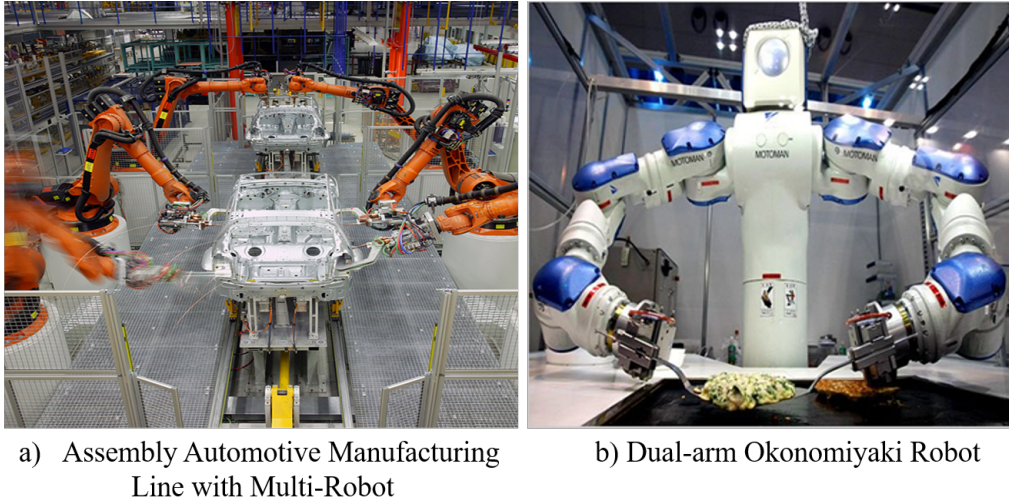
*“Simplicity is the final achievement.” - Frederic Chopin*



# Chapter 1

## Introduction

Today many robotic applications involve multi-arm manipulator (MaM) platforms [1–6] (Fig. 1.1). Most MaM platforms share a common workspace to increase efficiency in manipulation relative to single-arm counterparts. They can easily work in environments too harsh for humans, such as in hospitals, where they can obtain virology samples from highly contagious patients, or in steel production and smelting plants, where they can operate high temperature, iron melting furnaces. Importantly, these robots can work either independently or collaboratively, alongside other robots or human partners, in known or unstructured environments. For instance, cooperative manipulators already perform cooperative tasks in construction and assembly lines, often helping to handle heavy objects or to transfer individual objects to desirable locations. However, simultaneous movement of MaMs alongside their human and robotic counterparts in shared workspaces is challenging; special planning is needed to avoid collisions with humans, objects, or the arms themselves, and to ultimately ensure workplace safety. For many of these tasks, a key requirement is the ability to plan arm motions that relocate an object from one location to another, within the workspace defined by the  $N$ -arm robotic system ( $N \geq 2$ , see Figure 1.2). When multiple objects must move concurrently, a task scheduling challenge also arises [7–9]. In industrial applications, efficient collaboration between multiple arms can dramatically increase productivity and overall cost-effectiveness. For instance, MaMs can share common tools, lift heavier objects together, and combine multiple tasks (e.g., one arm picks up assembly parts from remote workspace locations and passes them to another for assembly). The planning problem for MaM platforms is challenging for several reasons. First, it requires careful and precise decision making to coordinate which arm manipulates a given object and in which order. In the simplest scenario, a single arm may be able to relocate the object. In other scenarios, all  $N$  arms ( $N \geq 2$ ) may potentially be involved in relocating

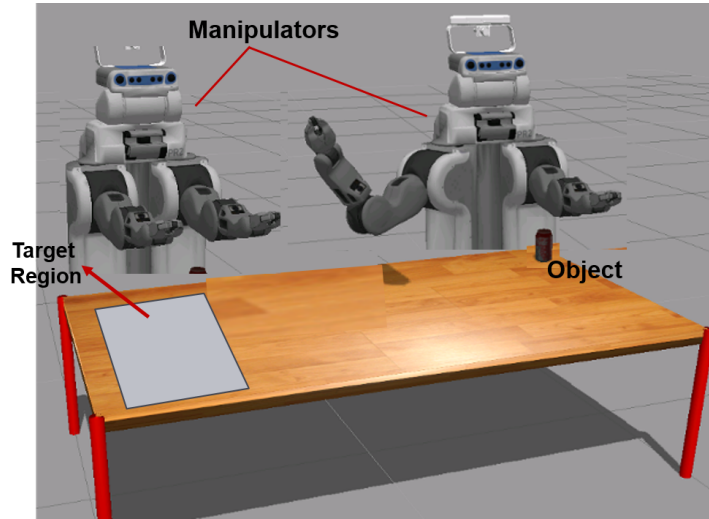


**Figure 1.1:** *Examples of MaM setups.*

an object from one end of the workspace to the other. In more complex cases, the same arm may have to re-grasp the object several times with the help of another arm in order to manipulate the object in a way that allows its placement at a target location, in the desired orientation. Second, the planning problem also involves computing valid locations for each of the handoffs between two consecutive arms. Figuring out these locations is non-trivial because the environment can be cluttered and the object itself can be large. Finally, the planner must also consider all possible grasping motions and plan each grasp in a way that allows for a successful sequence of handoffs, where the object is passed from one arm to another. Handoff planning is complex because the arms must orient and grasp objects that may be in arbitrary and unstable poses.

Despite growing use of robots in the manufacturing and service sectors, robotic platforms have seen limited deployment in the food and health service industries. Robotic systems are becoming very attractive to the restaurant sector, where they can help combat labour shortages and high labour turnover. There is also potential for dual-arm humanoid manipulators to support cooking and serving related tasks as they can be relatively easily implemented in facilities designed for human workers and chefs who use similar bi-manual skills [10–15].

However, to maintain high-level tasks like cooking or serving, as well as human assistive robotics, robot manipulators require efficient high-level symbolic task planning with low-level motion planning. In this integration, task planning algorithms decide the procedures or operations that each robot must execute at each step, while motion planning algorithms

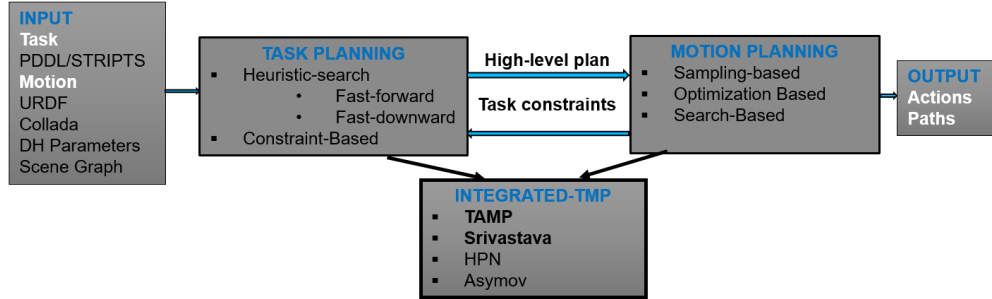


**Figure 1.2:** *An example with  $n = 4$  arms, where each arm has a fixed base. In this setup, no arm is able to bring the object from its initial location to the target region.*

coordinates the required motions. Integrating these planning algorithms is challenging since task planning specifications usually ignore the solid geometric constraints on the robotic motion planning problems. For instance, before deciding to pick up a cup, one should determine if it is geometrically feasible to move the robot base to a suitable location. In conventional approaches, such issues have been considered from the top down, isolating high-level task planning (e.g., sequencing pick-and-place actions) from lower-level motion planning (e.g., finding feasible paths for the arm). Task planning is then performed by ignoring low-level details, yet the subsequent plans may be ineffective or even infeasible because of missed lower level collaborations and conflicts.

In a typical task and motion planning scenario, the task planning layer calculates a high-level plan and the motion planning layer finds a corresponding path for each action in the high-level task plan. However, applying task and motion planning sequentially is very complicated in real scenarios since the task planner does not consider geometric locations of the robots, objects or grippers, and geometric preconditions and effects around them. Therefore, there is a need to integrate these two layers, rather than applying one after another, primarily for semi-structured or uncertain environments. As seen in Figure 1.3 (readers can refer to Chapter 2 for further details), some approaches exist to deal with these issues and integrate those layers. For instance, the study [16] tries to estimate what can go wrong in a pick and place task in advance and try to define a complete task definition in the higher-level task planning layer. However, it is hard to estimate what can go wrong in

real scenarios in advance and update the task planning layer accordingly. Also, when there exist multiple tasks, there will be many geometrical facts around them to be re-arranged.



**Figure 1.3:** *A general layout for task-motion and integrated task-motion planning problems in the literature*

Another approach is Task and Motion planning (TAMP), which searches for a discrete sequence of symbolic actions around a motion plan for each of them [17]. However, the discrete form of manipulation tasks, such as pick and place, is over defined in the symbolic domain, decreasing the complexity of the problem but increasing the computation time. The main disadvantage of TAMP is to avoid calling motion planner for unfeasible actions, especially for constraint manipulation planning problems. Since there is not enough literature on the integrated task and motion planning and available ones cannot provide a solution for robot manipulation problems for semi-structure or unstructured environments regarding generality, completeness, and performance, this thesis will focus on solving this problem.

Recent research has produced critical advances in the state-of-the-art of mobile manipulation, with new platforms such as Kinova-Movo, PR2, Intel HERB Personal Robot, ICub, the TUM Rosie robot, the HRP2, ARMAR, and Justin all demonstrating improved capability for combined manipulation tasks. More up-to-date platforms (e.g., the PR2) have coordinated 3D sensors to achieve consistent representation of their evolving surroundings. Meanwhile, 2D visual data has been used for object recognition and tracking, and as feedback for visual servoing controllers.

Manipulation planning focuses on automatic production of robot motion sequences for manipulation of movable objects among obstacles to accomplish a desired goal. Manipulation planning problems for MaM platforms involve objects which can only move when picked up by robots. In these cases, the order of pick-and-place actions is critically important to finding a feasible kinematic solution for object manipulation [18]. Therefore, geometric reasoning and motion planning alone are not sufficient to obtain a solution; and

planning of operations, for example the pick-and-place actions, should be integrated with the motion planning problem.

## 1.1 Scopes and Objectives

The scope of this research addresses challenges at the intersection of two fields. On one hand, motion planning methods focus on the problem of trajectory planning with numerous kinematic and dynamic constraints [15], [19]. [17]. Currently, these techniques can manage numerous constraints for a single task yet normally do not rapidly factor into comprehensive sub-tasks. On the other hand, the symbolic task planning research community has long realized that reasoning about robotic actions must consider geometric constraints at the planning level, which spurred development of the task and motion planning field [16], [8,20], [1]. While these methodologies have generated noteworthy showcases, and have demonstrated some capability to understand complex, puzzle-like situations, they are still significantly short of implementation for dynamic, real-world use cases. Since typical task planning approach for robotics applications aims to perform both symbolic- and geometric-level planning simultaneously, firmly relying upon an exact and reliable domain representation, these frameworks are vulnerable to small variations and uncertainties in parameterization.

Furthermore, interrelations between different sub-tasks should be displayed unequivocally in the symbolic domain or numerous potential sub-solutions will risk being dismissed later. In the latter case, the underlying backtracking-based search becomes ineffective and it is hard to exploit the local structure of a particular task. In this thesis, rather than tackling with such non-exclusive task problems, we focus on the common sub-problem of finding feasible sequences of trajectories given that the high-level action sequence is already known in advance. The assumed action sequence can be compared to the plan skeletons characterized in [1]. While the study in [1] presents strategies to change action sequences into discrete-space constraint satisfaction problems, this thesis utilize conventional motion planning algorithms to provide solutions in continuous space if the multi-layer planner fails.

This thesis addresses the robotics object manipulation problem in a multi-robot environment, focusing on the following three key objectives:

1. To effectively formulate the cooperative multi-robot manipulation problem. While formulation of motion and manipulation planning for individual robot is not challenging in general, the overall manipulation planning problem for MaM systems includes many difficulties as presented in the previous section.

2. To develop manipulation planning approaches for MaM systems. Manipulation planning for MaM is challenging since it requires the search and planning of the sequence of tasks to be performed by MaM systems with planning of the motions needed to succeed for a set of goals. Indeed, even in the most basic pick and place tasks, each sub-task can be accomplished in many possible ways, each having an impact on follow-on tasks.
3. To adapt the developed manipulation planning approaches to a variety of MaM system settings, involving robots with mobile bases as well as stationary ones, and uncertainties and variations in the environment. To be relevant in a wide variety of industrial scenarios, several types of robots must be considered.

## 1.2 Contributions

Modern robotics planning must consider dynamic and uncertain domains, in which the results of actions can give a reasonable chance of failure or non-deterministic effects. The need for integration of task and motion planning to accomplish complex manipulation tasks and to navigate in unpredictable environments (e.g., crowded spaces) is well understood in the robotics research community. While there exists a vast literature and a plethora of recent technological advances in both areas (individually), integrated motion and task planning studies are only just emerging, particularly for multi-robot manipulation. Indeed, multi-robot manipulation has been thoroughly studied for structured environments and repetitive tasks, but slight changes in the environment or task cause both task and motion planning layers to fail.

In most cases in the literature, failure action due to the geometrical change in the environment is taken care of in the high-level task planning layer by manually editing task definitions or constraints. One of the main focus of this thesis is to solve the failure of subtask/action in an integration layer automatically, instead of editing the higher-level task planning layer.

To deal with the complexities of uncertain environments, this thesis proposes a new approach; one that adds an interface layer between the task and motion planning layers, enabling nimble responses to changes through generation of alternate plans, while ensuring the higher-level goal is achieved. The interface layer includes a shared-space graph, in which each node stores some information regarding the robot base location, and states of the object(s) to be manipulated and the obstacles to be avoided in the environment. Note that uncertainty in this thesis means that the robots' base location and/or objects/obstacles location are subject to change in time, and/or a new object may appear in the environment.

It is always worth mentioning that we are not focusing on Navigation Among Movable Obstacles (NAMO) or grasping from cluttered environments.

This thesis focuses on an integrated motion and task planning approach for MaM system applications, including those for kitchen robotics and grocery setup environments. In addition to fundamental advances to robotics planning, the thesis results are applied in an innovative robot cooking system design involving multiple robotic arms, capable of implementing cooking recipes via picking and placing ingredients collaboratively and/or independently in an environment with intermittent variations.

The main contribution of this thesis is development of a general integrated motion and task planning algorithm for MaM systems with arbitrary number of robot-arms. The proposed interface layer permits robust and adaptive planning as well as re-planning when facing unexpected changes. Relevant geometric information about the environment is transferred to the task planner as logical predicates. The task planning layer then analyses if the preconditions necessary to execute the actions are satisfied, and extracts a plan, while the motion planning layer executes the plan. The contributions of this thesis are classified into three-fold:

1. A high level task planning layer is designed to identify the sequence of arms that would optimally transfer an object from its initial location to a specified goal location (each path for each arm in the arm sequence optimal-sub-task optimality). In the proposed task planning layer design, each primitive task is divided into subtasks before it is introduced as an input to the motion planning layer. The main novelty of this contribution is the introduction of a shared space graph (SSG) based representation, where vertices of the SSG represents the workspaces of the robot arms and the edges indicate workspaces shared between two arms, allowing the planner to check whether two arms share a part of the workspace. The SSG representation is further extended to apply for the cases involving, introducing a dynamic (time-varying) version, dynamic-SSG (D-SSG) in order to accommodate changes to base locations of robots in a sequence and the paths for each robot in the sequence. The SSG and D-SSG representations are helpful in effective handoff scheduling between robot arms and determining where to place or transfer the object.

2. A low-level motion planning layer is designed to calculate a collision-free robot-endpoint trajectory for each step of the sequence. This would include, for example, each handoff in a sequence of handoffs to move an object to its goal position. Heuristic, e.g. the  $A^*$  algorithm, are utilized to find the optimal path (probabilistic optimality) for each robot arm in the task sequence, allowing that robot arm to avoid obstacles and reach its goal location.

3. The proposed task and motion planning layers designed for MaM systems are integrated and adapted for applications with uncertain environments, which is presented in

Chapter 5. This contribution enables complex industrial and human assistive applications, as explained through a case study, in Chapter 6.

### 1.3 Outline of the Thesis

In this thesis, we address an integrated motion and task planning algorithm for multi-arm robot systems in a shared workspace, a roadmap of the overall thesis can be seen in Figure 1.2. In Chapter 2, we provide formal statements of the problems studied in the thesis and review of the relevant literature, including state-of-the-art motion and task planning methods for robotic manipulators in general. Chapter 3 introduces the proposed SSG representation and the base integrated task and motion planning based on this representation for MaM system manipulation problems in well defined and structured environments. Chapter 4 presents the modified notion of D-SSG application of this representation to manipulation (integrated task and motion) planning for cases involving some robot arms with mobile bases in a uncertain, semi-structured, environment. Chapter 5 provides an integrated high-level symbolic task planning and motion planning approach using the proposed D-SSG based algorithm as an interface in the case of failure in the high-level symbolic planner. Chapter 6 provides an application case study involving an automated MaM system in a partially uncertain kitchen environment. Chapter 7 provides conclusion and some potential future search direction.

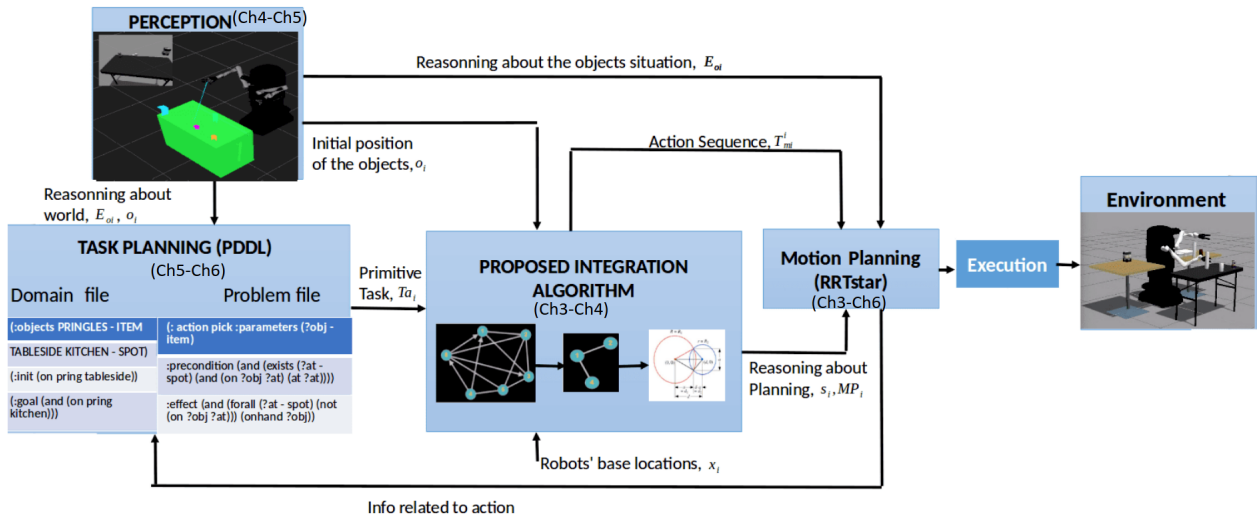


Figure 1.4: A roadmap of the thesis.



# Chapter 2

## Background, Problem Definition and Literature Review

This chapter presents a literature review on integrated task and motion planning for multi-robot manipulators. Although initial studies in this area began to appear in the late 1980s, many open problems remain [20–25]. Moreover, as computers are getting faster and are increasingly ubiquitous, the possibility of implementing a multi-robot planner is more feasible than ever before, for a wider range of industries. The objectives of this chapter are: 1) to provide an overall picture of this research field, 2) to present the current state of the art in research and technologies development, and 3) to provide a problem definition for integrating task and motion planning for multi-robot manipulators.

In this chapter, we provide a problem definition, notation and definition in Section 2.1. A general background on the motion and task planning of robot manipulators are presented in Section 2.2 and Section 2.3, respectively. Finally, in Section 2.4, manipulation planning techniques are introduced in detail.

### 2.1 Problem Definition, Notation, and Foundation

It is difficult to find solutions for mobile manipulation problems involving many robots, working with many target objects, due to the high dimensionality and multi-modality of their hybrid configuration spaces. Planners, from a purely geometric search perspective, are extremely slow at finding solutions for these problems because they are not capable of factoring the configuration space. While symbolic task planners can effectively obtain plans involving many variables, they cannot incorporate the geometric and kinematic constraints

required for manipulation. Our goal is address this shortcoming, and ultimately present an algorithm for solving task and motion planning problems.

Before presenting detailed problem statements, it will be useful to introduce some notations and definitions.

**Definition 2.1** (*Motion planning*). *Given a robot, a set of static obstacles, and an initial and a goal configuration of the robot, motion planning is the task of finding a collision-free path for the robot from the initial to the goal configuration.*

The Configuration Space representation introduced by [26] was the first major contribution to solving the motion planning problem. Denoted  $CS$ , it is the Cartesian product of the interval of the definition of each joint parameter. For a robot with  $n$  degrees of freedom (DOF),  $CS$  is an  $n$ -dimensional manifold containing all the robot configurations. It describes the pose of each body of the robot as a single point such that the robot's trajectory is a continuous path in  $CS$ . Therefore, the problem is transformed into finding a trajectory for a point in  $CS$  instead of a trajectory of several bodies in the Euclidean space.

The motion planning problem in general can be presented as finding a continuous path,  $p(t)$ , from a start configuration,  $p(0) = q_{init}$ , to a goal configuration,  $p(1) = q_{goal}$ , such that  $\forall t[0, 1], p(t) \in CS_{free}$ , where  $CS_{free}$  is the subset of collision-free configurations, i.e.  $CS_{free} = CS / CS_{obs}$ .

Next, definitions of key terms *transit-path*, *transfer-path* and *manipulation-path* will be introduced to ensure consistency with the relevant literature [7, 27–29].

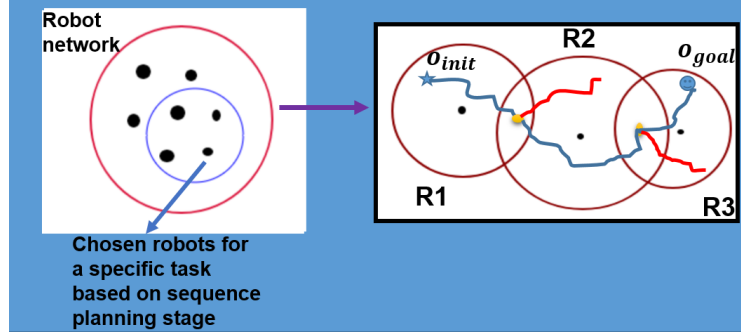
*Transit-path* is a path that describes the motion of an empty end-effector (i.e. one that is not grasping an object). This term is used to represent the path of an arm which avoids collision with other arms or which is moving toward an object in order to grasp it.

*Transfer-path* is a path that describes the motion of an arm which is grasping an object within its gripper. This term is used to represent the path of an arm that manipulates an object.

*Manipulation-path* is a set of alternating *transit* and *transfer* paths. This term is used to represent the sequence of motions of a set of arms that are moving an object from an initial position to goal position. In Figure 2.1, blue and red paths represent transfer and transit paths respectively. The manipulation path is the combinations of blue and red paths.

**Definition 2.2** (*State*) *A state  $S$  is a finite set of logical atoms.*

**Definition 2.3** (*STRIPS actions*) *A STRIPS action,  $o$ , is a triple, such that  $o = (pre(o), add(o), del(o))$ , where  $pre(o)$  are the preconditions of  $o$ ,  $add(o)$  is the add list of  $o$  and  $del(o)$  is the delete list of the action, each being a set of atoms. For an atom*



**Figure 2.1:** *Transfer, transit and manipulation path representations*

$f \in \text{add}(o)$ , we say that  $o$  achieves  $f$ . The result of applying a single STRIPS action to a state is defined as follows:

$$\text{Result}(S, \langle o \rangle) = \begin{cases} (S \cup \text{add}(o)) \text{ del}(o), & \text{if } \text{pre}(o) \in S \\ \text{undefined}, & \text{otherwise} \end{cases}$$

In the first situation, where  $\text{pre}(o) \in S$ , the action is supposed to be applicable in  $S$ . The outcome of implementing a sequence of more than one action to a state is recursively described as

$$\text{Result}(S, \langle o_1, \dots, o_n \rangle) = \text{Result}(\text{Result}(S, \langle o_1, \dots, o_{n-1} \rangle), \langle o_n \rangle)$$

From a set of actions, the goal is to find a sequence of actions that modify, by their addition and deletion lists, an initial set of atoms in a final set of atoms. A precise definition is reviewed below.

**Definition 2.4** (*Planning Task*). A planning task  $P = (O, I, G)$  is a triple where  $O$  is the set of actions, and  $I$  (the initial state) and  $G$  (the goals) are sets of atoms.

**Definition 2.5** (*Plan*). Given a planning task  $P = (O, I, G)$ , a plan is a sequence of actions in  $O$  such as  $P = \langle o_1, \dots, o_n \rangle$  that solves the task, i.e., for which  $G \subseteq \text{Result}(I, P)$  holds.

The task planning problem is to obtain a sequence of elementary operations that achieve a given task. Even though task planning is not the primary focus of this research, these definitions will be helpful later. One can refer [30] for more details.

**Definition 2.6** (*Manipulation planning problem*). Given a set of robots, objects, static obstacles, an initial configuration  $I$  for all robots and objects and a set of goal configurations

$G$ , obtain a path, for all robots and all objects, from  $I$  to a final configuration in  $G$ . The path must be collision-free and must satisfy the manipulation rules.

We define three main MaM manipulation problems to be solved in Chapters 3, 4, and 5 respectively.

**Problem 1:**

Consider a workspace containing a network  $\mathcal{S} = \{s_1, s_2, \dots, s_i, \dots, s_N\}$  of  $N$  robot arms with fixed base locations  $\{x_1, x_2, \dots, x_i, \dots, x_N\}$ , each of which are geometrically modeled as a monolithic rigid-body, and a set,  $O = o_1, o_2, \dots, o_m$ , of monolithic objects to be manipulated with 3D position and orientation- configuration space  $C_{o_k} \subset SE(3)$ . Each robot arm  $s_i$  has its own configuration space  $C_{s_i}$ , a fixed base and a workspace  $W$  which represents the space of the arm gripper positions and orientations. The initial configuration of the robot arm network  $\mathcal{S}$ , the initial and final positions of the objects,  $o_{i,init}$  and,  $o_{i,goals}$  are known. Also, the geometry, type and position of the obstacles,  $E_o$ , are available in advance. Also, note that  $O \subsetneq E_o$ . The exact configuration of tasks and their trajectories are not known in advance. The geometric configuration space of the whole problem is indicated as follows:

$$C = C_S \times C_{E_o},$$

where  $C_S = \prod_{i=1}^N C_{s_i}$  is the configuration space of all of the arms and  $C_{E_o} = \prod_{i=1}^n C_{E_{oi}}$  is the configuration space of all the objects. Next, define a sequence of tasks  $Ta_1, \dots, Ta_L$  with start times  $t_0^1 < \dots < t_0^L$  and start and goal locations  $p_0^1, \dots, p_0^L$  and  $p_t^1, \dots, p_t^L$  respectively. Each task  $Ta_i$  is defined in terms of pick and place action for a certain object  $o_k \in O$ , and an initially unknown sequence of subtasks  $T_1^i, \dots, T_{m_i}^i$ , with durations  $\Delta_j^i$  for  $j = 1, \dots, m_i$  and interim way-point waiting/process times  $\delta_1^i, \dots, \delta_{m_i-1}^i$ . The overall duration of task  $Ta_i$  is

$$\tau_i = \sum_{j=1}^{m_i} \Delta_j^i + \sum_{j=1}^{m_i-1} \delta_j^i \tag{2.1.1}$$

The values of  $\delta_j^i$ 's are fixed and pre-defined, but  $\Delta_j^i$  are not and can be optimized via optimal selection of the manipulator(s) to perform  $T_j^i$ . The objective is to develop an algorithm that allows robots to autonomously perform sequence planning (i.e., what robots should do to complete the goal) as well as path planning for each task to succeed a set of goals.

**Problem 2:**

Consider Problem 1, with the exception of some  $s_i$  in  $\mathcal{S}$  having mobile bases. Let  $x_i(k)$  denote the base location of Robot  $s_i$  at discrete time step  $k$ . Also, we know the geometry

and type of obstacles,  $E_o$ , but not their positions.

Our ultimate goal in solving Problem 2 is to develop an algorithm that allows robots to autonomously reason about a complex time-varying environment (a semi-structured environment) to perform high-level decision-making (sequence planning), and path planning for each robot, in order for each task to succeed a set of goals.

Traditional robotic motion planning approaches are followed to solve Problems 1 and 2. However, challenges arise in the translation of some kinematic and geometrical changes in the environment to a symbolic task planning layer (resources, availability, and time management concept-independent of manipulation in general) when arm(s) and/or object(s) locations change or a new object appears in the environment. This causes failures in the motion planning layer of a specific action and higher-level task planning, which can ultimately lead to failure in achieving the overarching goal (such as baking a turkey). Thus, all changes in the environment needs be translated into the task planning layer, and a third problem must be defined.

### Problem 3:

Consider Problem 2. However, we perform the planning first through a high-level symbolic task planner deciding on action specifications (AS) as a general purpose planning representation that supports arbitrary predicates as conditions in the form of PDDL (Plain Domain Definition Language). The objective is to design an algorithm that integrates motion planning and symbolic task planning (resources, availability, and time management concept) layers such that one can switch to alternate plans in case of kinematic and geometric changes in the environment and prevent failure in the high-level symbolic task plan.

In Problem 3, for instance, high level specifications like *pick* capture the logical preconditions of the physical actions. A more complete representation of the pick action can be written with predicates, such as *IsGP*, *IsMP* and *Obstructs* that capture geometric conditions:

*IsGP*( $p, o$ ) holds if  $p$  is a pose at which  $o$  can be grasped; *IsMP*( $traj, p_1, p_2$ ) holds if  $traj$  is a motion plan from  $p_1$  to  $p_2$  ; *Obstructs*( $obj, traj, obj$ ) holds if  $obj$  is one of the objects obstructing a pickup of  $obj$  along  $traj$ . The argument  $obj$  need not be an argument in *Obstructs*; we include it for clarity.

However, in the case where arm or object locations change, some of the arms in  $\mathcal{S}$  may not have access for the corresponding objects in  $O$  because they are located out of the corresponding robot’s reachability space. This causes not only failure in the motion planning layer of a specific action, but also the higher level task plan  $T_{ai}$  or  $T_{mi}^i$ .

In solution of all the three problems defined above, the following key questions need to be answered:

- 1 What is the minimal/path optimal robot arm sequence to complete a given manipulation planning problem in the case of failure, such as a failed pick action from the task planning layer, due to an unstructured environment?
- 2 What are the optimal positions along the manipulation path to perform transfers and/or cooperation?
- 3 What is the collision-free optimal trajectory of each arm throughout the sequence in Question 1?
- 4 What is minimal time required to complete each task defined above? (Do a time-indexed manipulator assignment for each task  $T_i$  among the available manipulators as of time  $t_0^i$  in order to minimize a cost function  $J = J_1 + \dots + J_N$ , and guarantee that manipulators do not collide each other other than the contacts during hand-overs.) In this thesis, we will focus on first three questions.

## 2.2 Motion Planning

Motion planning aims to obtain a collision-free path from an initial configuration to a desired goal, and is considered a continuous geometrical problem. The earliest formulations such as pianos mover’s problem [31] consider situations where the robot is a single unarticulated body, and have been seen in industrial applications for disassembly problems [32]. However, motion planning for articulated robots is more complicated.

The literature contains three classes of techniques that address motion planning problem: deterministic approaches, randomized approaches, and optimization-based approaches. For a broad overview of current techniques, the reader can refer to [33], [34], and [35]. Sampling-based techniques are the most effective of the modern motion planning approaches, and are classified into deterministic and probabilistic depending on the way the samples are generated.

Typical examples of the deterministic approaches are visibility graph, retraction algorithms [27], the A\* algorithm [36], the Dijkstra algorithm [37], and potential fields [38]. Among the most relevant probabilistic approaches are the Rapidly-exploring Random Trees planners (RRT) [35] and Probabilistic Road Map planners (PRM) [39]. However, the efficiency of these conventional probabilistic approaches drops dramatically in spaces with

narrow passages. Several variations have been developed to overcome this challenge, such as the multi-resolution PRM planner [40], dynamics domain RRTs [35], retraction base RRTs [41], and adaptive workspace biasing [42]. In order to speed up query path planning, some variants of PRM planners build a roadmap without checking for collisions. Then, once a potential solution path is found the existence of collisions is verified and if they occur the corresponding nodes and edges are removed from the roadmap and a new search is started; the process is repeated until a collision-free path is found (e.g. the Lazy PRM planner [5]). Some extensions of PRMs include the use of object symmetries in order to improve the performance of the planner [27], and consider scenarios where there are obstacles with known collision free paths around them which have to be connected to the general roadmap, to yield a solution path that skirts the obstacles [27]. The paths obtained with these planners can be optimized using post processing methods, which search for an optimal subset of samples in the configuration space that replace some samples from the initial path [27].

Finding a least-cost trajectory in a graph is a problem of many robotics related fields. Heuristic searches such as A\* search [36] have often been used to find such trajectories because they offer the strong theoretical guarantees such as completeness and optimality or bounds on suboptimality. Also, the generality of heuristic searches allows researchers to incorporate complex cost functions and complex constraints and to easily represent arbitrarily shaped obstacles with grid-like data structures [27]. Finally, heuristic searches provide good cost minimization and consistency in the solutions.

### 2.2.1 Deterministic methods

Deterministic techniques form and search for a systematically generated graph. Methods such as cellular decomposition, Voronoi diagrams, visibility graphs and the Cannys algorithm depend on the formation of a precise and accurate representation of  $CS_{obs}$  to construct a graph, or roadmap, that describes the connectivity of  $CS_{free}$  [43], [44], [45]. Other methods approach  $CS$  by discretizing it. Motion planning is then diminished to a search in a graph. Even though completeness and sub-optimality guarantee graph consistency, high dimensionality slows down the search for the problem. Potential field based algorithms are recognized as adequate to high-dimensional configuration space [38]; however, they are, in general, not complete because the planner can be stuck in local minima. Harmonic functions can formulate a potential with only one local minimum, which is the global minimum. However, such functions are represented only through a differential equation, and the straightforward solution is not known in the general case.

## 2.2.2 Sampling-based methods

Sampling-based approaches randomly sample  $CS$  and construct a graph of configurations connected by collision-free paths. This graph is commonly declared as a roadmap and approximate the connectivity of  $CS_{free}$ . Sampling-based algorithms are usually categorized into two groups: single query and multiple query algorithms.

Multiple query algorithms involve two steps: First, a roadmap representing all possible connectivity of  $CS_{free}$  is created. Second, motion planning queries are resolved by connecting the initial and final configurations to the roadmap. If the connectivity of  $CS_{free}$  is well-formed in the first step, the queries in the second step run fast. The best known multiple query algorithms are Probabilistic Road Maps (PRM) [9] and Visibility PRM (V-PRM) [46]. By contrast with PRM, V-PRM seeks to recognize configurations that add connectivity data to the roadmap. The last roadmap is smaller, which solves proximity queries quicker.

Single query algorithms do not seek to represent the connectivity of  $CS_{free}$  completely. Instead, the algorithms represent a piece of  $CS_{free}$  throughout the initial and goal configuration(s). A tree of configurations is built by iteratively extending the closest neighbour of a randomly sampled configuration in the direction of this random configuration. Although it is challenging to represent a short outline of the recent algorithms, the most popular is the variations of the Rapidly Exploring Random Tree (RRT) [35].

Randomized algorithms are highly effective in solving high-dimensional problems.

Nevertheless, there are a couple of disadvantages.

- These algorithms will only present probabilistic completeness. This implies that, if a solution exists, the probability of obtaining a solution converges to 1 as the amount of iterations increases. However, if no solution exists, they will work infinitely without identifying it. They depend on heavy randomization.
- There exists no explicit cost minimization. However, methods like PRM\* and RRT\* [21], provide an optimal guarantee in the limits of samples. These methods include an action to recompute the shortest path between sets of configurations in the roadmap. This rewiring action is of fixed time but time-consuming.

In [47], [48], and [9], the authors proposed the basis for probabilistic roadmap methods (PRM). PRM-techniques have two phases: a learning phase, and a query phase. First, the configuration space is sampled for collision-free configurations. Second, a graph known as a roadmap is created using these configurations as nodes. A basic local planner is applied to search for connections between nodes. If a connection is encountered, an edge is added



to the graph connecting the corresponding nodes. Eventually the roadmap will find an efficient representation of  $CS - free$ .

### 2.2.3 Optimization-based methods

Optimization-based approaches express motion planning as a trajectory optimization problem. Provided a naive initial trajectory, the approaches iteratively extract the trajectory from collision while optimizing a cost. The optimization handles both a pre-defined cost - path length, energy- and collision detection parts. Collision avoidance is represented as a constraint of the expressed distance function. This function is described as; a positive distance that resembles the shortest distance between non-colliding objects, while a negative one represents the penetration between colliding objects.

The two main optimization-based approaches in the literature are Covariant Hamiltonian Optimization for Motion Planning (CHOMP) [49] and Stochastic Trajectory Optimization for Motion Planning (STOMP) [50]. CHOMP decreases the overall cost based on covariant gradient data. STOMP produces noisy trajectories to search the space throughout the current trajectory. They are then merged to build an updated trajectory of lower cost. However, STOMP cannot be expected to solve typical motion planning problems like the alpha puzzle in a reasonable amount of time [50].

### 2.2.4 Constrained motion planning

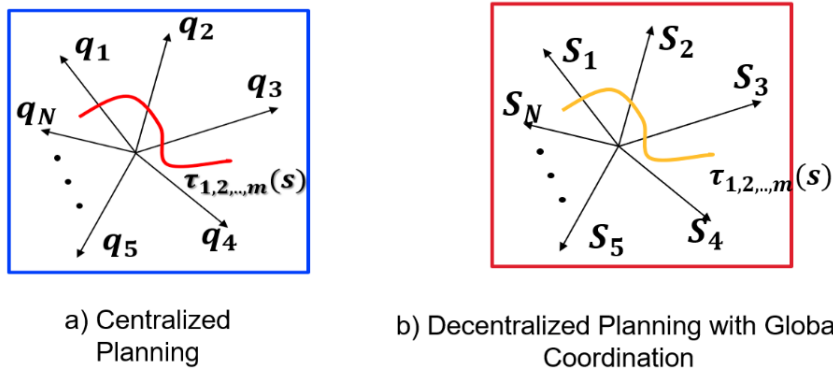
The feasible configuration space of closed-loop systems, such as humanoid robots and underactuated robots, is of the scheme  $q \in CS \mid f(q) = 0$ , where  $f$  expresses the constraints like a loop closure constraint, an equilibrium constraint, etc. The possible configurations are described only inherently, and it has stress zero in  $CS$  when  $f$  is non-trivial.

The motion planning problem is represented as obtaining a continuous path  $p$  such that  $\forall t \in [0, 1], p(t) \in CS_{free}$  and  $f(p(t)) = 0$  for a constraint  $f$  from  $CS$  to  $\mathbb{R}^n$  such that a configuration  $q$  is feasible if and only if  $f(q) = 0$ . Most methodologies obtained in the literature only try to solve a complex rendition of this issue. The studies [51], [52] have presented constrained versions of randomized planners yet their algorithms do not ensure continuity of the solution path.

Until now, we have presented motion planning approaches for a single arm. However, motion planning approaches for MaM in a shared environment creates highly complicated problems in high dimensional spaces since scaling up the number of DOF quickly complicates the search process. For example, a  $CS$  with 6 manipulators, each having 6 DOF, requires a motion plan for 36 DOF. We have examined the most common motion planners,

including RRT [35, 53], PRM [2, 53], and their more recent variants, and found them to be too computationally expensive to solve (exponential in time) for MaMs. The study [27] demonstrates that the motion planning problem for single and multiple robots is basically identical. Thus, theoretically they both have the same sampling based or combinatorial search algorithms. So, we can categorize the approaches for MaM’s problems into two main groups: 1) centralized and 2) decoupled planning

**Centralized Motion Planning:** Centralized Motion Planning approaches take the total DOF into account when searching for a solution. In this approach, the paths for all robots are planned simultaneously by searching the C-space of multi-arm robots. For example, with 6 robots (each with 6 DOF), the C-space will have 36 DOF. Advantages come with completeness, which means, if the underlying planner is complete, a solution is guaranteed to be found (if one exists). However, it is potentially expensive, and typically requires searching high-dimensional spaces and knowledge of goals and states of all robots.



**Figure 2.2:** *Multi Robot Planning Approaches: An initial and a goal configuration are given as input for each robot.*

**Decoupled Motion Planning:** The second approach, Decoupled Motion Planning, is more relevant in terms of time consumption but lacks the computational efficiency and optimality traits. As listed in [1, 4] decoupled planning is categorized into 3 main methods:

- **Prioritized Planning (PrP):** In this approach, we sort all the robots by priority and plan an individual path for each of them. Each path is calculated based on the hierarchy obtained by the sorting procedure such that the path of the higher ranked robot is being calculated first. The collision-free element is acquired by treating the higher-ranking robots as a moving obstacle.

- Fixed Path Coordination (FPC): This method is divided into 2 parts.
  - First Phase - a collision-free path  $t_i$  is generated for each robot considering only obstacles (ignoring other robots) in its space
  - Second Phase (Velocity Tuning) coordination of the robots' velocities along their pre-generated paths to prevent collisions between robots. The two coordination methods are
    - \* Pairwise Coordination
    - \* Global Coordination
  - Each robot is restricted to motion in its pre-generated path although it may stop, retreat or change velocity to allow coordination with other robots.
- Fixed Roadmap Coordination (FRC): This strategy extends the FPC by assuming that each robot is guided by a roadmap. This yields a wider set of routes to achieve a single robot goal (instead of one in the FPC). Thus timing of the overall coordination may be more accessible.

Studies validate the assumptions that loss of completeness with decoupled planning can be ignored in practice. SBL helps to make the usage of centralized planning for multi-robot systems practical. But centralized planning still requires knowledge of all robot states, which may be impossible in some settings. On the other hand, decoupled planning can be quite unreliable particularly in tight robot coordination. Centralized planning appears to have better reliability compared to the decouple approaches.

Although heuristic searches are widely popular, they have not been applied for motion planning of high- DOF robotic manipulators due to the high-dimensionality of the planning problem. In this research, we present a heuristic search-based planner for manipulation by reducing this high dimensionality problem.

In our research, we apply the same principles of *decoupled* planning approaches with the essence of the *prioritized* method. By resolving each primitive task, we obtain that it is associated with only a single arm (e.g *transfer arm*). This is the key feature that guides our search. Only a single arm gets the highest priority while all the others are non-prioritized.

## 2.3 Task planning

We must note that task planning or merely planning is not essentially related to robotics. Alternatively, this technique should be considered solely from an algorithmic

perspective, and one can prefer resolving problems without communicating with the real world. Planning, as noted by researchers of AI, is an alternative approach to solving some discrete problems without incurring ad-hoc algorithmic solutions. It can also be seen as the problem itself, and consequently, its computational complexity can also be analyzed. The solution is not unique because it depends on the considered paradigm, the permitted features and the assumptions created over the domain. A planning system generally uses the following input: a domain, i.e. the description of the problem and the possible actions, and a problem instance, i.e. the initial state and the desired (goal) state. Its output is a sequence of actions that can drive to the goal if applied from the initial state. Alternatively, it should tell if such a sequence does not exist.

The task planning problem is to obtain a sequence of elementary operations that achieve a given task. Even though task planning is not the primary focus of this research, these definitions will be helpful later. One can refer [30, 54] and definitions (Definition 2.2-2.5) presented earlier in the chapter for more details.

We can examine the case of The Towers of Hanoi. A potential ad-hoc solution for the variant of the problem in which the items are initially placed in the first branch, the classic recursive algorithm. Alternatively, a Breadth-First Search algorithm within adjacent configurations can also be used to solve the case of the problem initially from any arbitrary configuration. A planner is a multipurpose application that takes the description of the problem, in this example, the rules of The Towers of Hanoi, the initial configuration and the end configuration, and uses a general algorithm to solve the problem defined by these three elements. The price to pay is, generally, efficiency.

The study in [54] presented the Fast Forward (FF) heuristic. It applies the simple problem to guide their search algorithm named enforced hill-climbing. It is a forward search engine, which indicates it never does backtracking, instead, attempts to append actions to the recent task plan. Selections are never re-considered. The only way to continue after deciding an faulty action is to exchange the effects of this action. Hence, although it has been considered successful in various schemes, the output can include an arbitrarily large number of actions and their reversed actions, which makes task planning problem quite suboptimal. The method is complete if the input problem includes no reachable dead-end states [54]. A state is a dead-end if and only if no sequence of actions achieves the goal from it. Some practical measures, like invertible planning task [54], [55], can make some problems dead-end free.

### 2.3.1 The Planning Domain Definition Language

The Planning Domain Definition Language (PDDL) [15] is a planning language first developed by Drew McDermott in 1998, as an attempt to standardise previous existing Artificial Intelligence planning languages. All those planning languages are used to model a given planning problem with certain conditions, and then automatically generate a chain of actions that lead to a desired goal which can be expressed as a set of conditions. The described actions will also depend on conditions, which need to be fulfilled in order to be executed, and will produce effects in the environment once executed.

In the Planning Domain Definition Language the planning task that we want to solve is expressed as a triple  $\langle O, s_0, g \rangle$ , where  $O$  is the set of parametrised propositional actions defined by preconditions and effects,  $s_0$  is the initial state, and  $g$  is the goal. Each state  $s$  including the initial state and the goal are expressed as a set of logical atoms or conditions. At the same time the actions are defined as a triple  $o = \langle pre(o), add(o), del(o) \rangle$ ; where  $pre(o)$  represent the preconditions necessary to apply the action,  $add(o)$  is the list of added effects once the action is executed, and  $del(o)$  is the list of deleted effects. We say that a single condition or atom  $f$  is achieved by the action if  $f \in add(o)$ .

Then we say that an action is applicable if our current state  $s$  fulfils  $s \subseteq pre(o)$ . The result of applying action  $o_i$  in state  $s_i$  generates a new state  $s_{i+1}$ .

**Limitations** Task planning is of somewhat limited use in robotics if we consider it as a purely symbolic problem. Actions are considered to be always feasible, while the underlying geometrical problem may be conditionally feasible or even infeasible. Admittedly, an action such as "pick object" on the desk is feasible only if the object is at a reachable location on the desk. For instance, in continuous grasps or placements examples, it is symbolically impossible to describe each grasp and each placement separately, so the case is even more challenging. To solve these kinds of problems, task planning must be combined with a geometrical motion planner, as described in the next section.

## 2.4 Manipulation planning

Most recent approaches, however, consider manipulation planning as a combination of task and motion planning. As a computational geometry problem, it has raised a lot of interest for the past forty years. Pioneering works by author of [56] and [57] first considered low dimensional problems where robots and objects move in translation. [58] and [57] are the first works that apply random motion planning methods developed a few years earlier [9]

to the manipulation planning problem. Recently, the domain has regained interest with various variants where papers propose approaches that tackle the inherent complexity of manipulation planning. The domain is traditionally divided into several categories.

Rearrangement planning consists of automatically finding a sequence of manipulation paths that move several objects from initial configurations to specified goal configurations [7], [59]. Multi-arm motion planning has also given rise to a lot of papers [17], [28], [7]. From a geometric point of view, manipulation planning is a hybrid problem where discrete states (gripper A holds object B) are defined by continuous constraints on the positions of objects and robots. States are connected by manipulation trajectories that give rise to the underlying structure of a graph whose nodes are the discrete states [60]. This structure, although not expressed as such, is present in various papers [52], [61], [55,62]. The partially discrete nature of the problem has also given rise to integration of task and motion planning techniques [63], [64], [17], [16], [65].

Given a set of robots and objects, the problem is to obtain both a sequence of elementary actions and robots and objects paths for each action to fulfill a set of goals. Compared to task planning, this aims to solve the limitations introduced above, i.e. the output is guaranteed feasible. The added complexity remains in the fact that solving motion planning problems are time-consuming and most algorithms are just probabilistically complete. They cannot state a problem infeasibility, and it is challenging, if feasible at all, to estimate search progress in the general case.

The following sections introduce a manipulation planner classification in two sections: multi-layer planners and single layer planners. The latter use a unique data structure to organize the data gathered during the search, while the former use a hierarchy of data structure to represent them.

### 2.4.1 Multi-layer manipulation planners

These planners try to solve the problem with two or three planning layers. A high-level symbolic planner creates task plans. A low-level geometric planner produces paths for elementary actions. Unluckily, symbolic (discrete) and geometric (continuous) planners do not use the same language. For example, a symbolic action, such as "pick object" on the desk or "place object" on the desk, does not define how the object must be grasped or where it must be released. However, the geometric planner requires to know it. Decreasing these possibilities to one is not adequate because the selection is case-dependent. Hence, the problematic concern is the communication between the two layers.

In [16], the authors introduce an intermediate layer that defines task plans to the motion planning layer. For each action of the task planning layer, the intermediate layer

calls the motion planner for each possible value of the effects of the action until it has found one which accomplishes. The algorithm backtracks to the previous action when motion planning fails. If the result fails, a partial plan is obtained, and the algorithm seeks a possible reason, and then updates the task planner before re-starting. For example, the action "place object" on the desk generates a subset of object poses on the table and plan only consider this subset. If the issue cannot be handled, then a new subset of object poses is produced.

In [63], the authors present a multi-layer method considering similar perspective as single layer planning. The study applies the reduction property, reviewed below, which offers a suitable approach for continuous grasps and placements. [66] focuses on Navigation Among Movable Obstacles (NAMO), which consists of planning a path for a robot that needs to move obstacles in order to reach the goal configuration [66], [52]. The robot re-configures the environment by removing obstacles and clearing free space for a path. A heuristic obtains a path without counting collisions with movable obstacles. This path is used to decide what objects should be transferred. Objects are transferred when they allow two distinct, but connected components of the  $CS$  to merge. However, this approach will not move obstacles that do not block the robot, even if they prevent other movable objects from being transferred. An important problem of multi-layered approaches is that the motion planning algorithm cannot prove infeasibility. Most consider a problem infeasible when some threshold is reached in terms of the number of iterations or elapsed time. Although this is satisfactory for many problems, it is a very ineffective approach in general, and it needs parameter tuning.

Another problem comes from the task planning techniques most methods use: the FF heuristic or one modification of it. As asserted above, the task plan returned by FF can include an arbitrarily large number of actions and its inverse. In a pick and place scenario, it suggests the robot can pick up and put down the object several times in an inconvenient way. From a theoretical perspective, this aspect is not significant. However, in practice, this produces inconvenient and useless motions.

## 2.4.2 Single layer planners

Unlike multi-layer planners, single-layer planners consider the problem using the Cartesian product of the configuration space of robots and objects. As this is the structure that will be used later, we recall necessary notions in [46]. The reduction property: We have a problem with a robot and an object. The configuration space of the system is  $CS = CS_{robot} \times CS_{object}$ . The domain in  $CS$  corresponding to the object's proper placements, i.e. stable placements where the object can hold when delivered by the robot, is

expressed by  $CP$ . The domain in  $CS$  corresponding to proper grasps of the object by the robot, is expressed by  $CG$ . Both  $CG$  and  $CP$  are sub-manifolds of  $CS$ .

A solution to a manipulation planning problem corresponds to a constrained path in  $CS_{free}$ . Such a solution path is an alternate sequence of two types of sub-paths verifying the specific constraints of the manipulation problem and separated by grasp/ungrasp operations.

- Transit paths where the robot moves alone while the object stays stationary in proper placement. They rest in  $CP$ . Nevertheless, a path in  $CP$  is not usually a transit path since such path has to belong to the sub-manifold corresponding to a fixed placement of the object. They induce a foliation of  $CP$ .
- Transfer paths where the robot moves while holding the object with the same grasp. The position of the object concerning the robot's end-effector is fixed. They rest in  $CG$  and cause a foliation of  $CG$ .

For completeness, we recall the notion of foliation [67].

**Definition 2.3.4**(Foliation). Foliation of a  $n$ -dimensional manifold  $M$  is an indexed family  $L$  of arc-wise connected  $m$ -dimensional sub-manifolds  $m < n$ , called leaves of  $M$ , such that:

- $L_\alpha \cup L_{\alpha'} = \emptyset$  if  $\alpha \neq \alpha'$
- $\bigcup_{\alpha} L_\alpha = M$
- every point in  $M$  has a local coordinate system such that  $nm$  coordinates are constant.

An instance of a discrete manipulation problem is navigation inside a building that includes floors and staircases. To go from one floor to another, one must use a staircase. To go from one staircase to another, one must navigate on the floor.

Permitted motions induce a foliation of the set of floors, in which each floor is a leaf, and a foliation of the set of staircases. Navigating in this space needs to find an alternative sequence of floors and staircases. Two foliation structures are described in  $CG \cap CP$ , which indicates the following property, demonstrated in [68].

**Theorem 1.1** (Reduction property). *Any path lying in  $CG \cap CP$  where the robot is not in collision with static obstacles can be transformed into a finite sequence of transit and transfer paths.*



This feature decreases the manipulation problem to discover several components of  $CG \cap CP$  using transit and transfer paths. They give two multiple-query algorithms depending on PRM. This is the main theoretical result in manipulation planning.

Another successful algorithm is FFrob, introduced by authors of [17]. It is an expansion of the Fast Forward heuristic that considers geometrical data. A set of beneficial object poses and robot configurations are sampled offline and stored in a conditional reachability graph. When the planner fails to solve the problem, new object poses and robot configurations are sampled, and the graph is updated. Thanks to these offline computations, FFrob can find a solution for challenging problems in a fair amount of time. Nevertheless, multiple parameters are to be tuned. Most of them are very case-specific such as the number of sampled object poses, the number of grasp configurations, the number of iterations of the RRT to solve a motion planning problem.

The Diverse Action RRT (DA-RRT) algorithm, introduced by [64], considers the Diverse Action Manipulation problem. The inputs to the problem are a mobile robot, a set of movable objects, and a set of diverse, possibly non-prehensile manipulation actions. This algorithm obtains a high-level sequence by planning a path just for objects in the domain. It next tries to complete each transfer manipulation separately. A similar approach is the Sampling-based Motion and Symbolic Action Planner, introduced in [64]. The authors generate a tree of configurations by sequentially utilizing actions. The action is picked using a measure of the efficiency of actions.

Combining motion and task assignment (hybrid planning) is an open problem in robotics [55, 69]. In the literature there are several relevant algorithms that deal with hybrid planning. One of these works employs a high-level planner that acts as a constraint and provides a heuristic cost function in the search algorithm in order to speed up the motion planner [10]. Furthermore, a grid-based discrete representation can also be used to combine the information from the task planner and the information from the motion planner in order to obtain a continuous free-collision trajectory for the robot [11]. Another way to combine the task and motion planners is formulated as a representational abstraction between them, where, each action in the task planner can have multiple instantiations in the motion planner. This increases the possibility of finding a feasible trajectory for the robot [12].

The study [8] introduces the problem of combining motion and task assignment for a dual-arm robotic system. Each arm of the system performs independent tasks in a cluttered environment. Robot acts are considered to eliminate possible obstacles and obtain collision-free paths to grasp the target objects. In the proposed scenario, the motion planner provides data to create a graph structure to represent the obstacles to be removed. The graph is applied to decide which is the next motion path to be computed, and to assign the tasks to each arm of the robotic system.

In [70], authors introduce an approach for online integration of 3D perception and manipulation for personal robotics applications. They propose a modular and distributed architecture, which integrates the creation of 3D maps for collision detection and semantic annotations, with a real-time motion re-planning framework.

The studies [29, 71] introduce that an object is moved by multi-robotic arms from an initial to goal configuration by avoiding obstacles in the environment. They suggest a solution based on the generated path for the object while simultaneously checking whether it can be grasped, if it reaches a position where the grasp is not valid any more, it searches for how to grasp the object with another arm. This way ensures that the object continues the motion through a calculated path while being grasped.

In general, cooperating robots correspond to overactuated systems or redundant systems, where the effective degrees of freedom are higher than those strictly required to perform a given task. This capacity increases the dexterity of the mechanism, and can be used to avoid joint limits, singularities and workspace obstacles, as well as to minimize the energy consumption and joint torques or to optimize a performance index [72].

The study [73] proposed a pick-and-place planner suitable for use in animation of autonomous agents. The emphasis was put on natural-looking motions and near real-time behaviour. In one of the demonstration applications, a virtual chess player planned and executed commanded moves in near real-time. The decomposition of the task is the same as that in [93], but here all motions are planned using Rapidly exploring Random Trees (RRTs) [78, 72], leading to a more robust and less heuristic planner (no restrictions on the last three joints, no discretization of the configuration space).

The study [74] introduces a sampling-based mobile manipulator planner based on the base pose uncertainty and the impacts of this uncertainty on manipulator motions. The planner uses the Hierarchical and Adaptive Mobile Manipulator Planner (HAMP) plans for both the base and the arm reasonably. In addition, it utilizes localization-aware sampling and connection strategies to consider only those nodes and edges which contribute toward better localization. Moreover, it fuses base pose uncertainty near the edges (where arm stays static) and impacts this uncertainty on arm motion.

The study [39] extended the Probabilistic Roadmap (PRM) framework to handle manipulation planning. Their approach uses the manipulation graph, whose edges can be transfer paths, or transit paths. A transfer path corresponds to a movement where the task object is grasped and moves along with the robot. Accordingly, a transit path is a path where the object is left in a stable position and only the robot is moving. The generality of the approach makes it possible to solve tasks that require several re-grasping motions. However, the planner has to be initialized with a user-defined set of grasps and stable object placements. Whereas the approach in [39] used a discrete set of grasps and

object placements, the approach in [46] could handle continuous sets. A continuous set of grasps assumes that the set can be parametrized by some coordinates. This is easy in cases that involve, e.g., a parallel-jaw gripper and a bar with rectangular cross section. The example in [46] showed that the planner is capable of finding solutions to problems that require long sequences of re-grasping motions.

Many robotic applications include multiple arms in the same workspace, which are centrally coordinated to transfer individual objects to desirable locations and orientations using grasping. The focus is to codify the topology of multi-arm manipulation problems [12, 75] in a way that allows for efficient integrated task and motion planning [76–78].

Existing efforts generally do not explicitly construct the topology for multi-arm manipulation. Early work performed an implicit transfer-transit path search given simplifications, such as not allowing regrasps at stable poses [29].

Alternatives build a large sampling-based roadmap by composing graphs for each arm [11], [12]. A recent effort employs heuristic search to first plan an unconstrained object path, which informs the search for the arm paths [79], but this is limited to light objects transferable by a single arm, and does not allow regrasps at stable poses. It can however be used as a heuristic in the proposed framework. Another approach deals with the multi-modal nature of manipulation, but focuses on complex plans for a single arm [64]. Also, the study in [80] is the first used sampling based planners to solve manipulation planning problem for redundant robots.

Although the topology of dual-arm manipulation is explicitly defined in [28], the topology of multi-arm case was recently defined in [7], topologies for mobile multi-arm manipulation have yet to be studied. Our research in this report extends the field to the multi-arm case and to codifying the topology of mobile multi-arm manipulation. It fills gaps in recent efforts, such as in [1], which is restricted by the number of arms needed to transfer an object and the feasible handoffs given the spatial arrangement of the arms in the workspace. The authors also assumed key variables are already known, such as which robot will grasp or transfer the object, and, while static obstacles in the shared workspace were considered, the possibility of collision between arms was not.

In our research, described in Chapters 3 through 5, we apply a hierarchical approach to decide which robot will do the task and in which order. Our proposed graph representation is based on the shared workspace between arms for transferring an object as represented in Fig 3.1. In this case, each robot must work in their transit area if there is no transfer between arms.

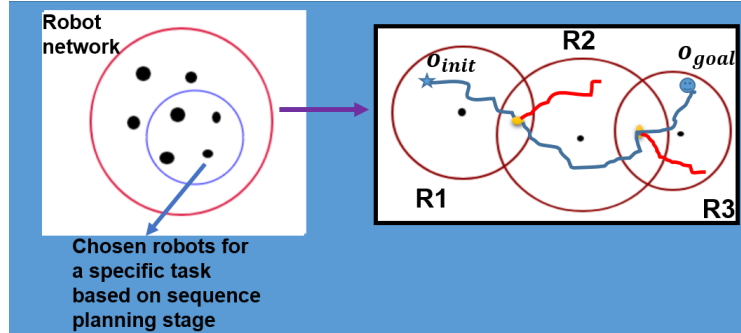
# Chapter 3

## Shared Space Graph Based Manipulation Planning

### 3.1 Introduction

In this chapter, we introduce an integrated motion and task planning methodology for multiple robot arm systems with high DOF applications performing different tasks simultaneously in a shared workspace as defined in Problem 1 in Section 2.1. In the proposed methodology, we introduce a high-level task planning layer and a low level motion planning layer. We develop an algorithm to integrate these two layers in order to achieve reliable and collision free task execution. At the low level motion planning layer, we extend heuristics based path planning algorithms for systems of multiple high DOF robotic arms. We introduce concept of shared space graph (SSG) to check whether two arms share certain parts of the workspace and quantify cooperation of such arm pairs, which is essential to the selection of arm sequences and scheduling of each arm in the sequence to perform a task or a sub-task.

In this chapter, we will focus on first three questions introduced in Section 2.1. To answer these first three questions, our approach first determines pairs of arms  $(A_i, A_j)$  in set  $\mathcal{S}$  with intersecting workspaces, i.e.  $W_{\mathcal{S}(i)} \cap W_{\mathcal{S}(j)} \neq \emptyset$ , to provide the topology of mobile multi-arm manipulation graph for any transfer that could appear. This topology can be represented by a graph  $G=(V,E)$ , where each node  $i \in V$  stores the base location of a particular arm  $A_i$  and the number of collision objects in the  $A_i$  robot's workspace and each edge  $(i, j) \in E$  represents a non-empty common workspace shared by the arms  $A_i$  and  $A_j$ . In this work, this graph is called *shared space graph* (SSG). Hence the SSG of a system of  $N$  arms has  $N$  nodes. The edges are found by checking the workspaces of



**Figure 3.1:** *Transfer, transit and manipulation path representations*

individual robot arms and their intersections as illustrated in Figure 3.2. The proposed SSG representation can be applied online as well as offline since the proposed idea does not require extensive sensory data rather than a vision system, except the base locations of the robots and objects, especially if the robots have mobile bases. Based on the notion of SSG, we propose the following 3-Step solution to the problem defined above:

Step 1: Generate the SSG with vision guide, corresponding to the shared workspace of the arms (Algorithm 1, line 1),

Step 2: Find the optimal arm sequence performing optimization based on the SSG as well as the individual robot arm workspaces (Algorithm 1- lines 2, 3, 4),

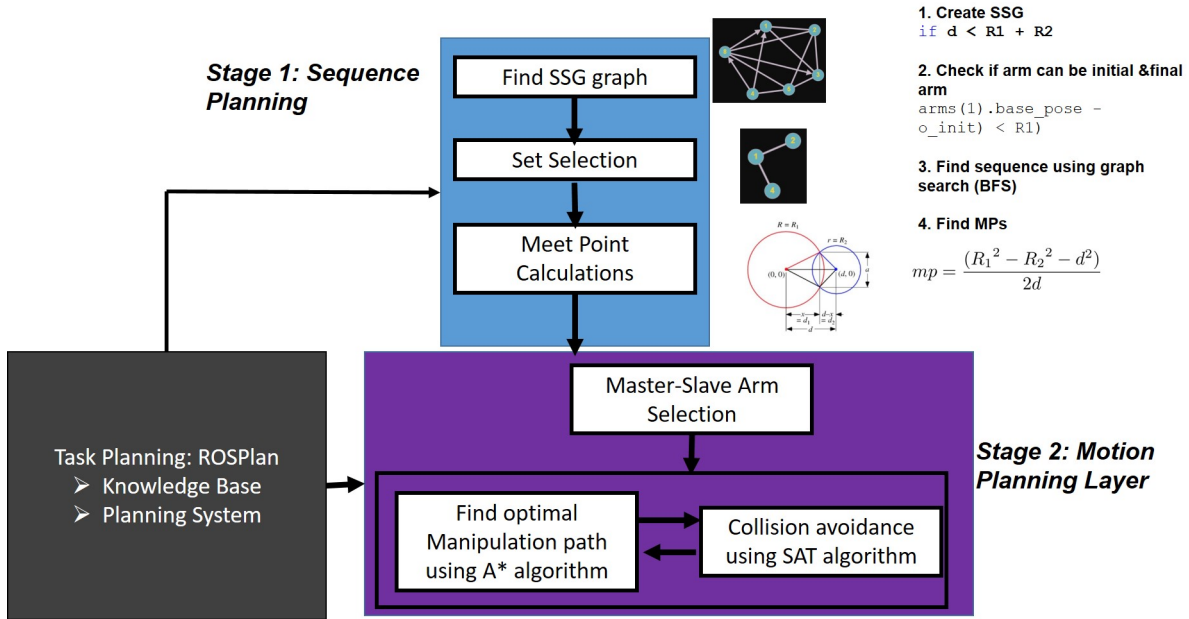
Step 3: Plan a collision-free manipulation path for each required task (Algorithm 1- lines 5,6)

Step 4: Integrate Step 1,2 and 3 with task planning layer to cope with failure in manipulation planning due to unstructured environment (Step 4 will be handled in Chapter 4 and Chapter 5).

## 3.2 Algorithm Development

Based on the problems described earlier, we consider both selection and motion planning attributes. For each attribute, a formal approach is followed. Our planning approach is guided by the principle of integrating path set  $\mathcal{M}$  and arm sequence  $S$  and their interpretation for primitive tasks. Hence, the algorithm is structured into two stages where Stage 1 maps elements of  $\mathcal{M}$  and elements of  $S$ , not necessarily one-to-one, and Stage 2 generates each corresponding task, and plan the arms' motions.

- Stage 1 = Step 1 + Step 2: Obtain a set of arms which can transfer an object from



**Figure 3.2:** A summary of proposed integrated task and motion planning

initial to goal positions. The main purpose of this stage is to obtain a series of arms where the object is transferred from one arm in the series to the next such that the last arm will place the object at its goal location (Fig. A.13).

- Stage 2 = Step 3: Calculate a collision free trajectory for each step of the arm series found in Stage 1. This stage is a more kinematic stage since the joint/work spaces of all of the arms are considered when a trajectory is calculated.

Our first objective is to combine the two stages by searching for applicable set in terms of kinematic feasibility. Since having a feasible set helps to grasp an object at each step by a different arm, this can be interpreted as a state, where a configuration of a single arm is a set. Our second objective is to calculate a transition function between those states. Here, a transition function is considered as a motion plan as detailed in the Stage 2. In the following subsections, details of the two algorithms stages are provided.

### 3.2.1 Stage 1: Multi Arm Sequence Planning

As mentioned above, a *manipulation-path* is defined by some primitive tasks, which come from task planning layer, such as pick and place operations. However, in this work,

we divide each primitive task to further subtasks before it is introduced to the motion planning layer. Therefore, the term primitive task is used to denote the  $k$ 'th procedure in  $\mathcal{M}$ , for convenience. To perform each primitive task separately, we simplify the problem to a single robotic arm pick and place problem. Still, there is no guarantee of completion of the whole task unless a binding condition is made between consequent primitive tasks. Such a criterion is described by the proposed *meet point* definition.

Let  $\mathcal{T}$  be a task, which involves  $K$  arms to bring  $o$  from  $o_{init}$  to  $o_{goal}$ .

$$\mathcal{T} = \bigcup_{j=1}^K T_j,$$

where each primitive task  $T_j$  represents movement of the object  $o$  along a portion of path,  $\mathcal{M}$  by a different arm. The goal position of a task,  $T_j(goal)$  is the initial position of next task,  $T_{j+1}(init)$  i.e.,  $T_j(goal) = T_{j+1}(init)$ . We define a *meet point* (MP) as a position where any  $T_j$  starts or ends. Any *MP* is located where an object is being picked or placed by a transferring arm. Any set  $S$  that involves  $K$  arms has to pass through  $o_{init}$ ,  $(K - 1)$  number of *MPs*, and  $o_{goal}$ , respectively, as illustrated in Figure 5, *Scheme 1*. MP is found by considering circle-circle and/or sphere-sphere intersection geometry by selecting the centre of the shared workspace (Figure 4.3).

By the fact that each of  $o_{init}$  and  $o_{goal}$  is in the workspace of one of the arms, we can reason about the starting and final nodes in the *SSG*. At this point, any graph search would give a solution in the form of a set of indices of arms  $\mathcal{S} = \{s_1, \dots, s_k, \dots, s_K\}$  connected by their shared workspaces such that:

$$o_{init} \in W_{S(1)} \tag{3.2.1}$$

$$o_{goal} \in W_{S(K)} \tag{3.2.2}$$

$$W_{S(k)} \cap W_{S(k+1)} \neq \emptyset \tag{3.2.3}$$

To complete the algorithm's first stage we still have to determine the positions of the *meet point series*. As shown in Figure 4.3, we can detect that any *meet point* is located in the shared work space of any two successive arms in  $\mathcal{M}$ .

Algorithm 2 shows the main procedure for generating a solution. Here, lines 1-4 represent the first stage of the algorithm for calculating the transfer arms and meet points and lines 5-6 represents the motion planning and assignment of the tasks to each robot respectively. The lines 1-3 are explained in detail above. Line 4 is finding optimal arm sequence by considering initial and goal workspace of the robots, as calculated in lines 2-3, and *SSG* graph based on Dijkstra's algorithm, as explained in [81]. The cost here is the

number of interactions between arms. We feed our motion planning ( $A^*$ ,  $RRT^*$ ) with  $MP$ ,  $S$  data to calculate transfer path for each robot individually (Line 5). Lastly, we map each  $S$  to each of  $\mathcal{T}$  by considering all  $\mathcal{M}$  in Line 6 (Figure 4.3).

---

**Algorithm 1** Multi Arm Sequence Planning

---

**Require:**  $o_{init}, o_{goal}, \mathcal{A}$

**Ensure:**  $\mathcal{M}$

- 1:  $S_{SG} \leftarrow \text{GenerateSharedSpaceGraph}(\mathcal{A})$
  - 2:  $S(1) \leftarrow \text{obtain}(S(1) \text{ such that } o_{init} \in W_{S(1)})$
  - 3:  $S(K) \leftarrow \text{obtain}(S(K) \text{ such that } o_{goal} \in W_{S(K)})$
  - 4:  $\mathcal{S} \leftarrow \text{Dijkstra}(s_{sg}, S(1), S(K))$  // generate ws sequence
  - 5:  $M \leftarrow \text{CalcTransferPath}$  using  $MP(\mathcal{S}, o_{init}, o_{goal})$
  - 6:  $\mathcal{T} \leftarrow \text{GeneratePrimitiveTasksSequence}(M, \mathcal{S}, MP)$
- 

### 3.2.2 Stage 2: Multi Arm Motion Planning

This section explains our proposed solution to calculate a collision-free *manipulation-path* for each of the arms. Still, a single arm motion planner is not applicable in a multi-arm scenario and a multi-arm motion planner solving for a large number of arms is highly complex.

In our problem, the overall goal is to obtain a multi-arm trajectory for each of the primitive tasks extracted by the proposed algorithm. Note that initial configuration of the each arm in  $\mathcal{S}$  and the transferring arm index are known. For simplicity, we assume that there are two types of obstacles: the arms themselves and the objects. The proposed planning scheme targets the following requirements: i) changing priority when needed, ii) plan a path for moving arms only.

Our planning scheme is built by the same principles of *decoupled* planning approaches with emphasis on the *prioritization* based on [27]. By resolving each primitive task, we obtain that it is associated with only a single arm (e.g. *transfer arm*). This is the key feature that guides our search. Only a single arm gets the highest priority while all the others are non-prioritized. Still, changing the priority of the obstructed arm means nothing without having a feasible target. Hence, in order to depict the procedure of changing priority it has to come with adding a target.



### 3.3 Simulation Results and Testing

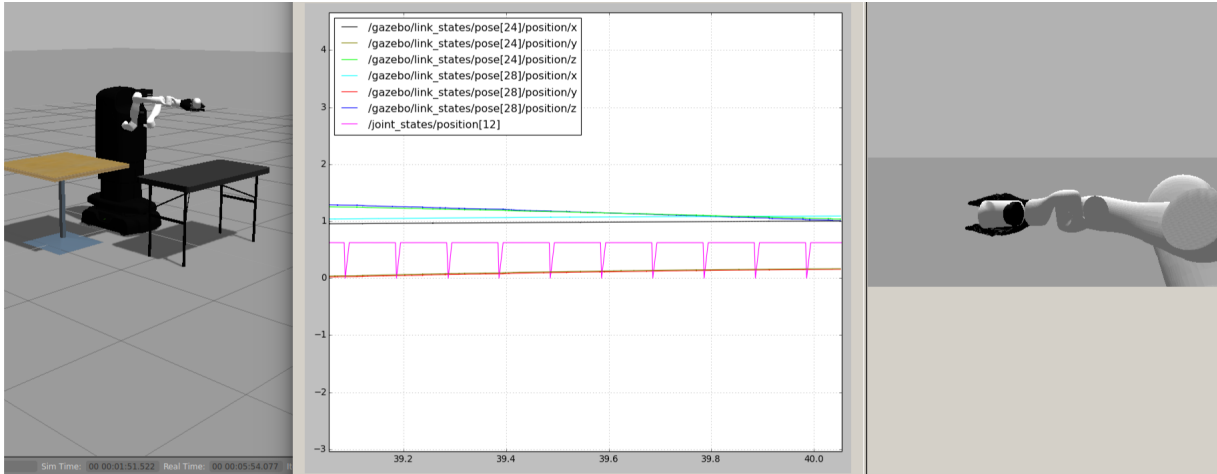
The designed planning algorithms that implement Stages 1 and 2 of the previous section are tested via simulation to analyse their capabilities of displacing one or multiple objects using multiple arms. The simulations are being performed under Matlab and ROS Rviz/Gazebo environments, which can also be used as a real experimental environment to test scenarios multiple robot arms sharing the same workspace. In this section, we consider 2 different setups: (i) single link, planar- 3, 4 and 6 robots cases, (ii) a dual arm robot (Kinova-Movo), whose specifications can be found in [82]. Figures 3.5 and 3.4 are representing the setup (i), where the link length is 0.3 m and the joint angle range is  $[-\pi, \pi]$  for each robot. Note that arms, tables and the object are the only possible collision objects. For setup (ii), we used following design parameters:

$$\begin{aligned} o_{init} &= [0.9 \quad 0.5 \quad 0.75], \\ o_{goal} &= [0.4 \quad -0.5 \quad 0.85], \\ movo_{homed} &= [-1.5, -0.2, -0.15, -2.0, 2.0, -1.24, -1.1, \\ &1.5, 0.2, 0.15, 2.0, -2.0, 1.24, 1.1, 0.35, 0, 0], \\ table_1 &= [0 \quad -0.8 \quad 84], \\ table_2 &= [1.1 \quad 0 \quad 0.74], \end{aligned}$$

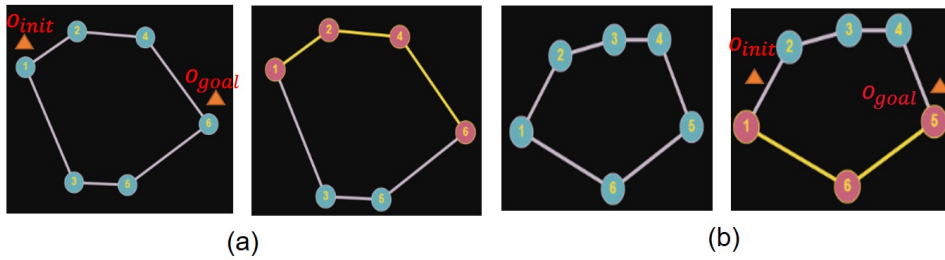
In this scenario, Kinova-Movo, 2 tables and a cylindrical object are the only collision sources.

In Figure 3.4 (a), there exist 6 single link robots with SSG representation and initial and goal location location of an object. Figure 3.4 (b) illustrates the optimal number of the robot and their identity based on SSG,  $o_{init}$  and  $o_{goal}$  and Dijkstra's algorithm. In this scene, robots 1, 6 and 5 number robots are responsible to bring the object from  $o_{init}$  to  $o_{goal}$ . *Scheme 1* in Figure 3.5 indicates that any set  $S$  that includes  $K$  arms has to move through  $o_{init}$ ,  $K - 1$  *MPS*, and  $o_{goal}$  and  $K$  arm's shared and independent working regions, respectively. In Figure 3.5 *Scheme 2*, we see that two tasks, carrying the objects,  $o_{1init} = [-0.2 \quad 0.8]$  and  $o_{2init} = [-0.2 \quad -0.15]$  to  $o_{1goal} = [0.6 \quad 0.8]$  and  $o_{2goal} = [0.6 \quad -0.15]$ , can be handled by robots  $R_1$ ,  $R_2$  and  $R_3$  and  $R_4$  in parallel, respectively. Here, when we calculate the path for the first object, robots  $R_3$  and  $R_4$  are ignored. Similarly, when we calculate a path for second object from  $o_{init}$  to  $o_{goal}$ , robots  $R_3$  and  $R_4$  are ignored. Similarly, in *Scheme 1*, workable regions for each robot is represented as follows:

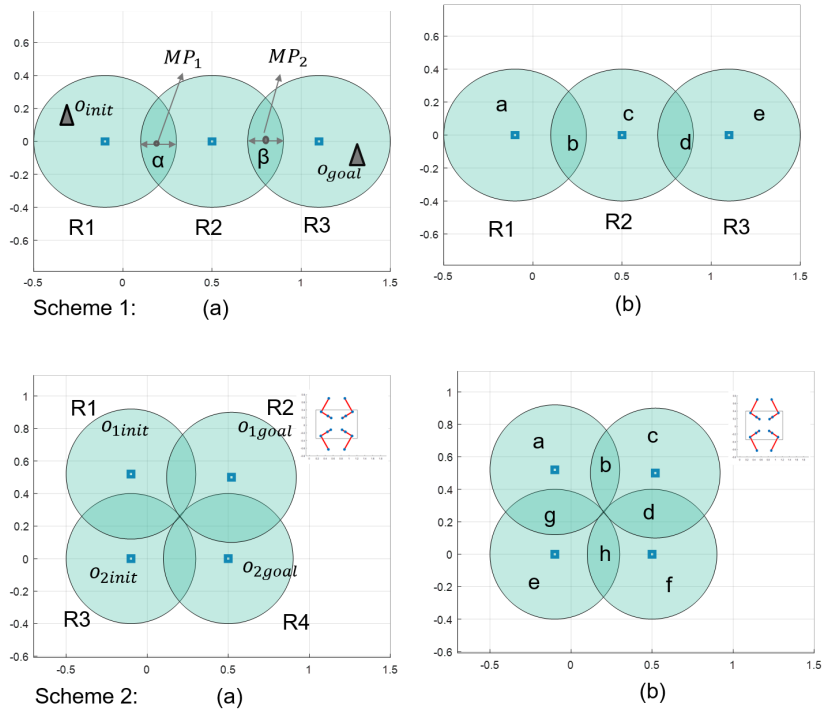
$$\begin{aligned} a &= R_1 \overline{R_2 R_3}, b = R_1 R_2 \overline{R_3}, c = R_2 \overline{R_1 R_3}, \\ d &= \overline{R_1} R_2 R_3, e = \overline{R_1} \overline{R_2} R_3. \end{aligned} \tag{3.3.4}$$



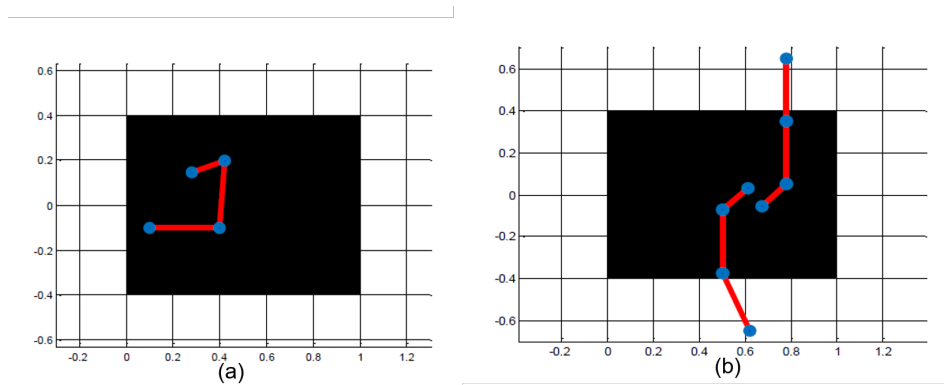
**Figure 3.3:** The object and right gripper's trajectory while object is moved from  $MP$  to  $o_{goal}$  where `gazebo/link_states/pose[24]`, `gazebo/link_states/pose[28]` and `joint_states/position[12]` are `movo::right_gripper_finger1`'s pose, `pringles_small::link`'s pose, and `right_gripper_finger1_joint`'s pose respectively.



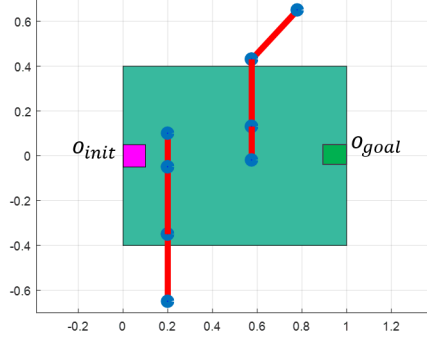
**Figure 3.4:** a) *SSG* representation based on shared workspace. b) The shortest path is found using Dijkstra.



**Figure 3.5:** Scheme 1: a) An example of MP representation for three robots, where R1, R2 and R3 indicate workspace of the each robot. b) The shared and individual working regions. Scheme 2: Parallel working principle based on shared and independent working areas



**Figure 3.6:** a) A 3-DOF single arm in 2D. Path planning time is around 70.19 seconds via  $A^*$  b) 2 arms, 3-DOF each, case. Planning time is more than 30 minutes



**Figure 3.7:** A setup consist of two arms with 3 DoF each. The first robot brings the  $o$  from its initial to a MP, and the second robot takes the  $o$  to  $o_{goal}$

where  $\overline{R_i}$  means the specific robots cannot work in the indicated region. Note that  $o_{init} = [-0.4 - 0.185]$  and  $o_{goal} = [1.40.15]$ .

When we consider there is not any research in the literature on the topology of mobile multi-robotic arms and manipulation graph representation based on shared workspace between arms to have an insight when some of the robots' shared space changed, the proposed representation is quite useful to extend the efforts on the topology of the mobile multi-arm systems. We define the robot parameters in Figure 5 (a) as follow;

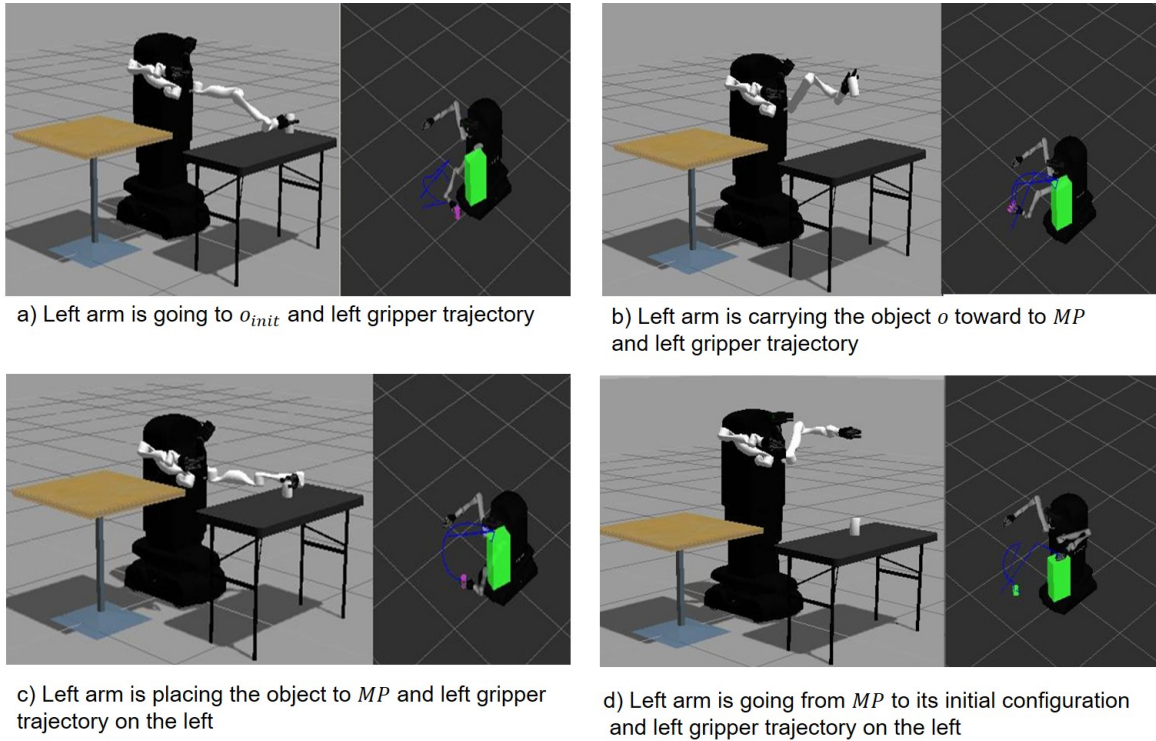
$$\begin{aligned}
 links\_length &= [0.3 \quad 0.3 \quad 0.15], \\
 joints\_lim &= [-\pi \quad \pi], \\
 arm1\_base\_pose &= [0.1 \quad -0.1 \quad -0.5], \\
 desired\_pose &= [0.3 \quad 0.3 \quad \pi], \\
 proximity &= 0.05, \\
 angle\_res &= 0.05.
 \end{aligned}$$

Similarly, robots' parameters for Figure 5 (b) are;

$$\begin{aligned}
 arm1\_base\_pose &= [0.68 \quad -0.65 \quad \pi/2], \\
 arm2\_base\_pose &= [0.78 \quad 0.65 \quad -\pi/2].
 \end{aligned}$$

Also, the proximity, resolution values, and initial and goal positions of the robots, for the  $A^*$  algorithm are 0.1, 0.1,  $o_{init} = [0.85, 0, 0]$ ,  $o_{goal} = [0.55, 0, 0]$ , respectively.

In Figure 6 (a), a single 3-DOF robotic arm with its initial and goal position is shown. We create a path using  $A^*$  and proposed SSG representation for the each arm and bring the



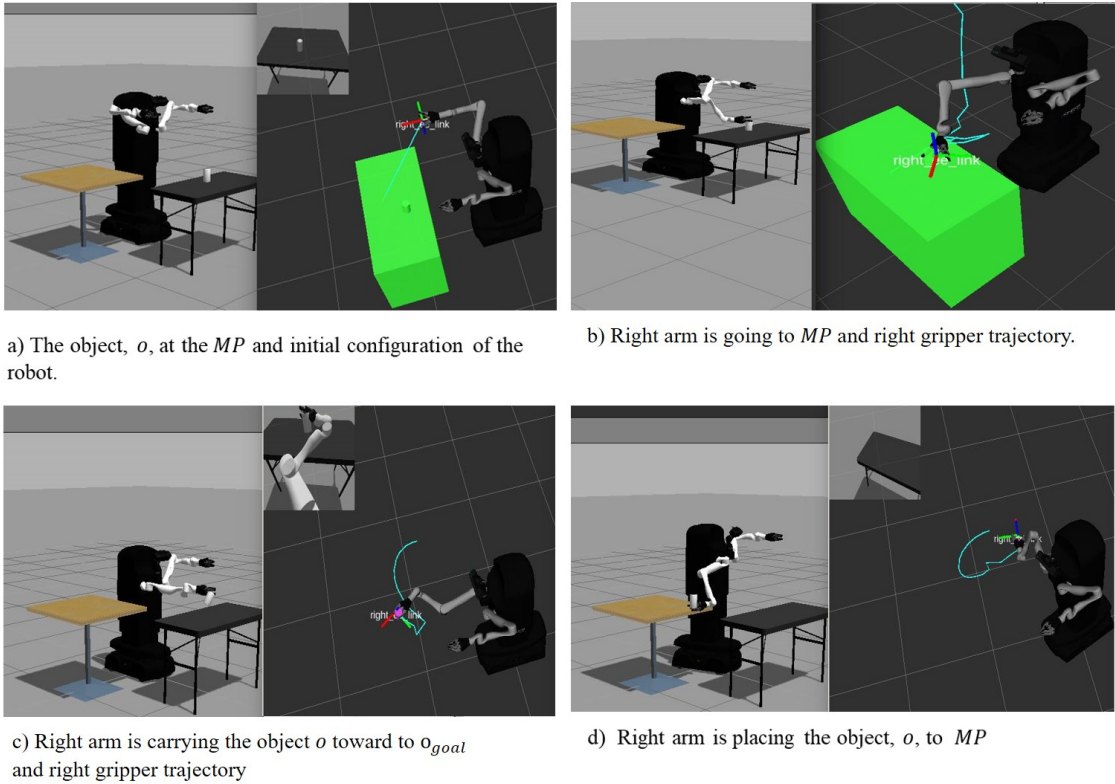
**Figure 3.8:** *Proposed scenario implementation using the Kinova Movo: The first Jaco arm brings the  $o$  from its initial to a meet point  $MP$ . Left side of the each figure are representing the Gazebo model and right side are showing RViz model created by mapping from gazebo model. Note that initial and goal location of the object detected by a kinect sensor on the movo's head.*

arms to their destination both in cartesian and joint space. The average planning time is 1 minutes 20 seconds in the one arm case. However, the planning time for two arms' case is more than 30 minutes as seen in the Figure 3.7. By applying our proposed algorithm with  $A^*$  for two arm case, the total planning time is around 3 minutes 10 seconds. In Figures 3.3, 3.8 and Figure 3.9, proposed scenario implementation using the Kinova Movo with a pair of 7-DOF 2-Jaco arms have been used to move an object from an initial location on the first table to another location on the second table. The first Jaco arm brings the  $o$  from its initial to a meet point  $MP$ , and ii) the second Jaco arm takes the  $o$  to  $o_{goal}$  by applying RRT based path planning algorithm in Gazebo/ROS environment. Note that the initial pose of the object is found by a kinect sensor's data, and the goal pose are given to the planner. The planner returns a sequence of trajectories, labeled with which arms they are for. Note that the initial state of all of the arms is for them to be in their

safe configurations. The whole planning is taken a maximum of 2 minutes to compute a solution. It is because we plan each arm path individually (6 DOF) rather than planning a path with two arm (6x6 DOF).

As it is illustrated in the Fig. 3.8, the dark blue path is the left Jaco arm’s gripper’s trajectory while the arm is bringing the object from its initial location to its meet point. Also, the light blue path in the Figure 3.9 is representing the right Jaco arm’s gripper’s trajectory while the right arm taking the object from its MP to final location. In the Figure 3.9, on the Rviz model which is placed on the right hand side of the figure, left top corner is demonstrating the kinect-camera’s view from the head of the Kinova-Movo. We get the initial location data of the object from the camera. Since the object’s initial location is inside the left arm’s workspace, left arm is responsible to take the object to the meet point and similarly, a given final location of the object is inside the right arm’s workspace, so second arm is responsible for carrying the object its final location. In this scenario, both arm is taking the role of placing the object from its initial to goal. Since the planning process for the each arm is calculated once at a time, the planning time is more efficient than planning the paths for the both arm at the same time. Also, simulation results indicate that third dimension and increased DOF is more complex to solve and increase the computation time for the planning process as seen in Fig. 3.8, and 3.9. This scenario can be easily extended more than two arms’ cases. As previously discussed, any primitive task is obligated to a single arm pick and place operation no matter the number of arms in the set. An interesting case emerges when only a few arms obstruct the path of the *transfer* arm. This notion simplifies the solution by planning with less DoF which drastically reduces computation time.

All experiments were performed on a computer with an Intel i7 CPU (2.8Ghz), 16GB of RAM, running Ubuntu 14.04. Note that we assumed all robots in the environments have fixed-bases. In this simulation, we use RRTStar to plan all of the required paths. RRTStar is one of the most common and fastest sampling based methods used today. We use the highly optimized implementation found in the OMPL [83]. Given that this method is not limited to two arms but, we are only able to run it on the tabletop scene with two arms’ case in this paper.



**Figure 3.9:** *Proposed scenario implementation using the Kinova Movo: The second Jaco arm brings the  $o$  from its meet point  $MP$  to goal. Left side of the each figure are representing the Gazebo model and right side are showing RViz model created by mapping from gazebo model. Note that initial location of the object detected by a kinect sensor on the movo's head.*

### 3.4 Summary

In this paper, we have developed an integrated motion and task planning algorithm for multi-arm robot systems in a shared workspace. We have utilized SSG idea not only to decide, among a certain number of robots within a cooperative multi-arm robotic system, which ones are the best fit for a given task, but also to distinguish the regions based on independent working areas and cooperation areas. Simulation results verify that SSG-MP based sequence and motion planning not only helps to find optimal arm sequences for a given task, and generate the corresponding optimal trajectories, but also decreases the planning time dramatically for search based planning algorithms.

However, some limitations exist in the techniques. First, in the case of a non-empty meet point, the algorithm should find an alternative. This problem is considered as future work.

Also, we consider that pre-defined grasps are available for every defined task. However, if there exists a failure from the task planning layer regarding grasping, our algorithm cannot generate an alternative grasping.

In the next chapter, the proposed technique will be extended for cases involving more than two robot arms with mobile and fix bases. Moreover, we will consider the problem of cluttered environment. The proposed SSG representation reduces the dimensionality of the problem in defining the tasks and motion paths for each robot in the workspace.

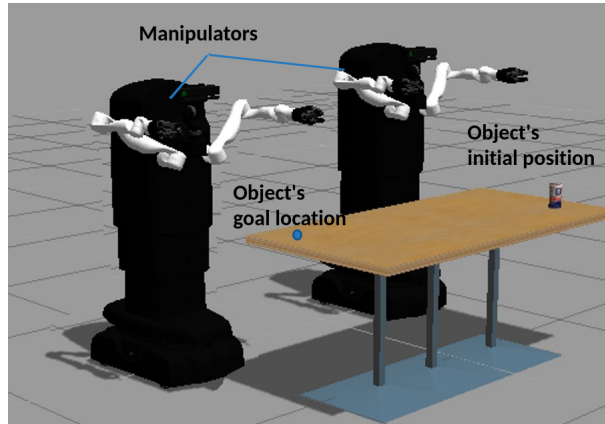


# Chapter 4

## Vision-guided Dynamic-Shared Space Graph Based Manipulation Planning

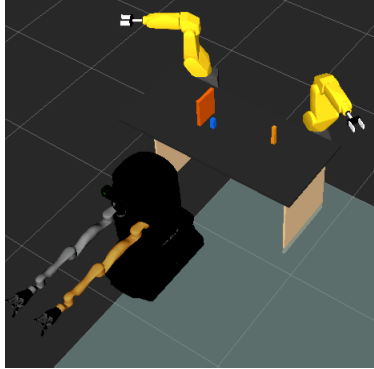
### 4.1 Introduction

Multi-robot manipulation tasks can be highly challenging to plan and execute correctly, especially in the presence of a mobile manipulator in the environment. In other words, the main challenge in this is to find a feasible plan to solve a given task under uncertainties, e.g., the exact location of the object to be manipulated may not be accurately known or some object properties such as geometry may not be known by the robots at the start of the task. To deal with uncertainties while planning robot(s) motion in the literature, generally robots planning algorithms search a sequence of actions and apply a replanning process in the case of failure or uncertain situations. This process of motion replanning can be computationally expensive and may not satisfy real time operation requirements. In the previous chapter, which is based on our work in [18], we proposed a shared space graph based integration algorithm for multi-robot manipulation problem. However, that approach assumes no change in the base locations of the robot arms or change in the environment (structured) during task(s) execution. The main objective of the task and motion planning algorithm proposed in [84] and their extensions in this chapter is to solve the problem of picking and placing one or more objects using a system of multiple robotic arms, including mobile and fixed based robots, aided by visual sensing. The proposed planning approach in this chapter is able to create alternative solutions which can effectively perform motion and task planning when robots base locations change throughout the task or uncertainties associated with location of obstacles in the workplace exist at the onset task execution.



**Figure 4.1:** *An example with  $n = 4$  arms, where each arm has a fixed base. In this setup, no single arm is able to bring the object from its initial location to the target region*

Significant progress has been made in recent years in advancing the state of the art in mobile manipulation, with versatile manipulation platforms which can work collaboratively in human-centric environments such as such as the Kinova-Movo (see Fig. 1), PR2, Intel HERB Personal Robot, ICub, the TUM Rosie robot, the HRP2, ARMAR, and Justin. Newer systems such as the Kinova-Movo have integrated 3D sensors to enable identification of changes in the location of objects of interest within the robots work environment. However, cooperation between mobile and fixed based robots in hybrid scenarios similar to the one depicted in Figure 4.2 have not been studied much in the literature. In Figure 4.3, we introduce the framework of the proposed integration algorithm for task and motion planning when multi-robots similar to the ones in Fig. 4.2 share the same manipulation workspace. We present higher level task planning layer in PDDL format. Reasoning about the world- robots, objects and obstacles situations in the world- situation is provided to task, motion and our integration algorithm as an input. In the case of failure of a specific primitive task, such task is introduced as an input to our integration algorithm. The integration algorithm provides robot-arm sequence and waypoints (meet points) for that specific task. These waypoints are then given to the motion planning layer which in response generates new motion plans utilizing this information. If the new plan can be successfully executed, it is translated to the task planning layer before moving on the next scheduled task. For the collision avoidance part of motion planning module, we use open source collision avoidance libraries. Details on the design of each block of the proposed framework in Figure 4.3 are provided in the following sections.



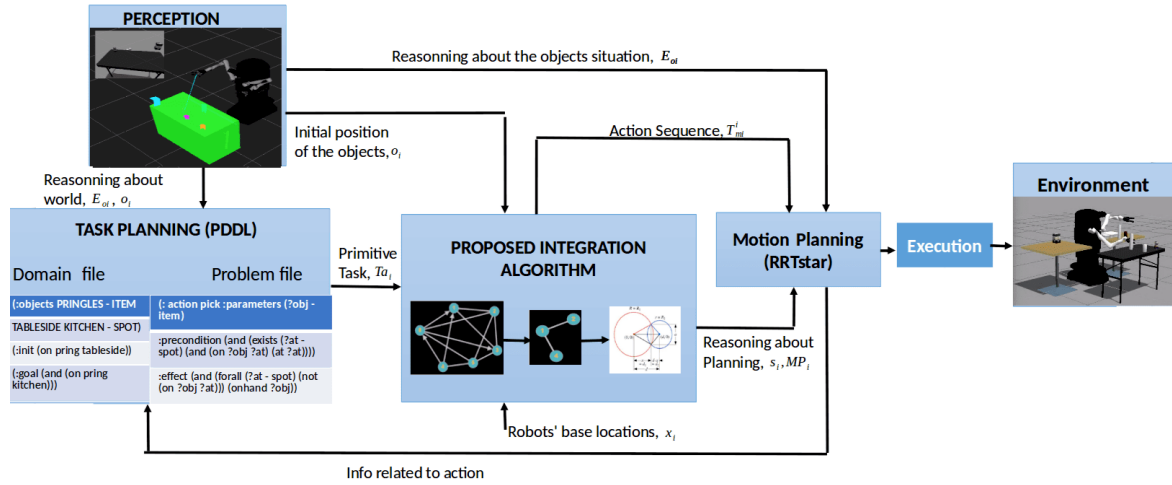
**Figure 4.2:** *An example of mobile and fixed base robots in a shared environment. In this setup, we have a kinova-movo, mobile based robots, and two Fanuc LRMate 200ic, fixed-base arms.*

## 4.2 Algorithm Development

Problem 2, described in Chapter 2, includes both sequence planning and motion planning steps. Hence, we introduce a two layer solution; 1) An integration layer following high level task planning layer which aims to find a series of arms, where the object can be passed from one arm to the next such that the last arm is able to place the object at it's target location (in case of failure of a primitive task). Here, a dynamic shared space graph, D-SSG, representation is introduced to provide the topology of interaction of fixed or mobile-based multi-arm systems. The D-SSG determines a pair of arms within the workspace where handover of the object being manipulated can take place from one to the other. The D-SSG with the proposed meet point, MP, representation, which is located in the shared workspace of any two adjacent arms, is used to find a sequence of arms, as well as the initial and final location of each path for each arm involved in the execution of this given task. 2) A low level motion planning layer, D-SSG and MP attributes is used to calculate a collision free trajectory for each step of the sequence.

Based on Problem 2 described earlier, we consider both selection (lines 1-4 of Algorithm 2 - page 54) and motion planning attributes Algorithm 2(lines 5-6). For each attribute, a formal approach is followed. Our planning approach is guided by the principle of integrating path set  $\mathcal{M}(k)$  and arm sequence  $S(k)$  and their interpretation for primitive tasks. Hence, the algorithm is broken down two stages where Stage 1 maps elements of  $\mathcal{M}$  and elements of  $S$ , and Stage 2 generates each corresponding task, as well as plans the arms' motions.

- Stage 1 = Step 1 (Finding arm sequence) + Step 2 (Finding meet points (MPs)) : Obtain a set of arms which can transfer an object from initial to goal positions. The



**Figure 4.3:** *The implementation process of the algorithm*

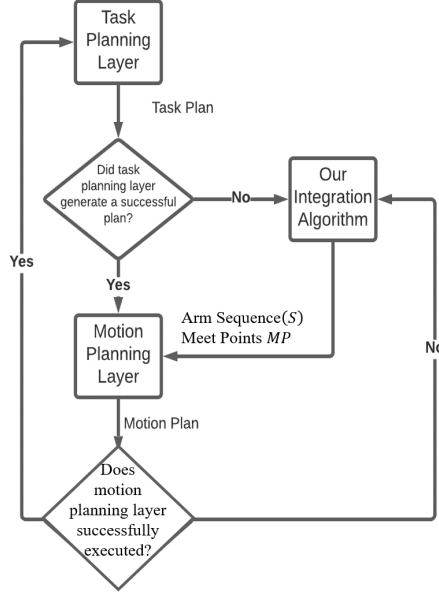
main purpose of this stage is to obtain a series of arms where the object is transferred from one arm in the series to the next such that the last arm is able to place the object at its goal location.

- Stage 2 = Step 3: Calculate a collision free trajectory for each pair of arms at each step in the series identified in Stage 1.

In the following subsections, details of the two algorithms stages are provided.

### 4.2.1 Stage 1: Multi Arm Sequence Planning

A *manipulation-path* is defined by some primitive tasks, which are transferred from task planning layer, such as pick and place operations. In our integration algorithm, we divide each failed primitive task to further subtasks before it is introduced to the motion planning layer. Therefore, the term primitive task is used to denote the  $i$ 'th procedure in  $\mathcal{M}$ . To perform each primitive task individually, the problem is simplified to a single robotic arm pick and place operation. Still, there is no guarantee of completion of the whole task unless a binding condition is defined which links the sequential primitive tasks. Such a criterion is described by the proposed *meet point* definition. Therefore, we define the positions of the *meet point series*. Any *meet point* is located in the shared work space of any two successive arms in  $\mathcal{S}$  [18].



**Figure 4.4:** Flow diagram of overall process.

### Arm Selection Based on Graph Search

Let  $D-SSG = (V, E)$  be a connected, undirected, simple graph. Here,  $V = \{v_1, v_2, \dots, v_n\}$  the graph's vertex set, and each edge in  $E$  has form of  $(v_i, v_j)$ , where  $i, j = 1, 2, \dots, n$ . Let  $\mathcal{S} = \{s_1, s_2, \dots, s_k, \dots, s_n\}$  be a set of robots with base position  $x_i$ , and sum of the links length  $l_i$  for each  $s_i$ , and initial location  $o_{init}$  and goal location  $o_{goal}$  on SSG given by the injective maps  $o_{init}, o_{goal} : SV$ , respectively. Each node  $v_i \in V$  represents the workspace of a particular arm  $s_i$  and each edge  $(v_i, v_j) \in E$  represents a non-empty common workspace shared by arms  $s_i$  and  $s_j$ . The domains of the base position  $x_i$  are changing in discrete time step  $k$ . Therefore,  $SSG$  become a dynamic graph  $SSG(k)$ , where edges might appeared or disappear between any two nodes based on the state of  $x_i(k)$ .

**Definition 4.2.1** (Node construction). Shared workspace between each robot in  $S$  makes vertices connected ( $E = (v_i, v_j)$ )

$$e_{ij}(k) = \begin{cases} (v_i, v_j), & \text{if } d_{ij}(k) < l_i + l_j \\ \text{undefined}, & \text{otherwise} \end{cases}$$

where  $d_{ij}(k) = \|x_i(k) - x_j(k)\|$ ,  $i, j = 1, 2, \dots, n$  and  $i \neq j$ .

**Definition 4.2.2** (*Initial arm set discovery*). Let  $o_{init}$  and  $o_{goal}$  be initial and goal locations of the object, which needs to be transport by two or more  $s_i$ , respectively.

$$ArmSet_{init} = \begin{cases} v_i, & \text{if } \|x_i - o_{init}\| < l_i \\ \text{undefined}, & \text{otherwise} \end{cases}$$

**Definition 4.2.3** (*Goal arm set discovery*). Let  $o_{init}$  and  $o_{goal}$  be initial and goal locations of the object, which needs to be transport by two or more  $s_i$ , respectively.

$$ArmSet_{goal} = \begin{cases} v_j, & \text{if } \|x_j - o_{goal}\| < l_j \\ \text{undefined}, & \text{otherwise} \end{cases}$$

**Definition 4.2.4** (*Shortest path*). A shortest path in an unweighted graph is the path with least number of edges (*Breadth First Search*). For each edge  $(v_i, v_j) \in E$ , if  $v_j$  is unvisited;

$$dist_{shortest}(v_j) = dist_{shortest}(v_i) + 1 \quad (4.2.1)$$

## Arm Selection with Environmental Obstacles

Consider the same environment given previous subsection above with some environmental obstacles  $E_o = \{E_{o1}, E_{o2}, \dots, E_{ol}\}$  with their locations  $e_p = \{x_p, y_p, z_p\}$ , where  $p = 1, \dots, l$  and obstacles might appeared or disappeared under any arm  $S_i$  workspace.

**Definition 4.2.5** (*Cost construction*). Obstacles under robot  $S_i$  makes vertices connected ( $E = (v_i, v_j)$ )

$$c_{ij}(k) = \begin{cases} c_{ij}(k) + 1, & \text{if } d_{ip}(k) < l_i \\ \text{undefined}, & \text{otherwise} \end{cases}$$

where  $d_{ip}(k) = \|x_i(k) - e_p(k)\|$ ,  $i = 1, 2, \dots, n$  and  $i \neq p$ .

**Definition 4.2.6** (*Initial arm set discovery*). Let  $o_{init}$  and  $o_{goal}$  be initial and goal locations of the object, which needs to be transport by two or more  $s_i$ , respectively.

**Definition 4.2.7** (*Shortest path*.) A shortest path in a weighted graph is found by Dijkstra. For each edge  $(v_i, v_j) \in E$ , if  $v_j$  is unvisited;

$$dist_{shortest}(v_j) = dist_{shortest}(v_i) + c_{ij} \quad (4.2.2)$$

## 4.2.2 Stage 2: Multi Arm Motion Planning

In this section, we introduce our proposed solution to calculate a collision-free *manipulation-path* for each arm. Still, a single arm motion planner is not applicable in a multi-arm scenario and a multi-arm motion planner solving for a large number of arms is highly complex. In our problem, the overall goal is to obtain a multi-arm trajectory for each of the primitive tasks extracted by the proposed algorithm (Algorithm 2). Note that initial configuration of the each arm in  $\mathcal{S}$  and the transferring arm index are known. A RRTstar based path planning algorithm is applied to the MaM environment, which includes a dual arm robot as specified in the following section. In the following algorithm, as one can see, RRT searches for a path from the initial configuration to the goal configuration by expand a search tree (details of RRTstar can be found in the Appendix A). For each step, the algorithm determines a target configuration and expands the tree towards it. The target can either be a random configuration or the goal configuration itself, depending on the probability value defined by the user. During the search tree evolution,, the algorithm only needs to determine if each step is collision free but does not need to plan for avoiding obstacles [36, 85].

Algorithm 2 shows the main procedure to generate a solution. Here, lines 1-4 represent the first stage of the algorithm which calculates the transfer arms and meet points and lines 5-6 represents the motion planning and assignment of the tasks to each robot respectively. Line 4 is finding optimal arm sequence by considering initial and goal workspace of the robots, as calculated in lines 2-3, and using SSG graph based on Dijkstra 's algorithm explained in [81].

### Minimization problem for finding MP poses for passing an object from one arm to another

A common task that requires manipulation by two or more robots from  $S_i$  is to move an item from one arm to the next in minimal time for any initial poses of each end-effector. To solve this complex task, it broken down into three steps:

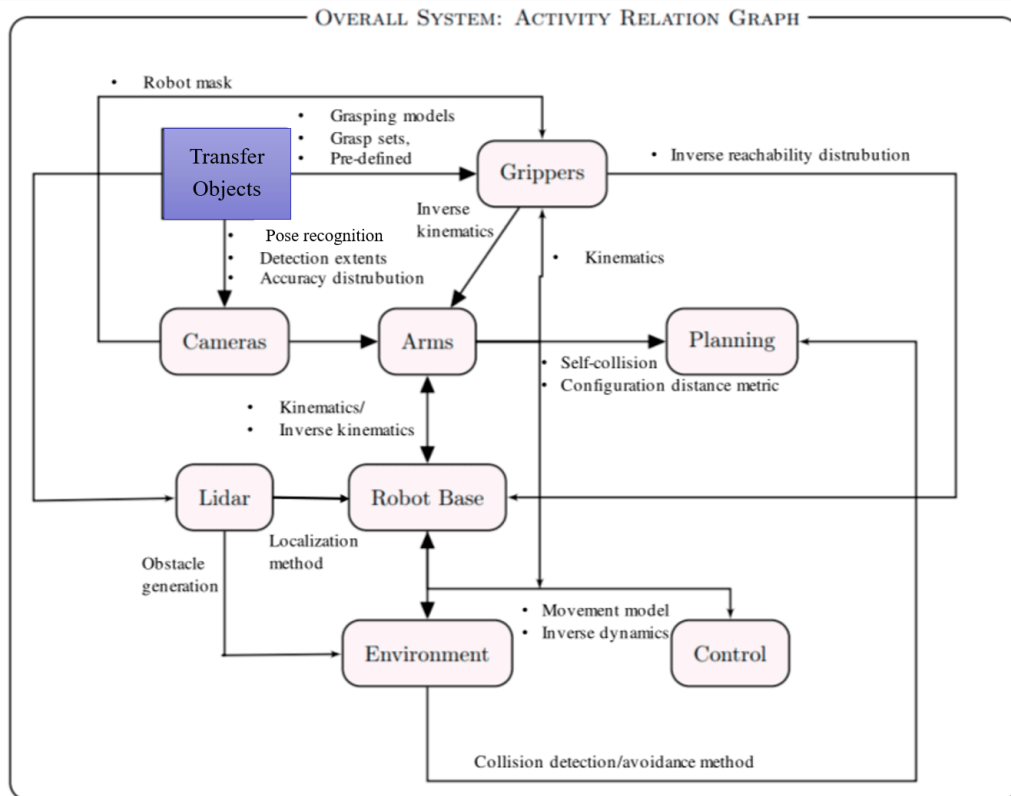


Figure 4.5: Roadmap of the motion planning layer.

- Minimization problem for finding MP poses for each gripper, which has to be at a specific distance and rotation from another gripper
- Using robot path planning tools to generate collision-free paths for both arms from initial to MP pose.
- Performing the first and second steps above repeatedly to find the path that requires minimum time to execute.

As mentioned above, the first step in solving a minimization problem for finding an MP pose for each gripper. The `scipy.optimize.minimize()` function is used to solve this problem, using Sequential Least-Squares Programming, SLSQP method, ( further details can be found at [86]).

Instead optimizing only gripper pose, the minimization problem is solved in the domain of joint angles ( $q$ ). The pose used is a vector of six components, position in the  $x, y, z$



---

**Algorithm 2** Multi Arm Planning Integration

---

**Require:**  $o_{init}, o_{goal}, \mathcal{A}$ **Ensure:**  $\mathcal{M}$ 

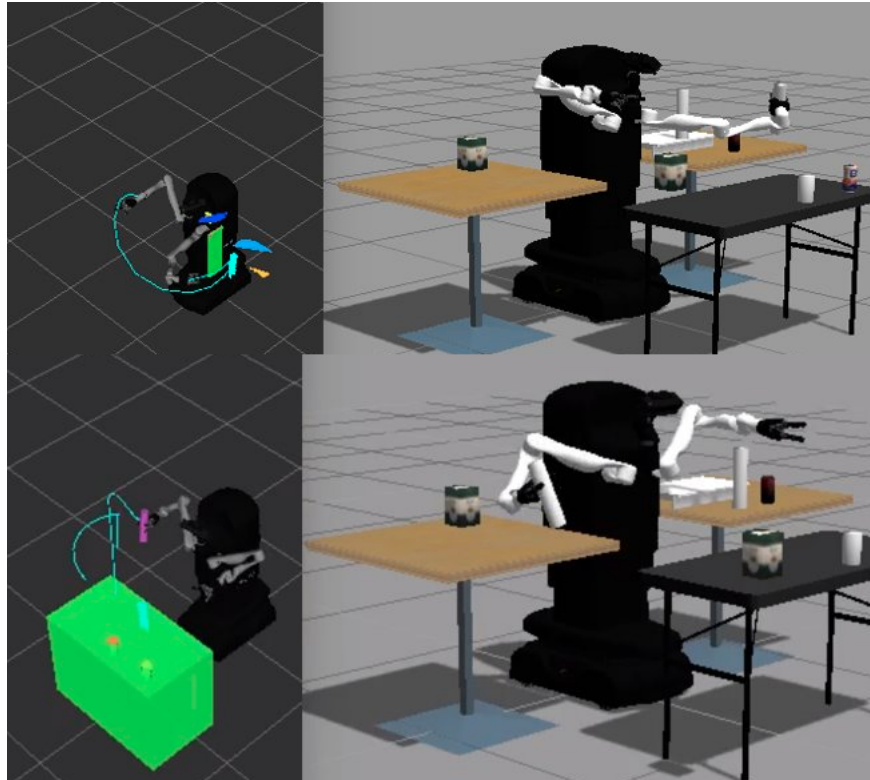
- 1: **for**  $k \leftarrow 1$  to  $N$  **do**
  - 2:    $D - SSG(k) \leftarrow D - SSG(\mathcal{S})$
  - 3:    $s(1) \leftarrow \text{obtain}(s(1) \text{ such that } o_{init} \in W_{f(1)})$
  - 4:    $s(K) \leftarrow \text{obtain}(s(K) \text{ such that } o_{goal} \in W_{f(K)})$
  - 5:    $\mathcal{S}(k) \leftarrow \text{Dijkstra}(SSG(k), s(1), s(K))$
  - 6:    $M(k) \leftarrow \text{RRTstar}(\mathcal{S}, o_{init}, o_{goal}, MP(k))$
  - 7:    $\mathcal{T}(k) \leftarrow \text{PrimitiveTasksSequence}(M(k), \mathcal{S}(k), MP(k))$
  - 8: **end for**
  - 9: **return**  $\mathcal{T}(k)$
- 

directions, and XYZ Euler angles, more commonly called the roll, pitch, yaw. We use the norm between the current end-effector pose and the corresponding end-effector pose as an objective function for the set of  $q$ 's obtained by the minimization for each gripper. We use two functions in the objective function, respectively, (1) calculate the norm and (2) return the end-effector's pose with the current set of  $q$ 's. Inside the norm function, another function,  $JS_{toP(q, 'limb')}$  is called. This uses KDLKinematics from *pykdlutils* to solve forward kinematics for the set of joint angles, returning the rotation and translation components that describe the end pose [87].

Also, rotational limits for each joint, can be obtained in the Unified Robotics Description Format (URDF) file, and constraints to satisfy both grippers to stay at a specific distance from each other are give to the minimization function.

### 4.3 Simulation Results and Testing

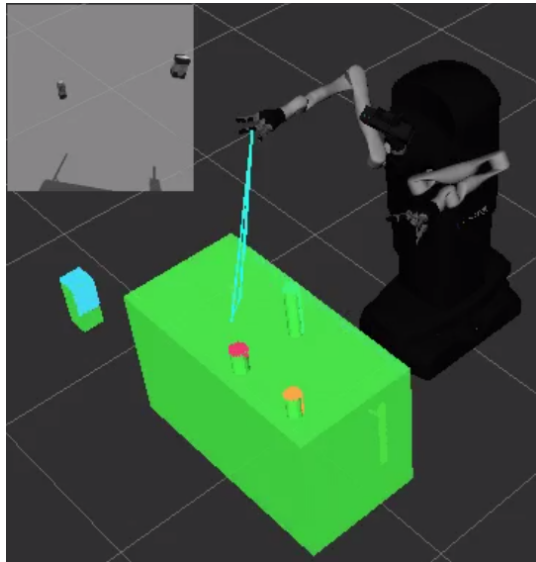
In this section, we evaluate the proposed planning algorithm that applies Stages 1 and 2 of described in section 4.2 via simulation. The simulations are being performed under ROS Rviz/Gazebo environments. Here, we consider 2 different setups: (i) a dual arm robot (Kinova-Movo) and ii) four arm robots environment, includes 2 fixed (Lrmate 200ic, Fanuc) and 2 mobile arms arm (Kinova-Movo)- robots specifications can be found in Appendix A. In the first scenario, Kinova-Movo, 3 tables and cylindrical and rectangle objects on the tables are described as part of the workspace as shown in Figure 4.6. In the second scenarios, 2 fanuc arms, Kinova, Movo, a table and cylindrical and rectangle objects on the tables are the collision sources as seen in Figure 5.11 and Figure 5.14 respectively.



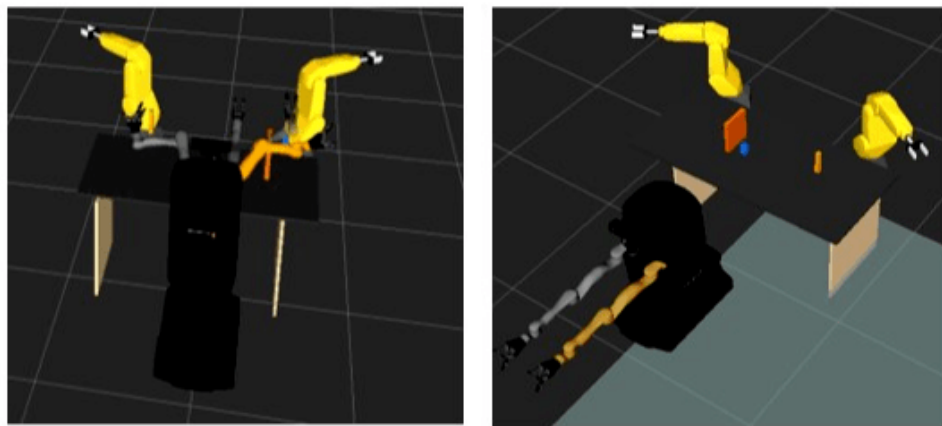
**Figure 4.6:** *Proposed scenario implementation using the Kinova Movo: The second Jaco arm brings the o from its initial to goal position under clustered environment.*

The simulation scenarios are implemented using the Kinova Movo with a pair of 7-DOF 2-Jaco arms (Fig. 5.11) and 6-DOF 2-Fanuc arms and a mobile based Kinova-Movo robot in Fig. 5.14 to move an object from an initial location to a goal location. In each of those scenarios, the task cannot be accomplished only one arm. In the both cases, the first Jaco arm brings the  $o$  from its initial to a meet point  $MP$ , and ii) the second Jaco arm takes the  $o$  to  $o_{goal}$  by applying RRTstar based path planning algorithm in Gazebo/ROS environment, after a systematically created SSG search and deciding  $S_i$  for particularly given task. As seen Fig. 5.14, when the base location of the arm changes, the SSG graph is changing simultaneously as well. In the Fig 4.6, sub-figures a) with c) and a) with d) are associated. Note that the initial pose of the object is found by a kinect sensor's data, and the goal pose are given to the planner . The planner returns a sequence of trajectories, labeled with which arms they are for. Note that the initial state of all of the arms is for them to be in their safe configurations. The whole planning is taken a maximum of 2 minutes to compute a solution.

Our ROS-perception methodology begins by building a predictable model of the environment using input from kinect sensor as seen in Figure 5.6. The environment is represented using a mix of pre-created mesh models for known objects and a voxel-based representation for the other parts of the environment. The environment representation serves as input to motion planners that generate collision-free, smooth motions for the robot. Fast, real-time trajectory controllers are used to perform the planned trajectories on the robot. Grasp action of objects consists of selecting an appropriate grasp from a set of possible grasps actions computed a priori.

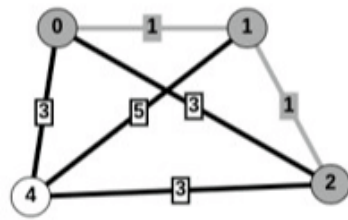


**Figure 4.7:** Color and shape features from the objects that were sitting on the table are extracted using object segmentation on 3D point cloud data from gazebo world.

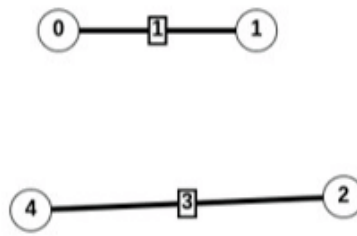


a)

b)



c)



d)

**Figure 4.8:** Proposed scenario implementation using the 2 Fanuc arms and a Kinova Movo

All simulations were performed on a computer with an Intel i7 CPU (2.8Ghz), 16GB of RAM, running Ubuntu 14.04. Note that we assumed all robots in the environments have the fixed-bases. In this simulation, we use RRTStar to plan all of the required paths. RRTStar is one of the most common and fastest sampling based methods used today. We use the highly optimized implementation found in the OMPL reference to paper [83] which describes the RRTstar algorithm.

## 4.4 Summary

In this chapter, an integrated motion and task planning algorithm for multi-arm robot systems in a shared workspace under heavy obstacles is introduced. D-SSG idea have been applied not only to determine, among a certain number of robots within a cooperative multi-arm robotic system, which ones are the best fit for a given task, but also to distinguish the regions based on independent working areas and cooperation areas. Simulation results verify that D-SSG-MP based sequence and motion planning not only helps to find optimal arm sequences for a given task, and generate the corresponding optimal trajectories, but also eliminate the failure of the given task in the case of robot base location change or environmental obstacles change. In the case of environmental changes, the algorithm is able to find alternate plans, which is helpful to the task planning layer in long term task success.

However, our algorithm is vulnerable in very cluttered environments. Our algorithm may not find an arm sequence and a proper meet point due to obstacles in these cases, such as the amazon grasping challenge. Also, SSG based planning cannot solve some problems if the robot cannot find the manipulation object, such as if the object in a drawer. One can combine our method with one of the available literature solutions for solving such a problem in those scenarios.

Another limitation is that the weights on the SSG are calculated based on the number of objects under each arm's workspace. However, considering the volumes of the objects can give a better path regarding a collision-free trajectory. This problem will be handled as future work.

# Chapter 5

## Integrated Task and Motion planning with Dynamic Shared Space Graph

### 5.1 Introduction

Solving robotic manipulation problems such as preparing a pasta dish or setting the table with a different type of item must have a planning system that can find a sequence of actions around a feasible path. However, these kinds of problems are more challenging in uncertain environments, especially in the MaM's environment, such as each robot's reachability changes in time.

Several versions of PRM have been proposed, such as manipulation graph by the authors of [46], or multimodal planning techniques [55, 62] to solve the manipulation planning problems. However, those approaches cannot deal well with the dimensionality to manipulation problems with many objects. Another possible approach is Task and Motion planning (TAMP), that searches for a discrete sequence of symbolic actions around a motion plan for each of them. However, the discrete formation of manipulation tasks, such as pick and place, is over defined in the symbolic domain, that decrease the complexity of the problem. The main disadvantage of TAMP is to avoid calling motion planner for unfeasible actions, especially for constraint manipulation planning problems.

There exist two primary approaches in the literature to solve integrating high level symbolic task planning and low-level motion planning problems; simultaneously [17, 63], interleaved [16, 88]. Simultaneous approaches consider geometric data by calling a motion planner while task planning is running. However, the interleaved methods decouple motion planning from the task planner. First, task planning is pursued, then a motion planning layer is called to evaluate the feasibility of the plan. If the plan fails, geometric constraints

are fed into the task planner, and the same steps are followed until a feasible plan is found. The integration of task and motion planning layers is complicated when the MaMs is in uncertain environments, and robot's and objects' locations are subject to change in time. These challenges are the subject of the present chapter.

In the previous two chapters (Chapter 3, 4), we have considered the manipulation planning problem as a single layer manipulation planning problem. In contrast, each robot's path planning was considered individually for a specific task based on SSG search for MaM in both structured and semi-structured environments. This chapter combines the aforementioned single layer planner through an SSG search with a high-level symbolic task planner. In other words, we combine the mentioned single layer planning approach with the multi-layer manipulation planner approach. This approach is helpful since it creates alternate plans in case a task or subtask fails due to some geometrical change in the environment.

Based on the Problem 3 defined in Chapter 2, first, we have created a "library" of basic motions for the robots. To integrate task planning and motion planning, the geometric information needs to be translated into the task planner that checks the motions' feasibility and creates a plan. This step is done by the PDDL that models the motions as geometric prerequisites and effects in order that the task planner checks the feasibility. It is necessary to model the actions according to the problem we are going to solve. As our purpose is to create an algorithm capable of solving the maximum variety of planning problems, we have to model it, including all the possible conditions. Although we include all possible conditions into the task planning problem, if a minor change happened in the environment, such as a change in the robot's or the object's base location or an appearance of a new obstacle in the environment, both task and motion plans fail. In this case, the specific task or subtask is translated into our dynamic SSG-MP based interface layer. This section explains in detail how the actions have been modelled in PDDL and how the failure is handled in semi-structured environments.

First, we start describing the predicates and the type of objects, which define the problems we are going to solve. Since we have computed the actions such as move base, move the arm, pick and place, we can solve planning problems that consider moving objects from a place to another. Solving other kinds of problems like cooking pasta, doing the dishes, etc. requires extra actions as press the accelerator and/or requires unique modelling as, for example, *is\_dirt?* dishes. Our problems have to be as real and general as possible, so we have to include blocking situations. If we want to solve problems in which objects are hidden within a drawer, we must create the motion open/close a drawer and model it on PDDL.

## 5.2 Combining Task and Motion Planning Through Our Interface Layer

The relationship between the high-level PDDL actions and low-level control has been established through our translation layer, which is instantiated as separate node. The Planning System will dispatch PDDL actions using a message type defined by ROSPLAN to the intermediate layer, where the translation can be carried out. Furthermore, ROSPLAN includes several modules, similar to those provided by the Knowledge Base, that correspond to common actions found in robotics domains. In this chapter, we combine our integration algorithm presented in the previous chapters (Chapter 3, and 4) with a PDDL based task planner. To take advantage of pre-existing work on the integrated motion and task planning and make the communication easier between different layers and nodes, we integrated Algorithm 2 with the algorithm used in [89] as follows: The Planning System

---

### Algorithm 3 Integrated SSG-ROSPlan

---

```

procedure PROCEDURE PLAN DISPATCH(Domain D,Task T)
  while  $T$  contains goals do do
3:   I:=generateProblem(D,T);
   P:=plan(D,I);
   F:=constructFilter(D,I,P);
6:   while execute do do
   subtask:=pop(P);
   dispatch(subtask);
9:   while actionExecuting do do
   if filterViolated then then
   execute :=  $\perp$ ;
12:   call := (MultiArmPlanningIntegration);
   end if
   end while
15:   execute := execute
   actionSuccess(subtask);
   end while
18: end while
end procedure

```

---

constructs a PDDL problem instance, sends this to an external PDDL planner, extracts a filter from the plan, dispatches the actions, and handles re-planning. To generate the



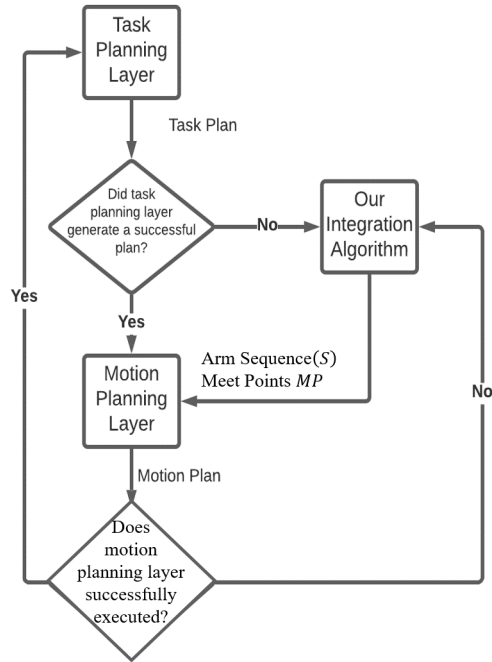
PDDL problem (line3 of Algorithm 3), the Planning System parses the supplied domain file, then queries the Knowledge Base for the object instances, facts, and fluents that comprise the initial state also the current goals. This problem file is then handed to a PDDL planner (line4), which produces a plan. ROSPLAN can be used with any planner, as long as it can handle the domain’s syntactic requirements. In our case study, we use an any-time version of POPF, a temporal and numeric planner. The amount of time allowed for planning is provided as a parameter to the node. Once the plan has been found, it can be validated using the Plan Validation Tool (VAL), which is included in ROSPLAN. The filter is constructed (line5) by taking the intersection of static facts in the problem instance with the union of all preconditions of actions in the plan. The action is dispatched by linking the PDDL action to a ROS action message (lines7-8). In ROSPLAN, re-planning depends on the reformulation of the problem. There are three reasons that the system may require a re-plan:

1. the current action fails;
- 2 the Knowledge Base notifies the planner of a change that cancels the plan or of new data critical to the goal;
3. the current action, or plan, has gone significantly over or under budget (such as time or energy).

In the case of (1), the dispatch loop is broken (line 15), the problem instance is re-generated based on the current model of the environment, and a new plan is produced. In the case of (2) or (3), the current action may still be executing. In these situations, the initial plan is re-validated, and then, if obtained to be not valid, the current action is cancelled, and the dispatch loop is broken as before (lines10-13)

### 5.3 Implementation Framework

The proposed approach for task and motion planning, shown in Fig. 4.3 consists of our interface layer, a task planning module and a motion planning module. Since different programming languages and libraries are associated with planners and modules, one challenging issue is ensure task planning, integration, and motion planning layers communicate with each other. This can be establish through Robotic Operation System (ROS) communication layer. We define the task planning layer as client node and call services for different geometric reasoning process. When task and motion planning is successfully terminated, the complete manipulation plan is stored in the task planning layer, needs



**Figure 5.1:** *Flow diagram of overall process.*

to be executed in the real world or in the simulation. The task executive layer is used to deal with executing the plan by a robot—moreover, the sensing information required to monitor the result of the sensing actions in case of uncertainty. Regarding perception, Kinect camera is employed using ROS driver (<http://wiki.ros.org/kinect>). For performing a different type of low-level geometric reasoning and motion planning, open-source robotics libraries such as OMPL, Moveit etc. are used. Motion planning problems are described using XML, such as robots, obstacles, controls and planners. The main parameters are given by a description file, which includes limits of the motion if the robot has a mobile base, home position wrt world reference frame. A robot is defined as a kinematic tree with an optional mobile base. The kinematic structure of the robot is defined using the URDF (<http://wiki.ros.org/urdf>). It includes the visual robot model, the collision model, the transformation between links, joints and dynamic parameters such as damping and masses. The visualization model is defined with triangular meshes that can be represented in .wrl, .stl, .dae formats. The collision model can be represented either as a triangular mesh or by primitive shapes such as box, cylinder or sphere. At the beginning the PDDL contains the objects of the environment, the position of the objects, the arm of the robots,

and the initial position for the base and the arms. As initial conditions, we consider where the objects and the robots are located, and also that the grippers.

This section presents some details on the implementation of the proposed framework to solve the integrated motion and task planning problem. Also, it explains how the interface layer updates the geometric information when the motion planning layer cannot execute the action, or the task planner cannot obtain a solution, as seen in Fig. 5.1. If the solver cannot obtain a solution, it is because the problem is unsolvable or because position data around robots or objects are missing or wrong due to uncertain environment.

## 5.4 Algorithm Analysis

In this section, we demonstrate the completeness properties of our algorithm.

Given:

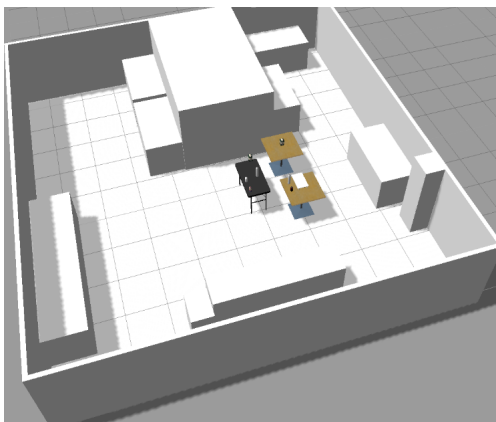
- The task planner (FF-Solver) alone is complete [54].
- The RRTstar motion planner is probabilistically complete [21], so given sufficient time, the probability of finding a plan, if one exists, approaches one.
- The Dijkstra algorithm is complete [90]

**Definition 5.4:** A set of actions is uniform regarding a goal in  $G$  and a set of predicates  $R$  if for every  $r \in R$ ,

**Theorem 5.4:** D-SSG based integrated task and motion planning is probabilistically complete.

Let  $P = (O, I, G)$  be a planning problem where  $O$  is the set of actions, and  $I$  (the initial state) and  $G$  (the goals) are sets of atoms such that there are no reachable dead-end states w.r.t.  $G$  and  $O$ 's descriptions are sound w.r.t. discrete effects and uniform w.r.t. the  $G$  and the geometric predicates used in the domain. **Assumption:**  $P$  is terminated or cannot be solved due to some change  $I$  or  $R$ .

Let  $p$  be the pose generator for the pose references used in  $I$  and  $G$ . If all the calls to the motion planner terminate, and  $P$  fail, then Alg. 1 will find a sequence of arms and corresponding motion plans for solving  $P$  if one exists using the SSG and graph search by calling the Dijkstra algorithm. Since the combination of Dijkstra,  $I$ ,  $G$  and the motion planner (RRTstar) is probabilistically complete (given sufficient time, the probability of finding a feasible pose instantiation when there exists one approaches one), then Alg. 1 and 2 will eventually terminate with a task and motion plan solving  $P$  if there exists one.



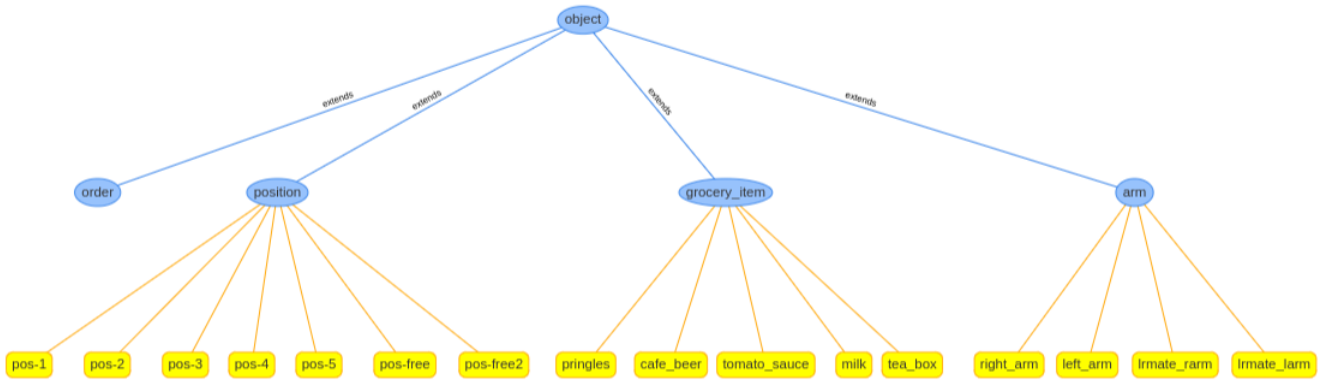
**Figure 5.2:** *Gazebo world of Uwaterloo Robohub is considered as grocery setup.*

The result follows that a D-SSG update occurs at every time step under the premises, and changed geometric facts are also updated accordingly in D-SSG in solving  $P$ . Our empirical evaluation also demonstrates the algorithm achieving in several tasks that do not satisfy the uniformity condition.

## 5.5 Simulation Results and Testing

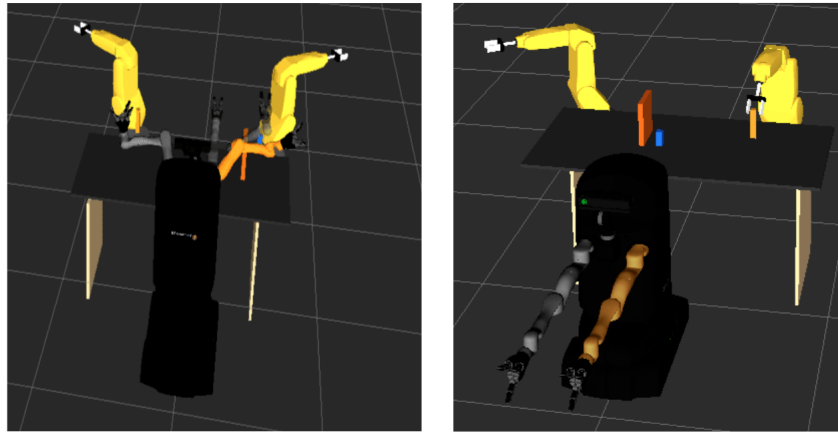
In this section, we analyze the efficiency of the integration planning algorithm through some simulation studies to evaluate its ability to displace one or multiple objects using multiple arms in case of failure in the motion or task planning layer of a specific problem. The simulations are being performed under ROS Rviz/Gazebo environments. Here, we consider two different case studies: (i) a dual-arm robot (Kinova-Movo) grocery packing setup and ii) a four-arm robot grocery packing setup that includes two fixed (Lrmate 200ic, Fanuc) and two mobile arms arm (Kinova-Movo). In the first scenario, Kinova-Movo, three tables and cylindrical and rectangle objects on the tables. In the second scenario, 2 Fanuc arms, Kinova-Movo, a table and cylindrical and rectangle objects(Fig 5.3) on the tables are the collision sources, as seen in Figure 5.11 and Figure 5.14 respectively.

Kinova-Movo [82] is a mobile robotic platform provided with two 7 DOF (degrees of freedom) arms, a Kinect and a 2D planar laser sensor. The Kinect for Xbox One is a time-of-flight sensor that gives both RGB images and distance measurements. Kinova-Movo is also coming with two Intel NUC5I7RYH computers. It is ready to use with Robot Operating System (ROS) and its motion planning framework MoveIt. All simulations and testing are performed on a computer with an Intel i7 CPU (2.8Ghz), 16GB of RAM,



**Figure 5.3:** *Objects of the world for the grocery store example(ii).*

running Ubuntu 14.04. In this simulation, we use RRTStar to plan all of the required paths. RRTStar is one of the most common and fastest sampling based methods used today. We use the highly optimized implementation found in the OMPL [83].



a) The initial task is assigned to movo right arm

b) Due to robot's base location change, the task is automatically assigned to the left fanuc arm

**Figure 5.5:** *Proposed scenario implementation using the Kinova Movo and 2 Fanuc LRMate 200ic robot: The left Fanuc arm brings the object from its initial to goal position.*

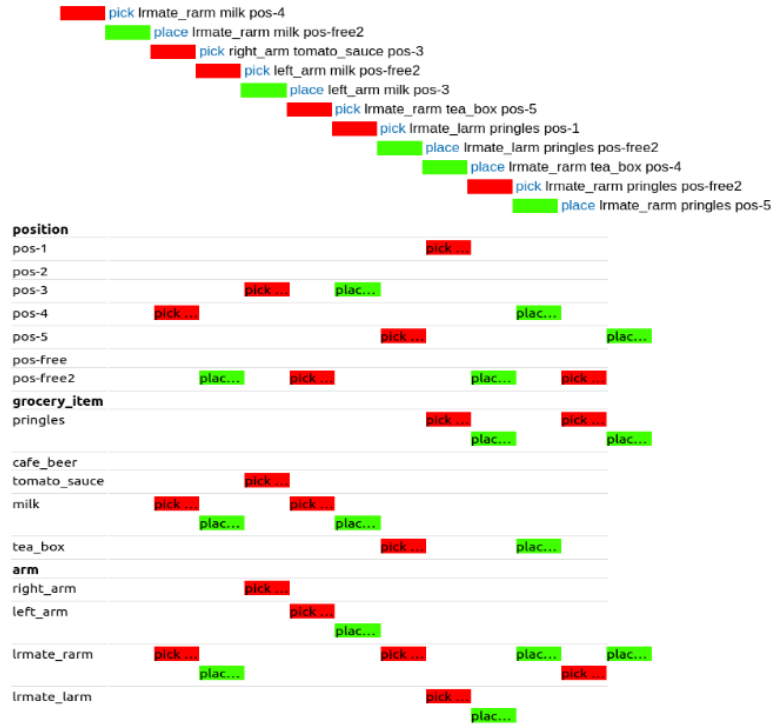


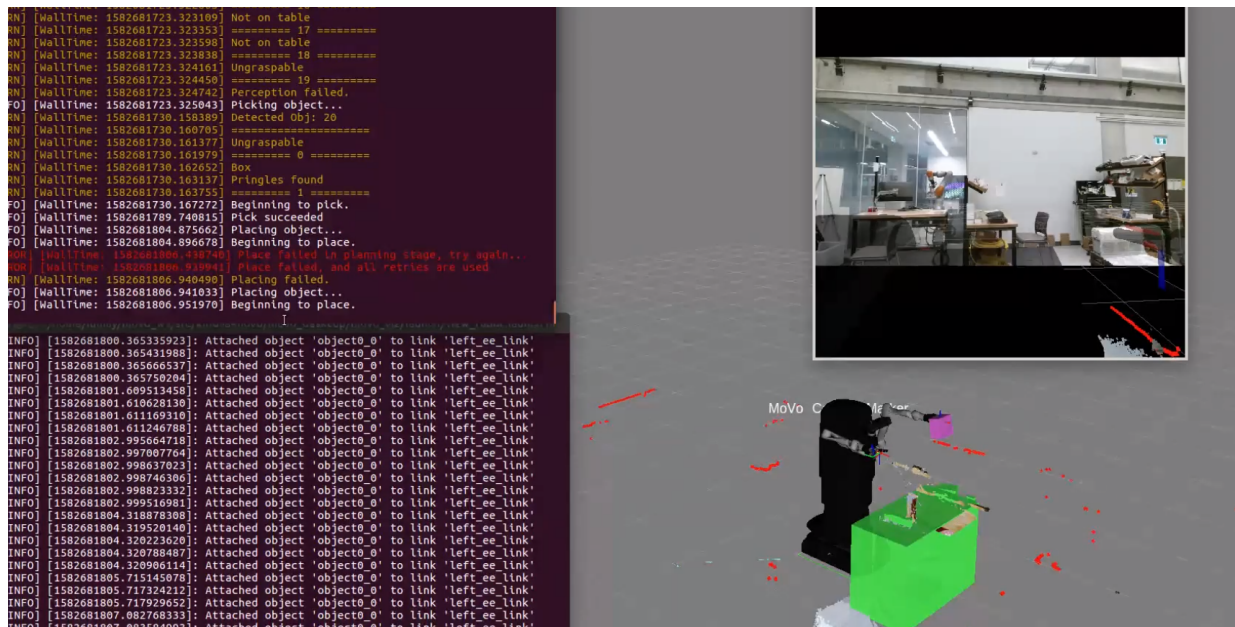
Figure 5.6: Output of movo-lrmate grocery setup.

The proposed scenarios implemented using the Kinova-Movo with a pair of 7-DOF 2-Jaco arms (Fig. 5.8) and 6-DOF 2-Fanuc arms and a mobile based Kinova-Movo robot (Fig. 5.4) to move an object from an initial location to a goal location. The proposed method is evaluated based on some clutter environments where the robot needs to sort objects. Fig 5.4, and 5.6 (namely case (i) and case (ii)) summarise the problems, initial states and desired goals, the number of objects in the environment and each object reachability situation by each arm, respectively. For the first case, the goal is to place the pringles in pos-free2 location. As shown in Fig. 5.7, the task planner returns with a plan to be executed. However, when the pringle’s initial location is changed to another location, as seen in Fig 5.8, the planers fail to find a solution. In both scenarios, the task and motion planning layers fail to find a solution since the object’s initial location (pringles) is changed and not reachable by the corresponding arm anymore as seen in Fig 5.7. Hence the specific problem is translated to our integration layer to solve it. Figure 5.9 depicts the successful execution results after the failed action is translated to integration layer.

In the the first setup, first Jaco arm brings the  $o$  from its initial to a meet point  $MP$ , and then the second Jaco arm takes the  $o$  to  $o_{goal}$  by applying RRTstar based path planning

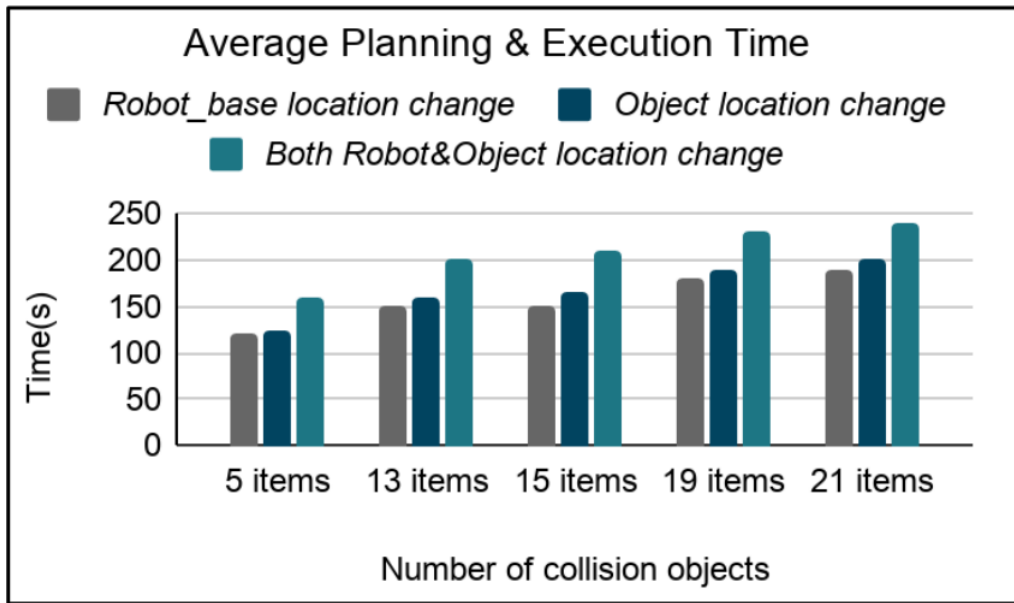
algorithm in Gazebo/ROS environment, after a systematically creating SSG search and deciding the robot ID for particularly translated problem. As seen Fig. 5.14, when base location of the arm is changed, the SSG graph is changed simultaneously as well. Note that the initial pose of the object is found by a kinect sensor's data, and the goal pose are given to the planner. The planner returns a sequence of trajectories, labeled with which arms they are for. Note that the initial state of all of the arms is to be in their safe configurations.

Our ROS-perception methodology begins by building a predictable model of the environment using input from kinect sensor as seen in Figure 5.6. To perform a scan for logos, Movo pans the Kinect over the scene and passes the RGB frames to the object detection CNN, which processes frames at a rate of 3 Hz. The environment is represented using a mix of pre-created mesh models for known objects and a voxel-based representation for the other parts of the environment. The environment representation serves as input to motion planners that generate collision-free, smooth motions for the robot. Fast, real-time trajectory controllers are used to perform the planned trajectories on the robot. Grasp selection for objects consists of selection of an appropriate grasp from a set of grasps pre-computed offline.



**Figure 5.12:** Proposed scenario experimentation using the Kinova Movo in Robohub environment

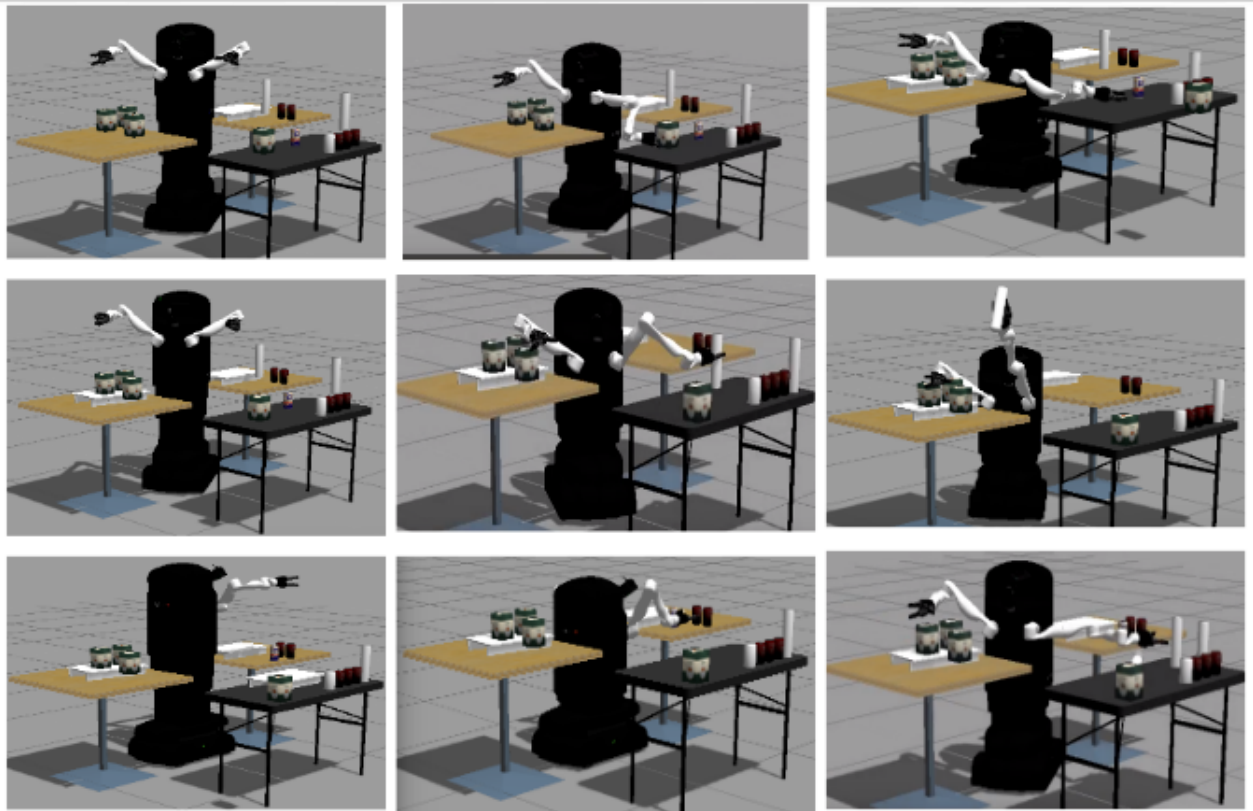
We demonstrate the summarized results for three different environmental changes, including the instances object, robot and simultaneous object and robot location change conditions for the different cluttered environment scenarios, in Fig. 5.14. For each case, average planning and execution time are calculated based on ten randomly generated simulation problems. To the best of our knowledge, no other method has been demonstrated performance at this level on randomly created constrained problems without utilizing specific geometric reasoning methods.



**Figure 5.13:** *Planning and execution times for problems solved within different types of environmental changes and different number of collision objects in the cluttered table domains. For each case, 10 different pick and place simulation studies are generated.*

In Fig. 5.15, the objective is to pick and place a target object from a cluttered table. In the figure, only cases of some of the objects' locations, or only robot base location or object and robot base location simultaneously changes. However, those changes are not reflected in the higher symbolic task planning layer. Instead, those handle on our integration layer. We also tested the integration layer's ability by increasing the number of objects up to 11 on three tables of fixed dimensions.





**Figure 5.14:** *Some of the pick-place tasks executed while solving an instance of the different number of objects cluttered table with the environmental changes.*

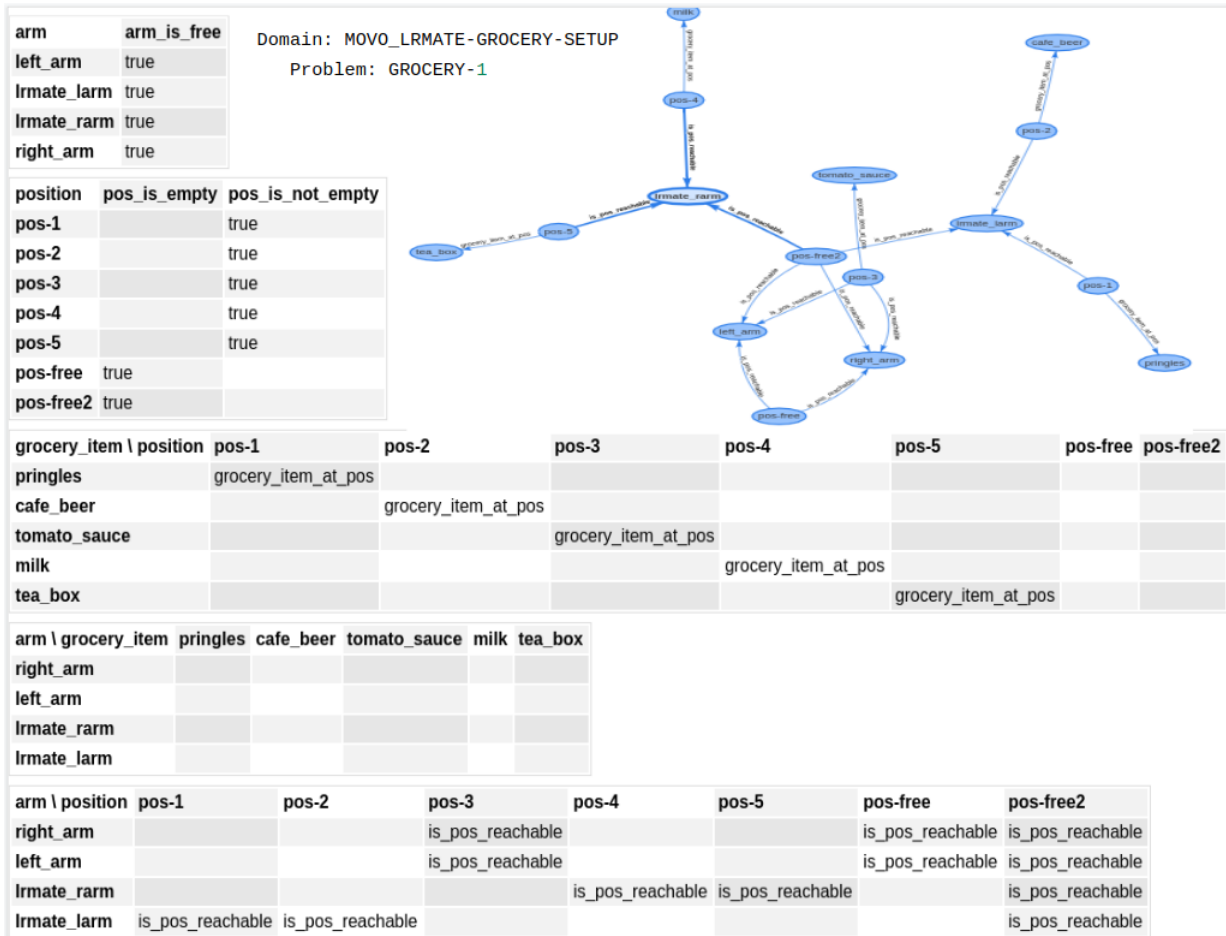


Figure 5.4: Case study(ii): The setup of the proposed grocery store example with Kinova-Movo and Fanuc-LRmate200ic arms.

```

(define (problem grocery-2)
  (:domain movo_lrmate-grocery-setup)
  Show hierarchy
  (:objects
    right_arm left_arm lrmate_rarm lrmate_larm - arm
    pringles cafe_beer tomato_sauce milk tea_box - grocery_item
    pos-1 pos-2 pos-3 pos-4 pos-5 pos-free pos-free2 - position)
  View
  (:init
    (arm_is_free right_arm)
    (arm_is_free left_arm)
    (arm_is_free lrmate_rarm)
    (arm_is_free lrmate_larm)
    (pos_is_empty pos-free)
    (pos_is_empty pos-free2)
    (pos_is_not_empty pos-1)
    (pos_is_not_empty pos-2)
    (pos_is_not_empty pos-3)
    (pos_is_not_empty pos-4)
    (pos_is_not_empty pos-5)
    (grocery_item_at_pos pos-1 pringles)
    (grocery_item_at_pos pos-2 cafe_beer)
    (grocery_item_at_pos pos-3 tomato_sauce)
    (grocery_item_at_pos pos-4 milk)
    (grocery_item_at_pos pos-5 tea_box)
    (is_pos_reachable pos-1 lrmate_larm)
    (is_pos_reachable pos-2 lrmate_larm)
    (is_pos_reachable pos-3 left_arm)
    (is_pos_reachable pos-3 right_arm)
    (is_pos_reachable pos-4 lrmate_rarm)
    (is_pos_reachable pos-5 lrmate_rarm)
    (is_pos_reachable pos-free right_arm)
    (is_pos_reachable pos-free left_arm)
    (is_pos_reachable pos-free2 right_arm)
    (is_pos_reachable pos-free2 left_arm)
    (is_pos_reachable pos-free2 lrmate_rarm)
    (is_pos_reachable pos-free2 lrmate_larm)
  )
  (:goal
    (and
      (grocery_item_at_pos pos-3 milk)
      (grocery_item_at_pos pos-5 cafe_beer)
      (grocery_item_at_pos pos-2 tomato_sauce))))

```

```

(define (domain movo_lrmate-grocery-setup)
  (:requirements :typing)
  Show hierarchy
  (:types order position grocery_item arm)
  (:predicates
    (arm_is_free ?a - arm) 1 1 1
    (pos_is_empty ?p - position) 1 1 1
    (pos_is_not_empty ?p - position) 1 1
    (grocery_item_at_pos ?p - position ?b - grocery_item) 1 1 1
    (arm_holds_grocery_item ?a - arm ?b - grocery_item) 1 1 1
    (is_pos_reachable ?p - position ?a - arm) 2)
  (:action pick
    :parameters (?a - arm ?b - grocery_item ?p - position)
    :precondition (and
      (arm_is_free ?a)
      (grocery_item_at_pos ?p ?b)
      (is_pos_reachable ?p ?a))
    :effect (and
      (not (arm_is_free ?a))
      (pos_is_empty ?p)
      (not (pos_is_not_empty ?p))
      (not (grocery_item_at_pos ?p ?b))
      (arm_holds_grocery_item ?a ?b)))
  (:action place
    :parameters (?a - arm ?b - grocery_item ?p - position)
    :precondition (and
      (arm_holds_grocery_item ?a ?b)
      (pos_is_empty ?p)
      (is_pos_reachable ?p ?a))
    :effect (and
      (not (arm_holds_grocery_item ?a ?b))
      (not (pos_is_empty ?p))
      (pos_is_not_empty ?p)
      (arm_is_free ?a)
      (grocery_item_at_pos ?p ?b))))

```

Figure 5.7: Problem and Domain generation for grocery store example(ii).

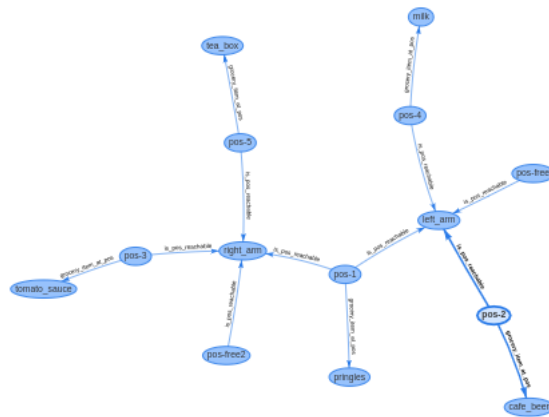


Figure 5.8: Initial reachability graph for case study (i).

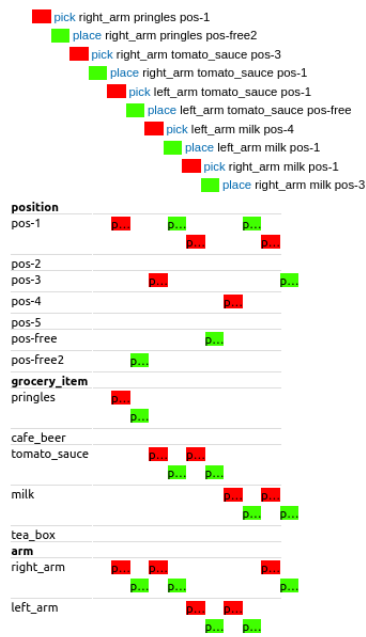
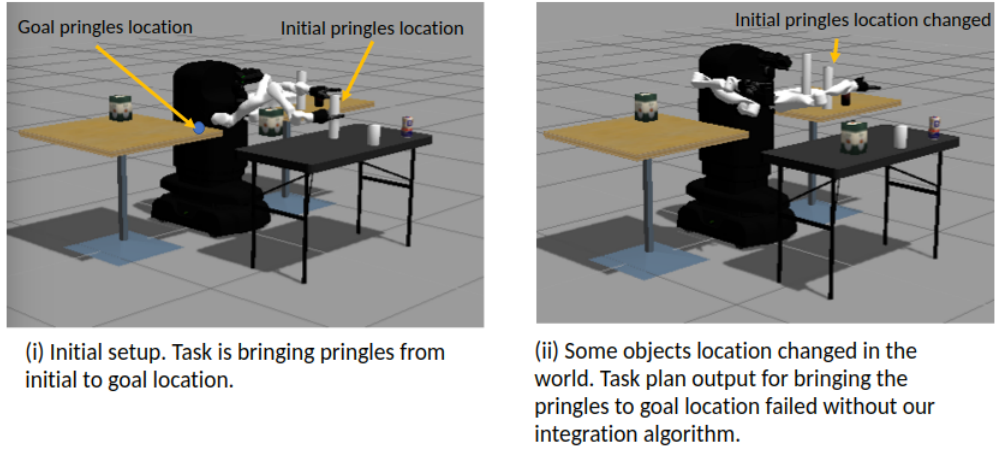
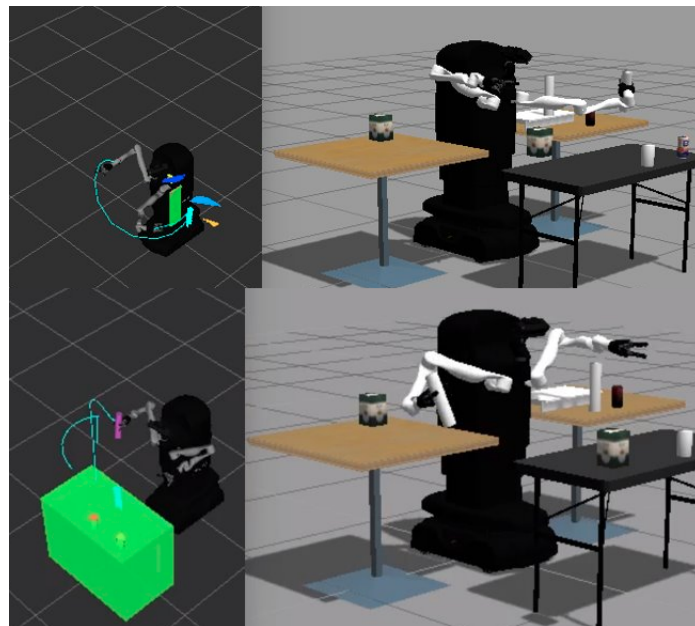


Figure 5.9: Output of case study(i) before environmental changes.



**Figure 5.10:** *Output of case study(i) without integration*



**Figure 5.11:** *Proposed scenario implementation using the Kinova Movo: The second Jaco arm brings the o from its initial to goal position under clustered environment.*

## 5.6 Summary

This chapter introduces a framework that combines high-level symbolic task planning and low-level motion planning, and is capable of solving non-trivial (either time consuming or difficult to accomplish) planning problems by just giving some goals in uncertain environments for MaMs. All the geometric information from the environment is modelled in a PDDL initially. Creating a single PDDL and giving an uncertain environment, the robot is capable of solving a variety of planning problems through our interface layer. The main novelty of this chapter is to integrate motion and task planning layers through an interface layer that is capable of handling these planning problems which includes robot(s) and/or object(s) location uncertainty.

“Sequence of Pick and Place” task in general is a good scenario for validation of the proposed algorithm. Because of that, these type of tasks was chosen since most of the manipulation problems also involve many pick-place sub-tasks. For instance, two arms will be involving in a collaborative mixing task in a pasta cooking scenario using our algorithm as shown in Fig. 6.2. In this case, we will be still using our algorithm to decide which arm will be responsible for a specific task and instead of performing sequence of pick-place tasks, we are performing a cooperative task using both arms at the calculated meet point.

Also, we have not tested the cases that include more than two mobile-based arms in the robot networks. One can further study those cases.

# Chapter 6

## Proposed System Design for Automated Kitchen Environment

In this chapter, we introduce a case study to implement the proposed contributions in Chapters 3-5 to design of a cooperative automated restaurant cooking robot system as part of an industrial collaborative research project. The main purpose of this research project is to develop an industrial automated dual-arm kitchen cooking environment for an industry partner. This project also enables us to introduce advanced MaM solutions to the food industry.

As we presented in the previous chapters (through Chapter 3-5), we first introduce a high-level symbolic task planner (Chapter 5) for various type of pasta dishes while the incoming order from a user is fed into the planning system. Then we create a library of basic motions for each actions defined in the task planning layer. However, only the proposed SSG approach presented in Chapter 3 is applied to the automated kitchen MaM system since the proposed kitchen environment is an overly structured environment and each arm tasks are defined in advanced. Therefore, we assume that only base locations of the robots changes and these inputs are provided to the algorithm by users in this case study.

### 6.1 Background and Literature Review

The history of robots and other automation technologies in the food industry goes back a few decades, mainly involved in palatalizing tasks; the so-called downstream applications. But as food industry giants continue converging and demand continues unabated from

large warehouse outlets, grocery chains and consumers for fresh products quickly, newly emerging configurations in robots with related automation technologies, including vision systems and processing software, are seeing robot applications move upstream for picking and packing [91, 92].

Food preparation using robotic machines has many challenges, such as accurate perception of ingredients in cluttered environments, manipulating objects, and engaging in complex activities such as mixing and chopping [10, 13, 93, 94]. Cooking can be categorized under three main tasks [95]

1. Perceive (vegetables, pans)
2. Decide (which recipe to use, what to do first)
3. Act (grab the apple, cut it)

Picking the right entry in a huge database, but perceiving and acting on ingredients is highly challenging since all ingredients, such as vegetables and fruits, have some similarities and differences regarding their shapes, colors and textures. While navigating in an environment, a vision system can recognize where it is and what the main objects in the scene are. Grasping a previously unknown object, whose 3D model is not available, is a challenging problem. Having a high level of manipulation of objects that are not variable at all is much easier. For instance, some industrial robots manipulate the objects with pre-defined and fixed shapes [76, 95, 96]. In study [5], BakeBot uses the low-level manipulation and perception system. The study [97] has demonstrated dispensing pancake batter from a premixed container and flipping the pancakes on a skillet.

Robot applications in food service industry can be categorized under kitchen operation and front staff operation [15]. A Chinese restaurant, Two Panda Deli, in Pasadena, California was one of the first to apply robotics for serving customers in 1983. They used two robots, Tanbo R1 and Tanbo R2. The robots were 4.5 feet tall and 180 pounds and would scoot around, bring trays of chow mein, spareribs and fortune cookies to customers tables [18]. FuA Men Restaurant in Japan utilized automated robots for preparing dishes. These robots use their manipulators to do tasks such as boiling water, pouring soup into bowls and placing the toppings on the dish. The two mechanical robots are working cooperatively to carry out the noodle preparation. Some benefit of using automated robotic arms compared to human chefs are precise movements in adding toppings and ingredients, consistency in taste and temperature, and accuracy of timing for food preparation. The sophisticated operation of the robotic arms in FuA-Men restaurant delivers efficient operation in the food preparation without intervention of a human chef. However there is little intervention of manual chef such as taking the order and serving the costumers [15]. Founders of Moley Robotics say that when the commercial version launches in 2017 users will be able to select one of 2,000 dishes from their phone and the robotic hands in the



automated kitchen will make it. According to the company, the user can select a dish from their phone, and the system will begin to prepare it. The robotic system does not use code to determine its action. Instead, to have high food quality, they try to match the level of skilled human chefs with the robotic system. This movement is achieved by using 3D camera to record a real chef preparing food and then convert the motions into algorithms to be recreated by the robot [10]. However, mentioned setup is not available today.

In the [98], the authors developed a mobile manipulator system which could pick an instructed object up, convey, and hand it to a person. However, these results achieved object manipulation in real environments, these technological elements are insufficient to apply to cooking tasks. The reasons arise from assumptions that conventional object manipulation by a robot has been assumed solid and distinguishable. However, foods such as vegetables can have various shape and appearance, and tools such as knife and bowl will not have rich texture. Moreover, the robot must recognize and manipulate several tools and foods at the same time. These facts make it difficult to achieve cooking task by an autonomous robot.

The Hierarchical Task Network (HTN) [10] describes the methods to achieve the high level task. Infact, it describes in a hierarchical way, all the possible choices that has the robot. The robot must use this data base to find a plan of primitive tasks representing the high-level recipe tasks. For that it will use the task planner. The primitive tasks have both a symbolic representation and a functional representation. The symbolic representation is used by the task planner and the functional one by the supervisor. The functional part of the primitive task is defined by a combination of modules of specific motion, motion planner, position generator, sensor actions and user interface. The supervisor has two fundamental roles. The first is to apply the action planned into the real world. The second is to report the real world modification into the state of the planners and then close the loop. For its first role, it is in charge of the execution of the primitive tasks by the use of their functional parts. The second role is to estimate the state of the system. Because we work in simulation it is not a difficult task, but in a real environment it is one of the most challenging aspects of the robotic system. It is also to ask the task planner to find a new plan in case of failure of the primitive task. It adds then a constraint to avoid the same choice.

The study in [99] introduces a hybrid planning approach for humanoid robotic cooking in a simulation environment. The authors implement a symbolic planner to plan sequences of motion primitives to execute a baking instruction and a separate geometric planner to execute the motion primitives. The hierarchical nature of the baking subtasks, each accomplished by motion primitives described in terms of their preconditions and effects, facilitates their future integration with hybrid symbolic and geometric planner.

## 6.2 Automated MaM Kitchen Environment

The process of using the robotic kitchen platform as seen is initiated with a set of ingredients, which are placed at certain locations within the workspace, and a set of instructions describing how to use those materials to cook certain dishes such as pasta. In Figure 6.1, the concept of the proposed robotic platform, i.e., robotic mini-kitchen for pasta cooking is demonstrated. The robot is provided an action sequence in the external world corresponding to the instructions. It then executes the action sequence on the robotic manipulation platform, performing the motion and task planning necessary to follow the recipe to create the appropriate dish.

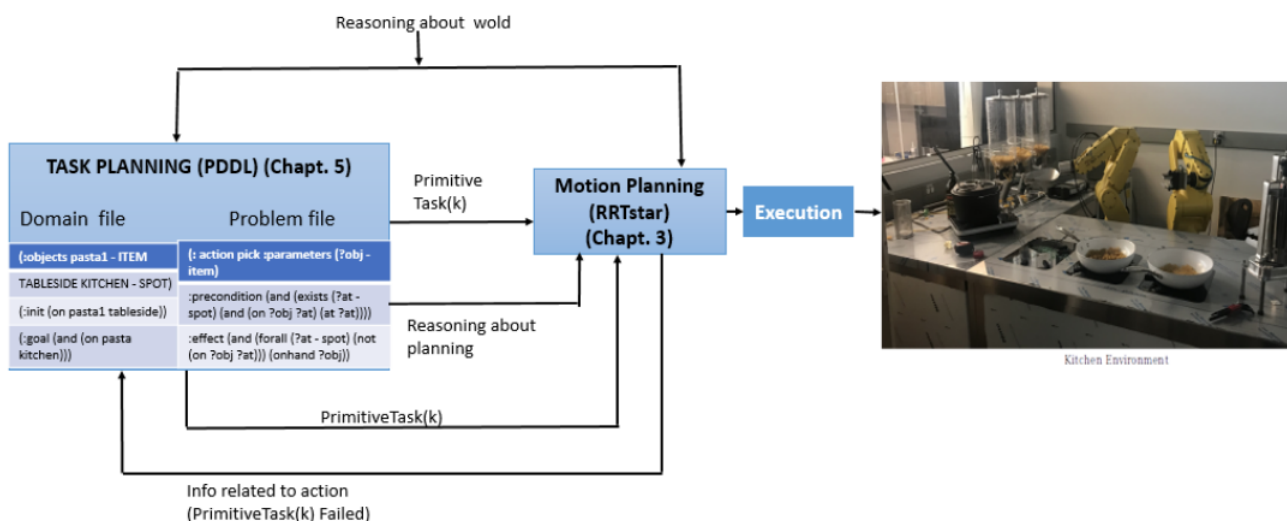


Figure 6.1: Detailed concept of the robotic platform for restaurants.

## 6.3 Task and Motion Planning

The proposed robotic kitchen environment requires detail work on the control techniques, motion planning, task planning, natural language processing, object recognition to execute recipes, and creating an app. Many existing research has been proposed robotic systems for natural language instructions. Previous studies focused on understanding natural language route directions [100–102] and mobile-manipulation commands [25]. However,

only motion planning and task planning layers, and integration between those layers of automated kitchen robots project is the focus of this case study.

As seen in the previous chapters, Chapter 3,4 and 5, we combine task planning layer with the motion planning layer which is able to run within the ROS environment for updating system state, executing joint trajectories, inverse kinematic planning, object perception and manipulation. The lowest level of software manages the sensor and actuator streams. These ROS topics are connected to nodes that integrate robot state information and control the manipulators.



**Figure 6.2:** *An example of collaborative pasta mixing task.*

### 6.3.1 Action Space for the Kitchen

A state-action space for the kitchen domain using primitive actions such as "mix", "flip", and "pour" is defined in PDDL format in a similar manner introduced in Chapter 5, objects in the kitchen environments can be seen in Fig. 6.4. Within this action space, many specific action trajectories are created, yielding a variety of different pasta types. Also, we have a mechanism to create the reference map for the workplace around the kitchen since location of containers, utensils, etc. and where to place them back after using them are extremely important. In order to follow a specific recipe, we applied both interval scheduling, partial-order planning (POPF-a forwards-chaining temporal planner)-two examples of durative-action defined in PDDL can be found in Fig. 6.3- and PDDL

via an online solver (<http://solver.planning.domains/>). The robot searches to find the sequence of states and actions. Then, it executes those actions in the external world as seen in Fig. 6.2. Also, we have built a vision system which can identify objects and their locations within the workspace of the MaM.

```

... (:durative-action move_to_pasta_dispenser
...   :parameters (?Arm - robot-arm ?p - pasta)
...   :duration (= ?duration 5)
...   :condition (and
...     (at start (free ?Arm))
...     (at start (pasta_notdispensed ?p))
...     (at start (notcarrying_pasta ?Arm))
...   )
...   :effect (and
...     (at start (busy ?Arm))
...     (at start (not (free ?Arm)))
...     (at end (at_pasta ?Arm ?p))
...   )
... )
... (:durative-action take_new_pan
...   :parameters (?Arm - robot-arm ?p - pasta)
...   :duration (= ?duration 2)
...   :condition (and
...     (at start (free ?Arm))
...     (at start (pasta_boiled ?p))
...   )
...   :effect (and
...     (at start (busy ?Arm))
...     (at start (not(free ?Arm)))
...     (at end (clean_pan-taken ?Arm))
...   )
... )

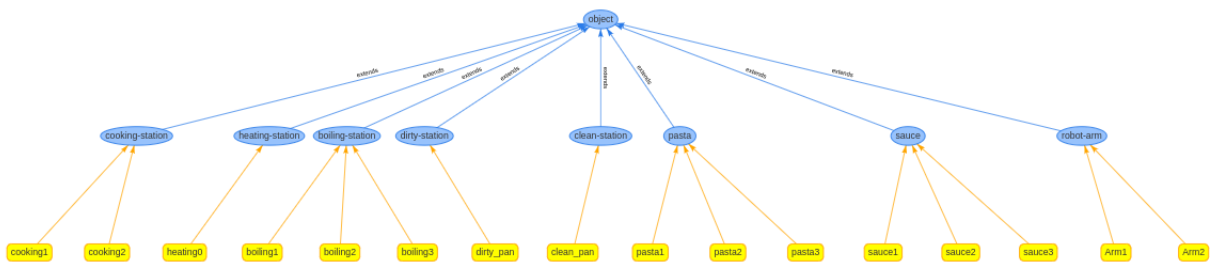
```

**Figure 6.3:** *Two examples of durative-action definition and their effect for automated MaM kitchen environment*

### 6.3.2 Assembly

The action sequence defined in Section 6.3.1 is interpreted into a robot action plan of pasta making primitives. Each pasta making primitive corresponds to a single cooking action and consists of a collection of motion primitives with preconditions and effects and a description of the goal (Figure 6.3)- (: goal(and(order<sub>1</sub>pasta<sub>1</sub>sauce<sub>1</sub>)(order<sub>2</sub>pasta<sub>3</sub>sauce<sub>2</sub>)))- in this case study, we consider serving two orders of pasta simultaneously. The action plan is represented as an action sequence of pasta making primitives. If no supported pasta making primitive exists for an action in the action sequence a user interaction node is created, enabling the system to follow recipes that might involve instructions for which no pasta making primitives exist. For example, if the system encounters a primitive for chopping walnuts, which it cannot execute itself, it asks its human partner for assistance (via text output). The Figure 6.6 represents the high-level task planner output from the POPF planner through ROSPlan as we described in the Chapter 5, Each block, and the text next to it indicate an action happening within the system, and the resources used by that action at the given time in the plan. In the figure, *arm1* and *arm2* represents Franka-Emika arms (7-DOF each). The output of the task planning module then is communicated to the motion planning layer through ROS-Moveit-Rviz to achieve and visualize the each action in the task planning layer. When we change the arm1 base location, the task plan-

ning layer return to failure since the reachability definition for each object in the task planning layer is changed. Then instead of re-defining each object reachability information in the task planning layer, we use communicate the integration layer approach (Ch3) to succeed overall goals. The simulation result shows that each action failure is prevented via proposed integration layer.



**Figure 6.4:** *The objects of the proposed robotic kitchen environment*

### 6.3.3 Plan Execution

Once a sequence of pasta making primitives has been inferred, the robot executes them eventually. When executing a pasta making primitive, the system infers a set of planning predicates that correspond to the world state, symbolically plans a sequence of motion primitives to execute the pasta making primitive from the current state, parametrizes the motion primitive sequence into an executable sequential state machine, executes the state machine. If execution fails, the system estimates the current state and replaces a new sequence of motion primitives to accomplish the pasta making primitive through our interface layer. This enables the system to recover from many errors experienced during recipe execution.

```

|move_to_pasta_dispenser arm1 pasta1
|dispense_pasta arm1 pasta1
|move_pasta_to_boiler boiling1 pasta1 arm1
|pasta_is_boiling pasta1 boiling1 arm1
|move_to_pasta_dispenser arm1 pasta3
|dispense_pasta arm1 pasta3
|move_pasta_to_boiler boiling2 pasta3 arm1
|pasta_is_boiling pasta3 boiling2 arm1
|take_new_pan arm2 pasta1
|move_to_sauce_dispenser arm2 sauce1
|move_to_sauce_dispenser arm2 sauce2
|dispense_sauce arm2 sauce1
|sauce_to_heating arm2 heating sauce1
|sauce_is_heating arm2 heating sauce1
|take_pasta_from_boiler_to_heating boiling1 heating pasta1 sauce1 arm1
|take_to_cooking cooking1 heating pasta1 sauce1 arm2
|take_pasta_from_boiler_to_heating boiling2 heating pasta3 sauce1 arm1
|take_new_pan arm2 pasta3
|dispense_sauce arm2 sauce2
|sauce_to_heating arm2 heating sauce2
|flip_1 cooking1 arm2 sauce1 pasta1
|flip_time_1 cooking1 arm2 sauce1 pasta1
|flip_2 cooking1 arm2 sauce1 pasta1
|flip_time_2 cooking1 arm2 sauce1 pasta1
|flip_3 cooking1 arm2 sauce1 pasta1
|flip_time_3 cooking1 arm2 sauce1 pasta1
|sauce_is_heating arm2 heating sauce2
|serve_order1 cooking1 pasta1 sauce1 arm2 dirty_pan
|take_to_cooking cooking1 heating pasta3 sauce2 arm2
|flip_1 cooking1 arm2 sauce2 pasta3
|flip_time_1 cooking1 arm2 sauce2 pasta3
|flip_2 cooking1 arm2 sauce2 pasta3
|flip_time_2 cooking1 arm2 sauce2 pasta3
|flip_3 cooking1 arm2 sauce2 pasta3
|serve_order2 cooking1 pasta3 sauce2 arm2 dirty_pan

```

**Figure 6.5:** *Output of task planning layer for two pasta orders*

## 6.4 Summary

In this chapter, we present a case study which allows us to implement the contributions we proposed in Chapters 3-5 in the design of task planner for a MaM system used in food cooking applications. We have introduced a high-level task planning layer and a low-level motion planning layer to perform each action defined in the task planning layer.

# Chapter 7

## Conclusions and Future Work

In this thesis, we have proposed and studied new integrated motion and task planning algorithm for multi-arm robot systems in a shared workspace. The contributions were based on the SSG notion introduced in the thesis. This notion is helpful to describe and quantify the interactions of the robots with the world, and to represent multi-robot-manipulator system manipulation planning problems using this graph.

In Chapter 3, we have considered fixed based multi-arm manipulator systems operating in a well defined, and structured environments. In this manner, SSG approach is helpful from the following aspects: s; i) Given a certain number of robots within a cooperative multi-arm robotic system, deciding which ones are the best fit for a given task, ii) distinguishing the regions based on independent working areas and cooperation areas, iii) providing useful information for planning a manipulation path. We have represented the task off-line in an abstract manner by the SSG-based approach.

In Chapter 4, we have extended the proposed SSG-based approach for cases involving robot arms with mobile and fix bases in a uncertain, and time-varying environments. The big advantage of this approach is that it is robust to changes in the conditions, like positions of the objects or robots in the environment.

In Chapter 5, the overall approach to solve a multi-arm manipulation problem was combined with a high-level symbolic planner. We first introduced the robots and their surroundings. Then, we defined some elementary actions for each task. While some environmental (world) parameters (objects or robots locations) were varying, and some of the pre-defined elementary action were failed, our SSG based approach was called. For all steps, simulation results verified that SSG based sequence and motion planning not only helped to find optimal arm sequences for a given task, generated the corresponding optimal trajectories, and decreased the planning time dramatically for search based planning



algorithms compared to single layer planning approaches, but also were able to create alternate plans for the integration between higher level-task planning and lower-level motion planning in case of failure due to uncertain environment.

As a potential future research direction, from a wider perspective, the proposed integration algorithm can be used to program robots for many industrial tasks. A graphical user interface can be formed based on this approach with a few modifications, where the users only need to enter the task, and the rest will be automated based on the SSG.

One of the limitations of this study is the assumption of a pre-defined grasping plan being available for each action. However, in real world, tasks without such plans such as pouring water, cutting some parsley or sweeping a surface have to be considered as well.

Another limitation is the problem of cluttered environment, limiting the arm motions and MP positions. One potential solution is combining MP approach with Reuleaux based approaches. Also, cluttered environment still makes the planning problem difficult due to strong geometric constraints such as occlusions, lack of space in placing objects, objects reachability, or kinematic constraints of the arms.

As another potential future work, SSG representation based on shared workspace between arms providing an insight when some of the robots' shared space changed, can be utilized in studying and selecting on the topology of the mobile multi-arm systems.

Collaborative multi-mobile or bi-manual robots manipulation problems are still challenging due to the coordination of sub-tasks and higher dimensionality. One can put further effort to extend the work.

# References

- [1] F. Basile, F. Caccavale, P. Chiacchio, J. Coppola, and C. Curatella. Task-oriented motion planning for multi-arm robotic systems. *Robotics and Computer-Integrated Manufacturing*, 28(5):569–582, 2012.
- [2] G. Sanchez, and J.-C. Latombe. Using a prm planner to compare centralized and decoupled planning for multi-robot systems. In *IEEE ICRA*, volume 2, pages 2112–2119, 2002.
- [3] G. Sanchez, and J.-C. Latombe. On delaying collision checking in prm planning: Application to multi-robot coordination. *The International Journal of Robotics Research*, 21(1):5–26, 2002.
- [4] S. Christian et al. Dual arm manipulationa survey. *Robotics and Autonomous systems*, 60(10):1340–1353, 2012.
- [5] R. Bohlin, and L. E. Kavraki. Path planning using lazy prm. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 521–528, 2000.
- [6] G. Yang, Z. Pang, M. Jamal Deen, M. Dong, Y. T. Zhang, N. Lovell, and A. M. Rahmani. Homecare robotic systems for healthcare 4.0: Visions and enabling technologies. *IEEE Journal of Biomedical and Health Informatics*, 24(9):2535–2549, 2020.
- [7] A. Dobson, and K. E. Bekris. Planning representations and algorithms for prehensile multi-arm manipulation. In *IEEE/RSJ IROS*, pages 6381–6386, 2015.
- [8] C. Rodriguez, and R. Suarez. Combining motion planning and task assignment for a dual-arm system. In *IEEE/RSJ IROS*, pages 4238–4243, 2016.
- [9] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. In *IEEE Transactions on Robotics and Automation*, volume 12, pages 566–580, 1996.

- [10] T. Thompson, M. Bollini, S. Tellex, N. Roy,, and D. Rus. Interpreting and executing recipes with a cooking robot. *Springer Tracts in Advanced Robotics*, (88):481–495, 2013.
- [11] Y. Sugiura, S.Daisuke, W. Anusha, I. Masahiko, and I. Takeo Igarashi. Cooking with robots: Designing a household system working in open environments. In *In Proc. CHI, ACM*, 2010.
- [12] M. Mizrahi et al. Digital gastronomy: Methods and recipes for hybrid cooking. In *UIST'16*, pages 16–19, 2016.
- [13] I. Daly. Just Like Mombot Used to Make. The NYTimes, Dining and Wine, 2010. [Online; accessed Oct. 20, 2016].
- [14] Technovelgy. AIC-CI Cookingrobot Chinese Robotic Chef. <http://www.technovelgy.com/ct/Science-Fiction-News.asp?NewsNum=770/>, 2006. [Online; accessed October 20, 2016].
- [15] M. Gibson. TIME Tech Innovation: Meet The Robot Chef That Can Prepare Your Dinner. <http://time.com/3819525/robot-chef-moley-robotics/>, 2015. [Online; accessed October 20, 2016].
- [16] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 639–646, May 2014.
- [17] C. R. Garrett, T. Lozano-Prez, and L. Kaelbling. Ffrob: An efficient heuristic for task and motion planning. *Algorithmic Foundations of Robotics (WAFR)*, (88):179–195, 2015.
- [18] I. Umay, B. Fidan, and W. Melek. An integrated task and motion planning technique for multi-robot-systems. In *IEEE International Symposium on Robotic and Sensors Environments (ROSE)*, pages 1–7, June 2019.
- [19] P. Svestka, and M. H. Overmars. Coordinated motion planning for multiple car-like robots using probabilistic roadmaps. In *IEEE International Conference on Robotics and Automation*, number 2, pages 1631–1636, 1995.
- [20] R. Diankov, and J. Kuffner. Openrave: A planning architecture for autonomous robotics. In *Tech. Rep. CMU-RI-TR-08-34, Robotics Institute, Pittsburgh*, July 2008.

- [21] L. E. Kavraki, and P. Švestka, J.C. Latombe, M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.
- [22] S. A. Tovar et al. A fully sensorized cooperative robotic system for surgical interventions. *Sensors*, 12(4):9423–9447, 2012.
- [23] J. A. Cascio. *Optimal Path Planning for multi-arm, multilink robotic manipulators*. Master of science in astronautical engineering, Naval Postgraduate School California, 2004.
- [24] L. E. Kavraki, and J.-C. Latombe. Randomized preprocessing of configuration space for fast path planning. In *IEEE International Conference on Robotics and Automation*, number 3, pages 2138–2145, 1994.
- [25] F. Caccavale, and M. Uchiyama. Cooperative manipulators. *Siciliano and O. Khatib (eds), Springer Handbook of Robotics, 1st edn, Springer Verlag Ltd*, pages 701–718, 2008.
- [26] T. Lozano-Pérez. *Spatial Planning: A Configuration Space Approach*, pages 259–271. Springer New York, New York, NY, 1990.
- [27] S. M. LaValle. *Planning algorithms*. Cambridge University Press, 2006.
- [28] K. Harada, T. Tsuji, and J. P. Laumond. A manipulation motion planner for dual-arm industrial manipulators. In *IEEE ICRA*, pages 928–934, May 2014.
- [29] Y. Koga, and J. Latombe. On multi-arm manipulation planning. In *IEEE ICRA*, pages 945–952, 1994.
- [30] R. H. Fikes, and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. In *Proceedings of the 2nd International Joint Conference on Artificial Intelligence*, 1971.
- [31] J. T. Schwartz, and M. Sharir. Motion planning and related geometric algorithms in robotics. In *In Proc. Int. Congress of Mathematicians*, page 15941611, 1986.
- [32] J. Laumond. Kineo cam: a success story of motion planning algorithms. *IEEE Robotics Automation Magazine*, 13(2):90–93, June 2006.
- [33] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

- [34] H. Choset, K. M. Lynch et al. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Cambridge, MA, June 2005.
- [35] S. M. LaValle. Rapidly-exploring random trees a new tool for path planning. 1998.
- [36] S. M. LaValle, and J. J. Kuffner. Rapidly exploring random trees: Progress and prospects. *Algorithmic and Computational Robotics: New Directions, Editors: Bruce Donald, Kevin Lynch, and Daniela Rus, A. K. Peters/CRC Press*, 12(4):293–308, 2001.
- [37] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.
- [38] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 500–505, 1985.
- [39] C. L. Nielsen, and L. E. Kavraki. A two level fuzzy prm for manipulation planning. In *IEEE International Conference on Intelligent Robots and Systems*, pages 1716–1721, Nov 2000.
- [40] Y. Yang, and O. Brock. Adapting the sampling distribution in prm planners based on an approximated medial axis. In *IEEE International Conference on Robotics and Automation*, volume 5, pages 4405–4410, 2004.
- [41] L. Zhang, and D. Manocha. An efficient retraction-based rrt planner. In *IEEE International Conference on Robotics and Automation*, pages 3743–3750, 2008.
- [42] M. Zucker, J. Kuffner, and J. A. Bagnell. Adaptive workspace biasing for sampling-based planners. In *IEEE International Conference on Robotics and Automation*, pages 3757–3762, 2008.
- [43] J. Canny. *The complexity of robot motion planning*, MIT Press. 1988.
- [44] J. ORourke and J. E. Goodman. *Handbook of discrete and computational geometry*. 2004.
- [45] P. Indyk and J. Matousek. *Low-distortion embeddings of finite metric spaces*. 2004.

- [46] T. Simeon, J. Cortes, A. Sahbani, and J.-P. Laumond. A manipulation planner for pick and place operations under continuous grasps and placements. In *IEEE International Conference on Robotics and Automation*, pages 2022–2027, May 2002.
- [47] L. E. Kavraki, and J.-C. Latombe. Randomized preprocessing of configuration space for fast path planning. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 2138–2145, 1994.
- [48] P. Svestka, and M. H. Overmars. Coordinated motion planning for multiple car-like robots using probabilistic roadmaps. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1631–1636, 1995.
- [49] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 489–494, 2009.
- [50] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4569–4574, 2011.
- [51] D. Berenson, and S. S. Srinivasaz. Probabilistically complete planning with end-effector pose constraints. In *IEEE International Conference on Robotics and Automation*, pages 2724–2730, May 2010.
- [52] S. Dalibard, A. E. Khoury, F. Lamiroux, A. Nakhaei, M. Taix, and J. Laumond. Dynamic walking and whole-body motion planning for humanoid robots: an integrated approach. In *The International Journal of Robotics Research*, volume 32(9-10), pages 1089–1103, 2013.
- [53] I. A. Sucan, and S. Chitta. Moveit. 2012.
- [54] J. Hoffmann, and B. Nebel. The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research (JAIR)*, pages 253–302, 2001.
- [55] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Integrating task and prm motion planning: Dealing with many infeasible motion planning queries. In *ICAPS Workshop on Bridging the Gap between Task and Motion Planning*, 2009.
- [56] G. Wilfong. Motion planning in the presence of movable obstacles. In *Proceedings of the Fourth Annual Symposium on Computational Geometry, SCG '88*, pages 279–288. ACM, 1988.

- [57] R. Alami, T. Simeon, and J.-P. Laumond. A geometrical approach to planning manipulation tasks. The case of discrete placements and grasps. In Hirofumi Miura, editor, *The fifth international symposium on Robotics research*, pages 453–463. MIT Press, 1990.
- [58] B. Dacre-Wright, J. Laumond, and R. Alami. Motion planning for a robot and a movable object amidst polygonal obstacles. In *Proceedings 1992 IEEE International Conference on Robotics and Automation*, pages 2474–2480 vol.3, 1992.
- [59] P. Lertkultanon, and Q. Pham. A single-query manipulation planner. *IEEE Robotics and Automation Letters*, (1(1)):198–205, 2015.
- [60] J. Mirabel, S. Tonneau, P. Fernbach, et al. Hpp: a new software for constrained motion planning. *IEEE/RSJ Intelligent Robots and Systems (IROS)*, 2016.
- [61] D. Berenson, S. Srinivasa, and J. Kuffner. Task space regions: A framework for pose-constrained manipulation planning. *Int. J. Rob. Res.*, 30(12):1435–1460, October 2011.
- [62] K. Hauser. The minimum constraint removal problem with three robotics applications. *Int. Journal of Robotics*, 19:72–32, 2013.
- [63] S. Cambon, R. Alami, and F. Gravot. A hybrid approach to intricate motion, manipulation and task planning. *Int. Journal of Robotics Research (IJRR)*, pages 104–126, 2009.
- [64] J. Barry, L. Kaelbling, and T. Lozano-Prez. A hierarchical approach to manipulation with diverse actions. *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.
- [65] G. Havur, G. Ozbilgin, E. Erdem, V. Patoglu. Hybrid reasoning for geometric rearrangement of multiple movable objects on cluttered surfaces. In *International Conference on Robotics and Automation (ICRA)*, pages 8705–8711, 2014.
- [66] M. Stilman. Task constrained motion planning in robot joint space. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3074–3081, Oct 2007.
- [67] A. Haefliger. Feuilletages sur les varits ouvertes. *Topology*, (9):183–194, 1970.

- [68] B. Dacre-Wright, J. Laumond, and R. Alami. Motion planning for a robot and a movable object amidst polygonal obstacles. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 5, pages 2474–2480, 1992.
- [69] L. Karlsson, J. Bidot, F. Lagriffoul, A. Saffiotti, U. Hillenbrand, and F. Schmidt. Combining task and path planning for a humanoid two arm robotic system. In *Combining Task and Motion Planning for Real-World Applications (ICAPS workshop)*, pages 13–20, 2012.
- [70] R.B. Rusu, I.A. Sutan, et al. Realtime perception-guided motion planning for a personal robot. In *Intelligent Robots and Systems*, pages 4245–4252, 2009.
- [71] Y. Koga, and J.-C. Latombe. Experiments in dual-arm manipulation planning. In *IEEE ICRA*, pages 2238–2245, May 1992.
- [72] L. Fernando, M. F. Gustavo, L. C. Antonio, and. Kinematic control of constrained robotic systems. In *Sba Controle and Automacao*, volume 22, pages 559–572, 2011.
- [73] J. Kuner, J. Latombe. Autonomous agents for real-time animation. 01 2000.
- [74] V. Pilania, and K. Gupta. Mobile manipulator planning under uncertainty in unknown environments. *The International Journal of Robotics Research*, 37(2-3):316–339, 2018.
- [75] M. Beetz, U. Klank, I. Kresse, et al. Robotic roommates making pancakes. In *International Conference on Humanoid Robots*, 2011.
- [76] C. Guo, and E. Sharlin. Exploring the use of tangible user interfaces for human-robot interaction: A comparative study. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 121–130, 2008.
- [77] E. Cheung. Hong Kong’s world-first surgery robot: ‘more precise, less invasive, operates without cutting patient’s body. <http://www.scmp.com/news/hong-kong/health-environment/article/1919679/new-robotic-surgery-tool-more-precise-less/>, March, 2016. [Online; accessed October 20, 2016].
- [78] L. Zongwei. Cooking navi: assistant for daily cooking in kitchen. In *Robotics, Automation, and Control in Industrial and Service Settings*, pages 293–294, 2015.



- [79] B. Cohen, S. Chitta, and M. Likhachev. Single- and dual-arm motion planning with heuristic search. *The International Journal of Robotics Research*, 33(2):305–320, 2014.
- [80] J.-M. Ahuactzin, K. K. Gupta, and E. Mazer,. Manipulation task planning for redundant robots: A practical approach. *International Journal of Robotics Research*, 17(7):734–747, 1998.
- [81] R. L. Rivest T. H. Cormen, C. E. Leiserson and C. Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009.
- [82] Kinova Robotics. Kinova-movo. <https://github.com/Kinovarobotics/kinova-movo>, 2018.
- [83] I. A. Sucas, M. Moll, L. E. Kavraki. The open motion planning library. *IEEE Robotics and Automation Magazine*, 19:72–32, 2012.
- [84] I. Umay, W. Melek, and B. Fidan. Vision-guided integrated task and motion planning for multi-robot-arms. In *Manuscript ready for submission*, 2020.
- [85] F. Bullo, and S. L. Smith. *Lectures on Robotic Planning and Kinematics*. 2015.
- [86] <http://www.pyopt.org/reference/optimizers.slsqp.html>, 2020.
- [87] [http://wiki.ros.org/pykdl\\_utils](http://wiki.ros.org/pykdl_utils), 2020.
- [88] K. He, M. Lahijanian, L. E. Kavraki, M. Y. Vardi. Towards manipulation planning with temporal logic specifications. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 346–352, May 2015.
- [89] M. Cashmore, M. Fox, D. Long, D. Magazzeni, B. Ridder, A. Carrera, N. Palomeras, N. Hurtos, M. Carreras. Rosplan: Planning in the robot operating system. In *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS-15)*, June 2015.
- [90] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.
- [91] R. Peters. Robotisation in food industry. In *in 5th Int Conference on the Food Factory for the Future*, pages 4245–4252, 2010.

- [92] P. Adl, Z. A. Memon, and R. T. Rakowski. Robot handling of food products. In *Sensors and Their Applications*, 1991.
- [93] R. Hamada, J. Okabe, I. Ide, S. Satoh, S. Sakai, and H. Tanaka. Cooking navi: assistant for daily cooking in kitchen. In *In Proc. of ACM*, pages 371–374, 2005.
- [94] Y. Nakauchi, T. Fukuda, K. Noguchi, T. Matsubara. Time sequence data mining for cooking support robot. In *In Proc. CIRA*, pages 3–4, 2005.
- [95] S. K. Gupta. When will we have robots to help with household chores? <http://spectrum.ieee.org/automaton/robotics/home-robots/when-will-we-have-robots-to-help-with-household-chores/>, 2014. [Online; accessed October 20, 2016].
- [96] D. Sakamoto, K. Honda, M. Inami, and T. Igarashi. Sketch and run: a stroke-based interface for home robots. In *In Proc. CHI, ACM*, pages 197–200, 2009.
- [97] D. Bertram, J. Kuffner, R. Dillmann, and T. Asfour. An integrated approach to inverse kinematics and path planning for redundant manipulators. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 1874–1879, May 2006.
- [98] L. Petersson. Systems integration for real-world manipulation tasks. In *Proc. of Int. Conf. on Robotics and Automation*, number 3, pages 2500–2505, 2002.
- [99] F. Gravot, A. Haneda, K. Okada, and M. Inaba. Cooking for humanoid robot, a task that needs symbolic and geometric reasonings. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, ICRA*, pages 462–467, 2006.
- [100] D. L. Chen, and R. J. Mooney. Learning to interpret natural language navigation instructions from observations. In *In Proc. of AAAI*, 2011.
- [101] T. Kollar, S. Tellex, D. Roy, and N. Roy. Toward understanding natural language directions. In *Proc. ACM/IEEE International Conf. on Human-Robot Interaction (HRI)*, pages 259–266, 2010.
- [102] C. Matuszek, D. Fox, and K. Koscher. Following directions using statistical machine translation. In *Proc. ACM/IEEE International Conf. on Human-Robot Interaction*, pages 251–258, 2010.
- [103] J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

- [104] M. Stilman and J. Kuffner. Planning among movable obstacles with artificial constraints. *The International Journal of Robotics Research*, (27(11-12)):481–495, 2008.
- [105] Hitachi Group. Autonomous Mobile Dual-Arm Robot Creating the Future of Logistic. [http://www.hitachi.com/businesses/innovation/topics/rd\\_logistics/](http://www.hitachi.com/businesses/innovation/topics/rd_logistics/).
- [106] R. B. Rusu, B. Gerkey, and M. Beetz. Drobots in the kitchen: Exploiting ubiquitous sensing and actuation. *Robotics and Autonomous Systems*, (56):844–856, 2008.
- [107] R. B. Rusu, B. Gerkey, and M. Beetz. Roboethics : A navigating overview. *Intelligent Systems, Control and Automation: Science and Engineering*, (79):1499–1524, 2008.
- [108] M. Novak. The Disco-Blasting Robot Waiters of 1980s Pasadena. <http://www.smithsonianmag.com/history/the-disco-blasting-robot-waiters-of-1980s-pasadena-70137340/>, April, 2012. [Online; accessed October 20, 2016].
- [109] ABB Robotics. The future of robotics and automation depends on humans and robots working together. <https://www.robots.com/articles/viewing/human-robot-collaboration-with-the-abb-yumi/>. [Online; accessed October 24, 2016].
- [110] R. S. Stone, and P. N. Brett. A novel tactile sensing technique for non-rigid materials. In *in Proceedings of Euriscon*, pages 1384–1524, 1994.
- [111] A. Saxena, J. Driemeyer, and Y. N. Andrew. Robotic grasping of novel objects using vision. *The International Journal of Robotics Research*, (27(2)):157–173, 2008.
- [112] Yang Yezhou, Li Yi, F. Cornelia, and A. Yiannis. Robot learning manipulation action plans by ”watching” unconstrained videos from the world wide web. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI’15, pages 3686–3692, 2015.
- [113] M. Strandberg. *Robot Path Planning: An Object-Oriented Approach*. Phd thesis, Royal Institute of Technology (KTH) Stockholm, Sweden, 2004.
- [114] K. Yamazaki, Y. Watanabe, K. Nagahama, K. Okada, and M. Inaba. Recognition and manipulation integration for a daily assistive robot working on kitchen environments. In *IEEE ICRB*, pages 196–201, 2010.

- [115] K. Yamazaki, M. Tomono, T. Tsubouchi, and S. i. Yuta. A grasp planning for picking up an unknown object for a mobile manipulator. In *IEEE Proceeding, ICRA*, pages 2143–2149, 2006.
- [116] N. Correll, N. Arechiga et al. Building a distributed robot garden. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1509–1516, 2009.
- [117] G. S. Sundaram, B. Patibandala et al. Bluetooth communication using a touchscreen interface with the raspberry pi. In *Proceedings of IEEE Southeastcon*, pages 1–4, 2013.
- [118] E. Sariyildiz, and H. Temeltas. A singularity free trajectory tracking method for the cooperative working of multi-arm robots using screw theory. In *IEEE International Conference on Mechatronics*, pages 451–456, 2011.
- [119] Y. Huang, M. Li, W. Lu, and G. Lu. Calibration of two cooperative manipulators via pseudo-closed-loop method. In *IEEE International Conference on Systems, Man and Cybernetics. Information Intelligence and Systems*, volume 2, pages 1465–1470 vol.2, 1996.
- [120] F. Sanfilippo, L. I. Hatledal, H. Zhang, M. Fago, and K. Y. Pettersen. Jopenshowvar: An open-source cross-platform communication interface to kuka robots. In *IEEE International Conference on Information and Automation (ICIA)*, pages 1154–1159, 2014.
- [121] F. Sanfilippo, L. I. Hatledal, H. Zhang, M. Fago, and K. Y. Pettersen. Controlling kuka industrial robots: Flexible communication interface jopenshowvar. *IEEE Robotics Automation Magazine*, 22(4):96–109, 2015.
- [122] M. Namvar, and F. Aghili. Adaptive force-motion control of coordinated robots interacting with geometrically unknown environments. *IEEE Transactions on Robotics*, 21(4):678–694, 2005.
- [123] T. Mehmood, U. Hashmi, A. Akhter, and A. Ajmal. Techniques and approaches in robocup@home - a review. In *International Conference on Information and Communication Technologies (ICICT)*, pages 1–6, 2015.
- [124] G. Chang, and D. Kulic. Robot task learning from demonstration using petri nets. In *IEEE Proceedings of RO-MAN*, pages 31–36, 2013.

- [125] A. Aldoma, and M. Vincze et al. Cad-model recognition and 6dof pose estimation using 3d cues. In *IEEE International Conference on Computer Vision Workshops*, pages 585–592, 2011.
- [126] A. Torralba, K. P. Murphy, W. T. Freeman, and M. A. Rubin. Context-based vision system for place and object recognition. *International Conference on Computer Vision*, (1):273–280, 2003.
- [127] R. Smits, H. Bruyninckx, and E. Aertbelien. Kdl: Kinematics and dynamics library. In *Available: <http://www.oroocos.org/kdl>*, 2011.
- [128] R Smits, H Bruyninckx, and E. Aertbelien. Openrave, ik fast module, openrave documentation. Jan 2014.
- [129] B. Asadi. Single and dual arm manipulator motion planning library. In *Motion Planning Library, Workshop on Task Planning for Intelligent Robots in Service and Manufacturing*, 2015.
- [130] N. Garca, R. Surez, and J. Rosell. Task-dependent synergies for motion planning of an anthropomorphic dual-arm system. *IEEE Transactions on Robotics*, 33(3):756–764, June 2017.
- [131] Z. McCarthy, T. Bretl, and S. Hutchinson. Proving path non-existence using sampling and alpha shapes. In *IEEE International Conference on Robotics and Automation*, pages 2563–2569, May 2012.
- [132] M. Helmert. The fast downward planning system. In *Journal Artificial Intelligence Resesearch (JAIR)*, pages 191–246, 2006.
- [133] H. Kautz, and B. Selman. Unifying sat-based and graph-based planning. *Int. Joint. Conf. on Artifial Intelligence (IJCAI)*, pages 318–325, 1999.
- [134] J. Rintanen. Planning as satisfiability: Heuristics. *Artificial Intelligence*, pages 45–86, 2012.
- [135] I. A. Şucan, M. Moll, and L. E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, December 2012.
- [136] P. Schuller, V. Patoglu, and E. Erdem. Levels of integration between low-level reasoning and task planning. In *AAAI Workshop on Intelligent Robotic Systems*, 2013.

- [137] H. K. Gazit, G. E. Fainekos, and G. J. Pappas. Translating structured english to robot controllers. *Advanced Robotics*, pages 1343–1359, 2008.
- [138] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. Temporal-logic-based reactive mission and motion planning. *IEEE Transactions on Robotics*, 25(6):1370–1381, 2009.
- [139] C. Matuszek, E. Herbst, L. Zettlemoyer, and D. Fox. Answer set programming for collaborative housekeeping robotics: representation, reasoning, and execution. *Experimental Robotics, Springer*, pages 403–415, 2013.
- [140] M. Tenorth, and M. Beetz. Answer set programming for collaborative housekeeping robotics: representation, reasoning, and execution. *In Experimental Robotics, Springer*, pages 275–291, 2015.
- [141] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. *In IEEE International Conference on Robotics and Automation (ICRA)*, pages 639–646, May 2014.
- [142] D. McDermott, M. Ghallab, et al. Answer set programming for collaborative housekeeping robotics: representation, reasoning, and execution. *PDDL the planning domain definition language, AIPS-98 Planning Competition Committee*, 1998.
- [143] D. M. Lyons, and M. A. Arbib. A formal model of computation for sensory-based robotics. *IEEE Trans. on Robotics and Automation*, page 280293, 1989.
- [144] G. Marani, G. Antonelli, and J. Yu. Marine robotic systems. *IEEE Robotics and Automation Magazine*, pages 18–28, 2010.
- [145] P. Cheng, V. Kumar, R. Arkin, M. Steinberg, and K. Hedrick. Cooperative control of multiple heterogeneous unmanned vehicles for coverage and surveillance. *IEEE Robotics and Automation Magazine*, pages 12–22, 2009.
- [146] C. C. Kemp, A. Edsinger, and E. Torres-Jara. Challenges for robot manipulation in human environments [grand challenges of robotics]. *IEEE Robotics Automation Magazine*, 14(1):20–29, March 2007.
- [147] N. G. Hockstein, C. G. Gourin, R. A. Faust, D. J. Terris. A history of robots: from science fiction to surgical robotics. *Journal of Robotic Surgery*, 1(2):113–118, 2007.

- [148] S. Thrun et al. Stanley: The robot that won the darpa grand challenge. *Journal of Field Robotics*, 23(9):661–692, 2006.
- [149] M. Tenorth, U. Klank, D. Pangercic, and M. Beetz. Web-enabled robots. *IEEE Robotics Automation Magazine*, 18(2):58–68, June 2011.
- [150] M. Toussaint. Logic-geometric programming: An optimization-based approach to combined task and motion planning. In *Proceedings of the 24th International JCAI, Argentina*, pages 1930–1936, 2015.
- [151] C. Boutilier, and R. I. Brafman. Partial Order Planning with Concurrent Interacting Actions. *Artificial Intelligence*, 14:105–136, 2001.
- [152] S. Cambon, R. Alami, and F. Gravot. A hybrid approach to intricate motion, manipulation and task planning. *I. J. Robotic Res.*, 28(1):104–126, 2009.
- [153] S. Cambon, F. Gravot, and R. Alami. A robot task planner that merges symbolic and geometric reasoning. In *ECAI*, volume 16, page 895, 2004.
- [154] V. R. Desaraju, and J. P. How. Decentralized path planning for multi-agent teams with complex constraints. *Autonomous Robots*, 32(4):385–403, 2012.
- [155] M. Fox, and D. Long. Pddl2.1: An extension to pddl for expressing temporal planning domains. *J. Artif. Int. Res.*, 2003.
- [156] F. Lagriffoul, D. Dimitrov, J. Bidot, A. Saffiotti, and L. Karlsson. Efficiently combining task and motion planning using geometric constraints. *I. J. Robotic Res.*, 33(14):1726–1747, 2014.
- [157] J. A. Shah, P. R. Conrad, and B. C. Williams. Fast distributed multi-agent plan execution with dynamic task assignment and scheduling. In *Proc. International Conference on Automated Planning and Scheduling*, 2009.
- [158] D. Katz, and O. Brock. Grounding relational reinforcement learning for manipulation in unstructured environments. In *Proceedings of Robotics: Science and Systems (RSS)*, pages 13–20, 2008.
- [159] D. Kulis, and E. A. Croft. Affective state estimation for human-robot interaction. *IEEE Transactions on Robotics*, 23(5):991–1000, Oct 2007.
- [160] R. E. Fikes, and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3):189 – 208, 1971.

- [161] J. Hoffmann, and B. Nebel. The ff planning system: Fast plan generation through heuristic search. *J. Artif. Int. Res.*, 14(1):253–302, 2001.
- [162] Bullet collision library. <https://github.com/bulletphysics/bullet3>, 2020.
- [163] Flexible collision library. <https://github.com/flexible-collision-library/fcl>, 2020.
- [164] Networkx. <https://networkx.github.io/>, 2020.
- [165] S. Haddadin, and L. Johannsmeier. The art of manipulation: Learning to manipulate blindly. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9, 2018.
- [166] V. Mokhtari, R. Manevich, L. S. Lopes, and A. J. Pinho. Learning the scope of applicability for task planning knowledge in experience-based planning domains. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3973–3979, 2019.
- [167] M. Grner, R. Haschke, H. Ritter, and J. Zhang. Moveit! task constructor for task-level motion planning. In *International Conference on Robotics and Automation (ICRA)*, pages 190–196, 2019.
- [168] C. Phiquepal, and M. Toussaint. Combined task and motion planning under partial observability: An optimization-based approach. In *International Conference on Robotics and Automation (ICRA)*, pages 9000–9006, 2019.
- [169] J. K. Behrens, R. Lange, and M. Mansouri. A constraint programming approach to simultaneous task allocation and motion scheduling for industrial dual-arm manipulation tasks. In *International Conference on Robotics and Automation (ICRA)*, pages 8705–8711, 2019.
- [170] A. S. Ioan, and C. Sachin. Moveit. <http://moveit.ros.org>, 2020.
- [171] F. Valenza N. Mansard and J. Carpentier. Pinocchio: Fast forward inverse dynamic for multibody systems. 2014.
- [172] E. Loutas, K. Diamantaras, and I. Pitas. Occlusion resistant object tracking. In *Proceedings 2001 International Conference on Image Processing (Cat. No.01CH37205)*, volume 2, pages 65–68, 2001.



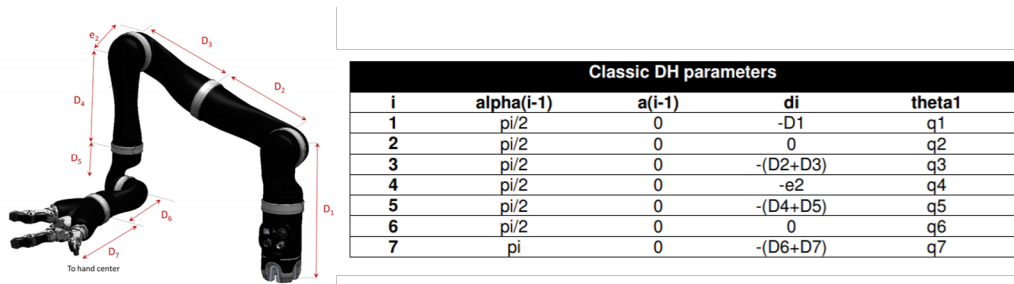
- [173] V. Lippiello, B. Siciliano, and L. Villani. Position-based visual servoing in industrial multirobot cells using a hybrid camera configuration. *IEEE Transactions on Robotics*, 23(1):73–86, 2007.
- [174] A. Hafez, and E. Cervera. Particle-filter-based pose estimation from controlled motion with application to visual servoing. *International Journal of Advanced Robotic Systems*, 11(10):177, 2014.
- [175] G. Flandin, F. Chaumette, and E. Marchand. Eye-in-hand/eye-to-hand cooperation for visual servoing. In *IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 3, pages 2741–2746, 2000.
- [176] M. Elena, M. Cristiano, F. Damiano, and M. Bonfe. Variable structure pid controller for cooperative eye-in-hand/eye-to-hand visual servoing. In *Proceedings of IEEE Conference on Control Applications.*, volume 2, pages 989–994, 2003.
- [177] A. Muis and K. Ohnishi. Eye-to-hand approach on eye-in-hand configuration within real-time visual servoing. *IEEE/ASME Transactions on Mechatronics*, 10(4):404–410, 2005.
- [178] T. Murao, H. Kawai, and M. Fujita. Passivity-based control of visual feedback systems with dynamic movable camera configuration. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 5360–5365, 2005.
- [179] T. Drummond, and R. Cipolla. Real-time visual tracking of complex structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):932–946, 2002.
- [180] A. I. Comport, D. Kragic, E. Marchand, and F. Chaumette. Robust real-time visual tracking: Comparison, theoretical analysis and performance evaluation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2841–2846, 2005.
- [181] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.
- [182] O. Kermorgant, and F. Chaumette. Multi-sensor data fusion in sensor-based control: Application to multi-camera visual servoing. In *IEEE International Conference on Robotics and Automation*, pages 4518–4523, 2011.

- [183] V. Lippiello, B. Siciliano, and L. Villani. 3d pose estimation for robotic applications based on a multi-camera hybrid visual system. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2732–2737, 2006.
- [184] K. Okada, M. Kojima, Y. Sagawa, T. Ichino, K. Sato, and M. Inaba. Vision based behavior verification system of humanoid robot for daily environment tasks. In *6th IEEE-RAS International Conference on Humanoid Robots*, pages 7–12, 2006.

# Appendix A

## A.1 Robot Specification Parameters

The following charts represents the classic DH-parameters of *JACO<sup>2</sup> Spherical 7-DOF* arm:



**Figure A.1:** *DH-parameters and robot's link-length values for JACO<sup>2</sup> Spherical 7 – DOF*

### Decoupled Planning

Decoupled approaches first design motions for the robots while ignoring robot- robot interactions. Once these interactions are considered, the choices available to each robot are already constrained by the designed motions. If a problem arises, these approaches are typically unable to reverse their commitments. Therefore, completeness is lost. Nevertheless, decoupled approaches are quite practical, and in some cases completeness can be recovered.

### Prioritized planning

A straightforward approach to decoupled planning is to sort the robots by priority and plan for higher priority robots first. Lower priority robots plan by viewing the higher priority robots as moving obstacles.

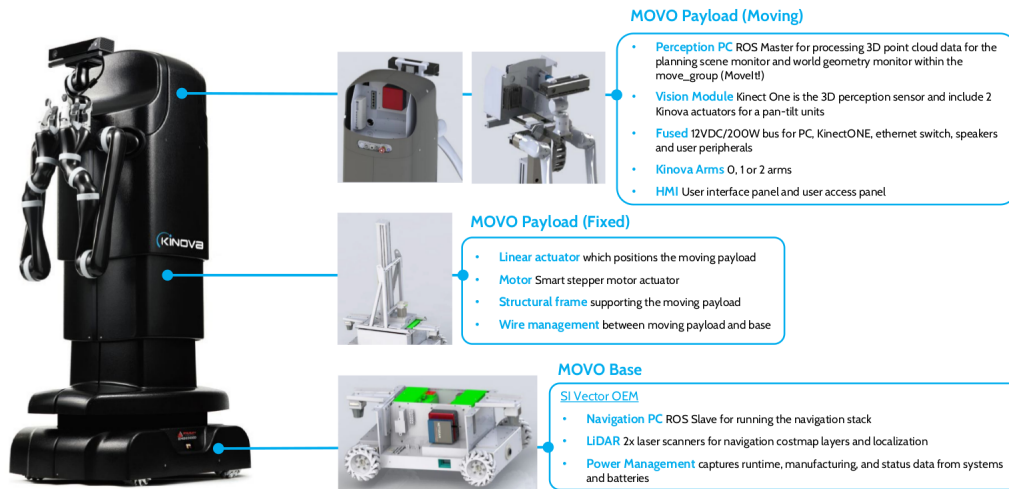
<b>Physical</b>	Variable height torso	1,100 mm to 1,580 mm (43 in to 62 in)
	Base footprint	508 mm width X 810 mm depth ( 20 in X 32 in)
	Operational environment	Indoor: 5 °C to 40 °C (41 °F to 104 °F)
	Run time	8 h Normal usage
	Recharge time	5 h Standard 2 h Fast charger
<b>Manipulator</b>	DOF	From 8 to 27 DOF
	Arm	0, 1 or 2 Kinova JACO <sup>2</sup> arms
	Gripper	2 or 3 fingers
	Maximum linear arm speed	20 cm/s (7.8 in/s)
<b>Mobile Base</b>	Max speed	2 m/s (78.8 in/s)
	Drive	Holonomic
<b>Sensors</b>	Platform	2D Planar laser (front and rear)
	Head	Kinect for Xbox One
	Actuator	Position, velocity, torque, temperature
<b>Computer</b>	Navigation and perception	2 dedicated PCs
	CPU	Intel NUC Kit NUC5i7RYH
	RAM	16 GB
	Hard disk	128 GB SSD
<b>Connectivity</b>	Interfaces	HDMI, USB 3.0
	Communication	Ethernet, Wi-Fi
	Audio	3.5 mm audio input

**Figure A.2:** *MovoBeta specifications*

*”Completeness: given a description of the robot and the environment as semi-algebraic sets, it is a collision-free path connecting the initial and goal configuration of the robot when it exists, and reports failure when it does not [?].”* Since completeness implies the algorithm is inapplicable to any realistic robot system with many degrees of freedom, we trade-off completeness for an algorithm that efficiently and reliably solves a majority of realistic manipulation tasks involving multiple arms.

**Complexity** has a direct link to solutions computation time. In most cases this parameter estimates the maximal operation to do in order to obtain a solution (if any). In practice complexity tend to over estimate the computation time of a solution which actually much more faster.

Redundant manipulators have more degrees of freedom that those strictly necessary to perform a given task. Redundancy is therefore a relative concept for a robot manipulator, depending on the particular type of task to be executed. For example, six DOF are necessary for positioning and orienting a robot end-effector, thus a 6-DOF manipulator is considered non-redundant. However,if only the positioning task is of concern, the same arm becomes redundant.The extra degrees of freedom provide more dexterity to the robot structure, and it can be used to avoid kinematic singularities.



**Figure A.3:** *Movo specifications*

The structure of a robot manipulator consists of a set of rigid bodies or links connected by means of revolute or prismatic joints, integrating a kinematic chain. In general, one end of the chain is fixed to a base, whereas an end-effector is connected to the other end. From a topological point of view, a kinematic chain can be classified in (i) open or serial, when there is only one sequence of links and joints connecting the two ends of the chain, and

(ii) closed or parallel, when a sequence of links and joints are arranged such that at least one loop exists.

The main disadvantages of using parallel robots are: workspace limitation, more complex forward kinematics maps and more involved singularity analysis [68]. For instance, in contrast to serial chain manipulators, the singularities in parallel mechanisms may have different characteristics.

In this context, the singularities can be classified into three basic types:

(i) configuration space singularities: when the rank of the structure equations drops and, thus, the endeffector loses the ability to move instantaneously in some directions;

(ii) end-effector singularities: when the end-effector loses degrees of freedom (DOF), that is, the motion of the active joints can result in no motion of the end-effector;

(iii) actuator singularities: when the actuator of the manipulator or the active joints cannot produce end-effector forces and torques in some directions.

## A.2 ROS

ROS is the acronym of Robot Operating System. This name, however, is misleading since ROS is not an Operating System in the traditional sense of the concept. Instead, ROS is an Open Source platform aimed at programming algorithms and drivers for application in Robotics. The desired functionality is wrapped in what is called a ROS package and then it can be shared with the community. The main objective of ROS is to facilitate the exchange and utilization of state-of-the-art algorithms and tools. There exist a high amount of useful packages being shipped for ROS. We will discuss some of these packages later. Next, we discuss some of the features that make ROS a valuable resource in the field of Robotics.

**Execution model:** In ROS, the minimum execution unit is the node. A node is a concurrent/parallel application or utility that performs certain task or functionality. In order to execute a node, there must be one ROS master running. A master handles several nodes and dispatches messages (more about messages below) among them. The master can be running on a remote machine. This execution model provides a natural mechanism to separate the functionality of a complex application into several modules.

**Topics:** Nodes can communicate between them by means of messages. Messages can be published by one or more nodes in what is called a topic. Other nodes may subscribe to those topics and receive the messages being published. This allows great flexibility since the subscribers do not have to worry about the implementation details of the publishers as long as they follow the same messaging convention. Applications do not have to be a monolithic pieces of code. Instead, this messaging mechanism allows nodes to be like small functional blocks whose inputs and outputs can be plugged in to other blocks or operate in isolation. Moreover, if the master is running in a remote machine, the messages are sent through the network transparently so the programmer does not have to worry about sockets. One of the most typical uses of topics is to stream a continuous flow of data (e.g. video from a camera or the joints state of a robot).

**Services:** Services have certain degree of similitude with regular functions in most programming languages. Nodes can offer services that may be called by other nodes. The specification of a service includes the input arguments and the output received by node that has called the service. Calls to services are blocking, so these are typically used for fast computations (e.g. an inverse kinematics query).

**Actions:** Actions resemble services in that they offer some kind of callable functionality for being used by other nodes. The difference is that actions are normally used to implement non-blocking functions that may span during a long interval in time. The calling node may preempt the action at any moment, and the node that offered the action can

publish a periodic feedback telling the caller about the current state of the action. Because of these characteristics one common use of actions is to implement robot movements.

Support for several programming languages: ROS provides support and an API for several programming languages, including C++, Python, Lisp and Java for Android. Even so, the documentation of ROS and most of its packages is focused mostly on the C++ API and the Python API. The languages used in our project are C++ and Python.

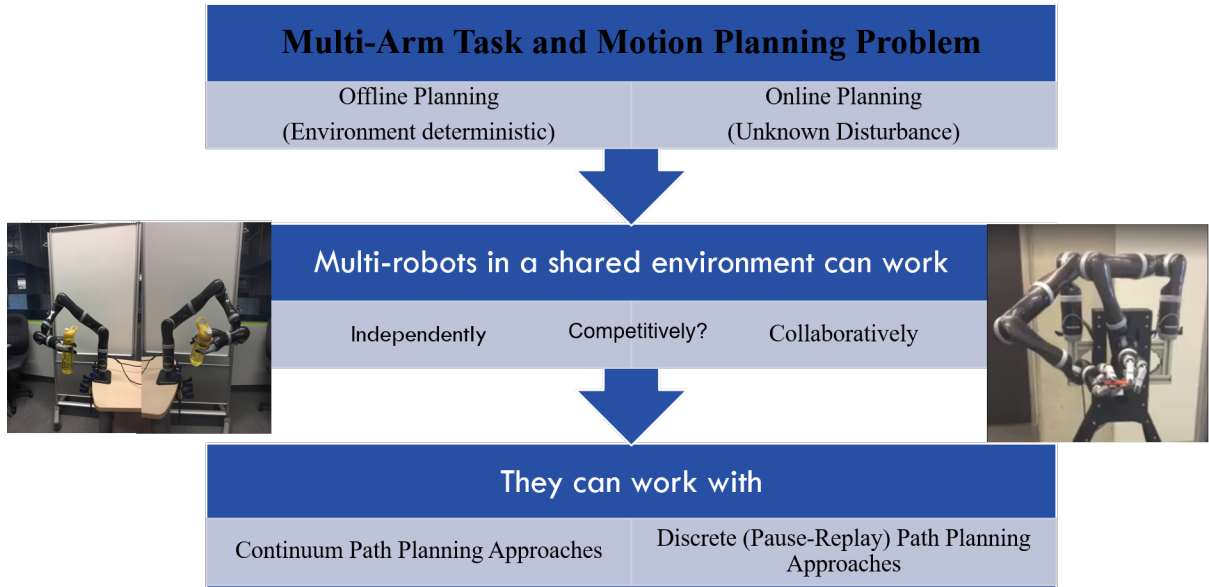
Debugging and configuration tools: ROS comes or can be extended with several debugging and configuration tools that may run from the command line or have graphical interface. One of these tools is `rqt-graph`, a package with contains a Qt application for visualizing the nodes, and publisher/subscriber relationship between them. We also have `rqt-reconfigure` which allows us to dynamically change the parameters of nodes that are already running. Next we present some example packages in order to give an idea of what kind of functionality can be offered by a ROS package (this is not intended to be an exhaustive list of all the packages we are going to use):

`tf`: `tf` takes care of publishing the transformations between different frames of reference (for example, between the joints of a robotic arm) at each instant of time. It can also be useful for obtaining the coordinate transformation between two frames that are connected indirectly, or for tracking the transformation between two frames in the past. In addition, the API of `tf` comes with some additional utilities (e.g. performing conversions between angles in RPY and quaternions).

`Rviz`: `Rviz` is a multipurpose visualization and interaction tool. It allows us to view the state of a robot, information from sensors that is published in a topic, etc. In this sense it is a very powerful monitoring tool.

`Moveit`: powerful geometrical planning framework for robots. `moveit` uses OMPL (another ROS package for path planning) for planning trajectories for robots. It also comes with a `rviz` plugin for visualizing plans and calculating new ones. `moveit` works for several robots. It takes an XML description of the robots joints and links and use them to calculate a plan without self-collision nor collisions against world objects.

`RRT*` (optimal RRT) is an asymptotically-optimal incremental sampling-based motion planning algorithm. `RRT*` algorithm is guaranteed to converge to an optimal solution, while its running time is guaranteed to be a constant factor of the running time of the RRT. The notion of optimality is with respect to a specified Optimization Objective (set in the Problem Definition). If a solution path's cost is within a user-specified cost-threshold, the algorithm terminates before the elapsed time [83,181].



**Figure A.4:** *Multi Robot Planning Approaches: An initial and a goal configuration are given as input for each robot.*

### A.2.1 Meet Points Calculation

Any *MP* is located where an object is being picked or placed by a transferring arm. Any set  $S$  that involves  $K$  arms has to pass through  $o_{init}$ ,  $(K - 1)$  number of *MPs*, and  $o_{goal}$ , respectively, as illustrated in Figure 5, *Scheme 1*. *MP* is found by considering circle-circle and/or sphere-sphere intersection geometry by selecting the centre of the shared workspace (Figure 4.3) .

Note that based on Figure A.6,

$$x = \frac{(r_1^2 - r_2^2 - d^2)}{2d}$$

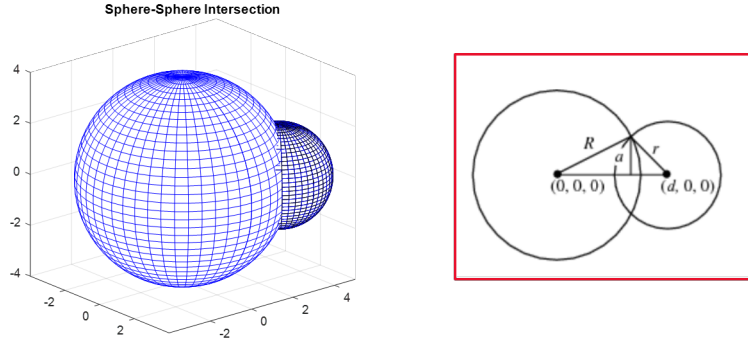
For instance, the analysis for 3-D case, sphere-sphere intersection for two arms as follow;

$$\begin{aligned} x^2 + y^2 + z^2 &= R^2 \\ (x - d)^2 + y^2 + z^2 &= r^2 \end{aligned}$$

Solving for  $x = MP$  gives;

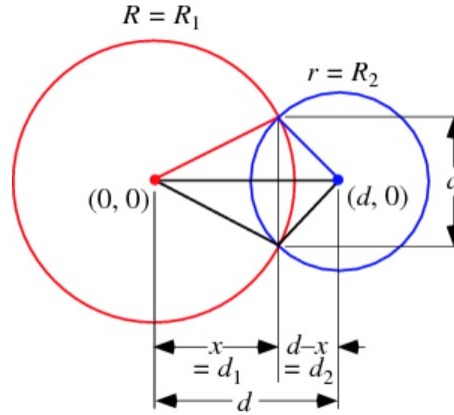
$$MP = \frac{d_{ij}^2 - r_i^2 + r_{i+1}^2}{2d_{ij}} \tag{A.2.1}$$





**Figure A.5:** *Intersection of two robots' workspace for 3D case*

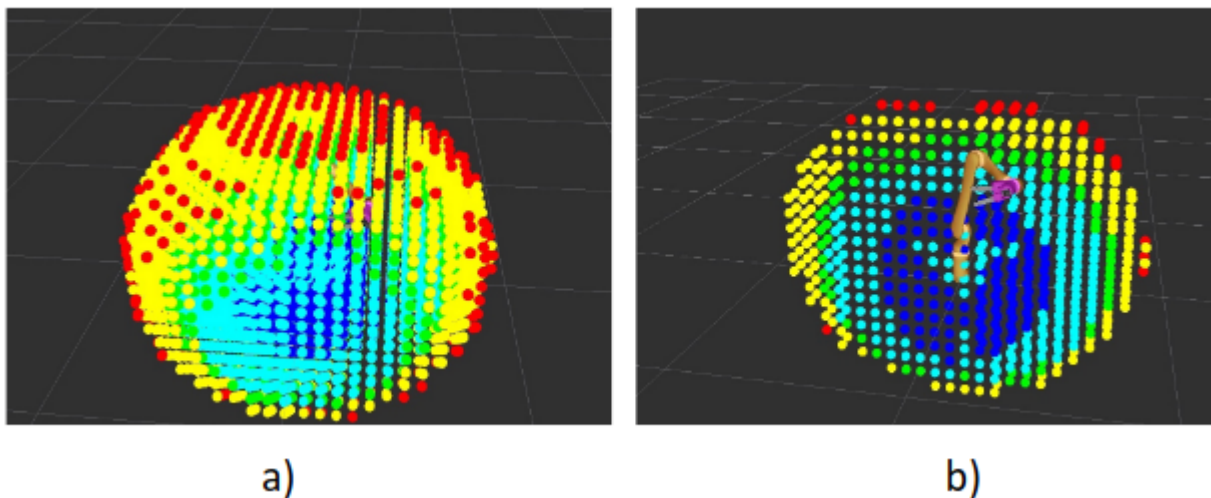
where  $d$ ,  $r$ ,  $R$  and  $MP$  are distance between 2-robots' base location, the sum of the link length of first robot, the sum of the link length of second robot and meet point respectively. Also,  $i, j$  are specific robots' IDs, and  $j = i + 1$ .



**Figure A.6:** *Meet point representation where,  $r_1$ , and  $r_2$ ,  $x$ ,  $d = d_1 + d_2$ , represent the first and second robot's workspaces, meet point, and distance between the two robots' bases, respectively. If  $d < r_1 + r_2$ , then there is a shared workspace between two robots.*

Note that meet point is a point that two consecutive arms are able to reach. However, in the case of calculated meet point is not empty or it is not possible to reach it by those consecutive arms, then we should be able to find a alternative meet points or a better candidate. Therefore, for future work, meet point will be calculated based on shared workspace between arms and the reachability database to be able to choose best handoff

location for a specific task. Figure A.7 shows collision-free reachability database for a jaco arm.



**Figure A.7:** Representation of down sampled version of our reachability database for Jaco arm. (a) Reachable poses represented by colored arrows, the color encodes the manipulability of the corresponding manipulator configuration. (b) The cross sectional view of the voxelized 3d workspace of a jaco arm, the color indicates the number of grasping poses contained in the 3d voxel.

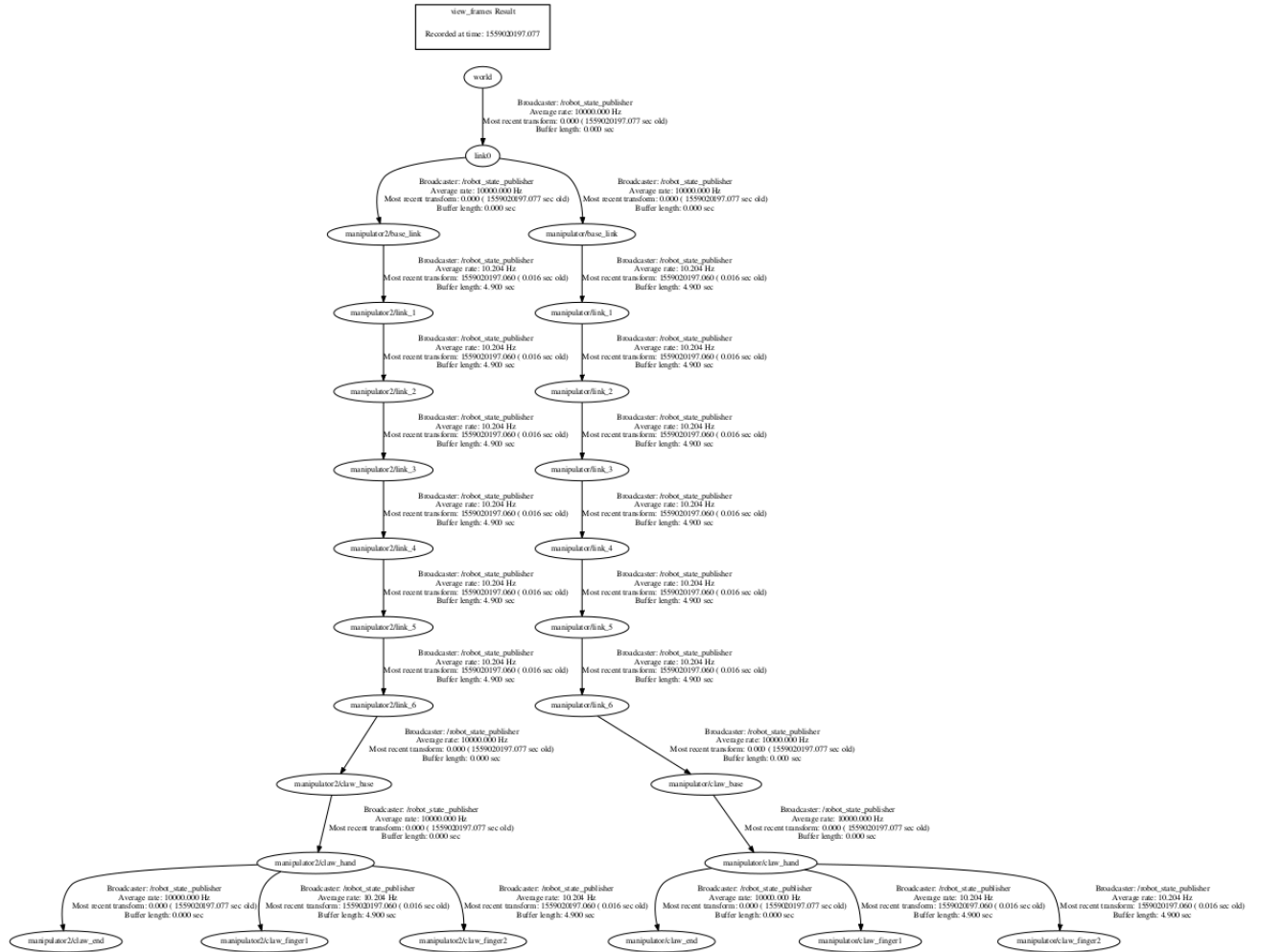
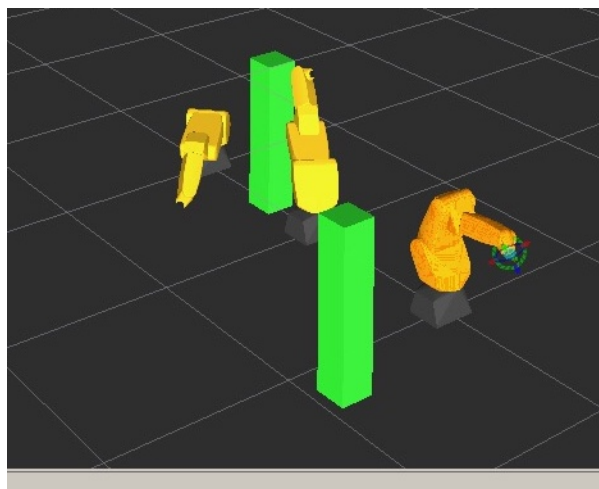
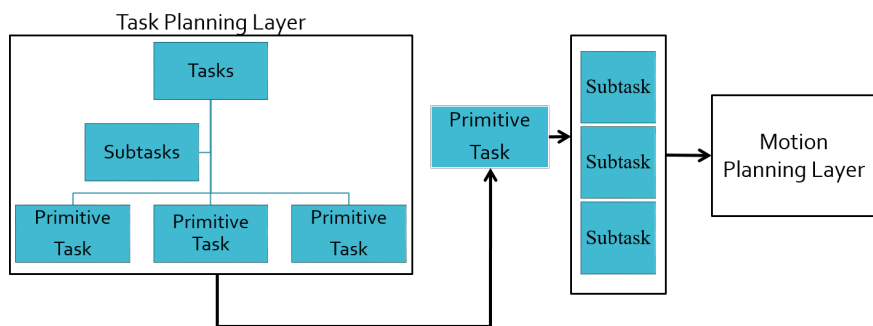


Figure A.8: A tree structure of the dual-fanuc-irmate200ic arms with gripper attached



**Figure A.9:** *Three fanuc arm environment*



**Figure A.10:** *Each action (primitive task) is introduced as a subtask to motion planning layer*

Robot Specifications		Robot Motion Speed		Robot Motion Range	
Axes:	6	J1	350 °/s (6.11 rad/s)	J1	±340°
Payload:	5.00kg	J2	350 °/s (6.11 rad/s)	J2	±200°
H-Reach:	704.00mm	J3	400 °/s (6.98 rad/s)	J3	388°
Repeatability:	±0.0200mm	J4	450 °/s (7.85 rad/s)	J4	±380°
Robot Mass:	27.00kg	J5	450 °/s (7.85 rad/s)	J5	±240°
Structure:	Articulated	J6	720 °/s (12.57 rad/s)	J6	±720°
Mounting:	Floor, Inverted, Angle				


  

Robot Controllers	Robot Applications
<a href="#">FANUC R-30iA Controller &gt;</a>	<a href="#">Arc Welding Robots &gt;</a>
<a href="#">FANUC R-30iA Mate Controller &gt;</a>	<a href="#">Assembly Robots &gt;</a>
	<a href="#">Foundry Robots &gt;</a>
	<a href="#">Milling Robots &gt;</a>
	<a href="#">Pick and Place Robots &gt;</a>
	<a href="#">Routing Robots &gt;</a>

**Figure A.11:** *Fanuc-LRMate200ic arm specifications*

**Panda**  
TECHNICAL DATA <sup>1,2</sup>

Arm	
degrees of freedom	7 DOF
payload	3 kg
sensitivity	torque sensors in all 7 axes
maximum reach	855 mm
joint position limits [°]	A1: -166/166, A2: -101/101, A3: -166/166, A4: -176/-4, A5: -166/166, A6: -1/215, A7: -166/166
joint velocity limits [°/s]	A1: 150, A2: 150, A3: 150, A4: 150, A5: 180, A6: 180, A7: 180
Cartesian velocity limits	Up to 2 m/s end effector speed
repeatability	+/- 0.1 mm (ISO 9283)
interfaces	<ul style="list-style-type: none"> <li>Ethernet (TCP/IP) for visual intuitive programming with Desk</li> <li>input for external enabling device</li> <li>input for external activation device or a safeguard</li> <li>Control connector</li> <li>Hand connector</li> </ul>
interaction	enabling and guiding button, selection of guiding mode, Pilot user interface
mounting flange	DIN ISO 9409-1-A50
installation position	upright
weight	- 18 kg
protection rating	IP30
ambient temperature	<ul style="list-style-type: none"> <li>+15°C to 25°C (typical)</li> <li>+5°C to + 45°C (extended) <sup>3</sup></li> </ul>
air humidity	20% to 80% non-condensing
Control	
interfaces	<ul style="list-style-type: none"> <li>Ethernet (TCP/IP) for Internet and/or shop-floor connection</li> <li>power connector IEC 60320-C14 (V-Lock)</li> <li>Arm connector</li> </ul>
controller size (19")	355 x 483 x 89 mm (D x W x H)
supply voltage	100 V <sub>AC</sub> - 240 V <sub>AC</sub>
mains frequency	47- 63 Hz
power consumption	<ul style="list-style-type: none"> <li>max. 600 W</li> <li>average - 300 W</li> </ul>
active power factor correction (PFC)	yes
weight	- 7 kg
protection rating	IP20
ambient temperature	<ul style="list-style-type: none"> <li>+15°C to 25°C (typical)</li> <li>+5°C to + 45°C (extended) <sup>3</sup></li> </ul>
air humidity	20% to 80% non-condensing

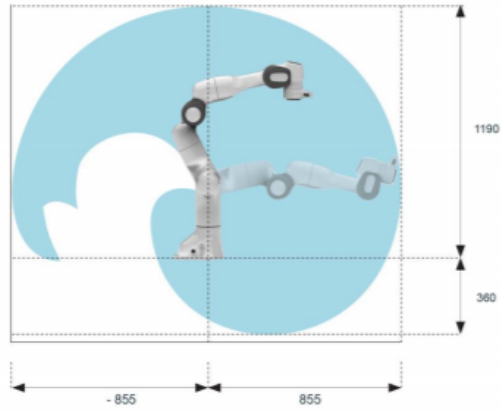
Hand	
parallel gripper	with exchangeable fingers
grasping force	continuous force: 70 N maximum force: 140 N
travel (travel speed)	80 mm (50 mm/s per finger)
weight	- 0.7 kg
Desk	
platform	via modern web browser
programming	visual & intuitive, dialog-based
Apps	can be composed into complex workflows to create Tasks and Solutions
CE out of the box solutions	
	
<p>Check <a href="http://www.franka.de">www.franka.de</a> to see if for your application a CE out of the box solution is available already. If so, you only have to go through an enclosed CE-checklist and you can start using Panda out of the box without performing an additional risk analysis.</p>	

<sup>1</sup> technical data is subject to change

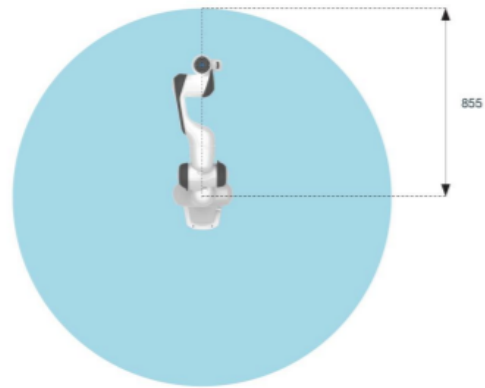
<sup>2</sup> if you have not purchased a Panda - CE out of the box solution, or don't comply with the CE-checklist the operator is responsible for the performance of a risk analysis and safe operation of the robot in accordance to its intended use and applicable standards and laws.

<sup>3</sup> performance can be reduced when operating outside the typical temperature range

Figure A.12: Panda arm specifications



1. Arm workspace side view



2. Arm workspace top view

**Figure A.13:** *Panda workspace views*