공학박사학위논문

Learning and Generalizing Complex Robot Motion Skills with Movement Primitives

모션 프리머티브를 이용한 복잡한 로봇 임무 학습 및 일반화 기법

2020년 8월

서울대학교 대학원
기계항공공학부
김 효 인

Learning and Generalizing Complex Robot Motion Skills with Movement Primitives

A Dissertation

by

HYOIN KIM

Presented to the Faculty of the Graduate School of

Seoul National University

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

Department of Mechanical and Aerospace Engineering

Seoul National University

Supervisor : Professor H. Jin Kim

AUGUST 2020

to my

FAMILY

with love

Abstract

Learning and Generalizing Complex Robot Motion Skills with Movement Primitives

Hyoin Kim
Department of Mechanical and Aerospace Engineering
The Graduate School
Seoul National University

Learning from demonstrations (LfD) is a promising approach that enables robots to perform a specific movement. As robotic manipulations are substituting a variety of tasks, LfD algorithms are widely used and studied for specifying the robot configurations for the various types of movements. This dissertation presents an approach based on parametric dynamic movement primitives (PDMP) as a motion representation algorithm which is one of relevant LfD techniques. Unlike existing motion representation algorithms, this work not only represents a prescribed motion but also computes the new behavior through a generalization of multiple demonstrations in the actual environment. The generalization process uses Gaussian process regression (GPR) by representing the nonlinear relationship between the PDMP parameters that determine motion and the corresponding environmental variables. The proposed algorithm shows that it serves as a powerful optimal and real-time motion planner among the existing planning algorithms when optimal demonstrations are provided as dataset.

In this dissertation, the safety of motion is also considered. Here, safety refers to keeping the system away from certain configurations that are unsafe. The safety criterion of the PDMP internal parameters are computed to check the safety. This safety criterion reflects the new behavior computed through the generalization process, as well as the individual motion safety of the demonstration set. The demonstrations causing unsafe movement are identified and removed. Also, the demolished demonstrations are replaced by proven

demonstrations upon this criterion.

This work also presents an extension approach reducing the number of required demonstrations for the PDMP framework. This approach is effective where a single mission consists of multiple sub-tasks and requires numerous demonstrations in generalizing them. The whole trajectories in provided demonstrations are segmented into multiple sub-tasks representing unit motions. Then, multiple PDMPs are formed independently for correlated-segments. The phase-decision process determines which sub-task and associated PDMPs to be executed online, allowing multiple PDMPs to be autonomously configured within an integrated framework. GPR formulations are applied to obtain execution time and regional goal configuration for each sub-task.

Finally, the proposed approach and its extension are validated with the actual experiments of mobile manipulators. The first two scenarios regarding cooperative aerial transportation demonstrate the excellence of the proposed technique in terms of quick computation, generation of efficient movement, and safety assurance. The last scenario deals with two mobile manipulations using ground vehicles and shows the effectiveness of the proposed extension in executing complex missions.

Keywords: Motion representation algorithm, Mobile manipulator, Manipulation planning, Learning from demonstration.

Student Number: 2014-21897

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Motivations

Robotic manipulations are substituting a wider variety of tasks. Various types of robot manipulators have been developed including the manipulators with mobile or ground vehicles. The manipulation parts also have been improved from single gripper [3], joints [3] or cable [4–6] to versatile robotic arm with multiple links [7–14]. The application of robot manipulators is no longer limited to fixed and structured environments in the past, according to the these development of robot manipulators. Now, the missions expand into unstructured environments where unknown situations arise.

It leaves a variety of challenges in terms of robot automation as robot manipulators increasingly applied to environments where various movement freedoms are exercised, not just do repetitive tasks. In addition to the essential requirement for the robot manipulator to complete the assigned task, the various properties of automation algorithms may be required. For example, more efficient movement generation and faster computational speed have been frequently addressed in the motion planning problem, such as optimal planning

[8, 15–18] or real-time planning [19, 20]. Moreover, there are a variety of issues faced by robot manipulators, such as via-points [15, 21, 22] or increased task complexity [23–26]. Assuming that robot operation is successfully controlled according to desired commands with excellent control technologies [7, 13, 14] that have been developed in recent years, the above problems can be adequately addressed in the stage of specifying the configurations of the robot. The problem of allocating robot configurations can be solved through existing motion planning techniques if it simply gives a starting point and a finishing point as a mission. However, the missions for robot manipulators often include a specific style of movement like stretching or twisting, which cannot be represented by starting or finishing points. That is, it may not complete the mission, which is the most basic requirement.

Recently, researchers have utilized learning from demonstration (LfD) [27–32] to specify configurations for robot manipulation. In the process of learning and generalizing motion from demonstrations, LfD can express tasks that require a specific style of motion as they are provided in the demonstrations while conventional planners cannot do. For example, via LfD stitching [30], knot tying [33] or swaying movements [31] can be represented, which are difficult to express formally and has a limitation in applying to other conventional planners. In particular, motion presentation algorithms among LfD techniques literally regenerate the movements provided by the demonstration, so if the demonstration has completed mission, the computed behavior will succeed. However, this application is limited to the environment where environmental settings do not change.

This dissertation is interested in resolving the following problems to assign the configurations of robot manipulators to carry out missions, including expressing certain style movements: (i) optimal movement, (ii) real-time adaptation, and (iii) motion safety. Moreover, (iv) complex assignments should be addressed through the proposed approach for practical applications in human life. Here, the environment can change continuously.

1.2 Literature Survey

This section provides the survey results of scholarly articles, books, and other materials relevant to this study. Most of the literature survey in this section refer to the papers [2,34].

1.2.1 Conventional Motion Planning in Mobile Manipulations

In terms of specifying the robot configurations, conventional planning approaches are investigated. Focusing on real-time and optimal motion generation, the relevant algorithms are offered in the following.

Optimization-based motion planning

Optimization techniques offer the explicit guarantee of the local optimality of the planned trajectory. Several constraints can be considered in a unified way so that the dynamic and kinematic feasibilities of the planned trajectory are ensured for mobile manipulation system.

As a straightfoward optimization-based motion planning, nonlinear programming (NLP) can be utilized. One easy way to formulate NLP for the mobile manipulation system is to assume a continuous-time trajectory of a specific representation such as a polynomial, B-spline, or Bézier curve using flat outputs. By doing so, the trajectory optimization problem is converted into a static nonlinear optimization problem on coefficients of the corresponding representation [15]. From this conversion, the search space of the optimization is confined to a span of bases defined in the representation space.

However, a simple NLP is not appropriate for generating suitable trajectories when the dynamics and constraints of the target system get complicated. The main reason is that a solution which satisfies complex constraints may not exist in the restricted search space. Especially for cooperative mobile manipulation, its complex dynamics and constraints further reduce the feasible search space for planning.

Another approach for trajectory optimization is model predictive control (MPC) [16],

[17]. MPC seeks the optimal trajectory from the current state to the target state over a finite time horizon, and the first optimal input is applied to the system. Although its computation time, which increases significantly as the dimension of the system increases, can be adjusted by modulating the length of the time horizon, the convergence domain becomes drastically contracted if the system is highly nonlinear or has many constraints. Also, the performance of MPC is highly dependent on the initial guess because underlying optimization algorithms used in every step usually give the local optimum. On the other hands, iterative linear quadratic regulators (iLQR) [35] can be implemented to generate feasible and optimal trajectories for robot manipulator [36]. Still, the representation of constraints is difficult to formalize the problem. In short, optimization-based motion planners have a possibility that they will not compute the global optimal solution, and thus they are sensitive to the initial guess. Besides, they cannot give a solution in a short time.

Sampling-based motion planning

Sampling-based techniques (SBP) [37] use random computations instead of solving difficult problems in motion planning. They utilize random sampling techniques and build a series of waypoints, providing a fast solution even for high-dimensional and constrained problems. Also, they need only the criterion to determine appropriate waypoints, making the algorithm is straightforward and easy to apply. Recently, with successful implementations of the optimizing process, robotic motion planning accelerates the utilization of SBP even to secure asymptotic optimality without converging to local minima.

Rapidly exploring random trees (RRTs) [38–40] is a widely used SBP algorithm thanks to the ability that it does not depend on the explicit representation of obstacles [1, 8]. RRT star (RRT*) [18], which is an extended form of RRT, has simple optimizing processes which connect or reconnect the waypoints as they improve the cost in a selected local area. By repeating those processes in every iteration, the algorithm guarantees asymptotic optimality.

For the cooperative mobile task, the planning process should handle the multiple kine-

Approach	Methods	Computing speed	Motion completion	etc
Optimization-based techniques	iLQR [35], NLP [15], MPC [16, 17]	Light or heavy	Hard formulation	Safety or complex task can be configured with constraints but cause Heavy computing load
Sampling-based optimal planners	RRT* [18], PRM* [18]	Heavy	Limited formulation (point-wise)	
Motion representation algorithms (optimal motion primitives)	DMPs [41], PDMPs [31]	Light	Easy formulation	Additional formulation is necessary to ensure safety

Table 1.1: Summary of optimal motion generation algorithms.

matic and dynamic constraints which are inherent from the interaction between agents. Using RRT* or other optimal planners, the solution is given within a reasonable time without worrying about local minima or difficult problem formulation. Moreover, the dynamics of the mobile manipulator can be considered in the local planning of RRT* in that the trajectory is computed using the dynamics of mobile manipulators and their properties such as actuation bounds. Also, RRT* consequently updates the connection of sampled nodes into a better one. During this process, RRT* asymptotically finds the optimal pose of each mobile manipulator along the trajectory, which allocates proper payload distribution.

RRT* is known to often demand much longer computational time to achieve a close-to-optimal solution than the time to obtain the initial solution. The recent research regarding both optimization and sampling based motion planning has improved the rate of convergence by utilizing smart sampling or developing the simple formulation of the problem. Still, neither optimization-based nor sampling-based planning alone is satisfactory for cooperative mobile tasks that require a fast solution to react appropriately to sudden risky situations.

1.2.2 Motion Representation Algorithms

Motion representation algorithms have been utilized to reproduce a given trajectory robustly. Among them, dynamic movement primitives (DMPs) [41–44] are powerful motion representation techniques in that they demand a low computational load during task execution as model-free approaches. By generating the trajectory from a prescribed movement,

DMPs can serve as a quick motion representation algorithm. In DMPs, the forcing term is computed to follow the demonstration in given DMPs equations. By using the supervised learning to the forcing term, the robotic system can quickly modify the trajectory so that the original demonstration can be robustly followed from perturbed configurations.

Some researchers utilize DMPs as tools for a motion generator within obstacles [29, 31, 45, 46]. In [29], [45], the new equations of DMPs are suggested including the term of repulsive potential fields, which guide the system to the opposite side from obstacles. However, the introduction of potential fields may cause inefficiency or degrade the optimality of the original movement primitives, and further, the possibility of local minima in potential fields still remains. On the other hands, via-points modulations can be used to avoid obstacles if via-points to avoid obstacles are specified. In [46], DMPs are combined with probabilistic movement primitives [47] to directly adapt to intermediate via-points such as configurations of collision-free state. This formulation has a limitation that the configuration of robot must be specified in advance for obstacle avoidance. Another approach for obstacle avoidance is shown in [31, 48], using parametric-DMPs (PDMPs) which allow the movement generalization by extracting features from multiple demonstrations. In [31], PDMPs provide the parametrized avoidance movements to new heights of obstacles. Similar models of PDMPs appear as task-parametrized [49–51] or stylistic form [52]. For those frameworks, finding parameters or styles is an important issue in order to produce suitable motion in a new circumstance.

In [53], successful combination of RRT* and DMPs are addressed. RRT* and DMPs are dealt in seperated manner where RRT* generates the trajectory avoiding known obstacles as an off-line motion planner while the DMPs technique with potential fields is applied to avoid unknown obstacles. However, the limitation exists as the optimality is lost when the environmental set-ups are significantly changed. Still, the successful combination of RRT* and DMPs can address the trade-off issue between motion optimality and quick reaction(or adaptation). Moreover, the robustness of DMPs is advantageous in the presence of perturbations such as internal forces between cooperating agents or their out-of-sync

movements.

1.2.3 Safety-guaranteed Motion Representation Algorithms

The performance of the resulting motion in motion representation algorithms depends on how appropriately demonstrations are provided for a given scenario. In other words, when the extracted motion primitives are not enough to generalize the desired motion in a given scenario, the computed motion may end up with the poor results. Thus, the skill of collecting suitable demonstrations for generalizing target scenario is important for motion representation algorithms. Moreover, the technique to identify safety-guaranteed demonstrations which provide safe primitives can be applied to the problem of deciding whether to reuse the demonstrations or not.

There are some related works regarding reasoning when the robot needs updated demonstrations [24, 54, 55]. In [55], the uncertainty of the trajectory is measured using the Gaussian process. When the uncertainty is higher than a certain level, a new demonstration is added. In [54], the update criterion is determined based on the information gain from the new data. Here, the information gain is the value that indicates the sufficiency of the demonstrations. [24] performs incremental learning for the success of task execution. During the task, each grasping motion and movement of end-effector is corrected by physical interaction with the object and by human, respectively. Above works focus on enhancing the performance by incrementally providing the demonstrations. For safety-critical situations, these criteria are not enough to determine which of the given demonstrations are bad and should be eliminated from the data-set.

1.3 Research Objectives and Contributions

The objective of this dissertation is to develop an integrated framework based on parametric dynamic movement primitives (PDMPs) so as to compute configurations of robot manipulators that performs the assigned mission. In particular, this work is interested in resolving

Methods	Initial/terminal offset generalization	Style generalization	Parameter selection process	Demonstration update	Safety guarantee
[29, 56]	O	X	X	X	X
[31, 49, 50]	O	O	X	X	X
[55]	O	X	X	O(Performance)	X
[24, 54]	O	X	X	O	O
The proposed	O	O	O	O	O

Table 1.2: The objective conditions in representative motion representation algorithms and the proposed algorithms.

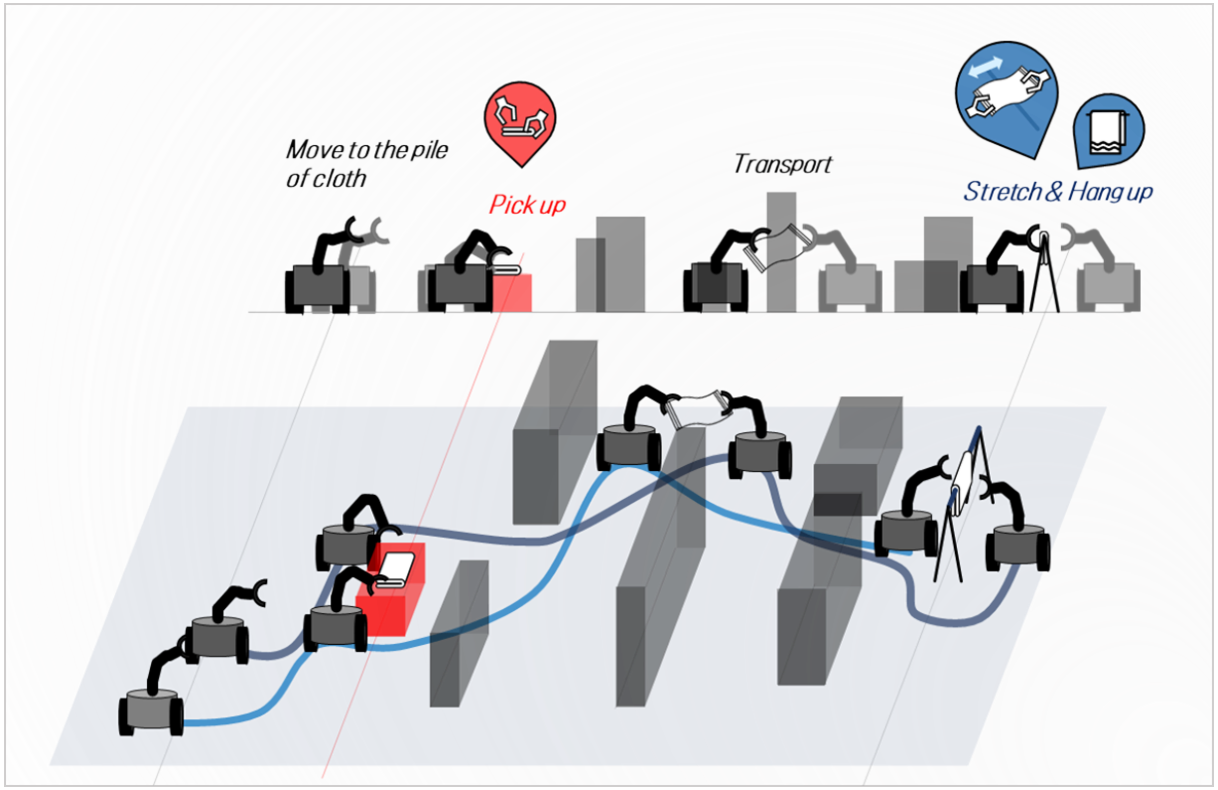


Figure 1.1: Cooperative mobile manipulation to hang a piece of cloth over the hanger (hang-dry mission). The hang-dry mission consists of sequential sub-tasks such as moving, grasping, and stretching. Each mobile manipulator starts to move to the cloth, first. When they reach the cloth, they grasp it cooperatively. While they approach a hanger, the robots should avoid multiple obstacles. Near the hanger, one robot moves to the opposite side of the hanger to easily stretch and hang up the cloth over the hanger.

the aforementioned problems in section 1.1. This dissertation concentrates on three parts in utilizing motion representation algorithm: (i) motion generalization, (ii) safety guarantee for successful operation, and (iii) applications to complex missions.

1.3.1 Motion Generalization in Motion Representation Algorithm

Objectives

This work first develops a base motion representation algorithm including the motion generalization process. The various styles of motion can be appropriately computed by learning multiple demonstrations. From the generalization process, the proper parameters in PDMPs and resulting motion are calculated corresponding to the new environment, even for the environment is never provided as a demonstration.

Contributions

- This dissertation presents a GPR formulation to express the implicit relationship between the parameters in PDMPs and environment. This formulation give reliable values for DDMP parameters to successfully execute the mission. The proposed framework serve as a powerful optimal planner. The performance is validated by comparing simulation results in terms of computational time and performance index with sampling-based planners.

1.3.2 Motion Generalization with Safety Guarantee

Objective

Second, this work presents the process for managing and improving the demonstration set to ensure the safety of motion. This process calculates the safety criterion in the PDMPs parameter value, and use it to identify demonstrations causing unsafe motion. By eliminating the unsafe demonstrations and adding safe ones, computed motion always ensures safety.

Contributions

- This work deduces the safety criterion for the parameter in the PDMP framework via optimization. Here, under the PDMPs dynamics, the parametric optimization problem is formulated.
- From the safety criterion, this work also suggest the process that manage demonstrations autonomously by directly eliminating the unsafe demonstrations. New demonstrations are provided in order to secure the safety of motion.
- Simulation results are given and show that the proposed process is also applicable where a new scenario is given and the previous demonstrations can be reused instead of computing all the demonstrations again.

1.3.3 Motion Generalization for Complex Missions

Objective

Third, this dissertation provides an extended approach that can efficiently reduce the number of required demonstrations for generalizations. In particular, this work is useful to the missions consisting of multiple sub-tasks.

Contributions

- This extension gives a new approach for configuring multiple PDMPs for each sub-task and autonomously selecting them according to the situation. This work effectively reduces the number of required demonstrations by eliminating the need to provide all combinations of sub-tasks in single PDMPs algorithm on the entire mission.

- This work also also proposes two processes that use Gaussian process regression (GPR) to assign generalized execution time and regional goal state in each sub-task, respectively. These processes enable flexible sub-task sequencing across the entire mission, even if the PDMPs of sub-tasks are configured independently.

- Various simulation results validate that the proposed extension is effective not only in reducing the number of required demonstrations but also in improving the generalization performance of each sub-task compared to applying a single PDMPs algorithm.

- This extension also show that the proposed framework can be extended to new scenarios by efficiently reusing sub-tasks, obviating the need to provide all demonstrations again.

1.4 Thesis Organization

The remainder of the thesis is organized as follows. Chapter 2 describes the necessary materials for this dissertation including DMPs and experimental setups for mobile manipulation systems. Chapters 3 and 4 describe the the proposed algorithms for generalizing and managing safety-guaranteed demonstrations. Chapter 5 presents the applicable extension of the proposed approach when performing complex missions with several sub-tasks. Chapter 6 verifies the algorithms by presenting the results of the actual experiments. Chapter 7 summarizes the papers.

2

Background

This section provides the background for this research, including DMPs, the mobile manipulation systems, and experimental setups.

2.1 DMPs

DMPs, proposed in [41,42], have received attention as model-free approaches which demand low computational load during task execution. DMPs can represent complex movements with incorporating sensory feedback in real-time. Detail about DMPs are described in [41, 42]. Here, basic information for DMPs are addressed only based on the formulation used in [29].

DMPs are defined based on the following attractor dynamics as,

$$\begin{aligned} \dot{\mathbf{v}}^d &= K_p(\mathbf{g}^d - \mathbf{q}^d) - K_D\mathbf{v}^d + K_p\mathbf{f}(\chi; \mathbf{w}) \\ \dot{\mathbf{q}}^d &= \mathbf{v}^d \end{aligned} \tag{2.1}$$

where $\mathbf{q}^d$ and $\mathbf{v}^d$ are the vectors of state variables and their time derivatives. $\mathbf{g}^d$ is the goal states for the system. Each K_p and K_d is the spring and damping coefficient in this dynamics. Superscript d is used to denote the variables are the state by DMPs. $\mathbf{f}(\chi; \mathbf{w})$ is a forcing term that leads the system to goal states $\mathbf{g}^d$. This forcing term describe the nonlinear motion as

$$\mathbf{f}(\chi; \mathbf{w}) = \frac{\sum_{i=1}^{N_w} \mathbf{w}_i \Phi_i^d(\chi)}{\sum_{i=1}^{N_w} \Phi_i^d(\chi)} \chi. \tag{2.2}$$

$\mathbf{w}_i$ is the weight of each basis function. The exponential basis function $\Phi_i^d(\chi)$ for $i = 1, ..., N_w$ can be defined to

$$\Phi_i^d(\chi) = \exp\left(-\frac{(\chi - d_i)^2}{2\sigma_i^2}\right), \tag{2.3}$$

where d_i and σ_i denote constants which determine the width and center of the basis function, respectively. $\mathbf{f}(\chi; \mathbf{w})$ does not depend on time as shown in (2.2). Instead, it depends on a external variable χ, which varies from 1 to 0 during a movement. The dynamics of χ is defined as

$$\dot{\chi} = -K_\chi \chi, \tag{2.4}$$

where K_χ is a constant.

2.2 Mobile Manipulation Systems

This work considers the mobile manipulators with two types of mobile platforms, ground and aerial vehicle. The manipulation parts of those manipulators are specified to robotic arm with multiple links, to operate dexterous manipulations. The tip of robotic arm is

equipped with a gripper that rigidly grasp the object. System variables are also configured for cooperative works.

2.2.1 Single Mobile Manipulation

The configuration space of single robot manipulator can be represented as $[\mathbf{p}_b^\top, \Phi^\top, \eta^\top]^\top$, where $\mathbf{p}_b = [x_b, y_b, z_b]^\top$ denotes the body position and $\Phi = [\phi, \theta, \psi]^\top$ represents the Euler angle. The manipulation part is specified to robotic arm so the variables $\eta = [\eta_{i,1}, ..., \eta_{n_m,2}]^\top$ indicates the set of arm joint angles for n_m number of links.

In the case of robot manipulator with ground vehicle, $\mathbf{p}_b$ and Φ is simplified to $\mathbf{p}_b = [x_b, y_b]^\top$ and $\Phi = \psi$. On the other hands, multi-rotor based aerial manipulator uses full states of above configuration space. The configuration space of the both ground-vehicle and aerial manipulators are shown in Fig. 2.1.

For those manipulation systems, the actual operation is performed through the position of the end effectors. The positions of the end effector in both types of robot manipulators are calculated as

$$\bar{\mathbf{p}}_e = g_{n_m}(\eta_{n_m})_1(\eta_1)g_0(\Psi)\bar{\mathbf{p}}_b \tag{2.5}$$

where g_i's for $i = 0, 1, ..., n_m$ are the transformation matrices in [57]. $\bar{\mathbf{p}}$ denotes a vector defined as $\bar{\mathbf{p}} = [\mathbf{p}^\top \ 1]^\top$.

2.2.2 Cooperative Mobile Manipulations

Cooperative manipulation systems are identified by addressing the constraints resulted from the motion of each robot manipulators. This section shows two cooperation states from two robot manipulators holding hard and elastic objects. The experiments in this dissertation deal both types of objects are handled in cooperation systems.

Rigid Object

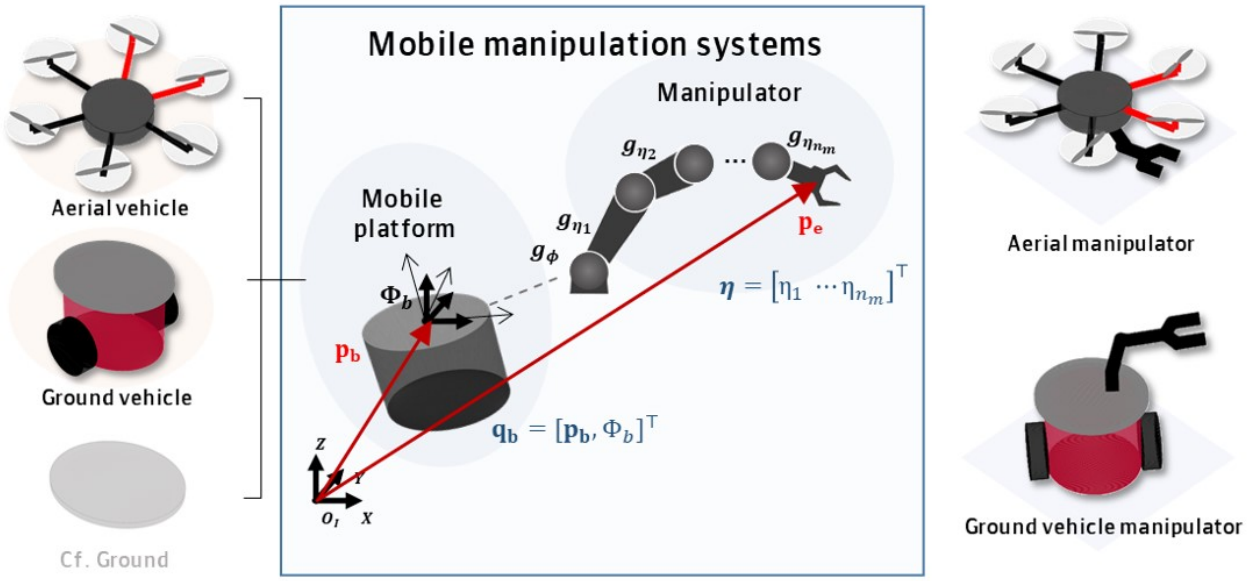


Figure 2.1: Various types of mobile manipulation systems and their coordinates.

When robot manipulators cooperatively transports a common object with rigid body with the rigid grasping condition, those robot manipulators and the object can be considered as one combined system. The cooperative system in cooperation can be described by the collection of state variables of each robot manipulators.

The configuration space of each robot manipulator can be represented as $\mathbf{q}_i = [\mathbf{p}_i^\top, \Phi_i^\top, \eta_i^\top]^\top$ for $i = 1, 2$, where $\mathbf{p}_i = [x_{b,i}, y_{b,i}, z_{b,i}]^\top$ denotes the body position and $\Phi_i = [\phi_i, \theta_i, \psi_i]^\top$ represents the Euler angle of i-th body platform. $\eta_i = [\eta_{i,1}, ..., \eta_{i,n_m}]^\top$ indicates the vector of joint angles of robotic arm as denoted in the right of Fig. 2.2. Subscript i represents the variable of the i-th aerial manipulator. The configuration space of the object is represented as $[\mathbf{p}_o, \Phi_o]^\top$.

In order to express the configuration space of the combined system, kinematic con-

straints from the grasping system are listed as below:

$$\begin{aligned} \mathbf{p}_{c_i} &= g_{o,i}\mathbf{p}_o \\ \mathbf{p}_{e_i} &= g_{i,n_m} \cdots g_{i,1} g_{i,\psi} \mathbf{p}_i \\ \mathbf{p}_{e_i} &= \mathbf{p}_{c_i}, \end{aligned} \tag{2.6}$$

where, $\mathbf{p}_o, \mathbf{p}_i, \mathbf{p}_{e,i}$ and $\mathbf{p}_{c,i}$ indicate the x, y and z positions of the center of object, body of the mobile platform, end-effector and tip of object. $g_{i,1}, \cdots, g_{i,n_m}$ and $g_{i,\Phi}$ denote the transformation matrices, which calculate the positions resulted by joint angles and Euler angle of robot manipulator. $g_{o,i}$ denotes the transformation matrix from the center of the object to the tip. The experiment in section 5.1 regards cooperative aerial transportation with rigid object. The experiments in this dissertation deal both types of objects are handled in cooperation systems. The property of differential flatness in the multi-rotor enables us to represent roll and pitch angles with other state variables and their time derivatives. From this property and (2.6), the configuration space of combined system can be represented as $\mathbf{q} = [x_o, y_o, z_o, \psi_o, \eta_{1,1}, \eta_{1,2}, \eta_{2,1}]^\top$. Here, the dimension of the configuration space has been reduced with observation that the grasping condition allows to know $\eta_{2,2}$ from $\eta_{1,1}, \eta_{1,2}$ and $\eta_{2,1}$.

Elastic Object

When the robot manipulators holding the elastic object, the tips of manipulators are freely move within the limited distance, the maximum length of the grasping parts of object. So during operation, these operation limit is added to check the cooperative stability. The experiment in section 5.2 deals with hang-dry work with elastic object, a cloth.

2.3 Experimental Setup

Two cooperative manipulation systems are employed to validate the proposed approach in this dissertation. This section describes the experimental setups.

2.3.1 Test-beds for Aerial Manipulators

The overall hardware setup of the aerial manipulator is shown in Fig. 2.3. Two identical multi-rotors are employed, and each is equipped with a 2-DoF robotic arm. In aerial vehicle, DJI E800 motors are controlled by 620S electronic speed controllers (ESCs). For onboard computation, Intel NUC7i7BNH running Ubuntu 14.04 and ROS Indigo is used. Also, a Pixhawk autopilot is attached at the center of the fuselage and connected to the onboard computer via USB.

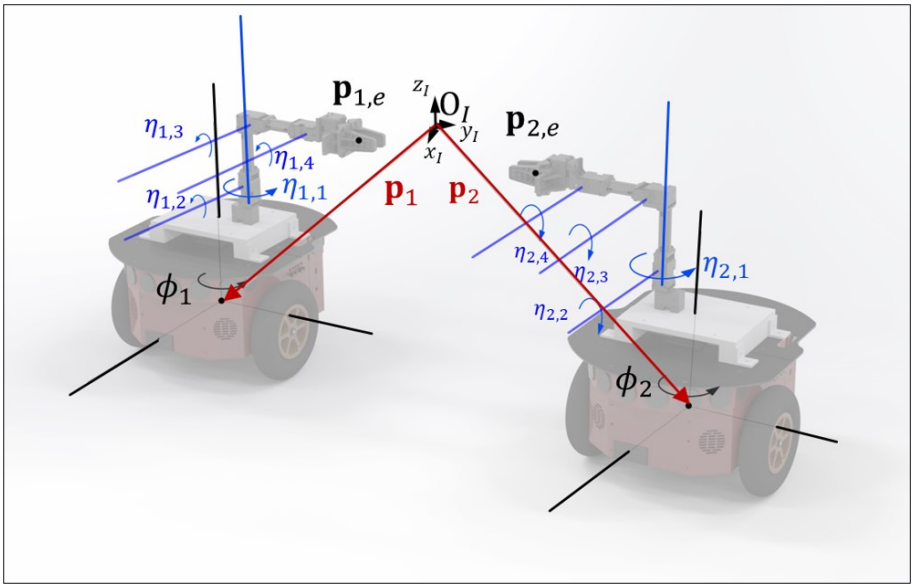
A trajectory tracking controller is a slightly modified version of [8] is used as a high-level controller to generate the desired net thrust and angular velocities. It is aimed to control the position and heading of each aerial manipulator in a decentralized manner. These set-points are sent to the autopilot, and customized the PX4 firmware computes the desired PWM signal of each motor by a low-level control algorithm [14] which tracks the desired angular velocities. For indoor tests, a motion capture system (VICON) is used. The ground control station receives pose measurements at 100 Hz, and sends them to the onboard computer via wireless communication. Lastly, MX-106 and MX-28 servos from ROBOTIS constitute the 2-DoF manipulator and are connected to the onboard computer via USB. An RGB-D or stereo camera can be used for detecting surroundings such as existence or location of the obstacle.

2.3.2 Test-beds for Robot Manipulators with Ground Vehicles

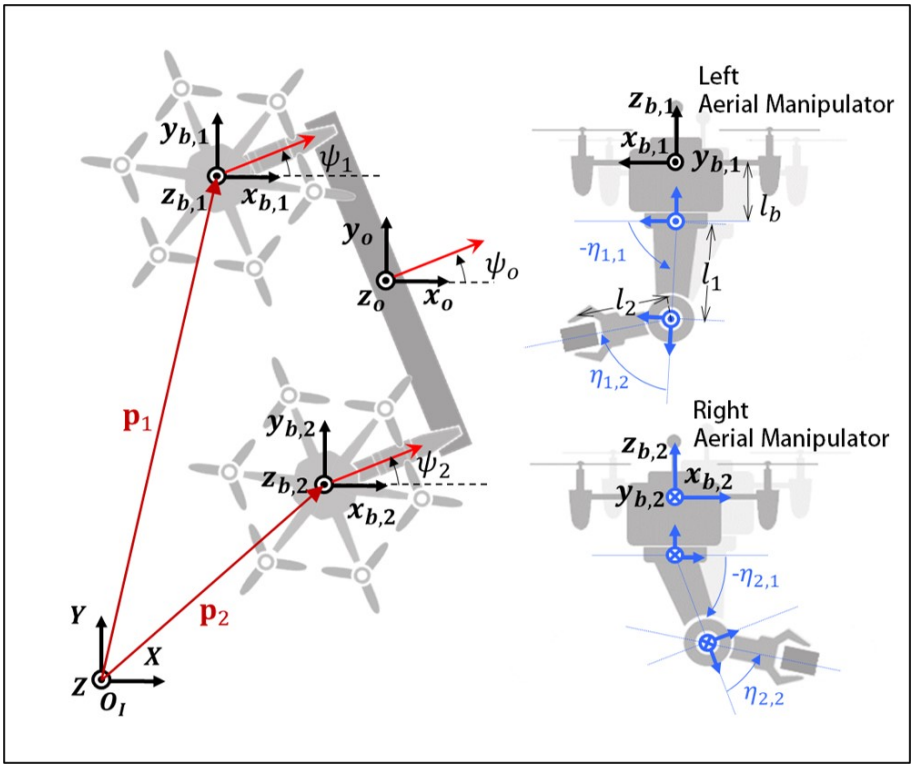
In experimental setups, each robot system consists of a computer, a ground robot, and a manipulator as described in section 2.2. Pioneer 3-DX is employed as a ground robot

platform. The ground robots are equipped with OpenMANIPULAOTR-X from ROBOTIS. Each manipulator is mounted to the middle top of a pioneer with the y-directional offset towards counterpart pioneer. For on-board computation, Intel NUC7i7BNHXG is used to operate Ubuntu 16.04 and ROS Kinect. The overall system is operated on ROS. Ground station is set to Intel Core i5-8256U laptop.

During the task, the ground robots are controlled to follow calculated trajectories. Similar to aerial transportation experiments, present positions are achieved by using the motion capture system, VICON. From the VICON system, current environmental information and the robot states are transferred to VICON computer. Using the proposed framework, the trajectories are calculated from the sensory feedback and are transmitted from ground station to the on-board computer via wireless communication. PI controller is designed to calculate control command from present positions and desired positions. The manipulator is connected to the on-board computer with USB and operated by proportional control based on joint position measurements. The low-level controller is implemented in C++. The control architecture is described in Fig. 2.4.



(a)



(b)

Figure 2.2: Configuration of the coordinates for the combined systems. (a) Cooperative mobile manipulation system with two mobile manipulators consisting of ground-vehicle and a 4-DOF robotic arm. (b) Cooperative aerial manipulation system with two aerial aerial manipulator consisting of a hexacopter and a 2-DOF arm.

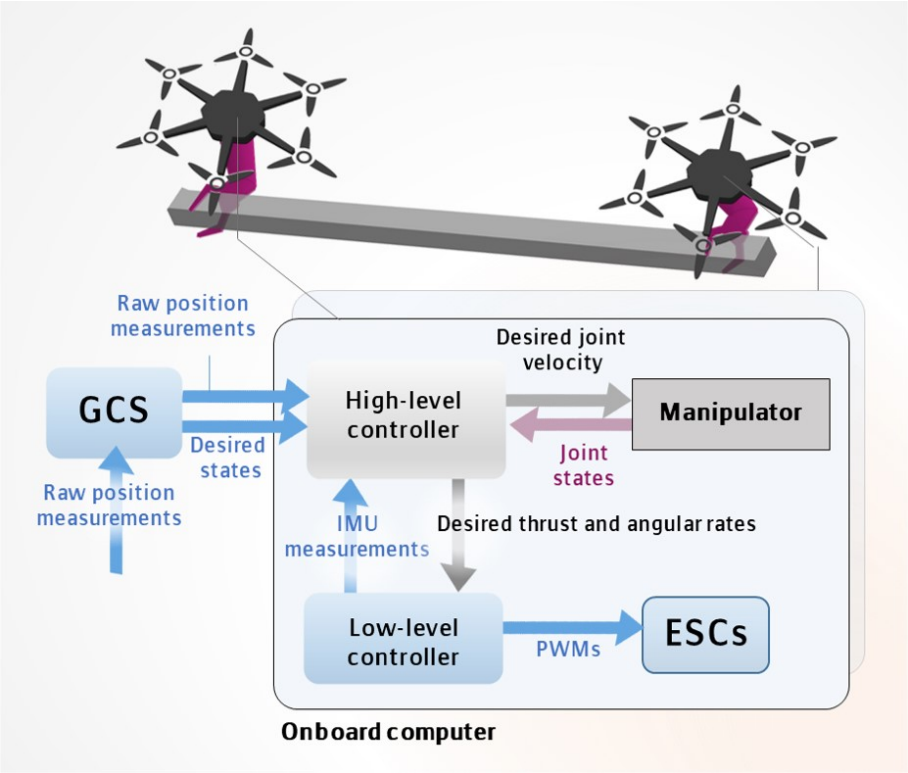


Figure 2.3: Experimental setups for cooperative aerial manipulation system in Fig. 2.2b.

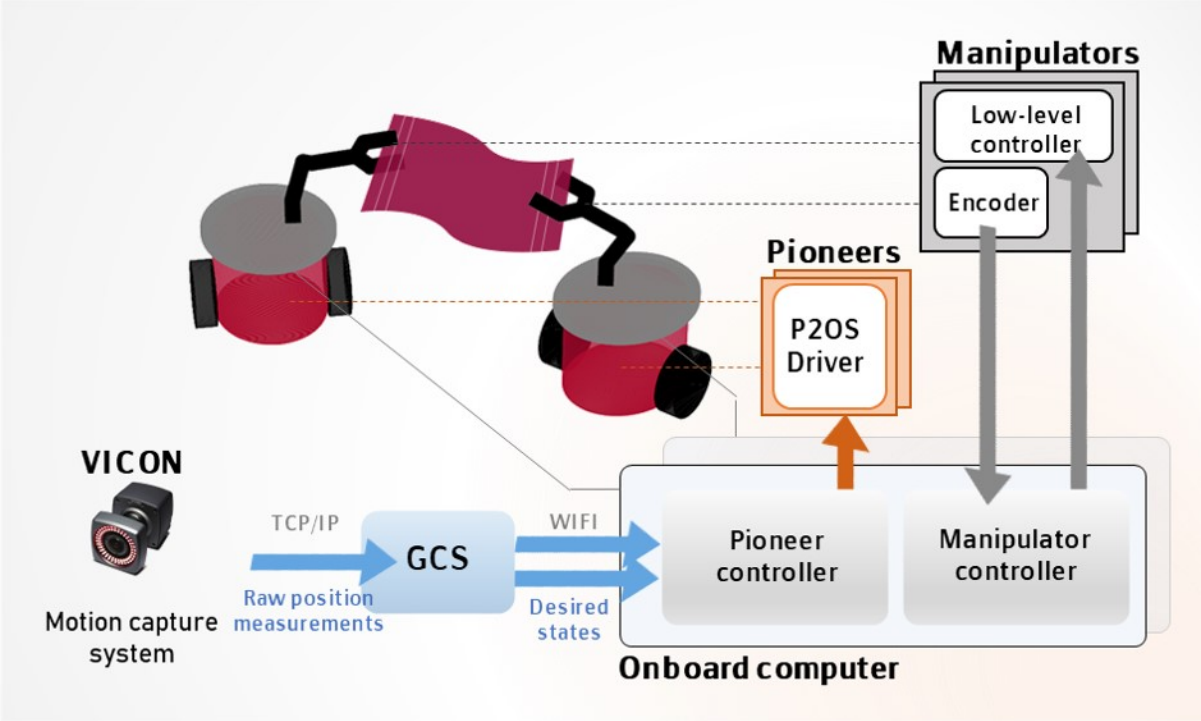


Figure 2.4: Experiment set-up for the hang-dry work in Fig. 2.2a.

3

Motion Generalization in Motion Representation Algorithm

In this section, an overview of the proposed framework is provided. PDMPs in [31] are described, and including the proposed generalization process, the modification of the framework is discussed in detail.

3.1 Parametric Dynamic Movement Primitives

Single demonstration in a specific environment can be reproduced using DMPs, but it is not sufficient for generalization in different environments. Although DMPs allow a certain level of initial or terminal offset [41, 58], their performance is degraded at large offsets. Moreover, DMPs cannot generalize motion styles themselves. In order to generalize the various movement styles to adapt to situations including large offset, DMPs can be modified to use multiple demonstrations that include all the required movements. The parametric-DMPs (PDMPs) [31], the extensions of DMPs, adopt style-variable modeling to DMPs. With the parametric skill, PDMPs have an advantage of generalizing motion from multiple

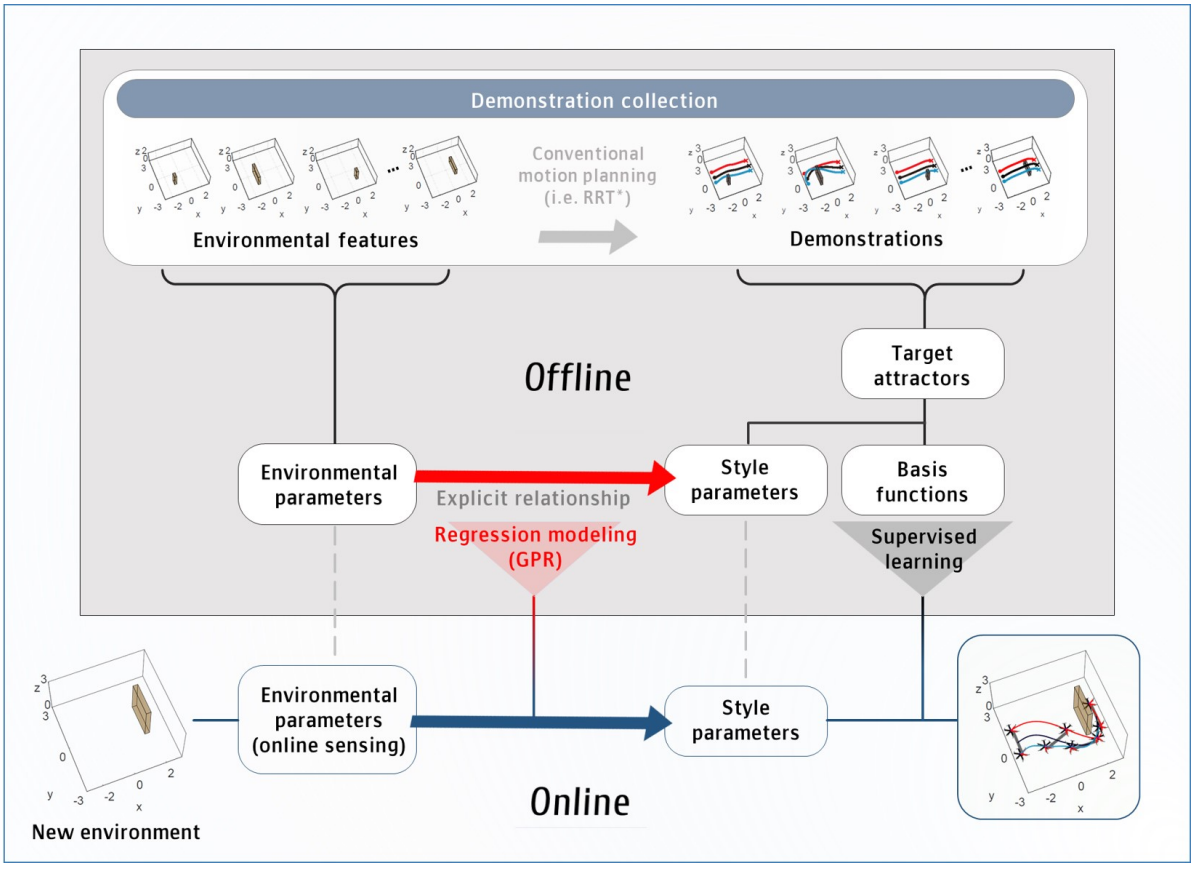


Figure 3.1: An overview of the proposed PDMPs framework with parameter selection process for cooperative aerial transportation.

demonstrations. Through the generalization, PDMPs generate the new movement that has never been learned.

PDMP has a structure that can generate various paths through a parameterized structure, but it is not known which value is suitable for a situation. In this work, an integrated framework is suggested to describe the process of selecting parameters by revealing the relationship between situations and PDMP parameter values. In Fig. 3.1, the overview of the proposed learning-based planning algorithm is shown. Except for the proposed generalization process, the detailed descriptions of PDMPs are presented as follows.

First, the equation of DMPs [29] is represented as

$$\alpha_3\ddot{\mathbf{q}} = \alpha_1(\mathbf{g} - \mathbf{q}) - \alpha_2\alpha_3\dot{\mathbf{q}} - \alpha_1(\mathbf{g} - \mathbf{q}_0)\chi + \alpha_1\mathbf{f}(\chi; \mathbf{w}), \tag{3.1}$$

where $\mathbf{q}$ denotes the state vector of the system, and $\mathbf{g}$ and $\mathbf{q}_0$ the goal and start configuration, respectively. In (3.1), χ and $\mathbf{f}(\chi; \mathbf{w})$ are introduced to make $\mathbf{q}$ asymptotically converge to the unique point $\mathbf{g}$. Here, χ is the phase variable whose dynamics is described as $\dot{\chi} = -K_4\chi$ in [56], which forms a canonical system. K_4 is the constant value. $\chi = 1$ indicates the start of the time evolution and χ close to zero means that the goal $\mathbf{g}$ has essentially been achieved. $\mathbf{f}(\chi; \mathbf{w})$ is an attractor function which are previously called the forcing term. The attractor function $\mathbf{f}$ forces the current state variable to the demonstrated trajectories by learning the weights $\mathbf{w}$. α_1, α_2, and α_3 are the time constants and $\mathbf{w}$ is the weight to be learned.

The difference of PDMPs from DMPs lies in formulating the attractor function $\mathbf{f}(\chi; \mathbf{w})$. In DMPs, the attractor function is formulated by supervised learning of the target attractor function, where the target attractor functions are computed by substituting $\mathbf{q}$ with demonstrated states $\mathbf{q}_d$ in (3.1). On the other hand, in PDMPs, to consider the multiple demonstrations in a unified framework, the style parameters and basis functions are introduced to the attractor function as

$$f_n(\chi, \mathbf{s}_n; \mathbf{w}_n) = \mathbf{s}_n^\top \mathbf{b}_n(\chi; \mathbf{w}_n). \tag{3.2}$$

The subscript $n = 1, \cdots, N$ indicates that the variable is concerned with the n-th state variable, where N is the number of state variables. Here, $\mathbf{s}_n \in \mathbb{R}^{L_n \times 1}$ denotes the *style parameter* corresponding to the n-th state variable and $\mathbf{b}_n(\chi; \mathbf{w}_n) = [b_n^1(\chi; \mathbf{w}_{n,1}), \cdots, b_n^{L_n}(\chi; \mathbf{w}_{n,l})]^\top \in \mathbb{R}^{L_n \times 1}$ represents the set of basis functions. The basis functions can be viewed as characteristic modes of movement that composes the movements to be generalized obtained from the whole demonstrations. L_n is the dimension of $\mathbf{s}_n$ to be explained below. In PDMPs, supervised learning is applied only to $\mathbf{b}_n(\chi; \mathbf{w}_n)$.

The target basis function and the corresponding style parameter are obtained as follows. Let $\mathbf{f}^m_{n,target} \in \mathbb{R}^{\bar{L}\times 1}$ for $m = 1, \cdots, M$ denote the series of target attractor functions along a trajectory, where $\bar{L}$ is the total number of time steps along the discrete-time trajectory and M is the number of demonstrations. The target matrix for the M demonstrations is formulated as $\mathbf{F}_n = [\mathbf{f}^1_{n,target}, \mathbf{f}^2_{n,target}, \cdots, \mathbf{f}^M_{n,target}]^\top \in \mathbb{R}^{M\times\bar{L}}$. The set of style parameters $\mathbf{S}_n$ and target basis functions $\mathbf{B}_n$ are obtained by taking the singular value decomposition (SVD) to the target matrix as,

$$\mathbf{F}_n = \mathbf{U}_n \Sigma_n \mathbf{V}_n^\top = \mathbf{S}_n^\top \mathbf{B}_n. \tag{3.3}$$

Here, $\mathbf{S}_n^\top = [\mathbf{s}_n^1, \mathbf{s}_n^2, \cdots, \mathbf{s}_n^M]^\top \in \mathbb{R}^{M\times L_n}$ is the first L_n columns of $\mathbf{U}_n$. The basis functions is composed to $\mathbf{B}_n = [\mathbf{b}^1_{n,target}, \mathbf{b}^2_{n,target}, \cdots, \mathbf{b}^{L_n}_{n,target}]^\top \in \mathbb{R}^{L_n\times\bar{L}}$ with the first L_n rows of $\Sigma_n \mathbf{V}_n^\top$. Each column of $\mathbf{B}_n$, i.e. $\mathbf{b}^l_{n,target}$, is the target vector of $\boldsymbol{\beta}^l_n(\chi_1; \mathbf{w}_{n,l}) = [b^l_n(\chi_1; \mathbf{w}_{n,l}), \cdots, b^l_n(\chi_{\bar{L}}; \mathbf{w}_{n,l})]^\top \in \mathbb{R}^{\bar{L}}$ for $l = 1, \cdots, L_n$, where χ_j for $j = 1, \cdots, \bar{L}$ is calculated from the dynamics of χ. The number L_n is selected by the singular value spectrum such as $\sum_{l=1}^{L_n} \sigma_{n,l} / \sum_{m=1}^{M} \sigma_{n,m} > 0.9$, where $\sigma_{n,m}$ indicates the m-th diagonal element of Σ_n. In each n-th variable, the weights for the l-th element of the basis function is obtained by supervised learning as below:

$$\mathbf{w}^*_{n,l} \leftarrow \arg\min_{\mathbf{w}_{n,l}} ||\mathbf{b}^l_{n,target} - \boldsymbol{\beta}^l_n(\chi; \mathbf{w}_{n,l})||^2. \tag{3.4}$$

The locally weighted regression is used for obtaining the weights of basis function $\mathbf{b}(\chi; \mathbf{w})$ as it is a non-parametric technique which has a low computational complexity and determines the necessary weights $\mathbf{w}$ automatically. Thus, in each demonstration, the specific set of style parameters and the weights for basis functions are obtained. Since basis functions map a subspace of all the training trajectories, the required movements corresponding situations are generalized by choosing the appropriate style parameters based on the environmental information.

3.2 Generalization Process in PDMPs

In order to reveal the relationship between the environment and the corresponding optimal motion, an explicit function is introduced, which is called *mapping function*. The mapping function is represented by solving a regression problem between *environmental parameters* and style parameters, where the environmental parameters are representative environmental information. Since the style parameter plays a dominant role in extracting the control policies, constructing the mapping function is a crucial part. In this section, the generalization process is presented, including settings of both environmental parameters and mapping function, which are important for the efficient description of style parameters.

3.2.1 Environmental Parameters

To represent the environment as a numerical variable, the environmental parameters are defined here. Environmental parameters serve as independent variables, and their distribution properties have great effects on the regression performance. Therefore, the distinctive features in environments are selected as environmental parameters. Let the environmental parameters as $\mathbf{r} = [\mathrm{r}_1, \mathrm{r}_2, ..., \mathrm{r}_k]^\top \in \mathbb{R}^{k\times 1}$. Each element, r_i for $i = 1, 2, ..., k$ is a distinctive feature of corresponding environments. For an example of the optimal avoidance scenario, environmental parameters consist of the obstacle information such as dimension or location of obstacle. Thus, in each demonstration, the specific set of style parameters and the weights for basis functions are obtained. Since basis functions map a subspace of all the training trajectories, the required styles of motion can be generalized by choosing the appropriate style parameters based on the environmental information.

3.2.2 Mapping Function

In order to present the explicit relationship between environmental and style parameters, regression problem is formulated here. When the compact sets of demonstrations are se-

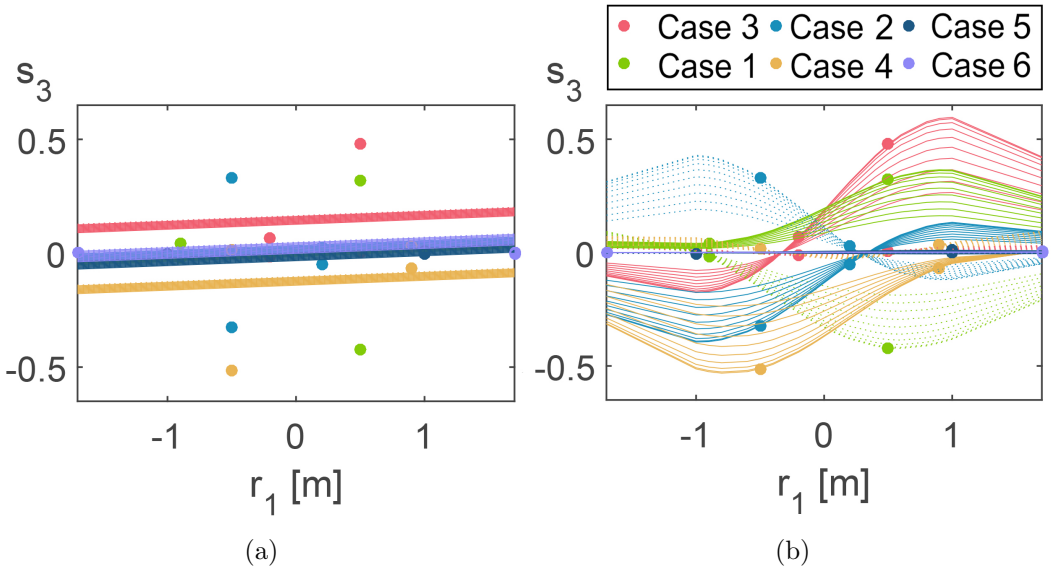


Figure 3.2: The results of regression through (a) LWR and (b) GPR for the example scenario of cooperative aerial transportation. In general, nonlinear relationship is observed between environmental and style parameters. Each point indicates the third style parameter with respect to the first environmental parameter, in demonstration set (see Fig. 3.5).

lected, the essential property for regression model is considered as a zero empirical error. The zero empirical error indicates that the regression function computes exact output style parameters from each set of environmental parameters in trained demonstrations.

Among the regression model, linear regression model gives direct and simple representations as $\mathbf{S}_{1:N} = \beta\mathbf{R}$. Each $\mathbf{S}_{1:N} \triangleq [\mathbf{S}_1^\top, \mathbf{S}_2^\top, ..., \mathbf{S}_N^\top]^\top \in \mathbb{R}^{L\times M}$ and $\mathbf{R} \triangleq [\mathbf{r}^1, \mathbf{r}^2, ..., \mathbf{r}^M] \in \mathbb{R}^{k\times M}$ indicates the matrix of stacked sets of style parameters and environmental parameters, respectively. The least-square solution for the constant matrix would be $\beta = \mathbf{S}_{1:N}\mathbf{R}^+$, where $\mathbf{R}^+$ denotes the pseudo-inverse of $\mathbf{R}$. Practically, the number of demonstrations M is bigger than k, making the $\mathbf{R}$ matrix fat. Therefore, the system is usually over-determined, which prevents from obtaining the exact solution for training data in general cases.

Therefore, for the zero empirical error, a nonlinear model is required. Obviously, this non-linear property is observed from visualized relationships between some elements of environmental and style parameter. For example of optimal aerial transportation scenario, these nonlinear relationship is also observed in Fig. 3.2. Thus, a nonlinear model is recommended for zero empirical error. Among the nonlinear modeling approaches, Gaussian process regression (GPR) is a highly recommended as its accurate and flexible regression performance for approximating unknown nonlinearities. Here, the complexity of GPR scores $\mathcal{O}(M^3)$ for M number of demonstrations. GPR is formulated using the matrix of stacked sets of style parameters, which is defined as $\mathbf{S}_{1:N} \triangleq [\mathbf{S}_1^\top, \mathbf{S}_2^\top, \cdots, \mathbf{S}_N^\top]^\top \in \mathbb{R}^{L\times M}$ and environmental parameters $\mathbf{R} \triangleq [\mathbf{r}^1, \mathbf{r}^2, \cdots, \mathbf{r}^M] \in \mathbb{R}^{k\times M}$. Here L is defined as $\sum_{n=1}^{n=N} L_n$. The below is the regression function in this formulation:

$$\begin{bmatrix} \mathbf{S}_{1:N}^\top \\ \mathbf{s}_{1:N}^{new\top} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \mathbf{K}(\mathbf{R},\mathbf{R}) & \mathbf{K}(\mathbf{R},\mathbf{r}^{new}) \\ \mathbf{K}(\mathbf{r}^{new},\mathbf{R}) & \mathbf{K}(\mathbf{r}^{new},\mathbf{r}^{new}) \end{bmatrix}\right), \tag{3.5}$$

where $\mathbf{K}$ indicates the matrix of Gaussian kernel [59]. $\mathbf{s}_{1:N}^{new} \triangleq [\mathbf{s}_1^{new\top}, \mathbf{s}_2^{new\top}, \cdots, \mathbf{s}_N^{new\top}]^\top \in \mathbb{R}^{L\times 1}$ is the column vector consisting of the new style parameters. $\mathbf{r}^{new}$ is the newly updated environmental parameters calculated from the actual environment. The vector of the style parameters $\mathbf{s}_{1:N}^{new}$ is obtained as

$$\mathbf{s}_{1:N}^{new} = (\mathbf{K}(\mathbf{r}^{new},\mathbf{R})\mathbf{K}(\mathbf{R},\mathbf{R})^{-1}\mathbf{S}_{1:N}^\top)^\top. \tag{3.6}$$

The offline work includes the calculation of the multiple trajectories for demonstrations and $\mathbf{K}(\mathbf{R},\mathbf{R}) \in \mathbb{R}^{M\times M}$ for the mapping function, which demands a relatively high computational load. During the online phase, the new style parameters $\mathbf{s}_{1:N}^{new}$ is obtained immediately from (3.6) with the calculation of kernel matrix $\mathbf{K}(\mathbf{r}^{new},\mathbf{R}) \in \mathbb{R}^{1\times M}$ and the current environmental parameters. Since a set of weights is already learned for basis function from (3.4), the final attractor function is obtained by simply multiplying new style

Offline demos	Offline learning	Online reproduction
$\mathcal{O}(M)$	$\mathcal{O}(M^2L)$ or $\mathcal{O}(M^3)$	$\mathcal{O}(ML)$

Table 3.1: Complexity in the proposed PDMPs.

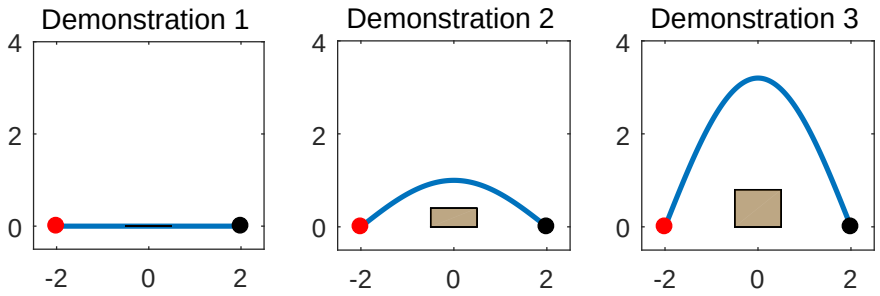


Figure 3.3: Three demonstrations are provided for two-dimensional hurdling motion.

parameters and the represented basis function. Hence, once the combined offline framework is built, the motion can be generated or modified quickly in the online phase. For both offline and online process, the complexity of GPR is analyzed in Table. 3.1. Be noticed that the complexity order is reduced from $\mathcal{O}(M^3)$ to $\mathcal{O}(ML)$ for online computation. Moreover, M scores relatively small number compared to other GPR problems, so our formulation is not dominantly affected by the complexity issue of the GPR. The problem of increasing M for complex missions will be addressed in Chapter 4 of the paper.

3.3 Simulation Results

3.3.1 Two-dimensional Hurdling Motion

Consider a simple generalization problem of two-dimensional hurdling motion. In this scenario, for a simple description, only the height of the hurdle is supposed to vary from zero to any value. The environmental parameter is the height of the hurdle. In order to

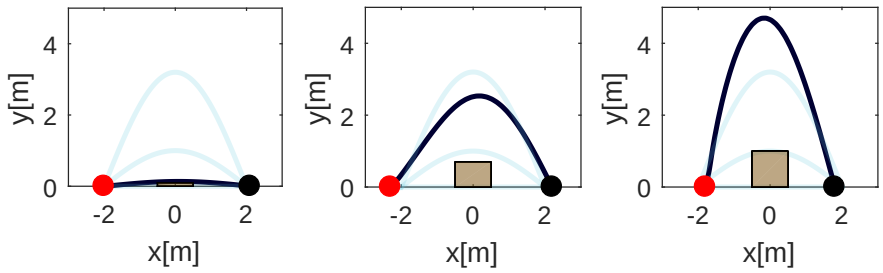


Figure 3.4: The simulation results for two-dimensional hurdling motion. Each red and black point indicates the start and goal states, respectively. The dark blue line shows the computed trajectory of the robot. Light blue lines show the demonstrations in corresponding PDMPs.

generalize the movement of the robot avoiding the hurdle, three different demonstrations are provided in Fig. 3.3. The generalization results are listed in Fig. 3.4. With only three demonstrations, the avoidance of any height of hurdle is possible.

3.3.2 Cooperative Aerial Transportation

In order to show that this proposed framework can serve as a real-time optimal motion planner, this section provides simulation results for cooperative aerial transportation scenario in obstacle environments. From that this framework generalizes the required behavioral style, the computed movements will sufficiently reflects the desired property of demonstrations. Thus, both properties of quickness and optimality are satisfied by providing optimal demonstrations to PDMPs using an optimal motion planner. Here, RRT* is used in the way that it achieves asymptotic optimality without the worry of converging local minima, eternally. Here, RRT* runs in off-line so the issue of slow convergence diminishes.

Ideally, it is desirable to configure the demonstrations by randomly changing the environmental parameters and extracting the characteristic data values observed among them. In the example scenarios in this dissertation, the demonstrations using trigonometric func-

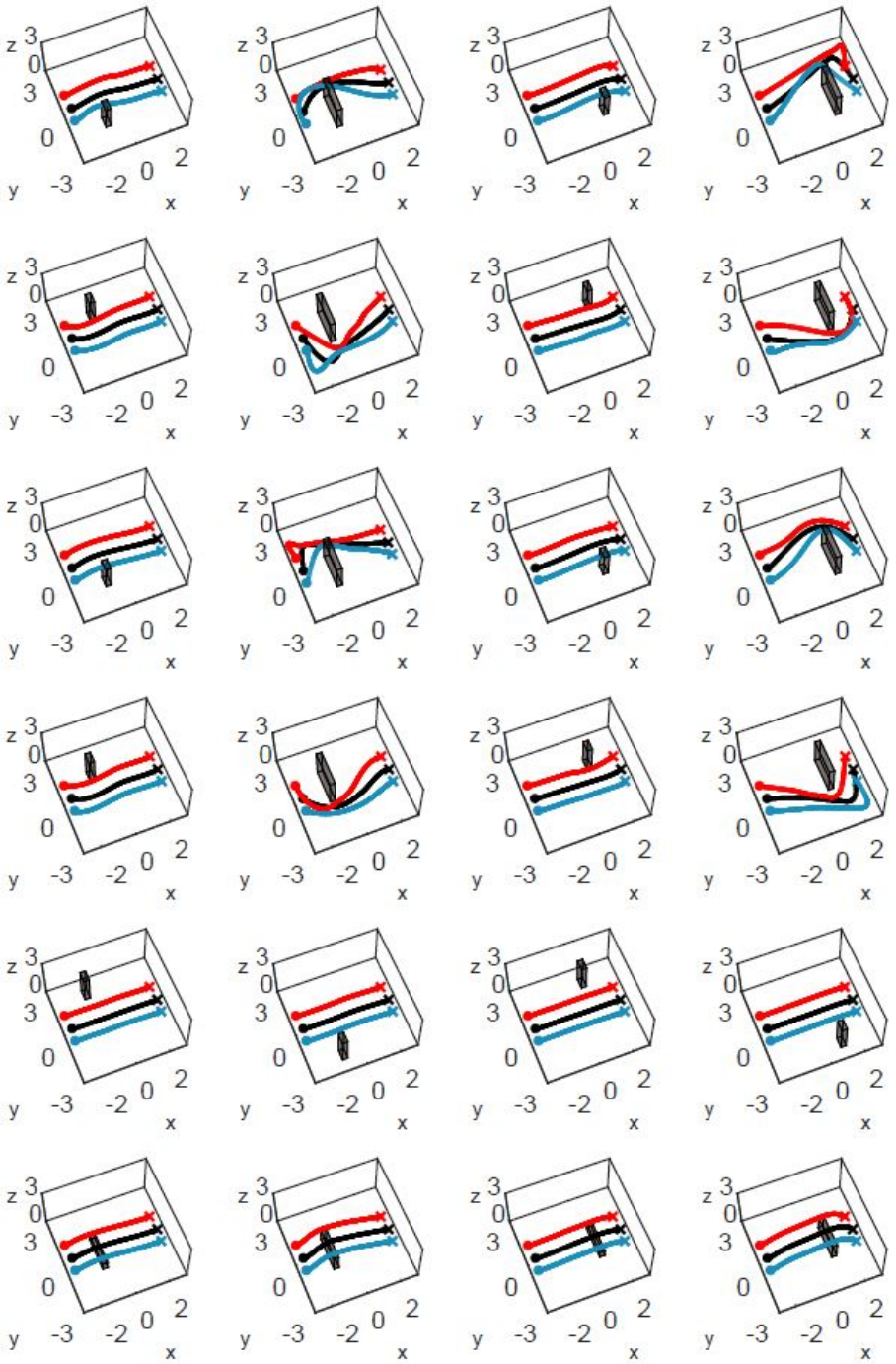


Figure 3.5: 24 Demonstrations using RRT*. In each environment, an obstacle having different dimension and location is located. Each black, red and blue line shows the trajectory of body positions of object, left and right aerial manipulator, respectively. The first two rows and the next two rows demonstrate negative and positive yawing motion for avoidance in y direction.

		Avoidance direction			
		Positive y	Negative y	Positive z	None
Yawing during the avoidance	Positive	Case 1	Case 2		
	Negative	Case 3	Case 4		
	zero			Case 5	Case 6

Table 3.2: The distinctive avoidance skills against ground obstacles in cooperative aerial manipulation.

tions, RRT* and iLQR methods were used as data for PDMPs. Each RRT* and iLQR is used to generate the trajectory for cooperative aerial transportation in sections 2.2.2 and 3.3.2 and hang-dry mission in section 6.2, respectively. However, getting demonstrations in mobile manipulators is not practical for random trial. (Be noticed that the method of obtaining the demonstrations is not strictly part of the PDMPs algorithm. PDMPs only generalize the movements from demonstrations set that previously obtained) When the generalization goal is specified, it is possible to specify distinctive demonstration settings and save the effort of obtaining all the demonstrations. From the RRT* results in Fig. 3.5, six distinct cases are observed for essential avoiding skills as listed in Table. 3.2. In order to represent the magnitude of specific motion, at least two demonstrations are required per each case. The distance from the obstacle is also considered by setting it at two different values of x coordinate, which leads to the total number of demonstration set for mimicking general environments as 24 ($= 6 \times 2 \times 2$). In Fig. 3.5, 24 distinctive results from RRT* are listed. 24 RRT* demonstrations (Fig. 3.5) and basis functions $\mathbf{b}_n$ for $n = 1, 2, ..., N$ are computed. Here, the demonstrations in the second row from bottom show similar movements but are included to training demonstrations. Since PDMPs learn not only individual movements in demonstrations but also corresponding environmental parameters, those demonstrations are treated as individual data. The explicit mapping function was obtained as described in section 3.2. Based on the above works, the motion trajectory is computed during the on-line phase. In the particular setting reported here, $k = 8$ for the environmental parameters including the closest x, y and z positions of obstacle edges. In a new environment, the information of obstacles is sensed in real-time and used to form a set of environmental

parameters. From the environmental parameters, the style parameters are obtained and used to generate motion skills based on attractor dynamics with $f_n = \mathbf{s}_n^T \mathbf{b}_n$.

In order to guarantee that the produced motion is collision-free to newly placed obstacle, algorithm runs in 2000 sets of environments containing obstacles with new dimension and location and all of these results are checked as collision-free. From the results, it is confirmed that avoidance is successfully performed from the obstacles located in space $\mathcal{C} = \{[x, y, z]^\top \in \mathcal{C} | -1.7 \leq x \leq 1.7, -2 \leq y \leq 2, 0 \leq z \leq 2\}$, quantitatively. In Figs. 3.7a – 3.7d, the examples are presented for a single and twin obstacle(s) environments. The black and grey lines indicate the produced trajectory and 24 demonstrations of the center of object. Each red and blue line is the body trajectory of left and right aerial manipulator, respectively. For the cases of twin obstacles in Figs. 3.7c and 3.7d, style parameter is updated right after the first obstacle is avoided and second obstacle is observed. If there are enough space to converge on a modified path to avoid second obstacle (with updated style parameter), the path is collision-free. Once the combined offline framework of RRT*-PDMPs is built, the motion can be generated or modified quickly in the online phase. The effectiveness of this learning-based motion planning, RRT*-PDMPs is compared with RRT* algorithm in Fig. 3.6 and Table. 3.3.

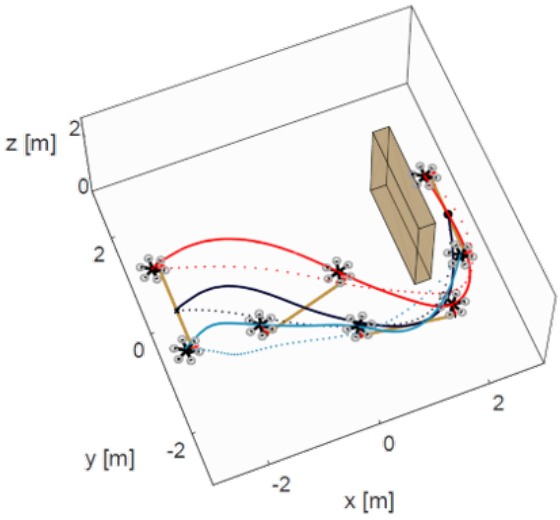
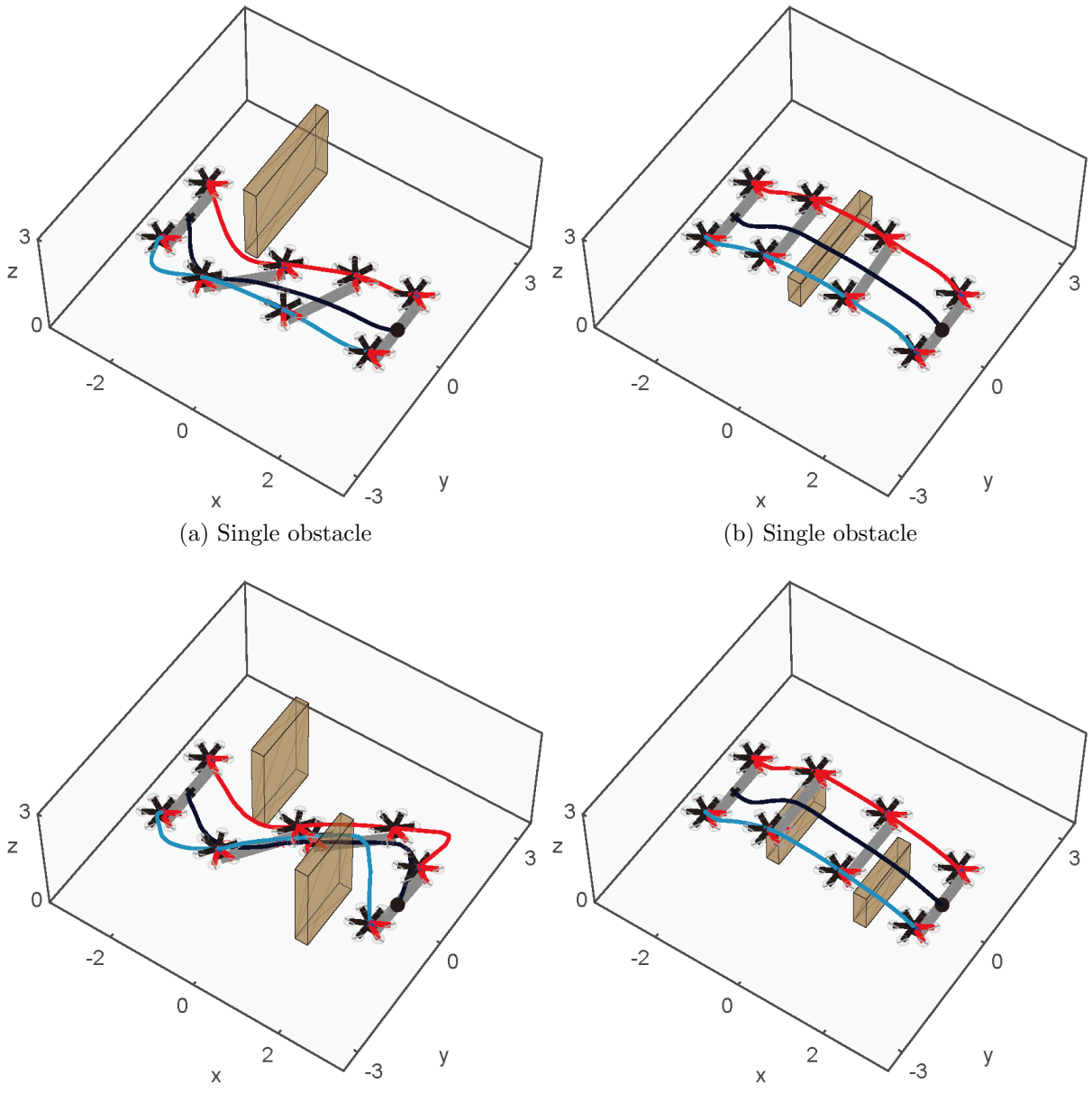


Figure 3.6: Comparison of the efficiency and computational time with sampling-based planning (RRT*) and learning-based planning (RRT*-PDMPs). The red and blue lines are the body trajectories of aerial manipulators, respectively. The black line indicates the produced trajectory of the center of object. Each dashed and solid line indicates the trajectory of aerial manipulator from RRT* and RRT*-PDMPs, respectively.

	RRT* only (dashed)		RRT*-PDMPs (solid)
	First trajectory	Optimized trajectory (30,000 nodes)	Online phase only
Computational time	121.23 s	1,830.53 s	**4.21 ms**
Travel distance	21.16	**13.16**	**13.94**

Table 3.3: Performance comparison between RRT* and RRT*-PDMPs (See Fig.3.6).



(a) Single obstacle

(b) Single obstacle

(c) Twin obstacles

(d) Twin obstacles

Figure 3.7: Simulation results of RRT*-PDMPs.

4

Motion Generalization with Safety Guarantee

Suppose that there are unsafe values of state variables. For example, some robot states result in exceeding the actuation limits and break the system stability. In robot manipulation, internal interference occurs between the robot arm links in the particular state variables. In order to address the safety of motion avoiding these types of unsafe states, this section addresses the additional process to be applied to the proposed framework in section 3. This process can be viewed as a way of managing the demonstrations in advance.

4.1 Safety Criterion in Style Parameter

Once the PDMPs framework is built on multiple demonstrations, the online behavior of the PDMPs is determined by calculating the style parameters for the current environment. Therefore, the safety of online motion, i.e., the motion computed during the online process, is verified by checking in advance whether the style parameter produces an unsafe behavior of the robot. Here, the safety criterion in the style parameter domain reflects the new behavior computed through the generalization process, as well as the individual motion safety of the demonstration set.

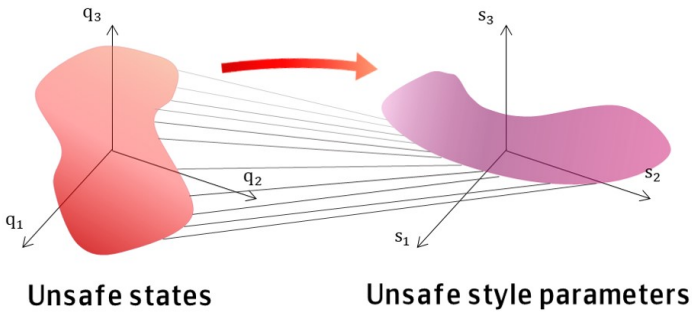


Figure 4.1: The set of unsafe states is mapped into the region of unsafe style parameters.

In the following, the optimization problem is described to obtain the boundary of the safe value of style parameters. Let $\mathbf{q}_{us}$ denote an unsafe state variable. The objective in this section is to obtain the style parameter which calculates the closest movement to the given $\mathbf{q}_{us}$. Here, the closest movement indicates the movement corresponding to the value of state variables very close to the unsafe range such as dangerous position, maximum velocity, and so on. Via optimization, the set of unsafe states is mapped to the region of unsafe style parameters (See Fig.4.1).

Consider the vector $\bar{\mathbf{q}}^t = [\mathbf{q}^{t\top}\ \dot{\mathbf{q}}^{t\top}]^\top$ that consists of the state vector at the t-the time step, $\mathbf{q}^t$, and the time derivatives of $\mathbf{q}^t$. From 4.1, attractor function for all state variables is represented as

$$\mathbf{f}(\chi, \mathbf{s}; \mathbf{w}) = \mathbf{B}(\chi; \mathbf{w})^\top \mathbf{s}, \tag{4.1}$$

where $\mathbf{B}$ is the basis matrix defined as,

$$\mathbf{B}_{target} = \begin{bmatrix} \mathbf{B}_{1,target} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_{2,target} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{B}_{N,target} \end{bmatrix}. \tag{4.2}$$

By applying the dynamics of PDMPs in 3.1, the dynamics of $\bar{\mathbf{q}}^t$ can be represented as

$$\dot{\bar{\mathbf{q}}}^t = \mathbf{A}\bar{\mathbf{q}}^t + \boldsymbol{\beta}^t\mathbf{s} + \mathbf{D}^t, \tag{4.3}$$

where,

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}_N & \mathbf{I}_N \\ -\alpha_1\mathbf{I}_N/\nu^2 & -\alpha_2\mathbf{I}_N/\nu \end{bmatrix},$$

$$\boldsymbol{\beta}^t = \begin{bmatrix} \mathbf{0}_N \\ \alpha_1\mathbf{B}^t(\chi^t;\mathbf{w}^t)/\nu^2 \end{bmatrix}, \tag{4.4}$$

$$\mathbf{D}^t = \begin{bmatrix} \mathbf{0}_N \\ \alpha_1(\mathbf{g} - \mathbf{g}\chi^t + \mathbf{q}_0)/\nu^2 \end{bmatrix}. \tag{4.5}$$

Here, each $\mathbf{0}_N$ and $\mathbf{I}_N$ represents $N \times N$ zero and identity matrix, respectively. The superscript $t = 1, \cdots, T$ indicates that the variables are concerned with the variables at the t-th time step.

Now, define the vector $\tilde{\mathbf{q}} = [\bar{\mathbf{q}}^{0\top} \cdots \bar{\mathbf{q}}^{T\top}]^\top$. Then, $\tilde{\mathbf{q}}$ can be simplified to

$$\tilde{\mathbf{q}} = \boldsymbol{\Phi}\mathbf{s} + \boldsymbol{\Psi} \tag{4.6}$$

where the t-th elements of each $\mathbf{\Phi}$ and $\mathbf{\Psi}$ matrix for $t = 1, \cdots, T$ are represented as

$$\Phi^t = \sum_{i=1}^{t} (\mathbf{A}\mathrm{dt} + \mathbf{I}_N)^{i-1} \boldsymbol{\beta}^{t-i} \mathrm{dt}, \tag{4.7}$$

$$\Psi^t = (\mathbf{A}\mathrm{dt} + \mathbf{I}_N)^t \bar{\mathbf{q}}_0 + \sum_{i=1}^{t} (\mathbf{A}\mathrm{dt} + \mathbf{I}_N)^{i-1} \mathbf{D}^{t-i} \mathrm{dt}. \tag{4.8}$$

In order to compute only the position difference to unsafe states rather than time derivatives, set the matrix $\mathbf{\Theta}^t = [\mathbf{I}_N \ \mathbf{0}_N; \mathbf{0}_N \ \mathbf{0}_N]$ and $\mathbf{\Theta} = diag([\mathbf{\Theta}^1 \cdots \mathbf{\Theta}^T])$. Then, the objective function can be formulated as

$$J = \frac{1}{2} \mathbf{s}^\top \mathbf{H} \mathbf{s} + \mathbf{h}^\top \mathbf{s} \tag{4.9}$$

where, $\mathbf{H} = 2\mathbf{\Phi}^\top \mathbf{\Theta}^\top \mathbf{\Theta} \mathbf{\Phi}$ and $\mathbf{h} = 2\mathbf{\Phi}^\top \mathbf{\Theta}^\top (\mathbf{\Psi} - \tilde{\mathbf{q}}_{us})$.

For all the values of the unsafe states, the corresponding optimal movements and their style parameters are computed. Here, optimal movements indicate the nearest motion. Those style parameters consist of the safety criterion for current PDMPs. Here, any optimization methods, including quadratic programming (QP), can be employed.

4.2 Demonstration Management

After the safety criterion of the style parameters is computed via optimization formulation, the demonstrations causing unsafe motion are eliminated based on this criterion. The overview of the proposed process is shown with the red arrows in Fig. 4.2.

The detail process is described in the following. First, there can be specific unsafe conditions that may have been unconsidered when obtaining the previous demonstrations in PDMPs (e.g. new environmental settings). Then, the safety criterion for style parameters is extracted as described in section 4.1. After the criterion is obtained, it is identified whether each given demonstration is safe or not. When the value of style parameter of a demonstration is outside the safety region, the corresponding demonstration is removed from the

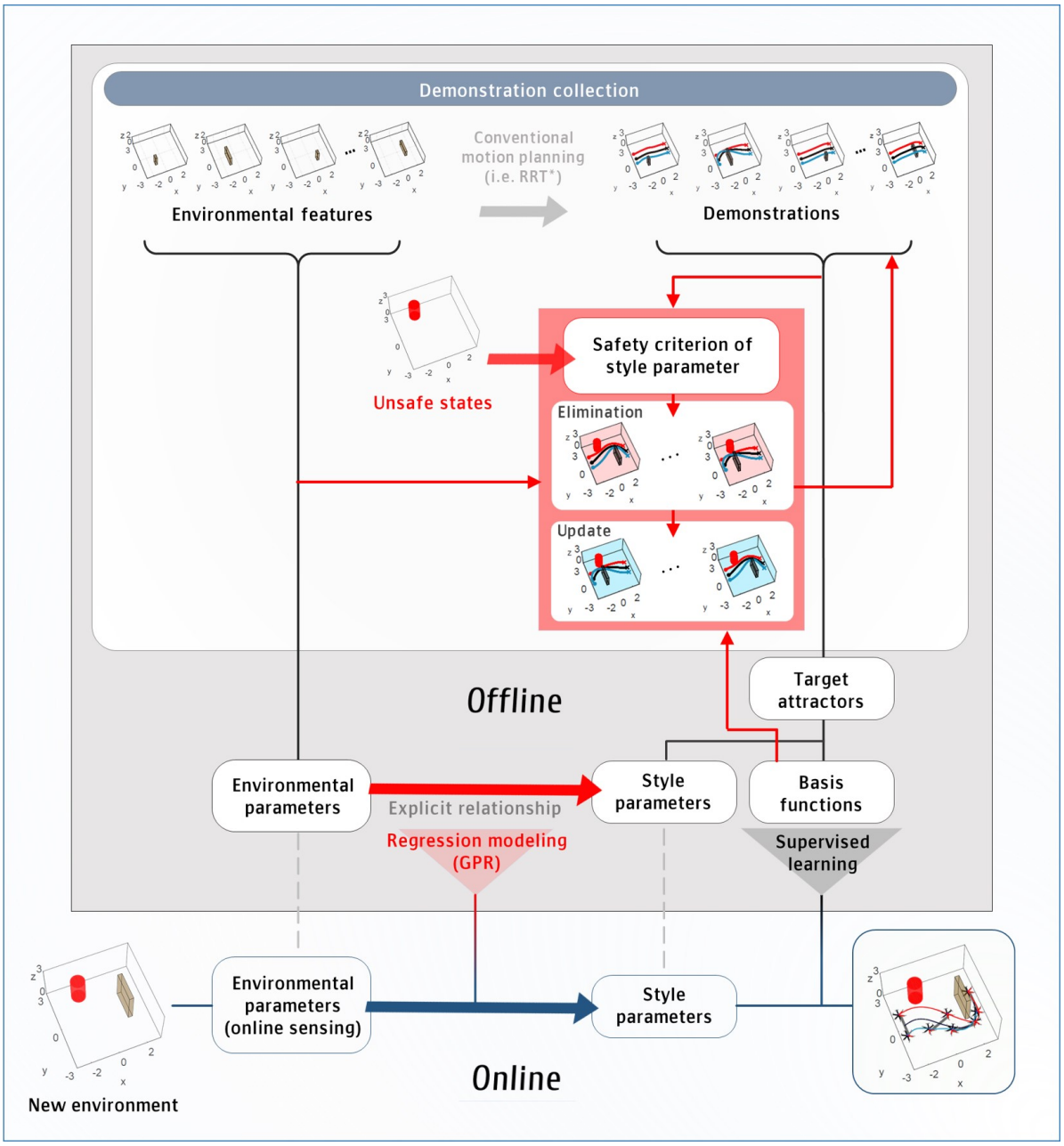
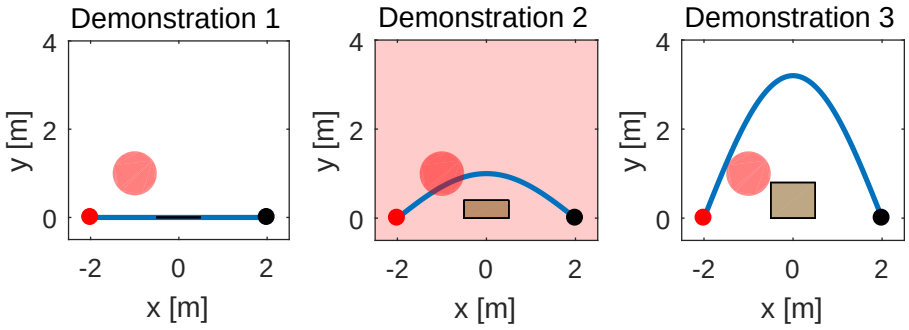


Figure 4.2: An overview of the proposed PDMPs framework with the safety guaranteeing process.

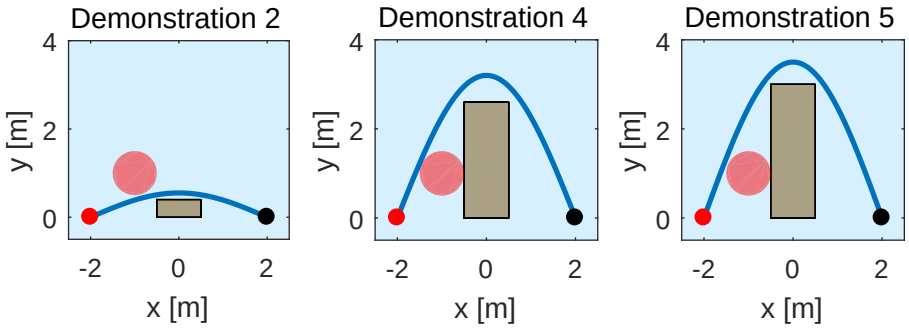
demonstration set. Instead, a new safe demonstration is provided for the same environmental setting. Moreover, to enhance the performance of representation near the unsafe set, additional demonstrations are computed from the style parameters at the boundary of safe region. It is possible that, although there is no unsafe demonstration in the dataset an unsafe motion is computed online for some untested environmental settings, because the motion computed via the generalization process in PDMPs. In such cases, trace back the style parameters that are at the boundary of the safe region and provide additional demonstrations. Here, new demonstrations are provided as follows:

1. Calculate the values of style parameters for all the possible range of environmental parameters.
2. Trace back the environments where some demonstrations caused unsafe motion online for the boundary of the unsafe style parameter values.
3. Provide new demonstrations for the particular environments in Step 2.
4. Check the safety of online motion and repeat Steps 1–3 until there remains no unsafe value of style parameters. (This step determines how many demonstrations are needed.)

After removing the unsafe demonstrations, in step 3, the demonstration set is complemented with new demonstrations. This is intended to maintain generalization performance, especially if the demonstrations removed are dominant data. Although the decrease in the number of demonstrations does not mean that PDMPs are not working, generalization performance is usually degraded. If the removed demonstrations include the data that played a dominant role in the generalization process, performance would degrade further. (See the generalization performance with insufficient data in the examples of driving multiple tracks in Section 5.5.) Thus, providing new demonstrations is important as well as elimination of unsafe demonstrations.



(a)

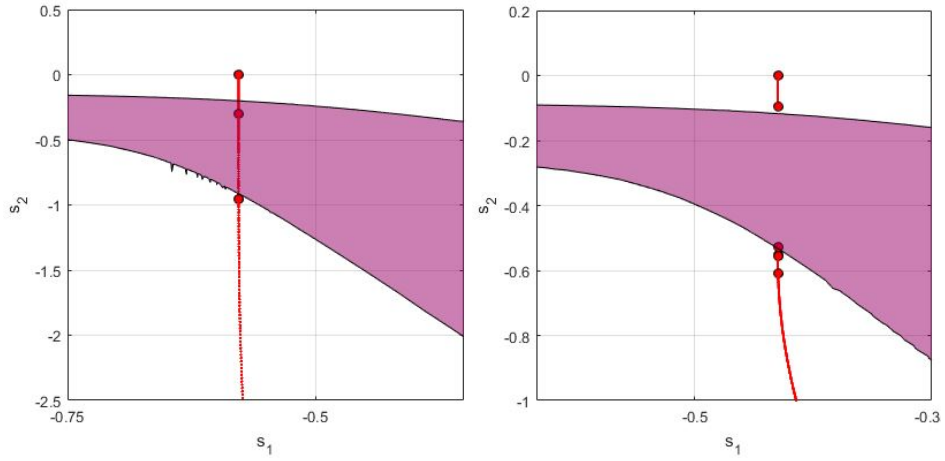


(b)

Figure 4.3: The demonstrations are given for the hurdling motion scenario where each x and y denotes forward and height, respectively. Each red and black point indicates the start and goal states, respectively. The red colored circle shows the unsafe region in state space. The blue line shows the trajectory of the robot. (a) Three demonstrations which are previously given. Based on the safety criterion, the second demonstration is concluded to (marked with red background) invade the given unsafe region. (b) Three newly provided demonstrations.

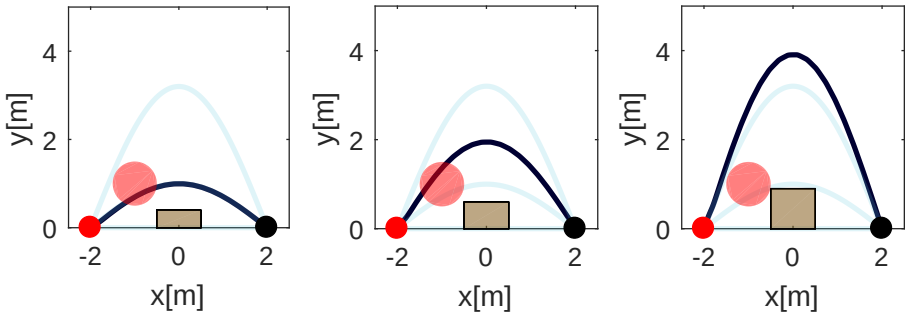
4.3 Simulation Validation

This section includes the simulation results for two different scenarios, similar to section 3.3. The additional setting here is a static obstacle intuitively representing unsafe states. The objectives in both scenarios are to avoid unsafe states occupying this new obstacle while moving to the goal state.

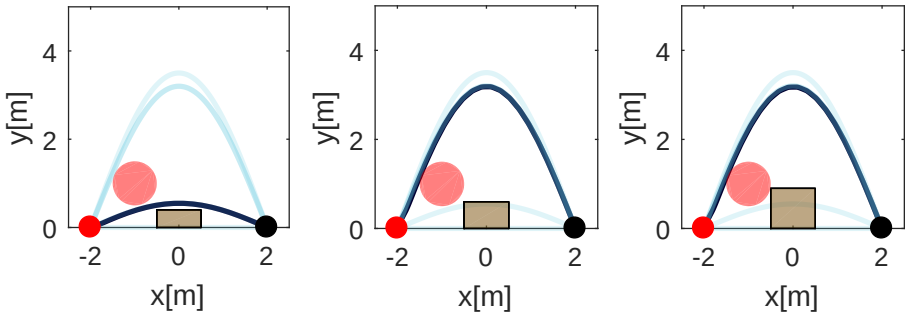


(a) (b)

Figure 4.4: The safety criteria for style parameters in the hurdling motion scenario (a) with the previous demonstrations and (b) with the updated demonstrations. The unsafe regions are colored in purple. Each red point indicates the style parameter obtained from the demonstration data. The red line shows the set of style parameters computed from the GPR function with all the possible obstacle configurations. In (a), there exists an unsafe demonstration, whereas it has been properly removed in (b).

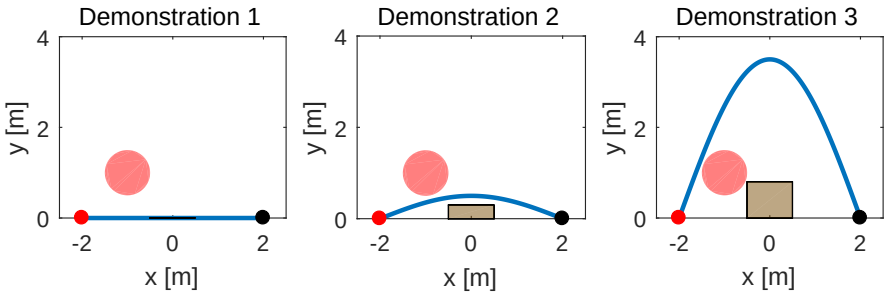


(a)

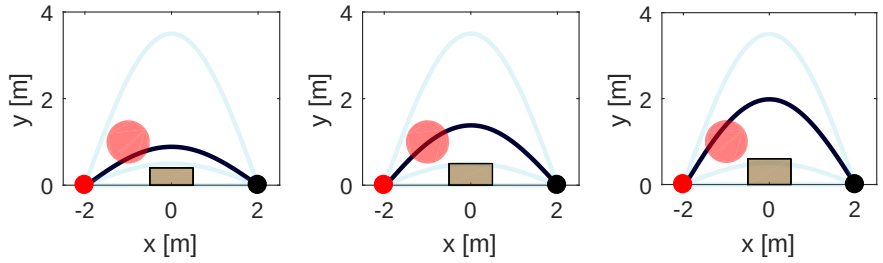


(b)

Figure 4.5: The simulation results for the hurdling motion scenario in the framework of PDMPs (a) without and (b) with the proposed process. Each red and black point indicates the start and goal states. The dark blue line shows the trajectory of the robot. Light blue lines show the demonstrations in corresponding PDMPs.



(a)



(b)

Figure 4.6: The simulation results to elucidate the issue that the resulting motion can be unsafe even though only safe demonstrations are included. (a) Provided demonstrations. (b) Resulting online motion.

4.3.1 Two-dimensional Hurdling Motion

Consider a robot avoiding a stationary ground obstacle. In this scenario, for simple description, only the height of the obstacle is supposed to vary from zero to any value.

First, in order to generalize the movement of the robot avoiding the obstacle, three different demonstrations are provided. If a new unsafe state is specified, the algorithm extracts the unsafe region of style parameters and removes the unsafe demonstration(s). Then, new demonstrations are provided to replace the unsafe motion and enhance the description of the motion near the unsafe region. In Figs. 4.3a and 4.3b, demonstrations are listed, which are previously provided and newly given according to the proposed process, respectively. The background of an unsafe demonstration is colored in red while the backgrounds of newly given demonstrations are marked with blue. The unsafe regions of style parameters are computed as the purple area in Fig. 4.4. Fig. 4.4a is computed from the PDMPs with the previous demonstrations while Fig. 4.4b shows the unsafe region of style parameter with updated demonstrations. As shown in Fig. 4.4a, it is obvious that the unsafe states are included in the second demonstration which is eliminated from the demonstration set. After providing new demonstrations, the values of style parameters are in the safe region for all the possible environmental settings (i.e. obstacle configuration) as shown in Fig. 4.4b. In Fig. 4.5, the online paths with various environmental settings in the PDMPs without and with the proposed process. The trajectories in Fig. 4.5a have been obtained from the original demonstrations only (i.e. Fig. 4.3a), whereas the trajectories in Fig. 4.5b have been computed from the demonstration set excluding unsafe one for the new environment and including additional ones (i.e. five demonstrations as in Figs. 4.3a and 4.3b) From the results, it is shown that the PDMPs successfully consider the unsafe states with the proposed process.

Even when PDMPs are constructed with safe demonstrations only, unsafe motion can still be produced. For the hurdling motion scenario, the demonstrations in Fig. 4.6a can be provided instead of the existing one. Resulting online motions in Fig. 4.6b are not safe

	Demonstration construction		
	w/ the proposed process		w/o the proposed process
	Safety criterion	RRT* demos (required #)	RRT* demos(required #)
Computational time	23.16 s	9,602,83 s (8)	736,533.12 s (28)
Total	**9,626.00 s**		736,533.12 s

Table 4.1: The computational times spent in obtaining demonstrations. Here, the demonstrations can be provided by techniques other than RRT* as before [1, 2]. Be noticed that demonstrations is obtained in the offline phase of PDMPs and the online motion generation is completed in a short time to be used in real-time.

even they are based on the PDMPs with safe demonstrations. By providing the sufficient demonstrations with the proposed process, the safety of motion is guaranteed as in Fig. 4.3b of the manuscript.

4.3.2 Cooperative Aerial Transportation

As described before, the online path is computed by using PDMPs even for the high-dimensional and complex dynamic system such as cooperative aerial manipulators. In section 3.3, PDMPs with the proposed generalization process is applied for cooperative aerial transportation. By providing optimal movements for demonstrations using RRT*, aerial manipulators adapts to the environments with the various size and location of obstacles. Now, a new cylinder-shaped obstacle is added as shown in Fig. 4.7 (compare it to Fig. 3.5). The black line represents x_o, y_o, z_o which are the position of the center of the common object. The unsafe set of states is specified with respect to x_o, y_o, z_o and yaw angle considering that not only the center position of the common object but the yaw angle in the cooperative pose should be considered to check the collision with the obstacle. The safety criterion of style parameters is computed as shown in Fig. 4.8a. Then, four demonstrations intrude the unsafe state. By replacing them with new demonstrations as listed in Fig. 4.7b, the proposed process set the demonstrations which are safe as shown in Fig. 4.8b. In Fig. 4.9, PDMPs successfully avoid the unsafe states with the proposed process. The efficiency of the proposed process is also confirmed as shown in Fig. 4.1.

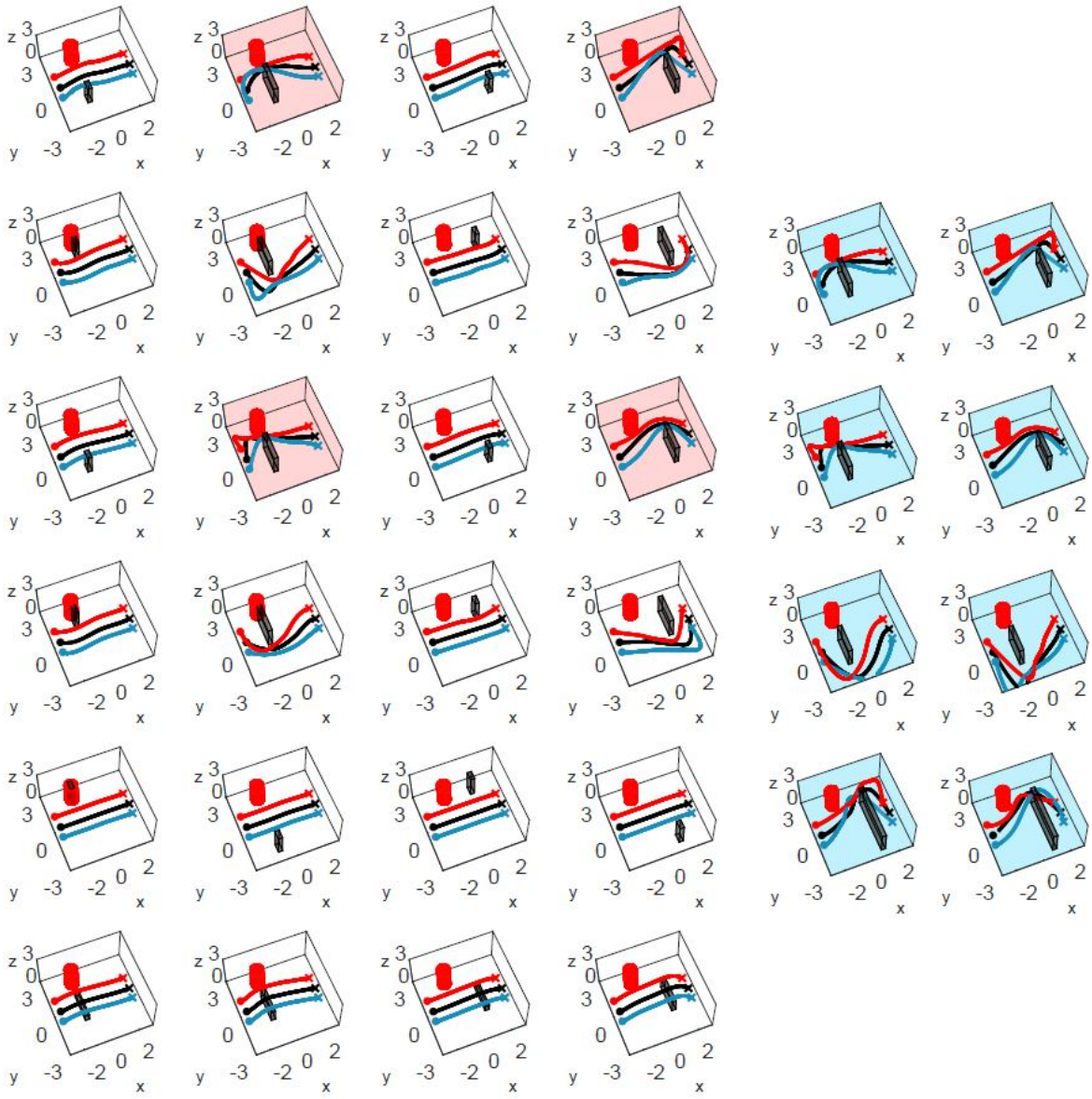
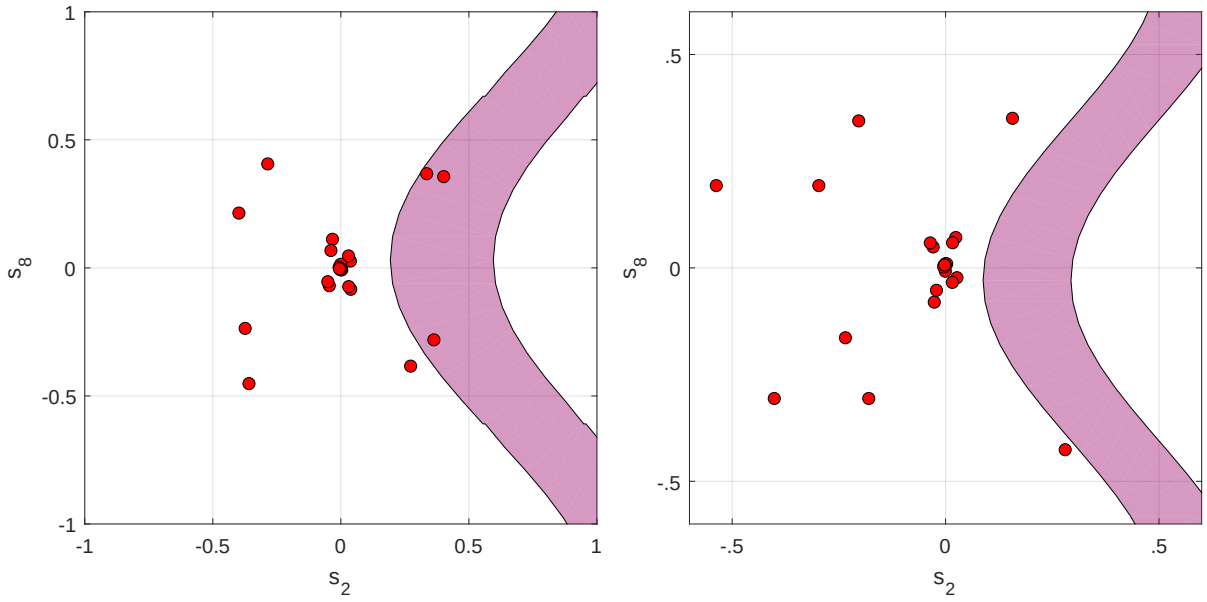


Figure 4.7: The demonstrations provided for the cooperative aerial transportation. Each red and blue line indicates the body trajectories of left and right aerial manipulator, respectively. The black line shows the trajectory of the center of the common object. (a) The firstly given demonstrations did not consider a cylinder-shaped obstacle marked with a red cylinder. With the safety criterion of the style parameter, unsafe demonstrations are marked with red background. (b) The additional demonstrations which are newly given. The first four demonstrations are computed from the same environments as the unsafe demonstrations in (a). Others are the safe demonstrations near the boundary of safety criterion.



(a) (b)

Figure 4.8: The safety criterion for style parameters in the cooperative aerial transportation scenario (a) with the previous demonstrations and (b) with the updated demonstrations. Each red point indicates the style parameter obtained from the demonstration data. The unsafe regions are colored in purple. In (a), there exist unsafe demonstrations in the demonstration, whereas they have been properly removed in (b).

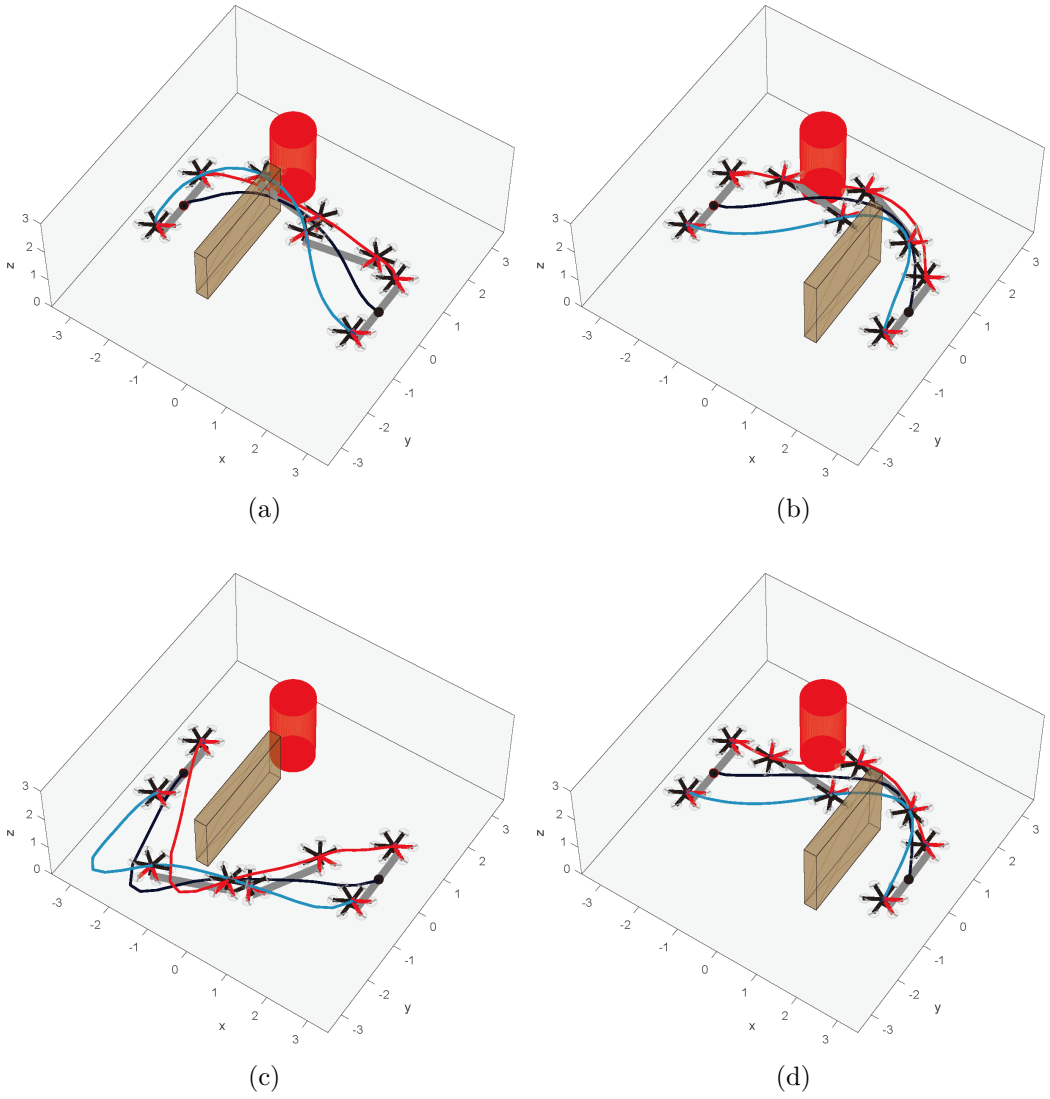


Figure 4.9: The simulation results for the cooperative aerial transportation scenario in the framework of PDMPs (a), (b) without and (c), (d) with the proposed process. Each red and blue line indicates the body trajectories of left and right aerial manipulator, respectively. The black line shows the trajectory of the center of the common object. In (a) and (b), aerial manipulators collide with the cylinder-shaped obstacle following the trajectory learned from the original demonstration set. On the other hand, safe motions are computed in (c) and (d) in the same environments as (a) and (b).

5

Motion Generalization for Complex Missions

This section presents the extension approach, seg-PDMPs, using the proposed PDMPs frameworks. This extension segments the trajectory of entire mission into unit movements and set multiple PDMPs. This is useful to reduce the number of demonstrations where the mission consists of multiple unit sub-tasks that should be generalized independently. For example, hang-dry mission (Fig. 1.1) has multiple sub-tasks (e.g., transporting, stretching, grasping) which can take various styles depending on the situation.

5.1 Overall Structure of Seg-PDMPs

The overview of the framework is shown in Fig. 5.1. First, the demonstrations for a complex mission are segmented into unit movements to represent multiple sub-tasks. Then, multiple PDMPs are formed independently in correlated sub-tasks. A period of performing sub-task is called as a phase in seg-PDMPs. The phase-decision process determines which sub-task and the associated PDMPs should be executed online, allowing multiple PDMPs to be configured within an integrated framework. GPR is applied to obtain reasonable execution time and regional goal configuration in each phase from the actual environment.

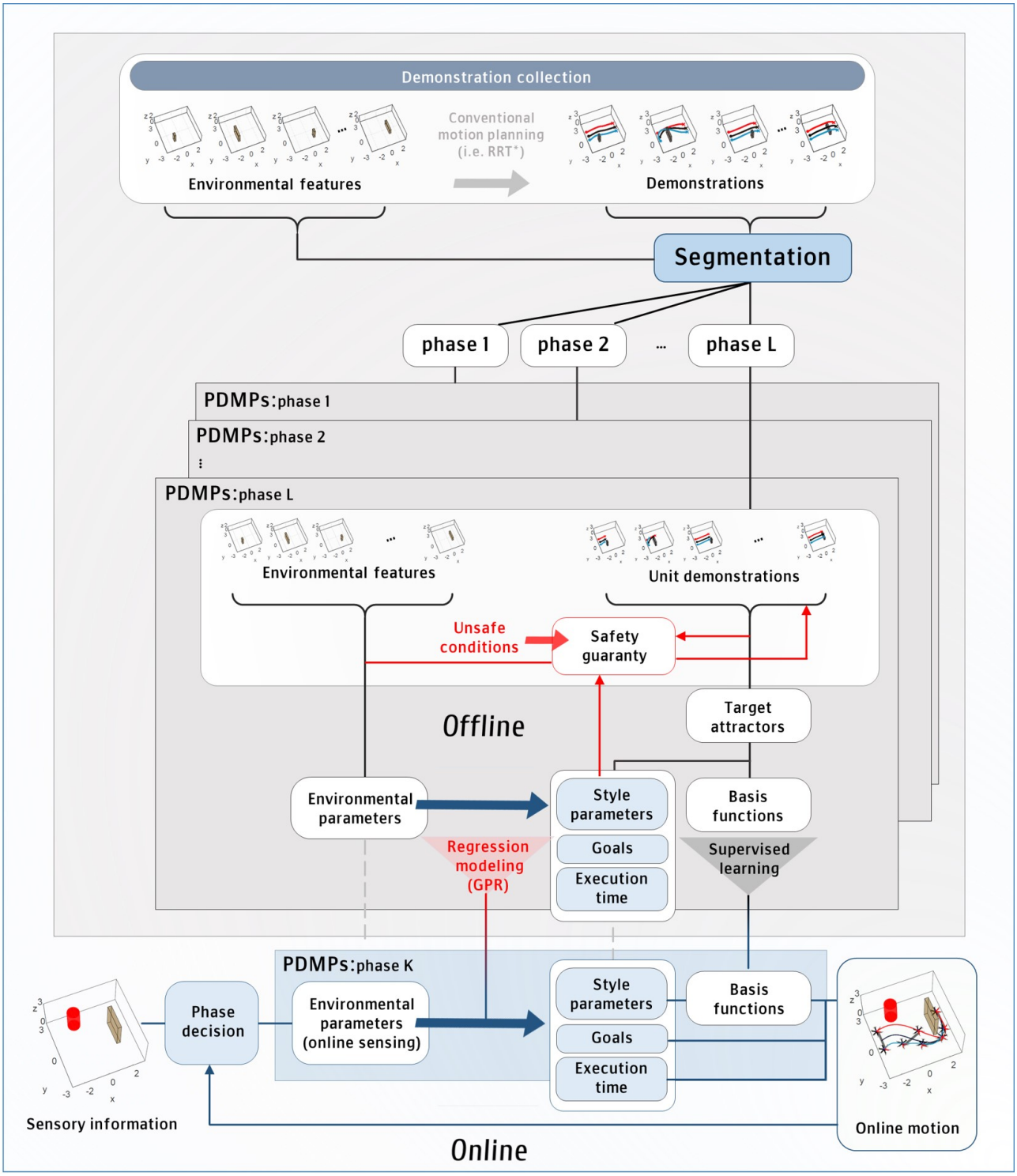


Figure 5.1: The overview of the proposed seg-PDMPs framework.

	PDMPs		Seg-PDMPs
	fixed sequencing	flexible sequencing	flexible sequencing
No. of demos	$M_1 \times M_2 \times \cdots M_{L^{ST}}$	$M_1 \times M_2 \times \cdots M_{L^{ST}} \times L^{ST}!$	$\text{Max}(M_1, M_2, \cdots, M_{L^{ST}})$

Table 5.1: Required number of demos for a complex mission.

5.2 Motion Segments

As mentioned before, M is the number of demonstrations and increasing M is not desirable from the perspective of the complexity of the algorithm (see TABLE. 3.1). If M gets significantly increased, the offline process requires a large amount of computational load in addition to much efforts for demonstration acquisition. Here, the number of required demonstrations is reduced by subdividing the recorded demonstrations rather than using them directly. Let L^{ST} be the number of sub-tasks and M_i for $i = 1, \cdots, L^{ST}$ denote the number of required demonstrations to generalize the styles of each i-th sub-task, respectively. In TABLE 5.1, the number of required demonstrations is listed with PDMPs and seg-PDMPs. Without segmentation, the sequential sub-tasks and their different styles are viewed in one PDMP framework. Thus, the number of required demonstrations for PDMPs will be $M_1 \times M_2 \times \cdots \times M_{L^{ST}}$ when their sequencing order is specified. If the sequencing order is not fixed, the number of possible sequences is multiplied to the number of required demonstrations. In contrast, seg-PDMPs only need fewer demonstrations where the styles of each sub-task are configured at least once. That is, the maximum value from M_1 to $M_{L^{ST}}$ is the number of required demonstrations in seg-PDMPs.

Instead of using segmented demonstrations, it is possible to provide demonstrations for each sub-task independently. However, using segmented demonstrations is preferred because it gives the values of execution time and regional goal state of sub-tasks when the robots perform sub-tasks sequentially. There are various segmenting algorithms such as Hidden Markov Models (HMMs) [60], transition state clustering [60], and zero crossing point(ZCP) or velocity(ZCV) [61]. This dissertation employ simple segmenting methods that focusing on motion segmentation like zero-crossing point or velocity.

5.3 Phase-decision Process

Each of multiple PDMPs generalizes the behavior of the corresponding sub-task in a phase. Seg-PDMPs autonomously determine which sub-task should be performed depending on the situation. This is called the phase-decision process and proceeds as follows.

Let $p = 1, 2, \cdots, L^{ST}$ denote a phase parameter. There are two different ways to specify a phase parameter according to the actual situation. One is using environmental information, and the other is using a segmentation criterion already used in the previous section. In the first approach, the clustering problem can be formulated to compute p from environmental information. The environmental parameters are used as samples and phase parameters are utilized as labels. Then, a clustering algorithm such as supervised k-means clustering is applied can be applied. In the second approach, the segmentation criterion can be used (e.g., ZCV or ZCP). Since the segmentation is already done with this criterion, the second approach gives more direct and accurate results for determining the phase.

Once the phase is determined, the corresponding PDMPs compute the generalized motion for the current sub-task. Whenever the robot completes the current sub-task, the phase-decision process deploys another PDMPs to be executed next. Until the robot reaches the goal or performs the final sub-task of the mission, multiple PDMPs will be configured in this way. The advantage of this architecture is that the robots can flexibly sequence sub-tasks depending on the situation. In order words, any mission with the structure of recombined sub-tasks is possible to be generalized.

5.4 Seg-PDMPs for Single Phase

The PDMPs framework in a single phase proceeds quite similarly to the general PDMPs except for two GPR processes. One process is to calculate the regional goal configuration, and the other is to allocate execution time in the current phase. The regional goals of sub-tasks are the terminal states of segmented demonstrations. The execution times are

the time lengths of segmented demonstrations. Naturally, the segmented demonstrations have different time lengths and goal states depending on the environmental settings and phase, and the GPR function is useful to express a nonlinear relationship between them as the style parameters are computed. Let $\mathrm{t}_{i,p}$ denote the time length for i-th demonstration in p-th phase, where $i = 1, \cdots, M_p$ and $p = 1, \cdots, L$. M_p is the number of demonstrations in p-th phase. By defining the time matrix as $\mathbf{T}_p^\top = [\mathrm{t}_{1,p}, \cdots, \mathrm{t}_{M_p,p}]^\top$, GPR function is formulated to

$$\begin{bmatrix} \mathbf{T}_p^\top \\ \mathrm{t}_p^{\mathrm{new}\top} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \mathbf{K}(\mathbf{R},\mathbf{R}) & \mathbf{K}(\mathbf{R},\mathbf{r}^{\mathrm{new}}) \\ \mathbf{K}(\mathbf{r}^{\mathrm{new}},\mathbf{R}) & \mathbf{K}(\mathbf{r}^{\mathrm{new}},\mathbf{r}^{\mathrm{new}}) \end{bmatrix}\right). \tag{5.1}$$

The Gaussian formulations for regional regional goals are similarly obtained by substituting $\mathrm{t}_{i,p}$, $\mathbf{T}_{i,p}$ for $\mathbf{g}$, $\mathbf{G}_{i,p}$, where $\mathbf{g}_{i,p}$, where $\mathbf{g}_{i,p}$ is the terminal state for $i = 1, \cdots, M_p$ and $\mathbf{G}_p^\top = [\mathbf{g}_{1,p}, \cdots, \mathbf{g}_{M_p,p}]^\top$. Finally, the execution time and regional goal are calculated as,

$$\mathrm{t}_p{}^{new} = (\mathbf{K}(\mathbf{r}^{\mathrm{new}},\mathbf{R})\mathbf{K}(\mathbf{R},\mathbf{r}^{\mathrm{new}})^{-1}\mathbf{T}_p^\top). \tag{5.2}$$

$$\mathbf{g}_p{}^{new} = (\mathbf{K}(\mathbf{r}^{\mathrm{new}},\mathbf{R})\mathbf{K}(\mathbf{R},\mathbf{r}^{\mathrm{new}})^{-1}\mathbf{G}_p^\top)^\top. \tag{5.3}$$

5.5 Simulation Results

Here, simulation results are provided for the mission of driving various tracks to compare the enhanced performance of seg-PDMPs with previous PDMP approaches.In this mission, a mobile vehicle drives P-, S-, and U-shaped tracks in sequence (Fig. 5.2). As mentioned earlier, general DMP can only generalize initial and terminal offset, while parametric skill enables PDMPs and seg-PDMPs to generalize the motion style of various demonstrations. Four demonstrations are provided on each track (two different curvatures $\times$ two different directions, $M_i = 4$ for $i = 1, 2, 3$). All four settings of each course should be considered for generalization in PDMPs or seg-PDMPs. When applying PDMPs for the entire mis-

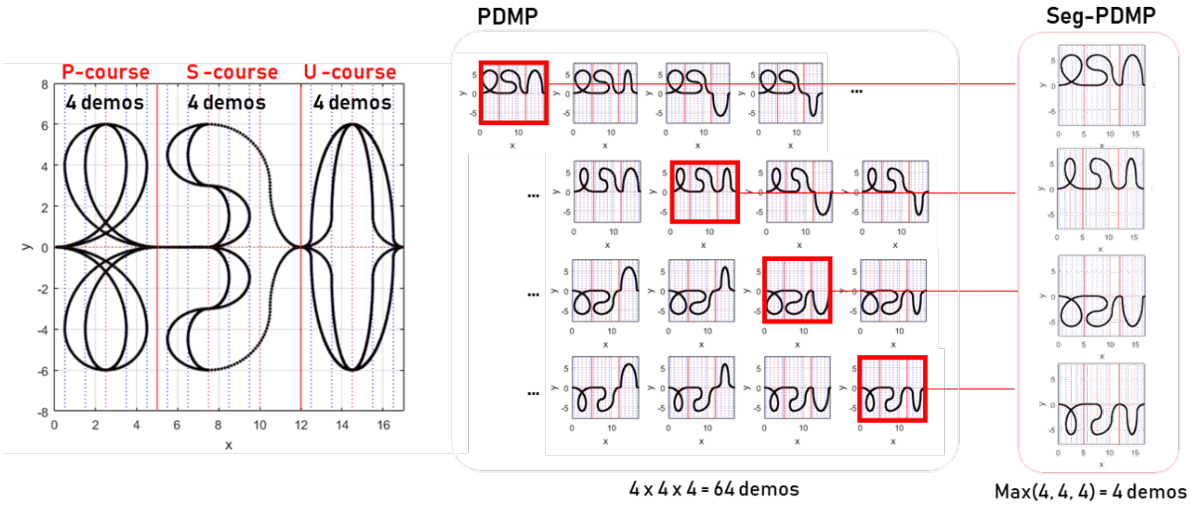
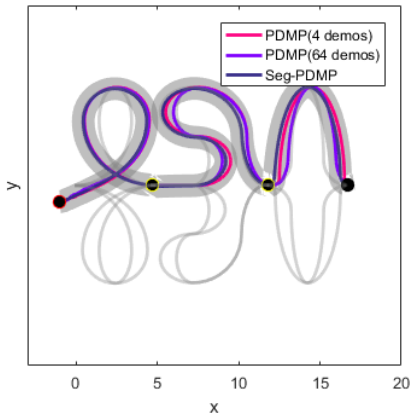


Figure 5.2: Required demonstrations for full generalization of driving multiple tracks scenario. In each demonstration, a mobile robot drives the course consisting of sequential P-, S-, and U-shaped courses. Each course may have various curvature and direction. Therefore, each course requires at least 4 demonstrations (2 directions $\times$ 2 curvatures).

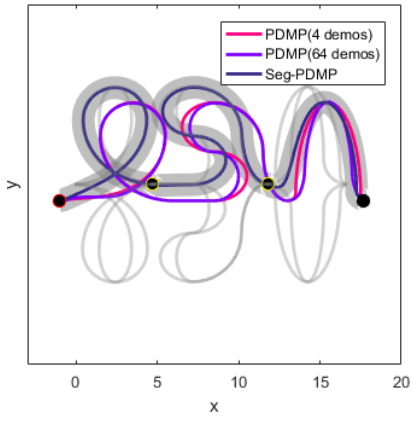
sion rather than individual tracks, the numbers of demonstrations are $M = 64$ for fixed sequencing and $M = 1536$ for flexible sequencing, respectively. On the other hand, seg-PDMPs require $M = 4$. In the following, the performances are compared between PDMPs with 64 demonstrations and the performance of PDMPs and seg-PDMPs with the same four demonstrations.

5.5.1 Initial/terminal Offsets

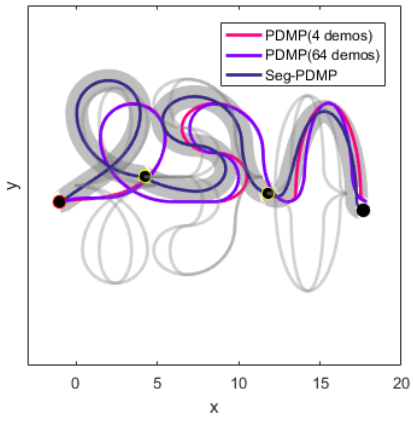
In the online processes of both DMPs and PDMPs, the overall representation of the reference trajectory is modified to deal with the offsets in initial and terminal configurations. Here, the reference trajectory indicates an expected trajectory without variations. The mission conditions, such as via-points or specific movements considered in the reference trajectory, may not be satisfied due to the offsets. For instance, in extending or shortening the reference trajectory, a new trajectory may deviate from via-points. This problem can be reduced to some extent in seg-PDMPs. Since the seg-PDMPs deal with sub-tasks indi-



(a)

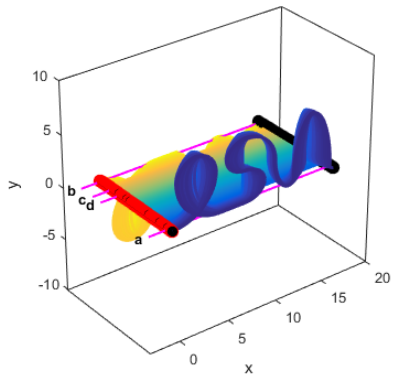


(b)

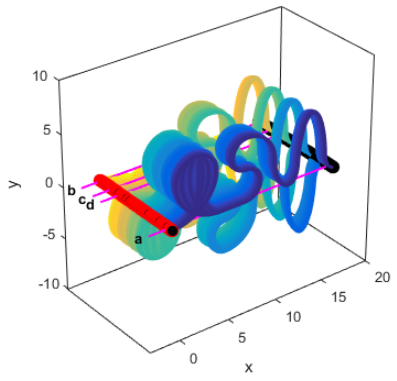


(c)

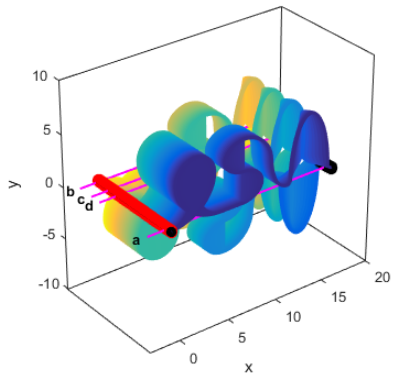
Figure 5.3: Simulation results for driving multiple tracks scenario with different initial/final offsets and via-points. Thick gray line shows the differed environmental settings including the initial, final, and via-point offset. Light gray lines are the demonstrations.



(a)



(b)



(c)

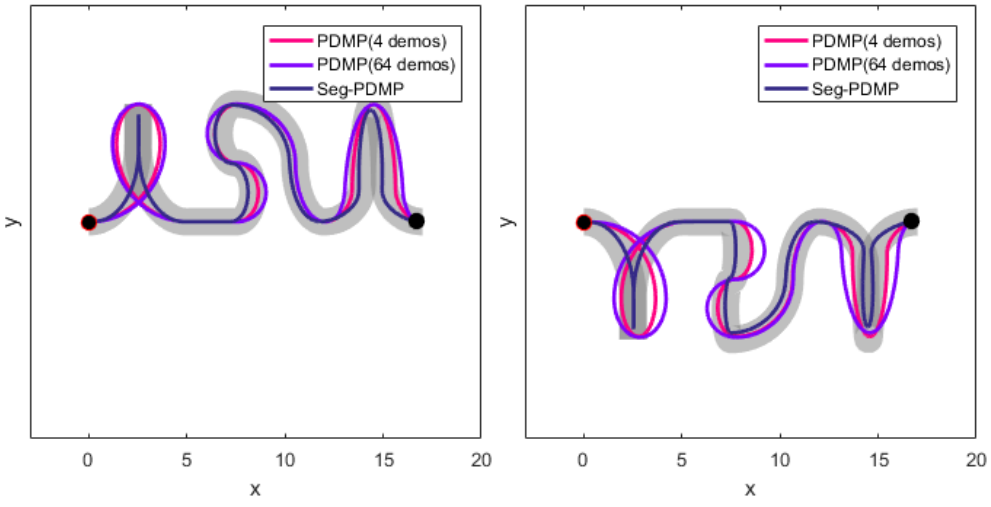
Figure 5.4: Generalization results of (a) PDMPs with four demonstrations, (b) PDMPs with 64 demonstrations, and (c) seg-PDMPs for all possible environmental parameter values (or possible environments). The results of each environment is represented along z-axis. In Fig. 5.5, the results for 'a', 'b', 'c', and 'd' environments are listed in detail.

vidually, maneuvers are modified where the current phase is directly involved in the offsets. Also, for via-point conditions given as intersection points between sub-tasks, the via-point can be satisfied by setting them to the regional goal state of the preceding phase. To afford via-points here, the GPR process to obtain the regional goal in Section 5.4 is ignored.

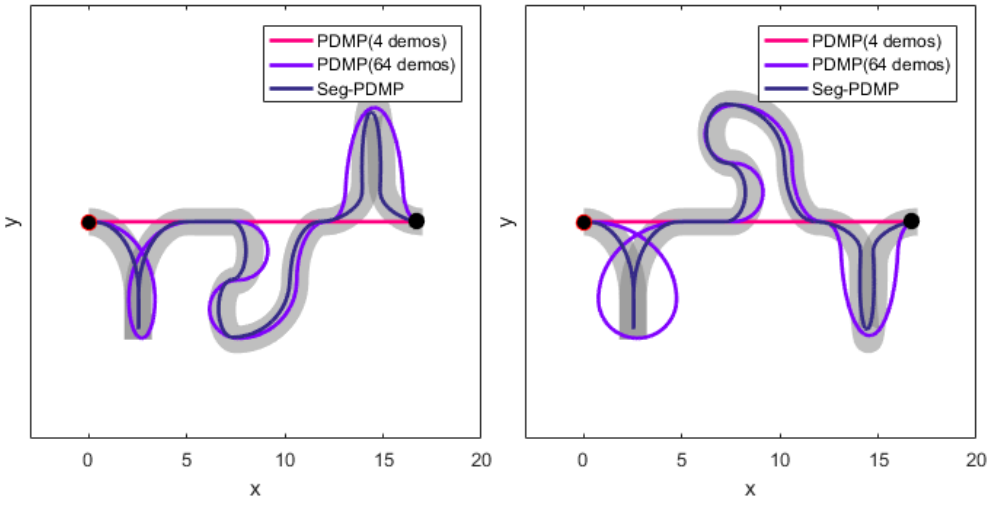
In Fig. 5.3, the representation results are provided for the driving mission using seg-PDMPs and two PDMPs with four and 64 demonstrations. Initial offsets are given for all cases and terminal offsets are given in Figs. 5.3b-5.3c. In Fig. 5.3a, only P-track in seg-PDMPs is affected by the initial offset while S- and U-tracks are additionally affected in PDMPs. Figs. 5.3b-5.3c show that the via-point conditions cannot be satisfied due to offset in some cases in PDMPs. In contrast, seg-PDMPs satisfy via-points by considering them as the regional goals of sub-tasks.

5.5.2 Style Generalization

With the parametric skills, motion styles as well as offsets can be generalized in PDMPs and seg-PDMPs. For the driving mission, the generalization performances on various tracks with different curvatures and directions are compared in Fig. 5.4. Simulation results for all the possible track settings are displayed along the z-axis where different curvature and direction values are set. Each of Fig. 5.4a, Fig. 5.4b, and Fig. 5.4c respectively shows the generalization results of PDMPs with four and 64 demonstrations and seg-PDMPs with four demonstrations. In Figs. 5.5c-5.5b, the generalization results for four different environments are presented. When the same set of demonstrations are provided to PDMPs and seg-PDMPs ($M = 4$), seg-PDMPs show better generalization performance. It is expected since at least 64 demonstrations need to be provided to PDMPs for full generalization. Even 64 demonstrations are provided for PDMPs, the generalization performance is poor regarding the S- and U-shaped tracks which are located much farther than the P-shaped track from starting point. Compared with PDMP results, seg-PDMPs keep the generalization performance for both S- and U-shaped tracks. This is because when extracting the attractor basis by taking SVD in PDMPs, the values related to P-track are relatively dom-



(a) (b)



(c) (d)

Figure 5.5: Reproduction results from two PDMPs and seg-PDMPs for (a) 'a', (b) 'b', (c) 'c', and (d) 'd' environment in Fig. 5.4. Each pink, purple, and navy line indicates the result of PDMPs with four and 64 demonstrations, and seg-PDMPs with four demonstrations, respectively. Thick grey line shows the differed environmental settings. Red and black circles are initial and goal configurations.

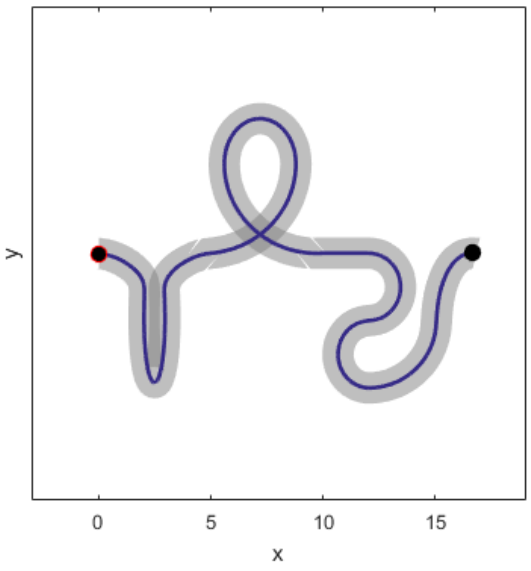
	DMP	PDMPs		Seg-PDMPs
of demonstrations	-	4	64	4
Initial/final offset	Task constraints may not be satisfied by stretching or shrinking to overcom offset			-
Generalization	unable	unable	able	able
Re-combination	unable	unable		able

Table 5.2: Algorithm summary of single DMP, PDMPs, and seg-PDMPs.

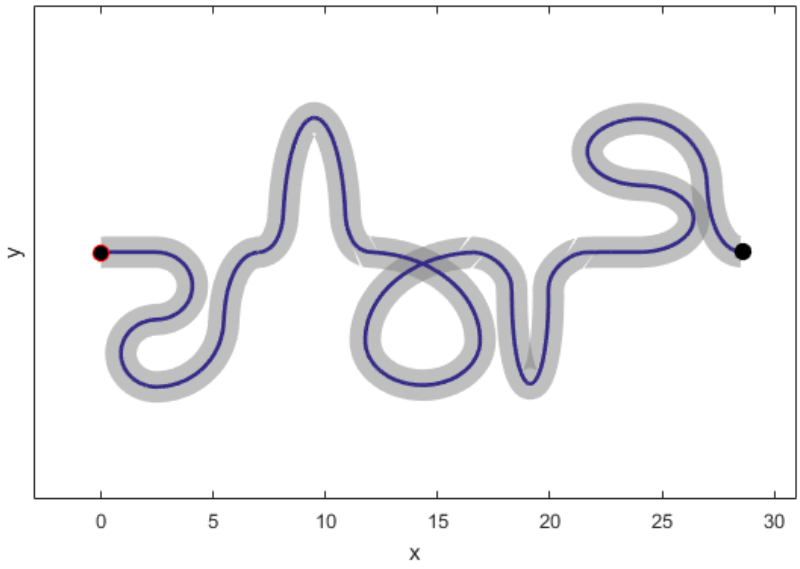
inant compared to other tracks. The PDMPs with four demonstrations show invalid results in some environments as shown in Figs. 5.5c-5.5d.

5.5.3 Recombination

Unlike the general PDMPs, seg-PDMPs has the structure where multiple PDMPs are configured on each sub-task individually and called autonomously according to the situations. Therefore, depending on the situation, the robot can recall the sub-task already done, so the overall mission composition may vary. Figs. 5.6a-5.6b show the results of different combinations of driving course compared to the demonstrations. Table 5.2 summarizes the previous simulation results.



(a)



(b)

Figure 5.6: Course reconstruction results using seg-PDMPs algorithm. In each simulation, the different settings are given from the learned demonstrations including (a) the course order and initial position, and (b) the course order, initial position, and final position, and the number of unit courses. Thick grey line shows the differed environmental settings.

6

Experimental Validation and Results

In this section, the proposed PDMPs algorithm and seg-PDMPs are validated with experiments of mobile manipulators with two types of vehicles. The first two scenarios regarding cooperative aerial transportation demonstrate the excellence of the proposed technique in terms of quick computation, generations of optimum movement, and safety assurance for the movement. The last two scenario deal with two mobile hang-dry missions using ground vehicles and show how the seg-PDMPs are applied to perform complex missions. The experimental setups and control architectures are presented in section 2.2 for both types of mobile manipulators. This part focuses on checking the reliability of the resulted motions and completeness of the missions in real environments.

6.1 Cooperative Aerial Transportation

The scenarios for aerial transportation have already been validated in sections 3.3 and 4.3 through the simulation results. Here, experiments verify the aerial manipulators also complete the transportation mission through the proposed PDMPs with real-time feedback. Since robot manipulation is affected by various kinds of perturbations during practical

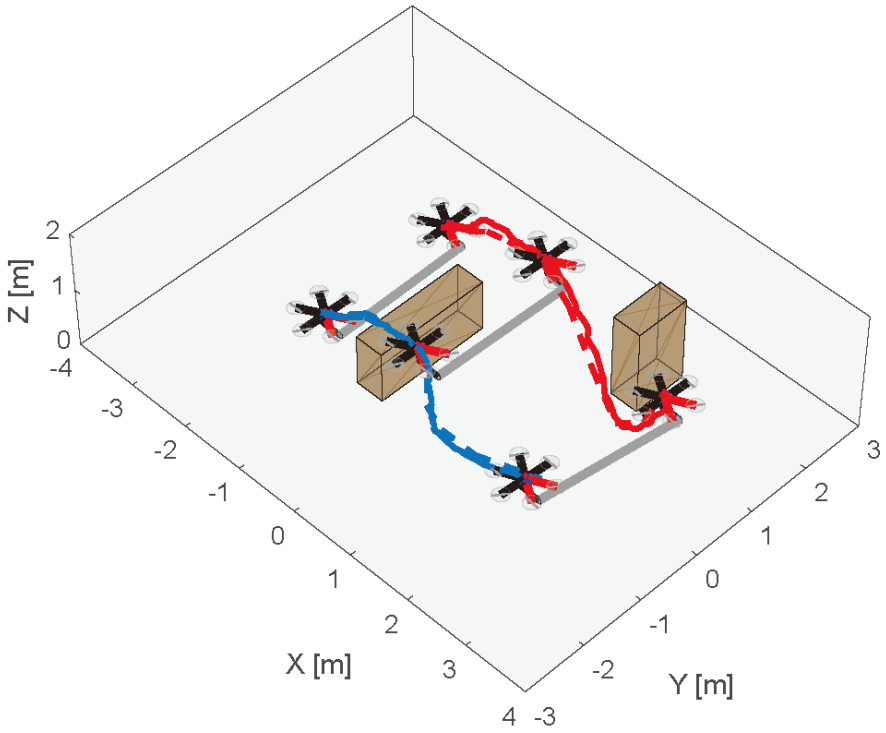
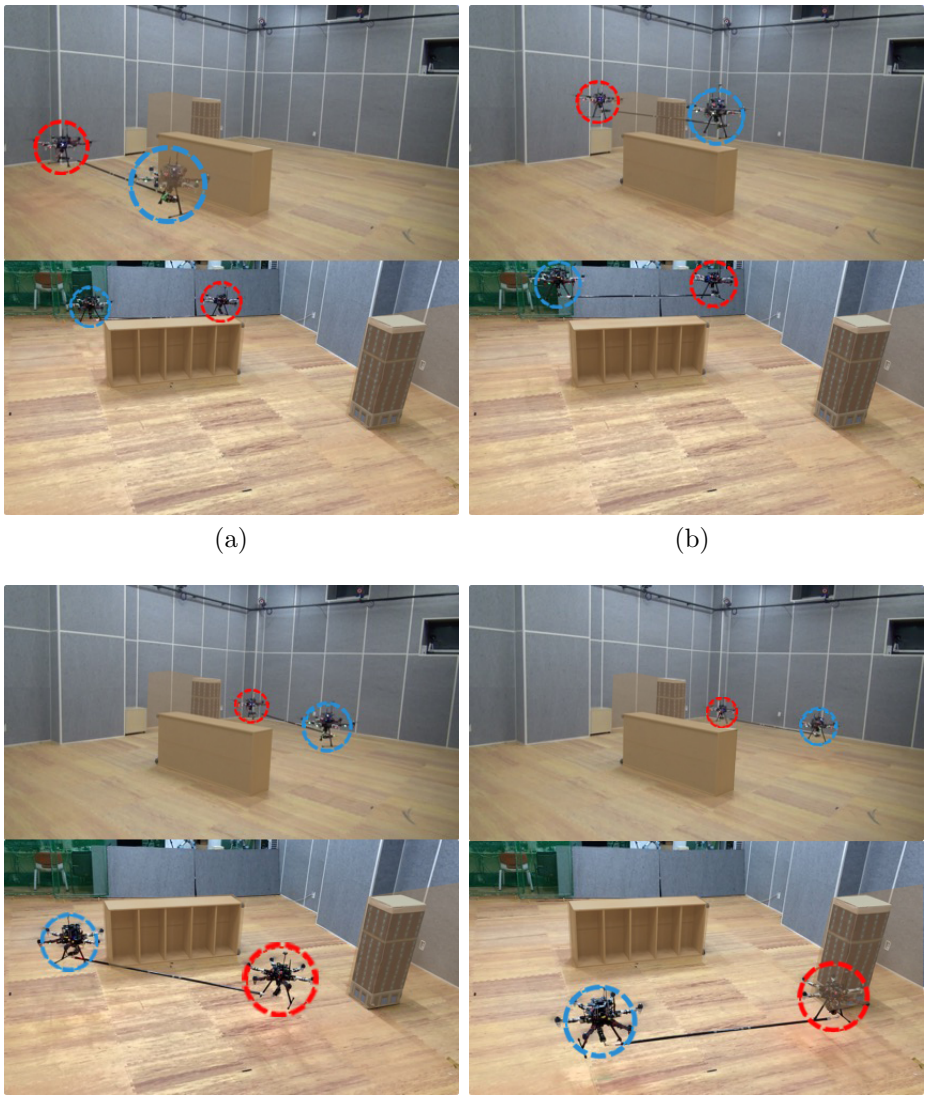


Figure 6.1: Resulting trajectories recorded from experiments for the first scenario. Desired trajectories are depicted in dashed lines, and actual trajectories are depicted in solid lines. Each red and blue line shows the body trajectory of the left and right aerial manipulator, respectively.

operation, this experimental validations show the effectiveness of PDMPs robustness to the perturbations.

As described above, the scenario settings are same as sections 3.3 and 4.3. The experimental results are reported in Figs. 6.1-6.6. The first scenario regards section 3.3 while the second scenario is related to section 4.3. The resulting trajectories from aerial manipulators for are shown in Figs. 6.1 and 6.4. Figs. 6.2 and 6.5 show the snapshots of the experiments. The time histories of the state variables are listed in Figs. 6.3 and 6.6. In the second scenario, the additional cylindrical obstacle is located near the starting point, which is never considered in the first scenario. As described in section 4.3, demonstrations are updated by applying the proposed safety guaranteeing process in section 4, and the aerial manipulators successfully avoid the obstacle.



(a) (b)

(c) (d)

Figure 6.2: Snapshots of the experiment for the first scenario. (a) starting operation, (b) avoiding first obstacle, (c) avoiding newly faced obstacle, and (d) terminating operation. Each red and red blue indicates left and right aerial manipulators, respectively.

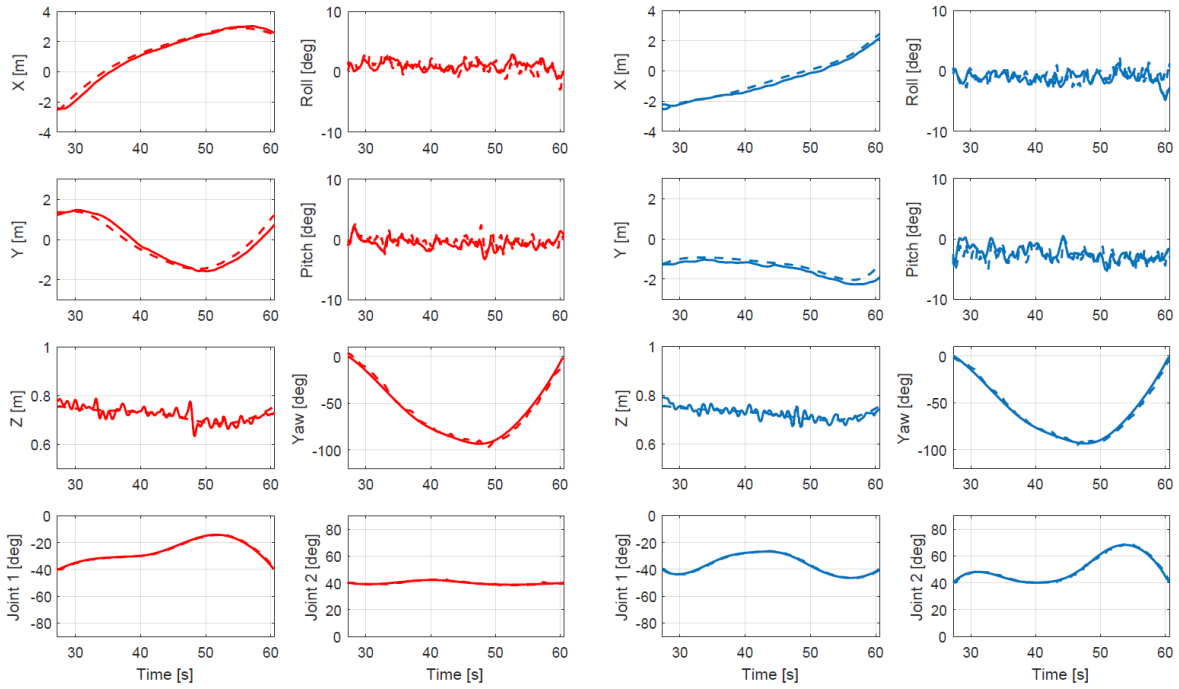


Figure 6.3: Time histories of the state variables of the aerial manipulators for the first scenario. (a) Position, attitude, and joint angles of the left aerial manipulator, and (b) the right manipulator. Desired trajectories are shown in dashed lines, and actual trajectories in solid lines.

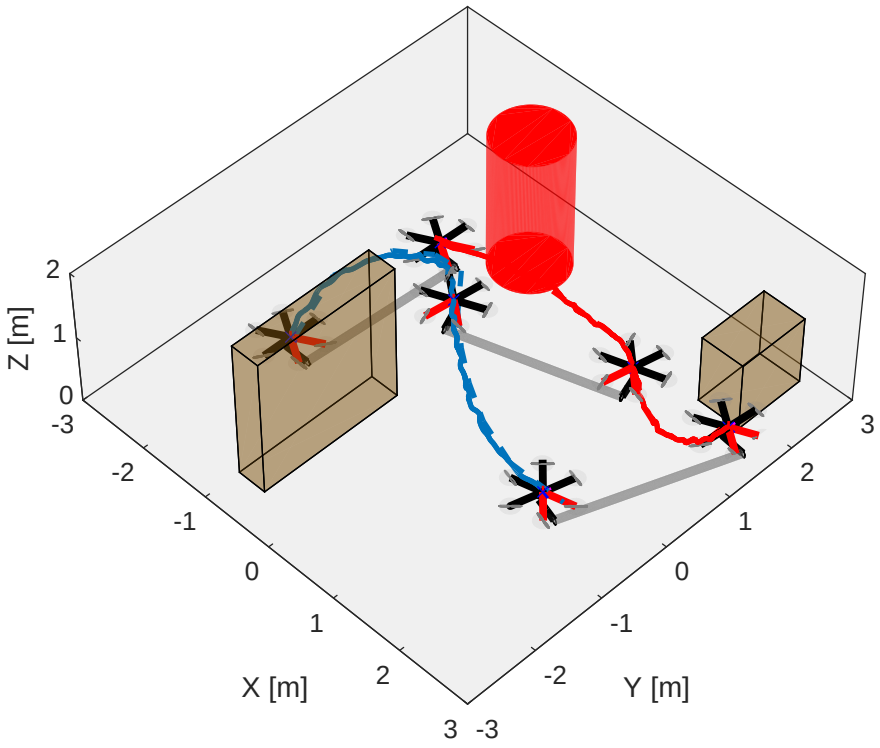
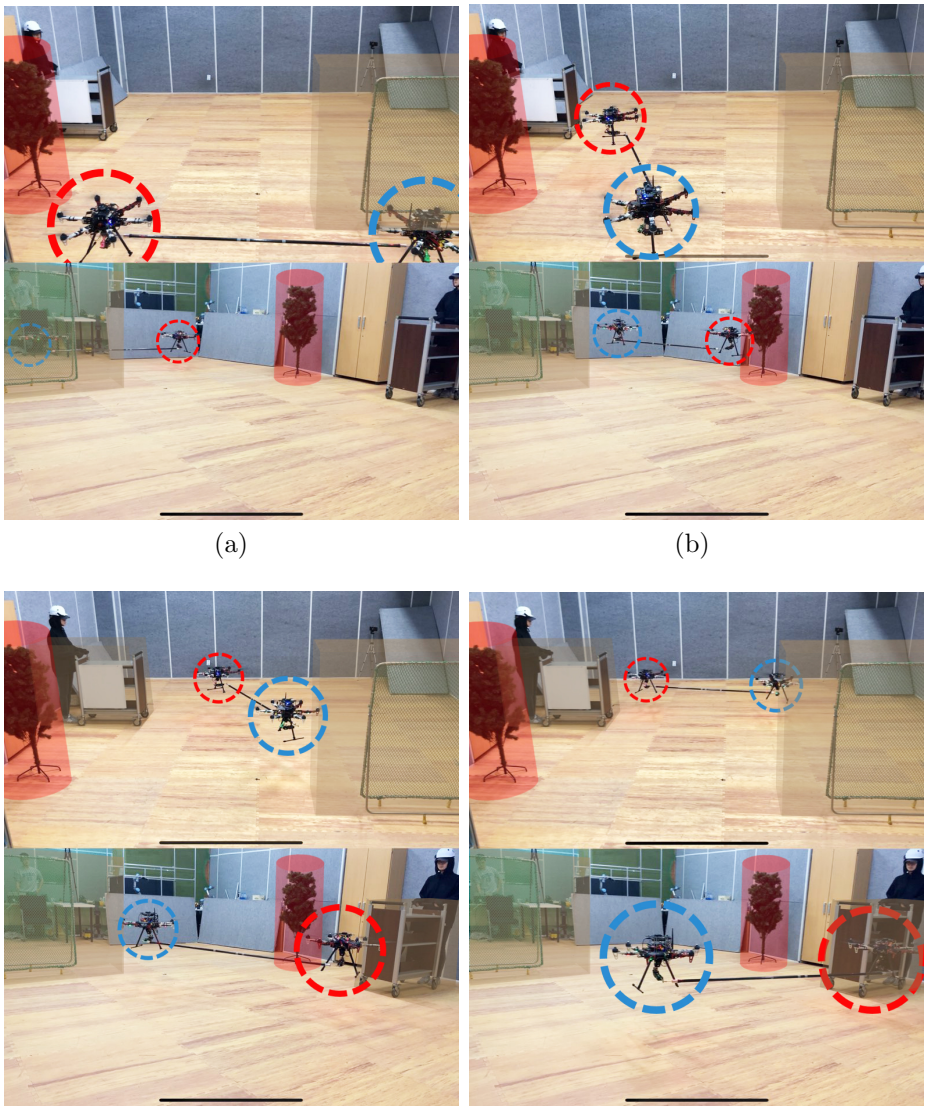


Figure 6.4: Resulting trajectories recorded from experiments for the second scenario. The cylindrical obstacle is added from the first scenario and other obstacle settings are also changed. Desired trajectories are depicted in dashed lines, and actual trajectories are depicted in solid lines. Each red and blue line shows the body trajectory of the left and right aerial manipulator, respectively.



(a) (b)

(c) (d)

Figure 6.5: Snapshots of the experiment for the second scenario. Snapshots at (a) starting operation, (b) avoiding first obstacle, (c) avoiding newly faced obstacle, and (d) terminating operation. Each red and red blue indicates left and right aerial manipulators, respectively.

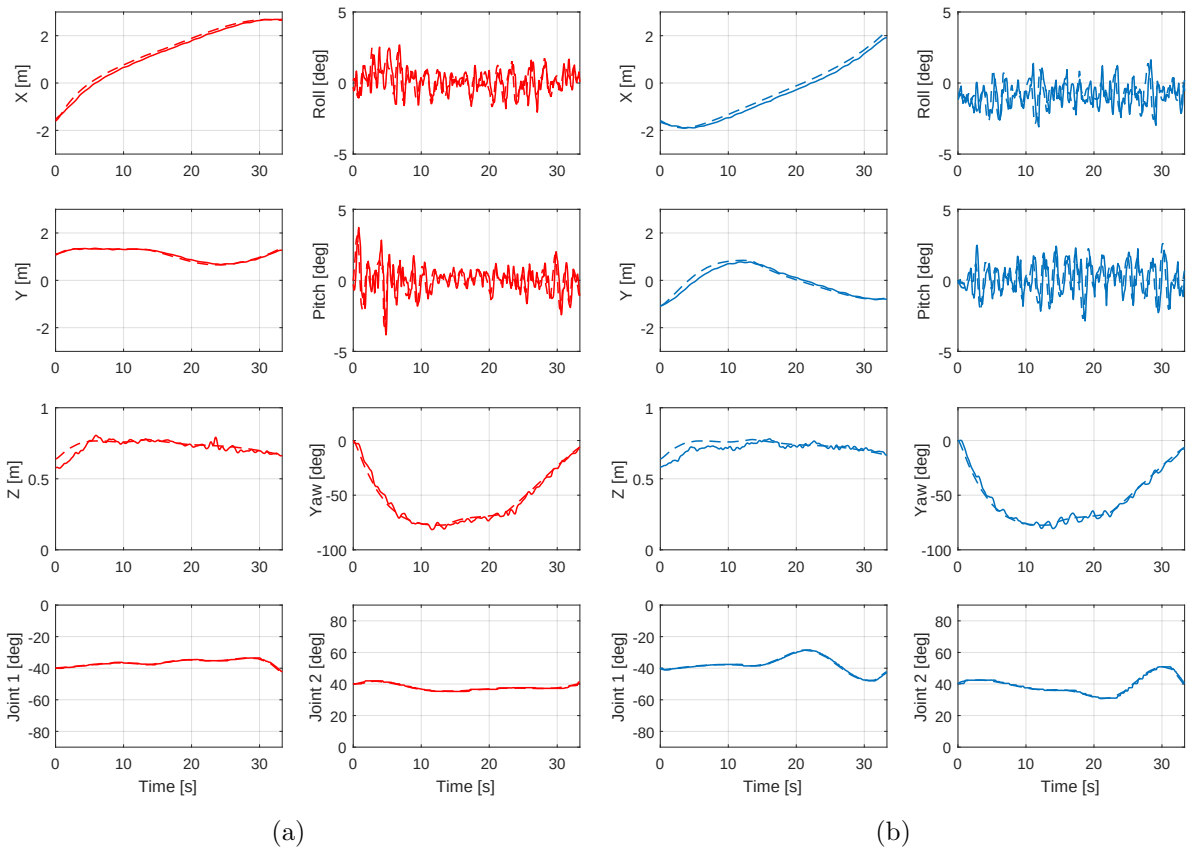


Figure 6.6: Time histories of the state variables of the aerial manipulators for the second scenario. (a) Position, attitude, and joint angles of the left aerial manipulator, and (b) the right manipulator. Desired trajectories are shown in dashed lines, and actual trajectories in solid lines.

6.2 Cooperative Mobile Hang-dry Mission

The experiments on hang-dry mission regard seg-PDMPs, which is the extension PDMPs algorithm in section 5. Two mobile manipulators are employed where each manipulator consisting of the two-wheeled mobile robots and four-DoF robotic arms. The detailed system descriptions and hardware setups are already presented in sections 2.2 and 2.3. Hang-dry missions for cooperative mobile manipulation system is also shown in Fig. 1.1. In the manipulation scenario, first, two mobile manipulators start to move towards a pile of clothes from different positions. Until they grasp each tip of a given object (cloth), they maneuver independently. The transportation process begins after both robots grab the cloth. In the process of transportation, the robots are constrained to each other by holding the cloth cooperatively, and sometimes they encounter obstacles to avoid. When they get near hanger, they stretch out the cloth and move to the each opposite side of the hanger. Finally, they pull down both ends of the cloth and finish the task. Thus, during the hang-dry mission, the robots perform sub-tasks including motions like grasping, stretching and releasing.

6.2.1 Demonstrations

There are various ways to provide demonstrations and they are affected by factors like complexity of the system or task [28]. While most of the work for a few demonstrations have made use of human demonstrators, the simulated planners is used to repeatably compute the demonstrations with a different environment. Moreover, objective of this work deals with complex motion skills which are typically difficult to control, the simulated planner would benefit various environments. The computational time to generate demonstrations does not need to be a real-time scale as the process of organizing a demonstration set is an offline process in PDMPs. Therefore, in order to generalize the efficient task execution, the trajectories are obtained from optimal motion planner. As the optimization technique, iterative linear quadratic regulator (iLQR) is applied for hang-dry mission. Unlike RRT* in cooperative aerial transportation scenario, iLQR can formulate via-points for hang-dry

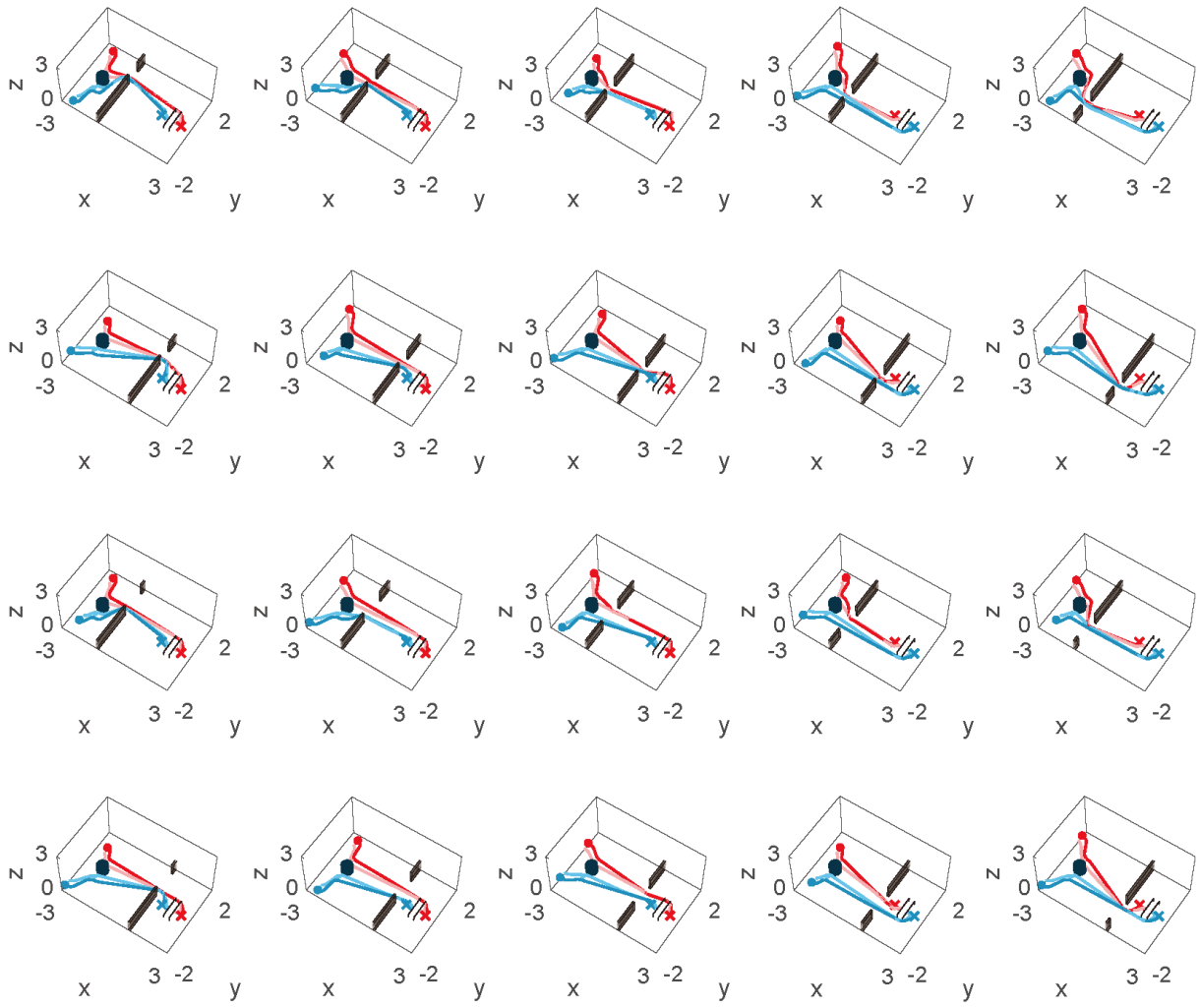


Figure 6.7: Twenty demonstrations computed from iLQR. The performance index of the trajectories are the travel length through the operation. The starting positions of robots and the locations of freeway against obstacles are different in each environment. Each red and blue color indicates that the trajectories are concerned with the left and the right mobile manipulator, respectively. The solid lines are the body trajectories of robots while light colored lines are end-effector trajectories.

M_1	M_2	M_3	M_4	M_5	M_6
16	2	20	20	4	4

Table 6.1: The number of required demonstrations in sub-tasks for hang-dry mission.

mission. As a result, twenty different environmental settings are selected to fully describe general environment for each sub-tasks. Hang-dry mission consists of six sub-tasks and the corresponding number of required demonstrations in each task are in Table. 6.1. Be notices that if PDMPs are applied instead of the seg-PDMPs, the total number of required demonstrations will be $16 \times 2 \times 20 \times 20 \times 4 \times 4 = 204,800$ here. In those environments, environmental settings are various with different starting positions, locations and dimensions of obstacles, heights of the first position of cloth or hanger. Fig. 6.7 shows the demonstrations of iLQR. The segmented demonstrations corresponding to each phase is shown in Fig. 6.8.

6.2.2 Simulation Validation

In order to show the seg-PDMPs allow repeated sub-tasks, two different environments are set for simulation by giving sequential one and three obstacles while the demonstrations only include one obstacle. Here, the obstacles are never given as demonstration settings before. The simulation results are shown in Fig. 6.9. Each green and yellow area shows starting and goal points in the entire task execution, respectively. The red lines show the trajectories of the left mobile manipulator while the blue lines show the trajectories of the right one. The phase bars are shown at the bottom of the figures indicating which phase is operated along the time. In the seg-PDMPs, repeatable tasks such as obstacle avoidance ($p = 3, 4$) can be automatically computed if the agents determine the task should be operated again based on the current situation. Thus, more than one obstacle can be deployed in the online seg-PDMPs. All the simulations also have different starting points and obstacle positions compared to demonstrations. From the results, the required sub-tasks are exactly executed in sequence at proper moments during the task.

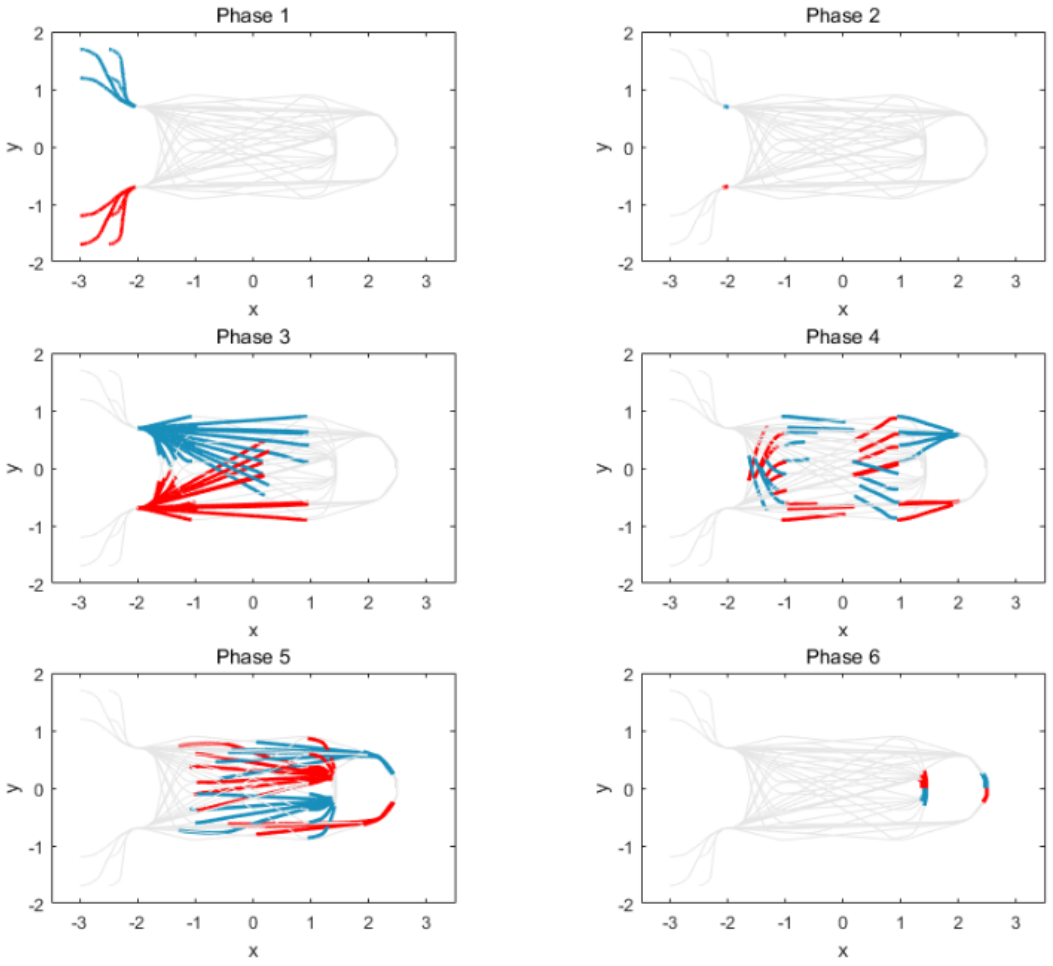
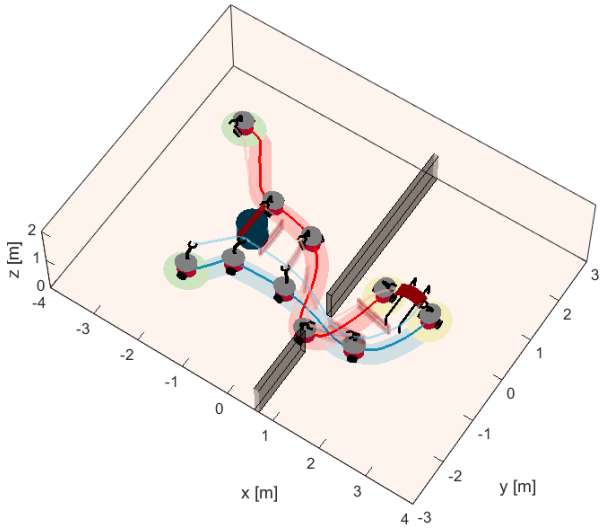
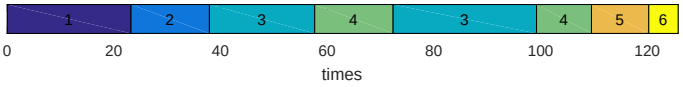


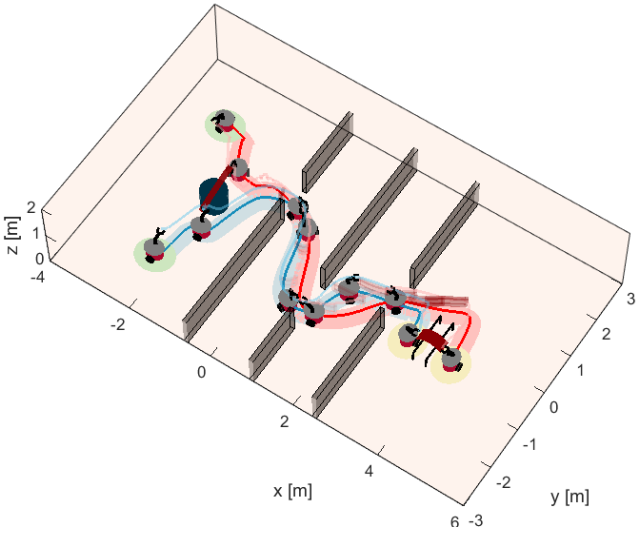
Figure 6.8: Six phases consist of the hang-dry mission in Fig. 6.7. Each red and blue line is the trajectories of the left and the right mobile manipulator, respectively. The grey lines show the rest trajectories in each phase.



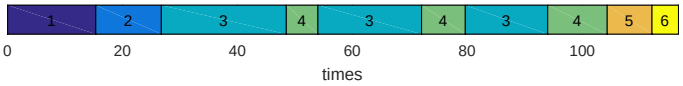
(a)



(b)



(c)



(d)

Figure 6.9: Computed trajectories of two mobile manipulators with differed environmental settings from demonstrations. (a,b) Single obstacle is considered and (c,d) three obstacles sequentially appear during transportation.

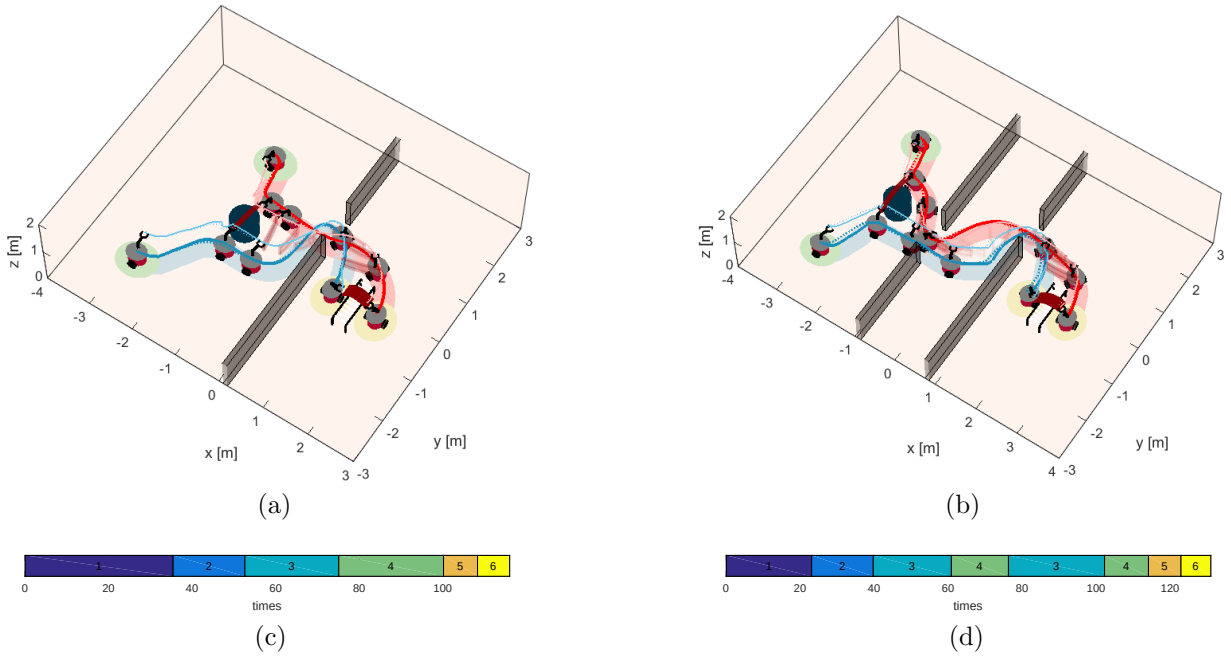
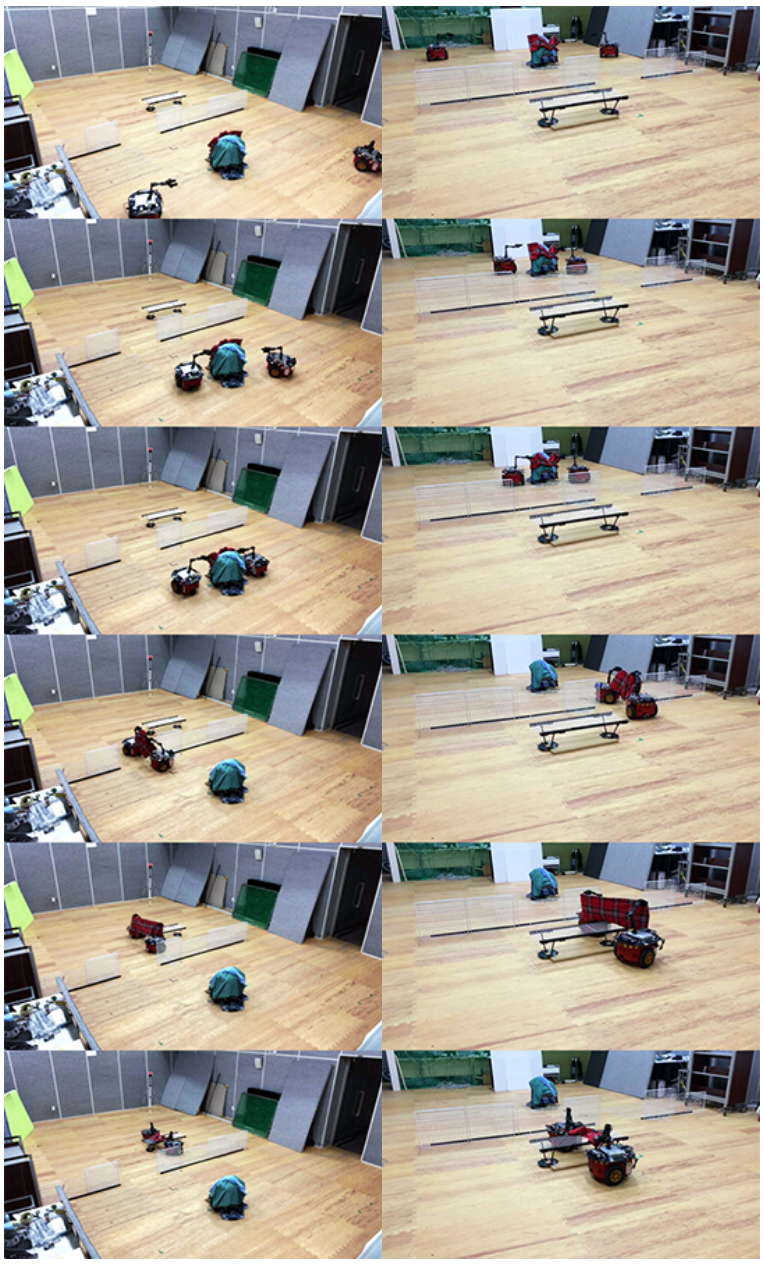
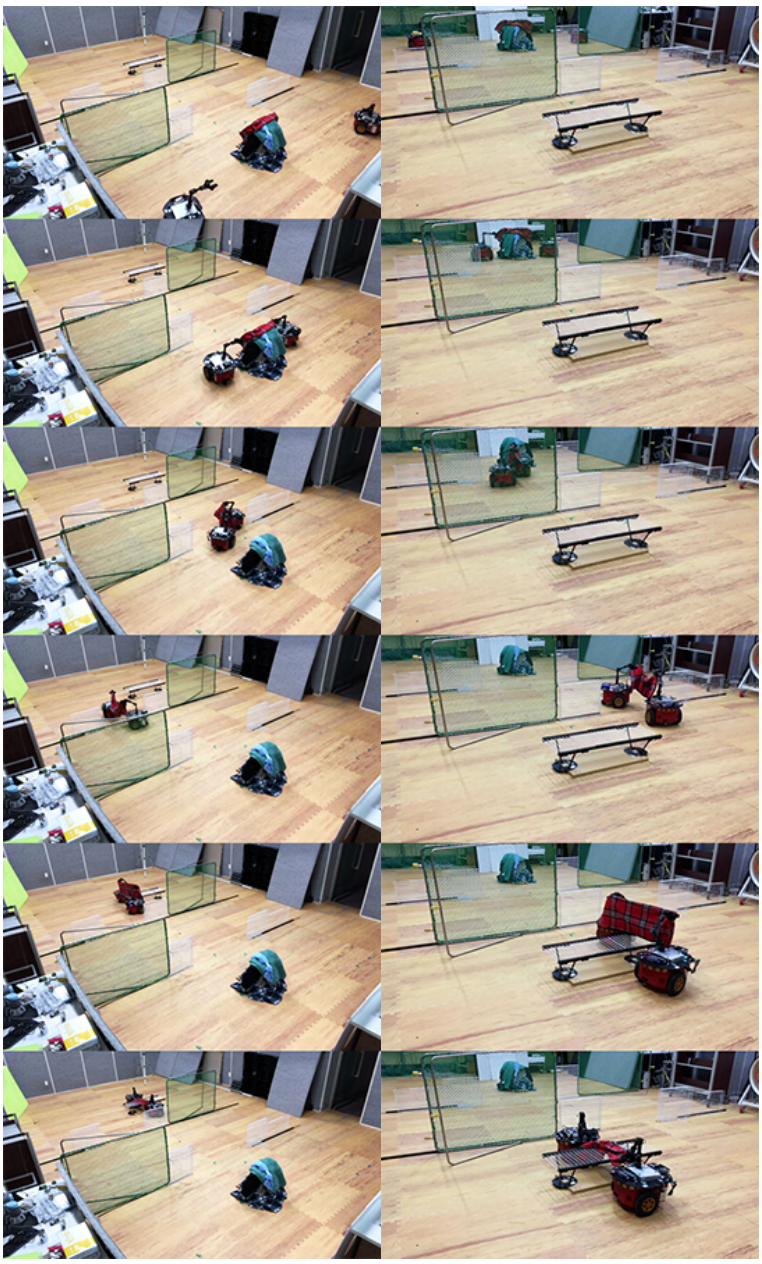


Figure 6.10: Resulting trajectory recorded from the (a,c) first and (b,d) second experiments. Two mobile manipulators move from the starting point (green circled area) to the hanger (yellow circled area). Each red and blue line shows the trajectory of the left and right mobile manipulator, respectively. The dark lines of each color denote the trajectories of the body position of mobile manipulators while the light lines represent the position of end-effectors. Each light red and light blue area shows the actual trajectories with the left and right platform dimension, respectively. The solid line indicates the recorded trajectories and dashed line indicates the desired trajectories computed from the proposed seg-PDMPs.



(a) (b)

Figure 6.11: Snapshots during the first experiment taken from two different perspectives (a) and (b). The robots start moving towards the pile of clothes (0 sec). The second robot first reaches the pile and grabs the fabric (red checkered blanket) located on top of the pile (20 sec). After that, the first robot also arrives in the pile and grabs the opposite end of the fabric (30 sec). The robots avoid obstacle (70 sec) while cooperatively holding the fabric (70 sec). When they reach the hanger, they stretch the fabric (90 sec) and hang it on the hanger (116 sec).



(a) (b)

Figure 6.12: Snapshots during the second experiment, taken from two perspectives (a) and (b). The robots start moving towards the pile of clothes (0 sec). Both robots reach the pile simultaneously and grabbing the each tip of the fabric (red checkered blanket) at the top of the pile (20 sec), respectively. In the second experiment, there are two obstacles. The robots avoid the first obstacle (70 sec) and the second obstacle (90 sec) while cooperatively holding the fabric. When they reach the hanger, they stretch the fabric (110 sec) and hang it on the hanger (131 sec).

<table>
<tr><td rowspan="2"></td><td rowspan="2">Phase</td><td rowspan="2">N</td><td rowspan="2" colspan="2"># of demos M</td><td rowspan="2">L</td><td rowspan="2" colspan="2">Time length</td><td colspan="4">Computational time</td></tr>
<tr><td>Getting demos (offline)</td><td colspan="2">Learning (offline)</td><td colspan="2">Reproduction (online)</td></tr>
<tr><td>Cooperative aerial transportation</td><td>Single</td><td>7</td><td colspan="2">24</td><td>25</td><td colspan="2">1001</td><td>602,342 s (RRT*)</td><td colspan="2">2.63 ms</td><td colspan="2">3.91 ms</td></tr>
<tr><td rowspan="6">Cooperative hang-dry mission</td><td>1</td><td rowspan="6">14</td><td rowspan="6">20</td><td>16</td><td>36</td><td rowspan="6">1855</td><td>230</td><td rowspan="6">156.24 s (iLQR)</td><td rowspan="6">44.57 ms</td><td>5.88 ms</td><td rowspan="6">16.55 ms</td><td>1.04 ms</td></tr>
<tr><td>2</td><td>2</td><td>4</td><td>50</td><td>1.25 ms</td><td>0.54 ms</td></tr>
<tr><td>3</td><td>20</td><td>58</td><td>750</td><td>12.29 ms</td><td>4.96 ms</td></tr>
<tr><td>4</td><td>20</td><td>79</td><td>525</td><td>12.50 ms</td><td>5.25 ms</td></tr>
<tr><td>5</td><td>4</td><td>53</td><td>125</td><td>9.17 ms</td><td>3.16 ms</td></tr>
<tr><td>6</td><td>4</td><td>43</td><td>175</td><td>4.47 ms</td><td>1.60 ms</td></tr>
</table>

Table 6.2: Detailed computational time.

6.2.3 Experimental Results

This section shows two online experiments involving one or two obstacles which are previously unknown. Similar to the simulations Fig. 6.9, the different positions of the obstacles and the robots are given from the prescribed demonstrations. During the task, the ground robots are controlled to follow calculated trajectories. From the VICON system, current environmental information and the robot states are transferred to VICON computer. This information is also used for seg-PDMPs to calculate trajectories. The calculated trajectories are transmitted from ground station to the on-board computer via wireless communication.

The experimental results are reported in Figs. 6.10-6.12. Each Fig. 6.10a and Fig. 6.10b shows the resulting trajectories recorded from the first and second experiments, respectively. Figs. 6.10c and 6.10d demonstrate the phase state along the time. Since there are two obstacles in the second scenario, the phases $p = 3, 4$ involving an obstacle avoidance task are repeated again during the task. Figs. 6.13-6.14 demonstrates the time histories of the state variables of the two agents. Fig. 6.13 shows the first experiment and Fig. 6.14 indicates the second experiment. Each Figs. 6.11-6.12 shows the snapshots during the first and second experiments, respectively. In both environments, the agents approach the pile of clothes and pick up the cloth(pink colored cloth) whenever they arrive at the pile. During the task, they encounter the obstacles once or twice. Based on the proposed framework, they generate the motion which can be compatible with the current environment. Be noticed that both environments have different environmental settings such as positions of starting points or

obstacles to show that the proposed PDMPs framework can allow any environments which are never given previously as one of the demonstration set.

The detailed computational times for experiments including 6.1 and 6.2 are listed in Table. 6.2

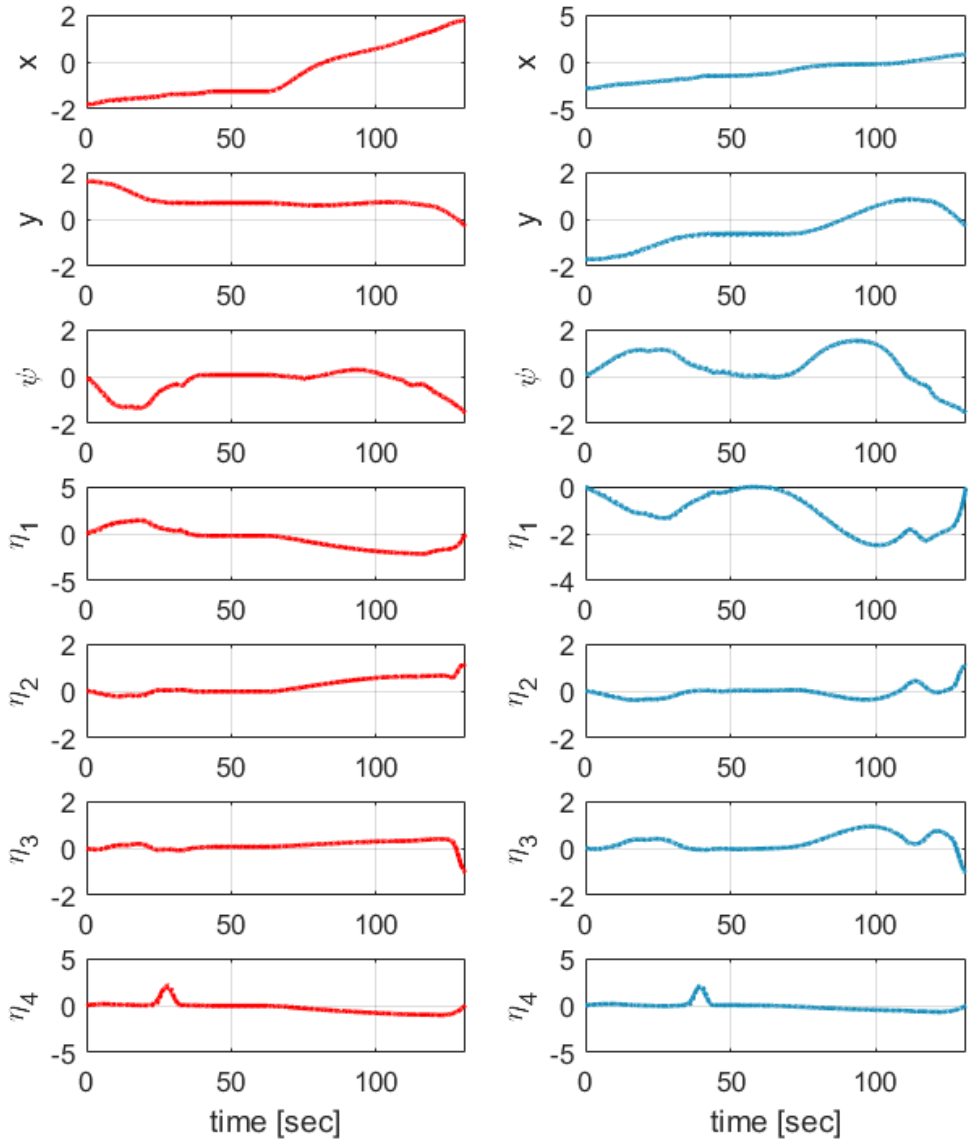


Figure 6.13: Time histories of the state variables of the mobile manipulators for the first experiment. Desired trajectories are shown in dashed lines, and actual trajectories in solid lines. Each red and blue line indicates the left and the right mobile manipulator, respectively.

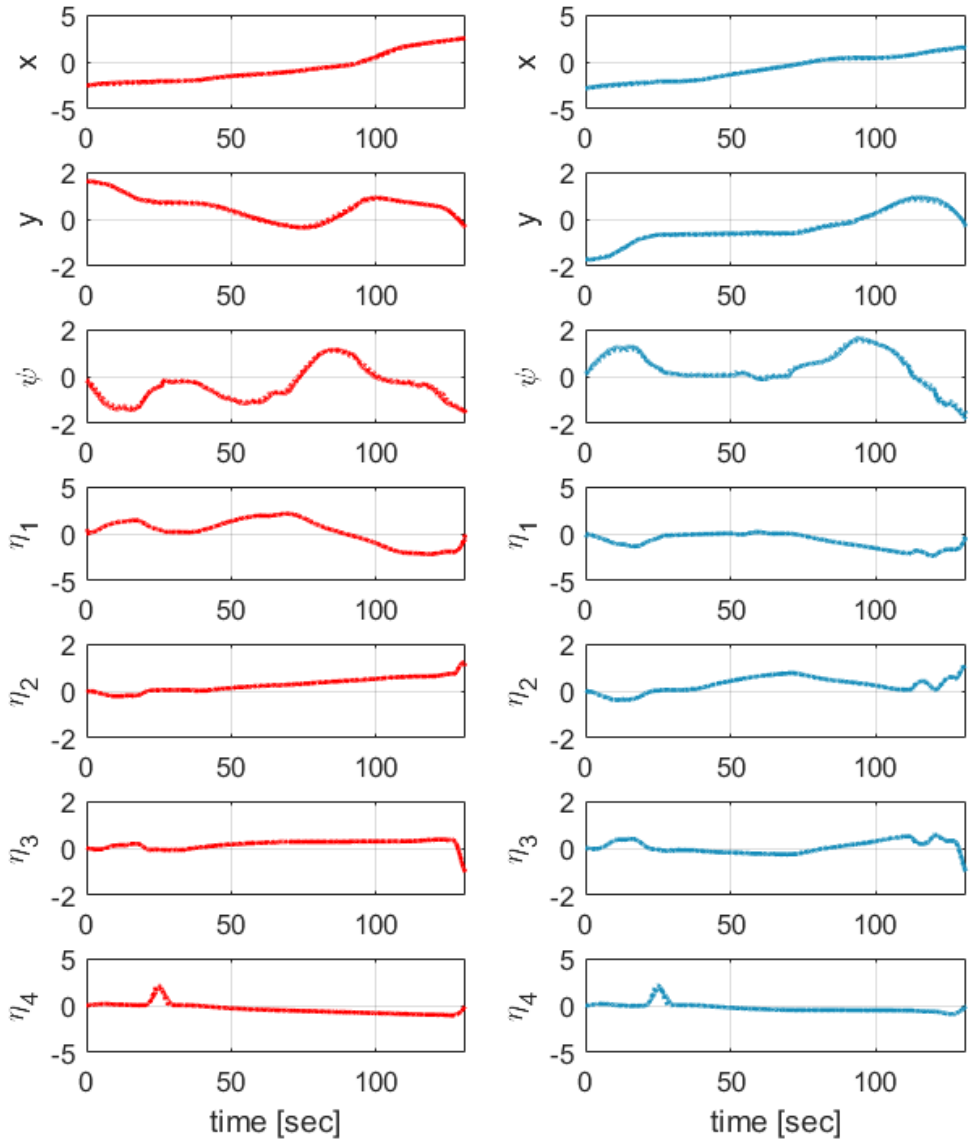


Figure 6.14: Time histories of the state variables of the mobile manipulators for the second experiment. Desired trajectories are shown in dashed lines, and actual trajectories in solid lines. Each red and blue line indicates the left and the right mobile manipulator, respectively.

7

Conclusions

Dynamic movement primitives (DMPs) are well known as a promising method to specify the robot movement particularly for robot manipulations. They guarantee quick computations and robust representation against the perturbations. Parametric DMPs (PDMPs) take advantages of these DMPs and furthermore, they provide parameterized representation from multiple demonstrations. The new representation of motion is possible in PDMPs by selecting the parameters. In this dissertation, the relevant research of PDMPs were addressed regarding (i) generalization of motion, (ii) safety guarantee, and (iii) managing the number of demonstrations.

- First topic was the generalization process in PDMPs. By learning the relationship between known environments and corresponding motions using Gaussian process regression(GPR), the proposed framework can serve as a real-time and optimal planner by learning multiple optimal movements. This dissertation provided the simulation results for aerial transportation scenarios within obstacle environments. By providing the optimal demonstrations of RRT*, the algorithm computed the motion emulating optimal-avoidance for a new environment. This indicates that the trade-off issue be-

tween the motion optimality and the computational time in the conventional motion planning problem is effectively relieved by using the proposed framework.

- Second, the safety of motion was addressed. This work suggested the process for calculating the safety criterion of the PDMP internal parameter value. The safety criterion reflected the new behavior computed through the generalization process, as well as the individual motion safety of the demonstration set. The demonstrations causing unsafe behavior were identified and removed through this safety criterion. Further, demolished demonstrations were replaced by proven demonstrations upon the safety criterion. Thus, the representation performance kept the previous level. This work also was utilized to reuse demonstrations when the static environmental settings have changed where all demonstrations should have been replaced.

- Third, this dissertation also presented an extension approach, seg-PDMP, which is beneficial for reducing the number of required demonstrations within the proposed framework. Especially when a single assignment consists of multiple unit sub-tasks and requires numerous demonstrations to generalize them, this approach showed great performance in reducing the number of demonstrations. Through this work, the whole trajectories were segmented into multiple sub-tasks representing unit motions. Multiple PDMPs were formed independently for the correlated-segments called phases. The phase-decision process allowed multiple PDMPs to be configured within an integrated framework. Gaussian process regression (GPR) was applied to obtain execution time and regional goal configuration within each phase from environmental variables.

- Finally, the proposed algorithm and its extension were validated in the experiments of two types of mobile manipulators. The first two scenarios showed the effectiveness of the proposed framework in terms of quick computation, generations of optimum movement, and safety assurance for the movement. The last scenario dealt with two mobile manipulations using ground vehicles and showed how the proposed extension of the proposed algorithm is applied to perform complex assignments. From this

experimental validation, the superiority of this research is evaluated by reducing over 200,000 number of demonstrations into 20 demonstrations.

In general, manipulation points can be assigned during operation and must be corrected in real time. In Seg-PDMPs, via-points can be considered unlike in PDMPs, but there is a limitation that they can be considered only as regional goals of each unit sub-task. Therefore, the possible future works of this study will be the development of a framework that can flexibly consider via-points.

References

[1] H. Kim, H. Lee, S. Choi, Y.-k. Noh, and H. J. Kim, "Motion planning with movement primitives for cooperative aerial transportation in obstacle environment," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2328–2334.

[2] H. Kim, H. Seo, C. Y. Son, H. Lee, S. Kim, and H. J. Kim, "Cooperation in the air : A learning-based approach for the efficient motion planning of aerial manipulators," *IEEE Robotics & Automation Magazine*, vol. 25, no. 4, pp. 76–85, 2018.

[3] D. Mellinger, M. Shomin, N. Michael, and V. Kumar, "Cooperative grasping and transport using multiple quadrotors," in *Distributed autonomous robotic systems*. Springer, 2013, pp. 545–558.

[4] K. Sreenath and V. Kumar, "Dynamics, control and planning for cooperative manipulation of payloads suspended by cables from multiple quadrotor robots," in *Proceedings of Robotics: Science and Systems*, 2013.

[5] C. Y. Son, D. Jang, H. Seo, T. Kim, H. Lee, and H. J. Kim, "Real-time optimal planning and model predictive control of a multi-rotor with a suspended load," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5665–5671.

[6] C. Y. Son, H. Seo, D. Jang, and H. J. Kim, "Real-time optimal trajectory generation and control of a multi-rotor with a suspended load for obstacle avoidance," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1915–1922, 2020.

[7] F. Caccavale, G. Giglio, G. Muscio, and F. Pierri, "Cooperative impedance control for multiple uavs with a robotic arm," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2015, pp. 2366–2371.

[8] H. Lee, H. Kim, and H. J. Kim, "Planning and control for collision-free cooperative aerial transportation," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 1, pp. 189–201, 2018.

[9] H. Yang and D. Lee, "Hierarchical cooperative control framework of multiple quadrotor-manipulator systems," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4656–4662.

[10] S. Kim, S. Choi, and H. J. Kim, "Aerial manipulation using a quadrotor with a two dof robotic arm," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 4990–4995.

[11] G. Heredia, A. Jimenez-Cano, I. Sanchez, D. Llorente, V. Vega, J. Braga, J. Acosta, and A. Ollero, "Control of a multirotor outdoor aerial manipulator," in *Intelligent Robots and Systems (IROS), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 3417–3422.

[12] V. Lippiello, J. Cacace, A. Santamaria-Navarro, J. Andrade-Cetto, M. A. Trujillo, Y. R. Esteves, and A. Viguria, "Hybrid visual servoing with hierarchical task composition for aerial manipulation," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 259–266, 2016.

[13] S. Kim, H. Seo, J. Shin, and H. J. Kim, "Cooperative aerial manipulation using multirotors with multi-dof robotic arms," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 2, pp. 702–713, 2018.

[14] S. Kim, S. Choi, H. Kim, J. Shin, H. Shim, and H. J. Kim, "Robust control of an equipment-added multirotor using disturbance observer," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 4, pp. 1524–1531, 2017.

[15] H. Seo, S. Kim, and H. J. Kim, "Locally optimal trajectory planning for aerial manipulation in constrained environments," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 1719–1724.

[16] G. Tartaglione, E. D'Amato, M. Ariola, P. S. Rossi, and T. A. Johansen, "Model predictive control for a multi-body slung-load system," *Robotics and Autonomous Systems*, vol. 92, pp. 1–11, 2017.

[17] G. Garimella and M. Kobilarov, "Towards model-predictive control for aerial pick-and-place," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4692–4697.

[18] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[19] O. Brock and L. E. Kavraki, "Decomposition-based motion planning: A framework for real-time motion planning in high-dimensional configuration spaces," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, vol. 2. IEEE, 2001, pp. 1469–1474.

[20] H. Najjaran and A. Goldenberg, "Real-time motion planning of an autonomous mobile manipulator using a fuzzy adaptive kalman filter," *Robotics and Autonomous Systems*, vol. 55, no. 2, pp. 96–106, 2007.

[21] H. Kim, H. Seo, J. Kim, and H. J. Kim, "Sampling-based motion planning for aerial pick-and-place," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 7402–7408.

[22] S. Calinon, F. D'halluin, D. G. Caldwell, and A. G. Billard, "Handling of multiple constraints and motion alternatives in a robot programming by demonstration framework," in *2009 9th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2009, pp. 582–588.

[23] D. Kulić, C. Ott, D. Lee, J. Ishikawa, and Y. Nakamura, "Incremental learning of full body motion primitives and their sequencing through human motion observation," *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 330–345, 2012.

[24] M. Saveriano, S.-i. An, and D. Lee, "Incremental kinesthetic teaching of end-effector and null-space motion primitives," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 3570–3575.

[25] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[26] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto, "Robot learning from demonstration by constructing skill trees," *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 360–375, 2012.

[27] C. G. Atkeson and S. Schaal, "Robot learning from demonstration," in *ICML*, vol. 97. Citeseer, 1997, pp. 12–20.

[28] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[29] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 763–768.

[30] B. Huang, M. Ye, Y. Hu, A. Vandini, S.-L. Lee, and G.-Z. Yang, "A multirobot cooperation framework for sewing personalized stent grafts," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 4, pp. 1776–1785, 2017.

[31] T. Matsubara, S.-H. Hyon, and J. Morimoto, "Learning parametric dynamic movement primitives from multiple demonstrations," *Neural Networks*, vol. 24, no. 5, pp. 493–500, 2011.

[32] D. Bruno, S. Calinon, and D. G. Caldwell, "Learning autonomous behaviours for the body of a flexible surgical robot," *Autonomous Robots*, vol. 41, no. 2, pp. 333–347, 2017.

[33] C. E. Reiley, E. Plaku, and G. D. Hager, "Motion generation of robotic surgical tasks: Learning from expert demonstrations," in *2010 Annual international conference of the IEEE engineering in medicine and biology*. IEEE, 2010, pp. 967–970.

[34] H. Kim, H. Seo, S. Choi, C. J. Tomlin, and H. J. Kim, "Incorporating safety into parametric dynamic movement primitives," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2260–2267, 2019.

[35] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 1168–1175.

[36] S. Lee, H. Seo, S. Choi, H. Kim, and H. J. Kim, "Smooth trajectory generation for soft catching a flying object with an aerial vehicle," in *2017 11th Asian Control Conference (ASCC)*. IEEE, 2017, pp. 790–794.

[37] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *Ieee access*, vol. 2, pp. 56–77, 2014.

[38] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., 1998.

[39] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Survey: Robot programming by demonstration," *Handbook of robotics*, vol. 59, no. BOOK_CHAP, 2008.

[40] G. Goretkin, A. Perez, R. Platt, and G. Konidaris, "Optimal sampling-based planning for linear-quadratic kinodynamic systems," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 2429–2436.

[41] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Learning movement primitives," in *Robotics research. the eleventh international symposium*. Springer, 2005, pp. 561–572.

[42] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," Tech. Rep., 2002.

[43] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural computation*, vol. 25, no. 2, pp. 328–373, 2013.

[44] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *2002 IEEE International Conference on Robotics and Automation*. IEEE, 2002, pp. 1398–1403.

[45] D.-H. Park, H. Hoffmann, P. Pastor, and S. Schaal, "Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields," in *Humanoids 2008-8th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2008, pp. 91–98.

[46] Y. Zhou, J. Gao, and T. Asfour, "Learning via-point movement primitives with inter- and extrapolation capabilities," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4301–4308.

[47] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," in *Advances in neural information processing systems*, 2013, pp. 2616–2624.

[48] T. Matsubara, S.-H. Hyon, and J. Morimoto, "Learning stylistic dynamic movement primitives from multiple demonstrations," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 1277–1283.

[49] S. Miller, M. Fritz, T. Darrell, and P. Abbeel, "Parametrized shape models for clothing," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 4861–4868.

[50] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-specific generalization of discrete and periodic dynamic movement primitives," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 800–815, 2010.

[51] A. Pervez and D. Lee, "Learning task-parameterized dynamic movement primitives using mixture of gmms," *Intelligent Service Robotics*, vol. 11, no. 1, pp. 61–78, 2018.

[52] M. Brand and A. Hertzmann, "Style machines," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 2000, pp. 183–192.

[53] H. Lee, H. Kim, and H. J. Kim, "Planning and control for collision-free cooperative aerial transportation," *IEEE Transactions on Automation Science and Engineering*, accepted for publication.

[54] F. Abi-Farraj, T. Osa, N. P. J. Peters, G. Neumann, and P. R. Giordano, "A learning-based shared control architecture for interactive task execution," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 329–335.

[55] G. Maeda, M. Ewerton, T. Osa, B. Busch, and J. Peters, "Active incremental learning of robot movement primitives," in *Conference on Robot Learning (CORL)*, 2017.

[56] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," in *Advances in neural information processing systems*, 2003, pp. 1547–1554.

[57] R. M. Murray, Z. Li, S. S. Sastry, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[58] I. I. San Juan, C. Sloth, A. Kramberger, H. G. Petersen, E. H. Østergård, and T. R. Savarimuthu, "Towards reversible dynamic movement primitives," in *Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019.

[59] C. E. Rasmussen, "Gaussian processes for machine learning," 2006.

[60] S. Krishnan, A. Garg, S. Patil, C. Lea, G. Hager, P. Abbeel, and K. Goldberg, "Transition state clustering: Unsupervised surgical trajectory segmentation for robot learning," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1595–1618, 2017.

[61] J. Barbič, A. Safonova, J.-Y. Pan, C. Faloutsos, J. K. Hodgins, and N. S. Pollard, "Segmenting motion capture data into distinct behaviors," in *Proceedings of Graphics Interface 2004*. Citeseer, 2004, pp. 185–194.

국 문 초 록

시연 학습 기법(Learning from demonstrations, LfD)은 로봇이 특정 동작을 수행할 수 있도록 하는 유망한 동작 생성 기법이다. 로봇 조작기가 인간 사회에서 다양한 업무를 대체해 감에 따라, 다양한 임무를 수행하는 로봇의 동작을 생성하기 위해 LfD 알고리즘들은 널리 연구되고, 사용되고 있다.

본 논문은 LfD 기법 중 모션 프리머티브 기반의 동작 재생성 알고리즘인 Parametric dynamic movement primitives(PDMP)에 기초한 알고리즘을 제시하며, 이를 통해 다양한 임무를 수행하는 모바일 조작기의 궤적을 생성한다. 기존의 동작 재생성 알고리즘과 달리, 이 연구는 제공된 시연에서 표현된 동작을 단순히 재생성하는 것에 그치지 않고, 새로운 환경에 맞게 일반화 하는 과정을 포함한다. 이 논문에서 제시하는 일반화 과정은 PDMPs의 내부 파라미터 값인 스타일 파라미터와 환경 변수 사이의 비선형 관계를 가우스 회귀 기법(Gaussian process regression, GPR)을 이용하여 수식적으로 표현한다. 제안된 기법은 또한 최적 시연를 학습하는 방식을 통해 강력한 최적 실시간 경로 계획 기법으로도 응용될 수 있다.

본 논문에서는 또한 로봇의 구동 안전성도 고려한다. 기존 연구들에서 다루어진 시연 관리 기술이 로봇의 구동 효율성을 개선하는 방향으로 제시된 것과 달리, 이 연구는 강한 구속조건으로 로봇의 구동 안전성을 확보하는 시연 관리 기술을 통해 안정성을 고려하는 새로운 방식을 제시한다. 제안된 방식은 스타일 파라미터 값 상에서 안전성 기준을 계산하며, 이 안전 기준을 통해 시연을 제거하는 일련의 작업을 수행한다. 또한, 제거된 시위를 안전 기준에 따라 입증된 시위로 대체하여 일반화 성능을 저하시키지 않도록 시위를 관리한다. 이를 통해 다수의 시연 각각 개별 동작 안전성 뿐 아니라 온라인 동작의 안전성까지 고려할 수 있으며, 실시간 로봇 조작기 운용시 안전성이 확보될 수 있다. 제안된 안정성을 고려한 시연 관리 기술은 또한 환경의 정적 설정이 변경되어 모든 시연을 교체해야 할 수 있는 상황에서 사용할 수 있는 시연들을 판별하고, 효율적으로 재사용하는 데 응용할 수 있다.

또한 본 논문은 복잡한 임무에서 적용될 수 있는 PDMPs의 확장 기법인 seg-PDMPs를

제시한다. 이 접근방식은 복잡한 임무가 일반적으로 복수개의 간단한 하위 작업으로 구성된다고 가정한다. 기존 PDMPs와 달리 seg-PDMPs는 전체 궤적을 하위 작업을 나타내는 여러 개의 단위 동작으로 분할하고, 각 단위동작에 대해 여러개의 PDMPs를 구성한다. 각 단위 동작 별로 생성된 PDMPs는 통합된 프레임워크내에서 단계 결정 프로세스를 통해 자동적으로 호출된다. 각 단계 별로 단위 동작을 수행하기 위한 시간 및 하위 목표점은 가우스 공정 회귀(GPR)를 이용한 환경변수와의의 관계식을 통해 얻는다. 결과적으로, 이 연구는 전체적으로 요구되는 시연의 수를 효과적으로 줄일 뿐 아니라, 각 단위동작의 표현 성능을 개선한다.

제안된 알고리즘은 협동 모바일 로봇 조작기 실험을 통하여 검증된다. 세 가지의 시나리오가 본 논문에서 다루어지며, 항공 운송과 관련된 첫 두 가지 시나리오는 PDMPs 기법이 로봇 조작기에서 빠른 적응성, 임무 효율성과 안전성 모두 만족하는 것을 입증한다. 마지막 시나리오는 지상 차량을 이용한 두 개의 로봇 조작기에 대한 실험으로 복잡한 임무 수행을 하기 위해 확장된 기법인 seg-PDMPs가 효과적으로 변화하는 환경에서 일반화된 동작을 생성함을 검증한다.

주요어: 동작 재생성 알고리즘, 모바일 매니퓰레이터, 동작 생성 기법, 시연 학습 기법

학 번: 2014-21897

공학박사학위논문

Learning and Generalizing Complex Robot Motion Skills with Movement Primitives

모션 프리머티브를 이용한 복잡한 로봇 임무 학습 및 일반화 기법

2020년 8월

서울대학교 대학원
기계항공공학부
김 효 인

Learning and Generalizing Complex Robot Motion Skills with Movement Primitives

A Dissertation

by

HYOIN KIM

Presented to the Faculty of the Graduate School of

Seoul National University

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

Department of Mechanical and Aerospace Engineering

Seoul National University

Supervisor : Professor H. Jin Kim

AUGUST 2020

to my

FAMILY

with love

Abstract

Learning and Generalizing Complex Robot Motion Skills with Movement Primitives

Hyoin Kim
Department of Mechanical and Aerospace Engineering
The Graduate School
Seoul National University

Learning from demonstrations (LfD) is a promising approach that enables robots to perform a specific movement. As robotic manipulations are substituting a variety of tasks, LfD algorithms are widely used and studied for specifying the robot configurations for the various types of movements. This dissertation presents an approach based on parametric dynamic movement primitives (PDMP) as a motion representation algorithm which is one of relevant LfD techniques. Unlike existing motion representation algorithms, this work not only represents a prescribed motion but also computes the new behavior through a generalization of multiple demonstrations in the actual environment. The generalization process uses Gaussian process regression (GPR) by representing the nonlinear relationship between the PDMP parameters that determine motion and the corresponding environmental variables. The proposed algorithm shows that it serves as a powerful optimal and real-time motion planner among the existing planning algorithms when optimal demonstrations are provided as dataset.

In this dissertation, the safety of motion is also considered. Here, safety refers to keeping the system away from certain configurations that are unsafe. The safety criterion of the PDMP internal parameters are computed to check the safety. This safety criterion reflects the new behavior computed through the generalization process, as well as the individual motion safety of the demonstration set. The demonstrations causing unsafe movement are identified and removed. Also, the demolished demonstrations are replaced by proven

demonstrations upon this criterion.

This work also presents an extension approach reducing the number of required demonstrations for the PDMP framework. This approach is effective where a single mission consists of multiple sub-tasks and requires numerous demonstrations in generalizing them. The whole trajectories in provided demonstrations are segmented into multiple sub-tasks representing unit motions. Then, multiple PDMPs are formed independently for correlated-segments. The phase-decision process determines which sub-task and associated PDMPs to be executed online, allowing multiple PDMPs to be autonomously configured within an integrated framework. GPR formulations are applied to obtain execution time and regional goal configuration for each sub-task.

Finally, the proposed approach and its extension are validated with the actual experiments of mobile manipulators. The first two scenarios regarding cooperative aerial transportation demonstrate the excellence of the proposed technique in terms of quick computation, generation of efficient movement, and safety assurance. The last scenario deals with two mobile manipulations using ground vehicles and shows the effectiveness of the proposed extension in executing complex missions.

Keywords: Motion representation algorithm, Mobile manipulator, Manipulation planning, Learning from demonstration.

Student Number: 2014-21897

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Motivations

Robotic manipulations are substituting a wider variety of tasks. Various types of robot manipulators have been developed including the manipulators with mobile or ground vehicles. The manipulation parts also have been improved from single gripper [3], joints [3] or cable [4–6] to versatile robotic arm with multiple links [7–14]. The application of robot manipulators is no longer limited to fixed and structured environments in the past, according to the these development of robot manipulators. Now, the missions expand into unstructured environments where unknown situations arise.

It leaves a variety of challenges in terms of robot automation as robot manipulators increasingly applied to environments where various movement freedoms are exercised, not just do repetitive tasks. In addition to the essential requirement for the robot manipulator to complete the assigned task, the various properties of automation algorithms may be required. For example, more efficient movement generation and faster computational speed have been frequently addressed in the motion planning problem, such as optimal planning

[8, 15–18] or real-time planning [19, 20]. Moreover, there are a variety of issues faced by robot manipulators, such as via-points [15, 21, 22] or increased task complexity [23–26]. Assuming that robot operation is successfully controlled according to desired commands with excellent control technologies [7, 13, 14] that have been developed in recent years, the above problems can be adequately addressed in the stage of specifying the configurations of the robot. The problem of allocating robot configurations can be solved through existing motion planning techniques if it simply gives a starting point and a finishing point as a mission. However, the missions for robot manipulators often include a specific style of movement like stretching or twisting, which cannot be represented by starting or finishing points. That is, it may not complete the mission, which is the most basic requirement.

Recently, researchers have utilized learning from demonstration (LfD) [27–32] to specify configurations for robot manipulation. In the process of learning and generalizing motion from demonstrations, LfD can express tasks that require a specific style of motion as they are provided in the demonstrations while conventional planners cannot do. For example, via LfD stitching [30], knot tying [33] or swaying movements [31] can be represented, which are difficult to express formally and has a limitation in applying to other conventional planners. In particular, motion presentation algorithms among LfD techniques literally regenerate the movements provided by the demonstration, so if the demonstration has completed mission, the computed behavior will succeed. However, this application is limited to the environment where environmental settings do not change.

This dissertation is interested in resolving the following problems to assign the configurations of robot manipulators to carry out missions, including expressing certain style movements: (i) optimal movement, (ii) real-time adaptation, and (iii) motion safety. Moreover, (iv) complex assignments should be addressed through the proposed approach for practical applications in human life. Here, the environment can change continuously.

1.2 Literature Survey

This section provides the survey results of scholarly articles, books, and other materials relevant to this study. Most of the literature survey in this section refer to the papers [2,34].

1.2.1 Conventional Motion Planning in Mobile Manipulations

In terms of specifying the robot configurations, conventional planning approaches are investigated. Focusing on real-time and optimal motion generation, the relevant algorithms are offered in the following.

Optimization-based motion planning

Optimization techniques offer the explicit guarantee of the local optimality of the planned trajectory. Several constraints can be considered in a unified way so that the dynamic and kinematic feasibilities of the planned trajectory are ensured for mobile manipulation system.

As a straightfoward optimization-based motion planning, nonlinear programming (NLP) can be utilized. One easy way to formulate NLP for the mobile manipulation system is to assume a continuous-time trajectory of a specific representation such as a polynomial, B-spline, or Bézier curve using flat outputs. By doing so, the trajectory optimization problem is converted into a static nonlinear optimization problem on coefficients of the corresponding representation [15]. From this conversion, the search space of the optimization is confined to a span of bases defined in the representation space.

However, a simple NLP is not appropriate for generating suitable trajectories when the dynamics and constraints of the target system get complicated. The main reason is that a solution which satisfies complex constraints may not exist in the restricted search space. Especially for cooperative mobile manipulation, its complex dynamics and constraints further reduce the feasible search space for planning.

Another approach for trajectory optimization is model predictive control (MPC) [16],

[17]. MPC seeks the optimal trajectory from the current state to the target state over a finite time horizon, and the first optimal input is applied to the system. Although its computation time, which increases significantly as the dimension of the system increases, can be adjusted by modulating the length of the time horizon, the convergence domain becomes drastically contracted if the system is highly nonlinear or has many constraints. Also, the performance of MPC is highly dependent on the initial guess because underlying optimization algorithms used in every step usually give the local optimum. On the other hands, iterative linear quadratic regulators (iLQR) [35] can be implemented to generate feasible and optimal trajectories for robot manipulator [36]. Still, the representation of constraints is difficult to formalize the problem. In short, optimization-based motion planners have a possibility that they will not compute the global optimal solution, and thus they are sensitive to the initial guess. Besides, they cannot give a solution in a short time.

Sampling-based motion planning

Sampling-based techniques (SBP) [37] use random computations instead of solving difficult problems in motion planning. They utilize random sampling techniques and build a series of waypoints, providing a fast solution even for high-dimensional and constrained problems. Also, they need only the criterion to determine appropriate waypoints, making the algorithm is straightforward and easy to apply. Recently, with successful implementations of the optimizing process, robotic motion planning accelerates the utilization of SBP even to secure asymptotic optimality without converging to local minima.

Rapidly exploring random trees (RRTs) [38–40] is a widely used SBP algorithm thanks to the ability that it does not depend on the explicit representation of obstacles [1, 8]. RRT star (RRT*) [18], which is an extended form of RRT, has simple optimizing processes which connect or reconnect the waypoints as they improve the cost in a selected local area. By repeating those processes in every iteration, the algorithm guarantees asymptotic optimality.

For the cooperative mobile task, the planning process should handle the multiple kine-

Approach	Methods	Computing speed	Motion completion	etc
Optimization-based techniques	iLQR [35], NLP [15], MPC [16, 17]	Light or heavy	Hard formulation	Safety or complex task can be configured with constraints but cause Heavy computing load
Sampling-based optimal planners	RRT* [18], PRM* [18]	Heavy	Limited formulation (point-wise)	
Motion representation algorithms (optimal motion primitives)	DMPs [41], PDMPs [31]	Light	Easy formulation	Additional formulation is necessary to ensure safety

Table 1.1: Summary of optimal motion generation algorithms.

matic and dynamic constraints which are inherent from the interaction between agents. Using RRT* or other optimal planners, the solution is given within a reasonable time without worrying about local minima or difficult problem formulation. Moreover, the dynamics of the mobile manipulator can be considered in the local planning of RRT* in that the trajectory is computed using the dynamics of mobile manipulators and their properties such as actuation bounds. Also, RRT* consequently updates the connection of sampled nodes into a better one. During this process, RRT* asymptotically finds the optimal pose of each mobile manipulator along the trajectory, which allocates proper payload distribution.

RRT* is known to often demand much longer computational time to achieve a close-to-optimal solution than the time to obtain the initial solution. The recent research regarding both optimization and sampling based motion planning has improved the rate of convergence by utilizing smart sampling or developing the simple formulation of the problem. Still, neither optimization-based nor sampling-based planning alone is satisfactory for cooperative mobile tasks that require a fast solution to react appropriately to sudden risky situations.

1.2.2 Motion Representation Algorithms

Motion representation algorithms have been utilized to reproduce a given trajectory robustly. Among them, dynamic movement primitives (DMPs) [41–44] are powerful motion representation techniques in that they demand a low computational load during task execution as model-free approaches. By generating the trajectory from a prescribed movement,

DMPs can serve as a quick motion representation algorithm. In DMPs, the forcing term is computed to follow the demonstration in given DMPs equations. By using the supervised learning to the forcing term, the robotic system can quickly modify the trajectory so that the original demonstration can be robustly followed from perturbed configurations.

Some researchers utilize DMPs as tools for a motion generator within obstacles [29, 31, 45, 46]. In [29], [45], the new equations of DMPs are suggested including the term of repulsive potential fields, which guide the system to the opposite side from obstacles. However, the introduction of potential fields may cause inefficiency or degrade the optimality of the original movement primitives, and further, the possibility of local minima in potential fields still remains. On the other hands, via-points modulations can be used to avoid obstacles if via-points to avoid obstacles are specified. In [46], DMPs are combined with probabilistic movement primitives [47] to directly adapt to intermediate via-points such as configurations of collision-free state. This formulation has a limitation that the configuration of robot must be specified in advance for obstacle avoidance. Another approach for obstacle avoidance is shown in [31, 48], using parametric-DMPs (PDMPs) which allow the movement generalization by extracting features from multiple demonstrations. In [31], PDMPs provide the parametrized avoidance movements to new heights of obstacles. Similar models of PDMPs appear as task-parametrized [49–51] or stylistic form [52]. For those frameworks, finding parameters or styles is an important issue in order to produce suitable motion in a new circumstance.

In [53], successful combination of RRT* and DMPs are addressed. RRT* and DMPs are dealt in seperated manner where RRT* generates the trajectory avoiding known obstacles as an off-line motion planner while the DMPs technique with potential fields is applied to avoid unknown obstacles. However, the limitation exists as the optimality is lost when the environmental set-ups are significantly changed. Still, the successful combination of RRT* and DMPs can address the trade-off issue between motion optimality and quick reaction(or adaptation). Moreover, the robustness of DMPs is advantageous in the presence of perturbations such as internal forces between cooperating agents or their out-of-sync

movements.

1.2.3 Safety-guaranteed Motion Representation Algorithms

The performance of the resulting motion in motion representation algorithms depends on how appropriately demonstrations are provided for a given scenario. In other words, when the extracted motion primitives are not enough to generalize the desired motion in a given scenario, the computed motion may end up with the poor results. Thus, the skill of collecting suitable demonstrations for generalizing target scenario is important for motion representation algorithms. Moreover, the technique to identify safety-guaranteed demonstrations which provide safe primitives can be applied to the problem of deciding whether to reuse the demonstrations or not.

There are some related works regarding reasoning when the robot needs updated demonstrations [24, 54, 55]. In [55], the uncertainty of the trajectory is measured using the Gaussian process. When the uncertainty is higher than a certain level, a new demonstration is added. In [54], the update criterion is determined based on the information gain from the new data. Here, the information gain is the value that indicates the sufficiency of the demonstrations. [24] performs incremental learning for the success of task execution. During the task, each grasping motion and movement of end-effector is corrected by physical interaction with the object and by human, respectively. Above works focus on enhancing the performance by incrementally providing the demonstrations. For safety-critical situations, these criteria are not enough to determine which of the given demonstrations are bad and should be eliminated from the data-set.

1.3 Research Objectives and Contributions

The objective of this dissertation is to develop an integrated framework based on parametric dynamic movement primitives (PDMPs) so as to compute configurations of robot manipulators that performs the assigned mission. In particular, this work is interested in resolving

Methods	Initial/terminal offset generalization	Style generalization	Parameter selection process	Demonstration update	Safety guarantee
[29, 56]	O	X	X	X	X
[31, 49, 50]	O	O	X	X	X
[55]	O	X	X	O(Performance)	X
[24, 54]	O	X	X	O	O
The proposed	O	O	O	O	O

Table 1.2: The objective conditions in representative motion representation algorithms and the proposed algorithms.

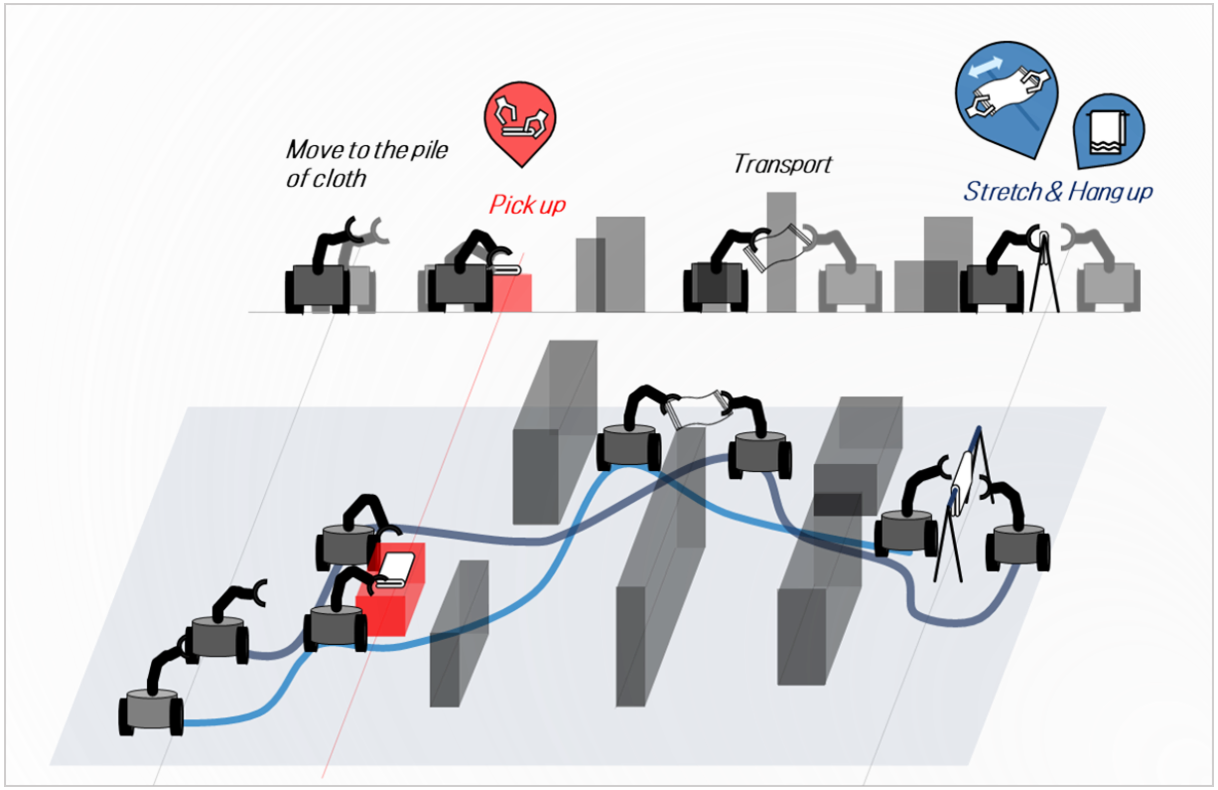


Figure 1.1: Cooperative mobile manipulation to hang a piece of cloth over the hanger (hang-dry mission). The hang-dry mission consists of sequential sub-tasks such as moving, grasping, and stretching. Each mobile manipulator starts to move to the cloth, first. When they reach the cloth, they grasp it cooperatively. While they approach a hanger, the robots should avoid multiple obstacles. Near the hanger, one robot moves to the opposite side of the hanger to easily stretch and hang up the cloth over the hanger.

the aforementioned problems in section 1.1. This dissertation concentrates on three parts in utilizing motion representation algorithm: (i) motion generalization, (ii) safety guarantee for successful operation, and (iii) applications to complex missions.

1.3.1 Motion Generalization in Motion Representation Algorithm

Objectives

This work first develops a base motion representation algorithm including the motion generalization process. The various styles of motion can be appropriately computed by learning multiple demonstrations. From the generalization process, the proper parameters in PDMPs and resulting motion are calculated corresponding to the new environment, even for the environment is never provided as a demonstration.

Contributions

- This dissertation presents a GPR formulation to express the implicit relationship between the parameters in PDMPs and environment. This formulation give reliable values for DDMP parameters to successfully execute the mission. The proposed framework serve as a powerful optimal planner. The performance is validated by comparing simulation results in terms of computational time and performance index with sampling-based planners.

1.3.2 Motion Generalization with Safety Guarantee

Objective

Second, this work presents the process for managing and improving the demonstration set to ensure the safety of motion. This process calculates the safety criterion in the PDMPs parameter value, and use it to identify demonstrations causing unsafe motion. By eliminating the unsafe demonstrations and adding safe ones, computed motion always ensures safety.

Contributions

- This work deduces the safety criterion for the parameter in the PDMP framework via optimization. Here, under the PDMPs dynamics, the parametric optimization problem is formulated.
- From the safety criterion, this work also suggest the process that manage demonstrations autonomously by directly eliminating the unsafe demonstrations. New demonstrations are provided in order to secure the safety of motion.
- Simulation results are given and show that the proposed process is also applicable where a new scenario is given and the previous demonstrations can be reused instead of computing all the demonstrations again.

1.3.3 Motion Generalization for Complex Missions

Objective

Third, this dissertation provides an extended approach that can efficiently reduce the number of required demonstrations for generalizations. In particular, this work is useful to the missions consisting of multiple sub-tasks.

Contributions

- This extension gives a new approach for configuring multiple PDMPs for each sub-task and autonomously selecting them according to the situation. This work effectively reduces the number of required demonstrations by eliminating the need to provide all combinations of sub-tasks in single PDMPs algorithm on the entire mission.

- This work also also proposes two processes that use Gaussian process regression (GPR) to assign generalized execution time and regional goal state in each sub-task, respectively. These processes enable flexible sub-task sequencing across the entire mission, even if the PDMPs of sub-tasks are configured independently.

- Various simulation results validate that the proposed extension is effective not only in reducing the number of required demonstrations but also in improving the generalization performance of each sub-task compared to applying a single PDMPs algorithm.

- This extension also show that the proposed framework can be extended to new scenarios by efficiently reusing sub-tasks, obviating the need to provide all demonstrations again.

1.4 Thesis Organization

The remainder of the thesis is organized as follows. Chapter 2 describes the necessary materials for this dissertation including DMPs and experimental setups for mobile manipulation systems. Chapters 3 and 4 describe the the proposed algorithms for generalizing and managing safety-guaranteed demonstrations. Chapter 5 presents the applicable extension of the proposed approach when performing complex missions with several sub-tasks. Chapter 6 verifies the algorithms by presenting the results of the actual experiments. Chapter 7 summarizes the papers.

2

Background

This section provides the background for this research, including DMPs, the mobile manipulation systems, and experimental setups.

2.1 DMPs

DMPs, proposed in [41,42], have received attention as model-free approaches which demand low computational load during task execution. DMPs can represent complex movements with incorporating sensory feedback in real-time. Detail about DMPs are described in [41, 42]. Here, basic information for DMPs are addressed only based on the formulation used in [29].

DMPs are defined based on the following attractor dynamics as,

$$\begin{aligned} \dot{\mathbf{v}}^d &= K_p(\mathbf{g}^d - \mathbf{q}^d) - K_D\mathbf{v}^d + K_p\mathbf{f}(\chi; \mathbf{w}) \\ \dot{\mathbf{q}}^d &= \mathbf{v}^d \end{aligned} \tag{2.1}$$

where $\mathbf{q}^d$ and $\mathbf{v}^d$ are the vectors of state variables and their time derivatives. $\mathbf{g}^d$ is the goal states for the system. Each K_p and K_d is the spring and damping coefficient in this dynamics. Superscript d is used to denote the variables are the state by DMPs. $\mathbf{f}(\chi; \mathbf{w})$ is a forcing term that leads the system to goal states $\mathbf{g}^d$. This forcing term describe the nonlinear motion as

$$\mathbf{f}(\chi; \mathbf{w}) = \frac{\sum_{i=1}^{N_w} \mathbf{w}_i \Phi_i^d(\chi)}{\sum_{i=1}^{N_w} \Phi_i^d(\chi)} \chi. \tag{2.2}$$

$\mathbf{w}_i$ is the weight of each basis function. The exponential basis function $\Phi_i^d(\chi)$ for $i = 1, ..., N_w$ can be defined to

$$\Phi_i^d(\chi) = \exp\left(-\frac{(\chi - d_i)^2}{2\sigma_i^2}\right), \tag{2.3}$$

where d_i and σ_i denote constants which determine the width and center of the basis function, respectively. $\mathbf{f}(\chi; \mathbf{w})$ does not depend on time as shown in (2.2). Instead, it depends on a external variable χ, which varies from 1 to 0 during a movement. The dynamics of χ is defined as

$$\dot{\chi} = -K_\chi \chi, \tag{2.4}$$

where K_χ is a constant.

2.2 Mobile Manipulation Systems

This work considers the mobile manipulators with two types of mobile platforms, ground and aerial vehicle. The manipulation parts of those manipulators are specified to robotic arm with multiple links, to operate dexterous manipulations. The tip of robotic arm is

equipped with a gripper that rigidly grasp the object. System variables are also configured for cooperative works.

2.2.1 Single Mobile Manipulation

The configuration space of single robot manipulator can be represented as $[\mathbf{p}_b^\top, \Phi^\top, \eta^\top]^\top$, where $\mathbf{p}_b = [x_b, y_b, z_b]^\top$ denotes the body position and $\Phi = [\phi, \theta, \psi]^\top$ represents the Euler angle. The manipulation part is specified to robotic arm so the variables $\eta = [\eta_{i,1}, ..., \eta_{n_m,2}]^\top$ indicates the set of arm joint angles for n_m number of links.

In the case of robot manipulator with ground vehicle, $\mathbf{p}_b$ and Φ is simplified to $\mathbf{p}_b = [x_b, y_b]^\top$ and $\Phi = \psi$. On the other hands, multi-rotor based aerial manipulator uses full states of above configuration space. The configuration space of the both ground-vehicle and aerial manipulators are shown in Fig. 2.1.

For those manipulation systems, the actual operation is performed through the position of the end effectors. The positions of the end effector in both types of robot manipulators are calculated as

$$\bar{\mathbf{p}}_e = g_{n_m}(\eta_{n_m})_1(\eta_1)g_0(\Psi)\bar{\mathbf{p}}_b \tag{2.5}$$

where g_i's for $i = 0, 1, ..., n_m$ are the transformation matrices in [57]. $\bar{\mathbf{p}}$ denotes a vector defined as $\bar{\mathbf{p}} = [\mathbf{p}^\top \ 1]^\top$.

2.2.2 Cooperative Mobile Manipulations

Cooperative manipulation systems are identified by addressing the constraints resulted from the motion of each robot manipulators. This section shows two cooperation states from two robot manipulators holding hard and elastic objects. The experiments in this dissertation deal both types of objects are handled in cooperation systems.

Rigid Object

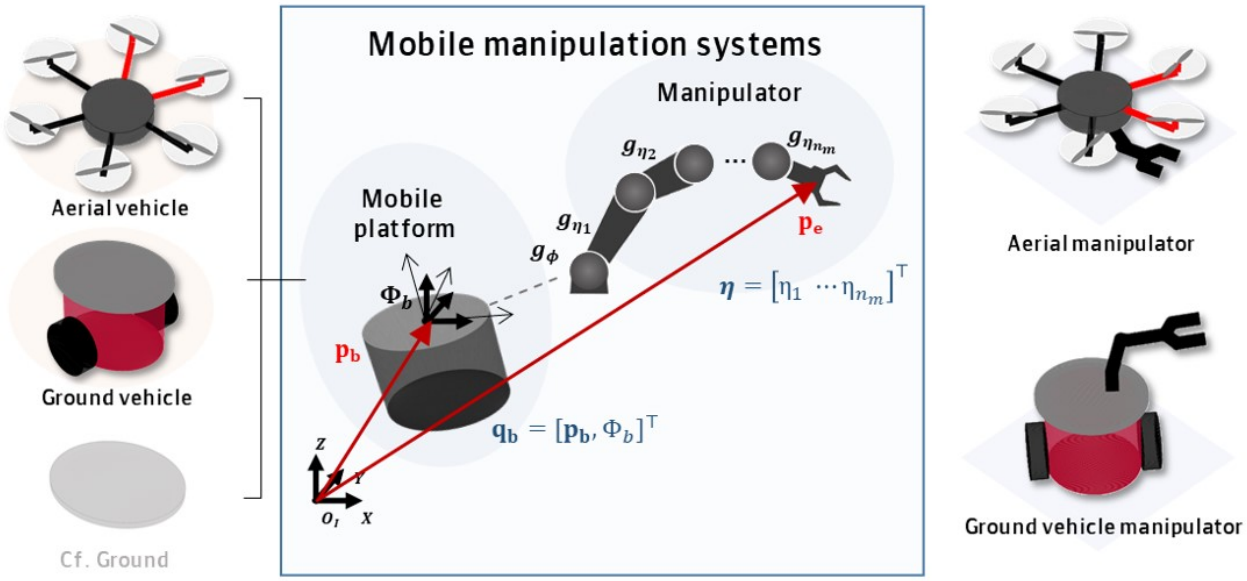


Figure 2.1: Various types of mobile manipulation systems and their coordinates.

When robot manipulators cooperatively transports a common object with rigid body with the rigid grasping condition, those robot manipulators and the object can be considered as one combined system. The cooperative system in cooperation can be described by the collection of state variables of each robot manipulators.

The configuration space of each robot manipulator can be represented as $\mathbf{q}_i = [\mathbf{p}_i^\top, \Phi_i^\top, \eta_i^\top]^\top$ for $i = 1, 2$, where $\mathbf{p}_i = [x_{b,i}, y_{b,i}, z_{b,i}]^\top$ denotes the body position and $\Phi_i = [\phi_i, \theta_i, \psi_i]^\top$ represents the Euler angle of i-th body platform. $\eta_i = [\eta_{i,1}, ..., \eta_{i,n_m}]^\top$ indicates the vector of joint angles of robotic arm as denoted in the right of Fig. 2.2. Subscript i represents the variable of the i-th aerial manipulator. The configuration space of the object is represented as $[\mathbf{p}_o, \Phi_o]^\top$.

In order to express the configuration space of the combined system, kinematic con-

straints from the grasping system are listed as below:

$$\begin{aligned} \mathbf{p}_{c_i} &= g_{o,i}\mathbf{p}_o \\ \mathbf{p}_{e_i} &= g_{i,n_m} \cdots g_{i,1} g_{i,\psi} \mathbf{p}_i \\ \mathbf{p}_{e_i} &= \mathbf{p}_{c_i}, \end{aligned} \tag{2.6}$$

where, $\mathbf{p}_o, \mathbf{p}_i, \mathbf{p}_{e,i}$ and $\mathbf{p}_{c,i}$ indicate the x, y and z positions of the center of object, body of the mobile platform, end-effector and tip of object. $g_{i,1}, \cdots, g_{i,n_m}$ and $g_{i,\Phi}$ denote the transformation matrices, which calculate the positions resulted by joint angles and Euler angle of robot manipulator. $g_{o,i}$ denotes the transformation matrix from the center of the object to the tip. The experiment in section 5.1 regards cooperative aerial transportation with rigid object. The experiments in this dissertation deal both types of objects are handled in cooperation systems. The property of differential flatness in the multi-rotor enables us to represent roll and pitch angles with other state variables and their time derivatives. From this property and (2.6), the configuration space of combined system can be represented as $\mathbf{q} = [x_o, y_o, z_o, \psi_o, \eta_{1,1}, \eta_{1,2}, \eta_{2,1}]^\top$. Here, the dimension of the configuration space has been reduced with observation that the grasping condition allows to know $\eta_{2,2}$ from $\eta_{1,1}, \eta_{1,2}$ and $\eta_{2,1}$.

Elastic Object

When the robot manipulators holding the elastic object, the tips of manipulators are freely move within the limited distance, the maximum length of the grasping parts of object. So during operation, these operation limit is added to check the cooperative stability. The experiment in section 5.2 deals with hang-dry work with elastic object, a cloth.

2.3 Experimental Setup

Two cooperative manipulation systems are employed to validate the proposed approach in this dissertation. This section describes the experimental setups.

2.3.1 Test-beds for Aerial Manipulators

The overall hardware setup of the aerial manipulator is shown in Fig. 2.3. Two identical multi-rotors are employed, and each is equipped with a 2-DoF robotic arm. In aerial vehicle, DJI E800 motors are controlled by 620S electronic speed controllers (ESCs). For onboard computation, Intel NUC7i7BNH running Ubuntu 14.04 and ROS Indigo is used. Also, a Pixhawk autopilot is attached at the center of the fuselage and connected to the onboard computer via USB.

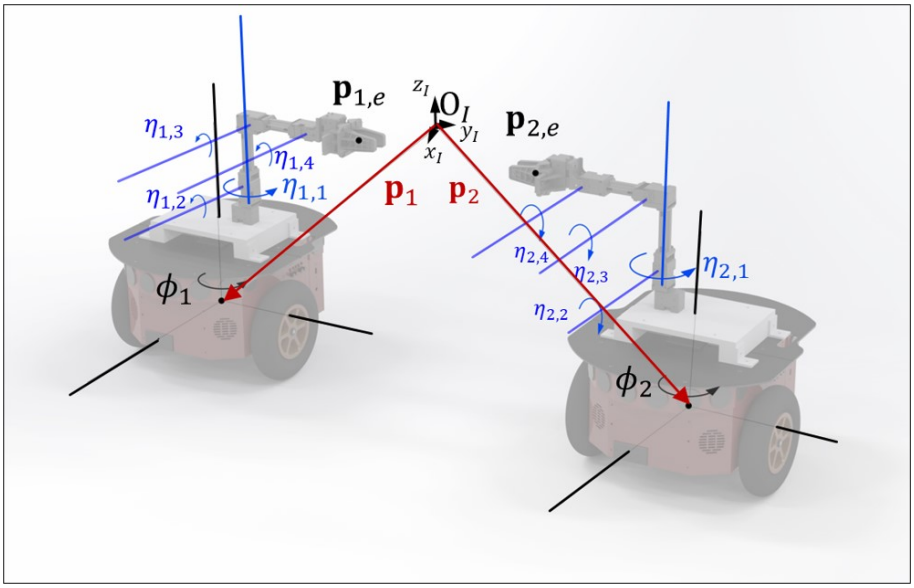
A trajectory tracking controller is a slightly modified version of [8] is used as a high-level controller to generate the desired net thrust and angular velocities. It is aimed to control the position and heading of each aerial manipulator in a decentralized manner. These set-points are sent to the autopilot, and customized the PX4 firmware computes the desired PWM signal of each motor by a low-level control algorithm [14] which tracks the desired angular velocities. For indoor tests, a motion capture system (VICON) is used. The ground control station receives pose measurements at 100 Hz, and sends them to the onboard computer via wireless communication. Lastly, MX-106 and MX-28 servos from ROBOTIS constitute the 2-DoF manipulator and are connected to the onboard computer via USB. An RGB-D or stereo camera can be used for detecting surroundings such as existence or location of the obstacle.

2.3.2 Test-beds for Robot Manipulators with Ground Vehicles

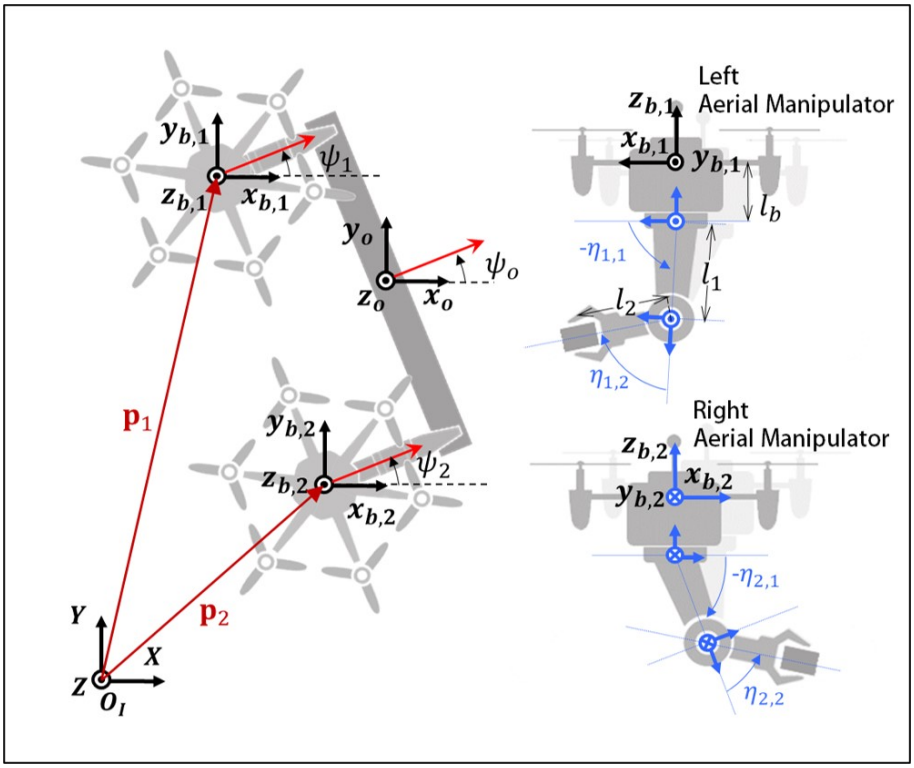
In experimental setups, each robot system consists of a computer, a ground robot, and a manipulator as described in section 2.2. Pioneer 3-DX is employed as a ground robot

platform. The ground robots are equipped with OpenMANIPULAOTR-X from ROBOTIS. Each manipulator is mounted to the middle top of a pioneer with the y-directional offset towards counterpart pioneer. For on-board computation, Intel NUC7i7BNHXG is used to operate Ubuntu 16.04 and ROS Kinect. The overall system is operated on ROS. Ground station is set to Intel Core i5-8256U laptop.

During the task, the ground robots are controlled to follow calculated trajectories. Similar to aerial transportation experiments, present positions are achieved by using the motion capture system, VICON. From the VICON system, current environmental information and the robot states are transferred to VICON computer. Using the proposed framework, the trajectories are calculated from the sensory feedback and are transmitted from ground station to the on-board computer via wireless communication. PI controller is designed to calculate control command from present positions and desired positions. The manipulator is connected to the on-board computer with USB and operated by proportional control based on joint position measurements. The low-level controller is implemented in C++. The control architecture is described in Fig. 2.4.



(a)



(b)

Figure 2.2: Configuration of the coordinates for the combined systems. (a) Cooperative mobile manipulation system with two mobile manipulators consisting of ground-vehicle and a 4-DOF robotic arm. (b) Cooperative aerial manipulation system with two aerial aerial manipulator consisting of a hexacopter and a 2-DOF arm.

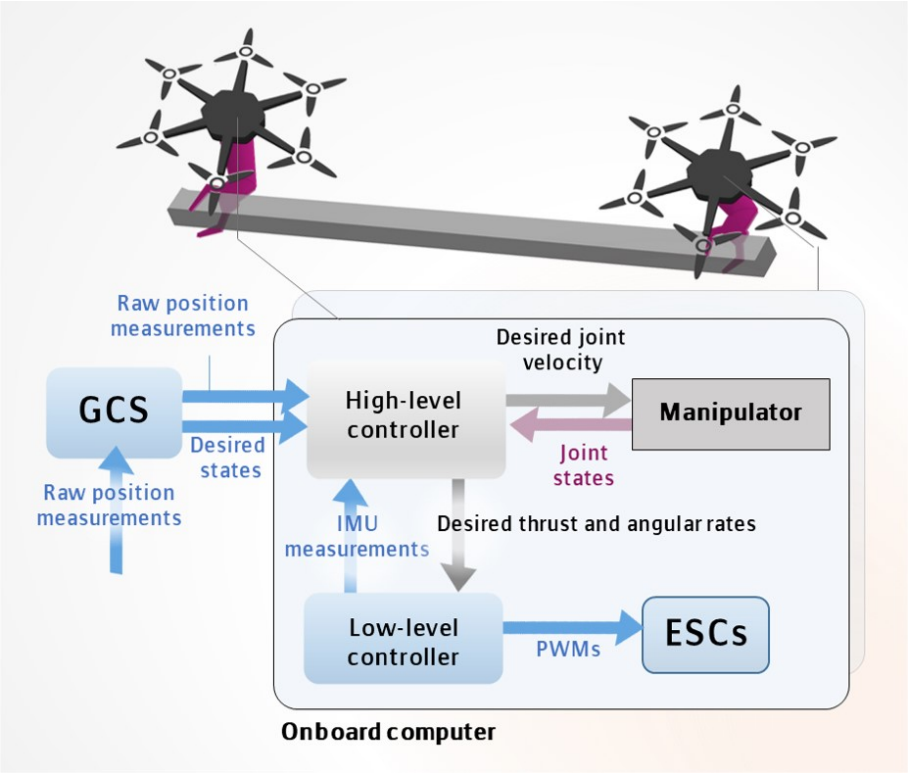


Figure 2.3: Experimental setups for cooperative aerial manipulation system in Fig. 2.2b.

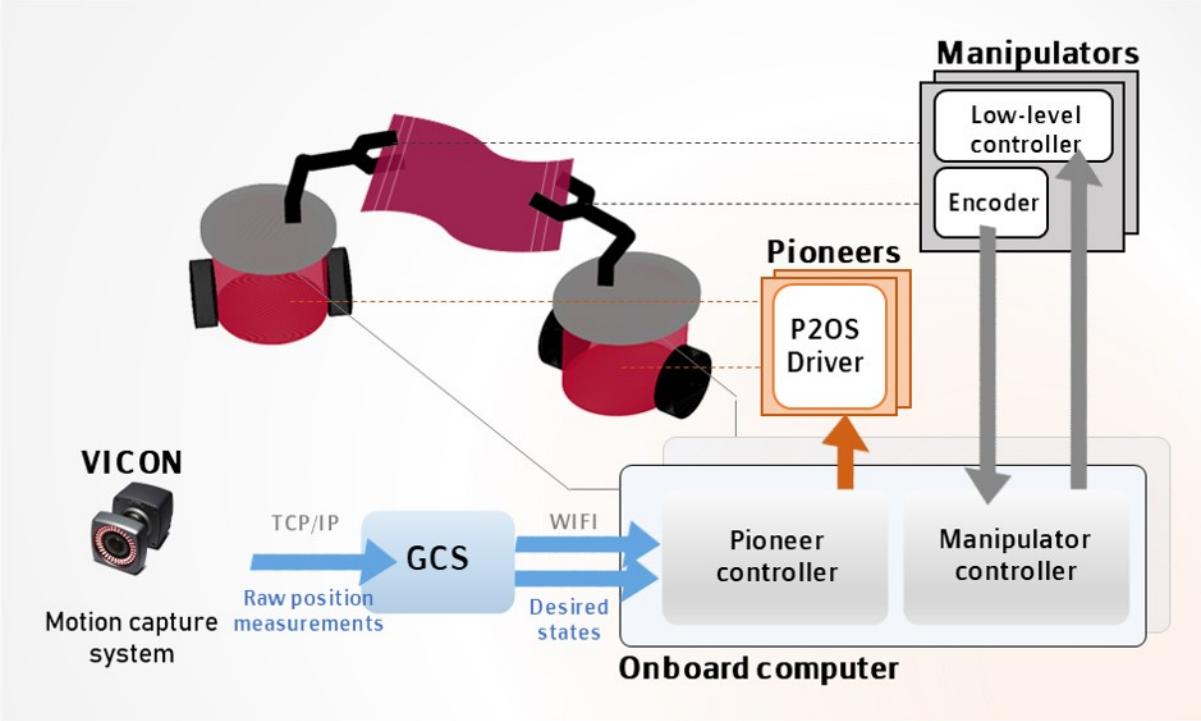


Figure 2.4: Experiment set-up for the hang-dry work in Fig. 2.2a.

3

Motion Generalization in Motion Representation Algorithm

In this section, an overview of the proposed framework is provided. PDMPs in [31] are described, and including the proposed generalization process, the modification of the framework is discussed in detail.

3.1 Parametric Dynamic Movement Primitives

Single demonstration in a specific environment can be reproduced using DMPs, but it is not sufficient for generalization in different environments. Although DMPs allow a certain level of initial or terminal offset [41, 58], their performance is degraded at large offsets. Moreover, DMPs cannot generalize motion styles themselves. In order to generalize the various movement styles to adapt to situations including large offset, DMPs can be modified to use multiple demonstrations that include all the required movements. The parametric-DMPs (PDMPs) [31], the extensions of DMPs, adopt style-variable modeling to DMPs. With the parametric skill, PDMPs have an advantage of generalizing motion from multiple

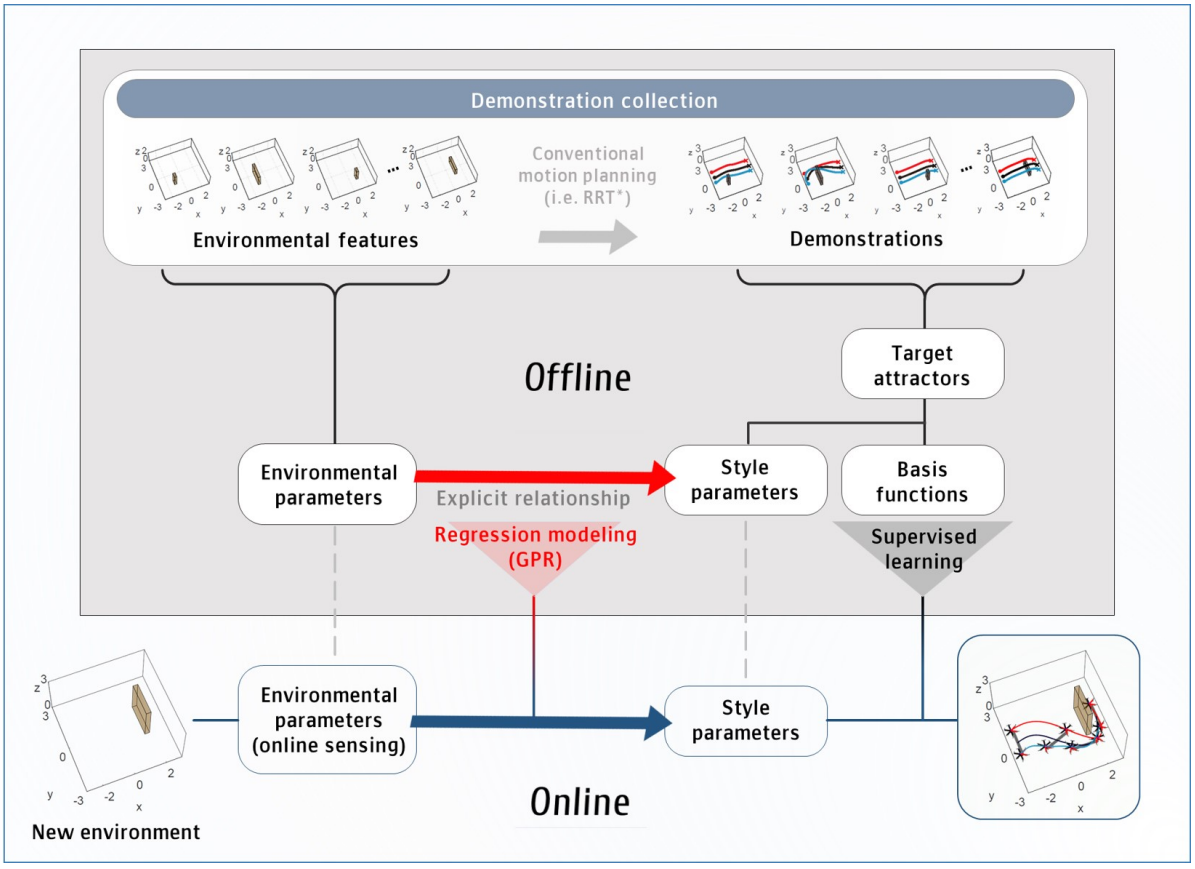


Figure 3.1: An overview of the proposed PDMPs framework with parameter selection process for cooperative aerial transportation.

demonstrations. Through the generalization, PDMPs generate the new movement that has never been learned.

PDMP has a structure that can generate various paths through a parameterized structure, but it is not known which value is suitable for a situation. In this work, an integrated framework is suggested to describe the process of selecting parameters by revealing the relationship between situations and PDMP parameter values. In Fig. 3.1, the overview of the proposed learning-based planning algorithm is shown. Except for the proposed generalization process, the detailed descriptions of PDMPs are presented as follows.

First, the equation of DMPs [29] is represented as

$$\alpha_3\ddot{\mathbf{q}} = \alpha_1(\mathbf{g} - \mathbf{q}) - \alpha_2\alpha_3\dot{\mathbf{q}} - \alpha_1(\mathbf{g} - \mathbf{q}_0)\chi + \alpha_1\mathbf{f}(\chi; \mathbf{w}), \tag{3.1}$$

where $\mathbf{q}$ denotes the state vector of the system, and $\mathbf{g}$ and $\mathbf{q}_0$ the goal and start configuration, respectively. In (3.1), χ and $\mathbf{f}(\chi; \mathbf{w})$ are introduced to make $\mathbf{q}$ asymptotically converge to the unique point $\mathbf{g}$. Here, χ is the phase variable whose dynamics is described as $\dot{\chi} = -K_4\chi$ in [56], which forms a canonical system. K_4 is the constant value. $\chi = 1$ indicates the start of the time evolution and χ close to zero means that the goal $\mathbf{g}$ has essentially been achieved. $\mathbf{f}(\chi; \mathbf{w})$ is an attractor function which are previously called the forcing term. The attractor function $\mathbf{f}$ forces the current state variable to the demonstrated trajectories by learning the weights $\mathbf{w}$. α_1, α_2, and α_3 are the time constants and $\mathbf{w}$ is the weight to be learned.

The difference of PDMPs from DMPs lies in formulating the attractor function $\mathbf{f}(\chi; \mathbf{w})$. In DMPs, the attractor function is formulated by supervised learning of the target attractor function, where the target attractor functions are computed by substituting $\mathbf{q}$ with demonstrated states $\mathbf{q}_d$ in (3.1). On the other hand, in PDMPs, to consider the multiple demonstrations in a unified framework, the style parameters and basis functions are introduced to the attractor function as

$$f_n(\chi, \mathbf{s}_n; \mathbf{w}_n) = \mathbf{s}_n^\top \mathbf{b}_n(\chi; \mathbf{w}_n). \tag{3.2}$$

The subscript $n = 1, \cdots, N$ indicates that the variable is concerned with the n-th state variable, where N is the number of state variables. Here, $\mathbf{s}_n \in \mathbb{R}^{L_n \times 1}$ denotes the *style parameter* corresponding to the n-th state variable and $\mathbf{b}_n(\chi; \mathbf{w}_n) = [b_n^1(\chi; \mathbf{w}_{n,1}), \cdots, b_n^{L_n}(\chi; \mathbf{w}_{n,l})]^\top \in \mathbb{R}^{L_n \times 1}$ represents the set of basis functions. The basis functions can be viewed as characteristic modes of movement that composes the movements to be generalized obtained from the whole demonstrations. L_n is the dimension of $\mathbf{s}_n$ to be explained below. In PDMPs, supervised learning is applied only to $\mathbf{b}_n(\chi; \mathbf{w}_n)$.

The target basis function and the corresponding style parameter are obtained as follows. Let $\mathbf{f}^m_{n,target} \in \mathbb{R}^{\bar{L}\times 1}$ for $m = 1, \cdots, M$ denote the series of target attractor functions along a trajectory, where $\bar{L}$ is the total number of time steps along the discrete-time trajectory and M is the number of demonstrations. The target matrix for the M demonstrations is formulated as $\mathbf{F}_n = [\mathbf{f}^1_{n,target}, \mathbf{f}^2_{n,target}, \cdots, \mathbf{f}^M_{n,target}]^\top \in \mathbb{R}^{M\times\bar{L}}$. The set of style parameters $\mathbf{S}_n$ and target basis functions $\mathbf{B}_n$ are obtained by taking the singular value decomposition (SVD) to the target matrix as,

$$\mathbf{F}_n = \mathbf{U}_n \Sigma_n \mathbf{V}_n^\top = \mathbf{S}_n^\top \mathbf{B}_n. \tag{3.3}$$

Here, $\mathbf{S}_n^\top = [\mathbf{s}_n^1, \mathbf{s}_n^2, \cdots, \mathbf{s}_n^M]^\top \in \mathbb{R}^{M\times L_n}$ is the first L_n columns of $\mathbf{U}_n$. The basis functions is composed to $\mathbf{B}_n = [\mathbf{b}^1_{n,target}, \mathbf{b}^2_{n,target}, \cdots, \mathbf{b}^{L_n}_{n,target}]^\top \in \mathbb{R}^{L_n\times\bar{L}}$ with the first L_n rows of $\Sigma_n \mathbf{V}_n^\top$. Each column of $\mathbf{B}_n$, i.e. $\mathbf{b}^l_{n,target}$, is the target vector of $\boldsymbol{\beta}^l_n(\chi_1; \mathbf{w}_{n,l}) = [b^l_n(\chi_1; \mathbf{w}_{n,l}), \cdots, b^l_n(\chi_{\bar{L}}; \mathbf{w}_{n,l})]^\top \in \mathbb{R}^{\bar{L}}$ for $l = 1, \cdots, L_n$, where χ_j for $j = 1, \cdots, \bar{L}$ is calculated from the dynamics of χ. The number L_n is selected by the singular value spectrum such as $\sum_{l=1}^{L_n} \sigma_{n,l} / \sum_{m=1}^{M} \sigma_{n,m} > 0.9$, where $\sigma_{n,m}$ indicates the m-th diagonal element of Σ_n. In each n-th variable, the weights for the l-th element of the basis function is obtained by supervised learning as below:

$$\mathbf{w}^*_{n,l} \leftarrow \arg\min_{\mathbf{w}_{n,l}} ||\mathbf{b}^l_{n,target} - \boldsymbol{\beta}^l_n(\chi; \mathbf{w}_{n,l})||^2. \tag{3.4}$$

The locally weighted regression is used for obtaining the weights of basis function $\mathbf{b}(\chi; \mathbf{w})$ as it is a non-parametric technique which has a low computational complexity and determines the necessary weights $\mathbf{w}$ automatically. Thus, in each demonstration, the specific set of style parameters and the weights for basis functions are obtained. Since basis functions map a subspace of all the training trajectories, the required movements corresponding situations are generalized by choosing the appropriate style parameters based on the environmental information.

3.2 Generalization Process in PDMPs

In order to reveal the relationship between the environment and the corresponding optimal motion, an explicit function is introduced, which is called *mapping function*. The mapping function is represented by solving a regression problem between *environmental parameters* and style parameters, where the environmental parameters are representative environmental information. Since the style parameter plays a dominant role in extracting the control policies, constructing the mapping function is a crucial part. In this section, the generalization process is presented, including settings of both environmental parameters and mapping function, which are important for the efficient description of style parameters.

3.2.1 Environmental Parameters

To represent the environment as a numerical variable, the environmental parameters are defined here. Environmental parameters serve as independent variables, and their distribution properties have great effects on the regression performance. Therefore, the distinctive features in environments are selected as environmental parameters. Let the environmental parameters as $\mathbf{r} = [\mathrm{r}_1, \mathrm{r}_2, ..., \mathrm{r}_k]^\top \in \mathbb{R}^{k\times 1}$. Each element, r_i for $i = 1, 2, ..., k$ is a distinctive feature of corresponding environments. For an example of the optimal avoidance scenario, environmental parameters consist of the obstacle information such as dimension or location of obstacle. Thus, in each demonstration, the specific set of style parameters and the weights for basis functions are obtained. Since basis functions map a subspace of all the training trajectories, the required styles of motion can be generalized by choosing the appropriate style parameters based on the environmental information.

3.2.2 Mapping Function

In order to present the explicit relationship between environmental and style parameters, regression problem is formulated here. When the compact sets of demonstrations are se-

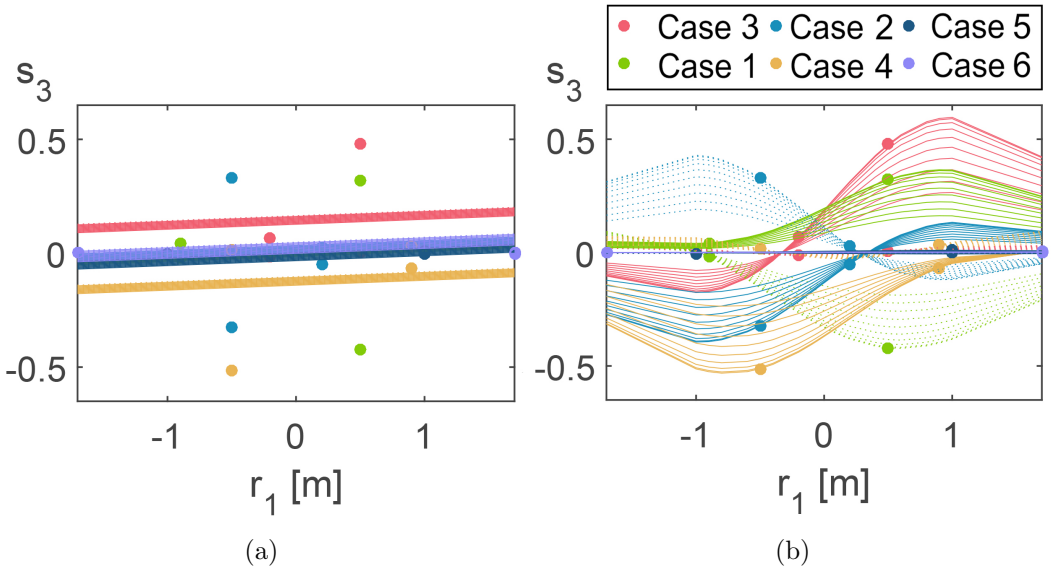


Figure 3.2: The results of regression through (a) LWR and (b) GPR for the example scenario of cooperative aerial transportation. In general, nonlinear relationship is observed between environmental and style parameters. Each point indicates the third style parameter with respect to the first environmental parameter, in demonstration set (see Fig. 3.5).

lected, the essential property for regression model is considered as a zero empirical error. The zero empirical error indicates that the regression function computes exact output style parameters from each set of environmental parameters in trained demonstrations.

Among the regression model, linear regression model gives direct and simple representations as $\mathbf{S}_{1:N} = \beta\mathbf{R}$. Each $\mathbf{S}_{1:N} \triangleq [\mathbf{S}_1^\top, \mathbf{S}_2^\top, ..., \mathbf{S}_N^\top]^\top \in \mathbb{R}^{L\times M}$ and $\mathbf{R} \triangleq [\mathbf{r}^1, \mathbf{r}^2, ..., \mathbf{r}^M] \in \mathbb{R}^{k\times M}$ indicates the matrix of stacked sets of style parameters and environmental parameters, respectively. The least-square solution for the constant matrix would be $\beta = \mathbf{S}_{1:N}\mathbf{R}^+$, where $\mathbf{R}^+$ denotes the pseudo-inverse of $\mathbf{R}$. Practically, the number of demonstrations M is bigger than k, making the $\mathbf{R}$ matrix fat. Therefore, the system is usually over-determined, which prevents from obtaining the exact solution for training data in general cases.

Therefore, for the zero empirical error, a nonlinear model is required. Obviously, this non-linear property is observed from visualized relationships between some elements of environmental and style parameter. For example of optimal aerial transportation scenario, these nonlinear relationship is also observed in Fig. 3.2. Thus, a nonlinear model is recommended for zero empirical error. Among the nonlinear modeling approaches, Gaussian process regression (GPR) is a highly recommended as its accurate and flexible regression performance for approximating unknown nonlinearities. Here, the complexity of GPR scores $\mathcal{O}(M^3)$ for M number of demonstrations. GPR is formulated using the matrix of stacked sets of style parameters, which is defined as $\mathbf{S}_{1:N} \triangleq [\mathbf{S}_1^\top, \mathbf{S}_2^\top, \cdots, \mathbf{S}_N^\top]^\top \in \mathbb{R}^{L\times M}$ and environmental parameters $\mathbf{R} \triangleq [\mathbf{r}^1, \mathbf{r}^2, \cdots, \mathbf{r}^M] \in \mathbb{R}^{k\times M}$. Here L is defined as $\sum_{n=1}^{n=N} L_n$. The below is the regression function in this formulation:

$$\begin{bmatrix} \mathbf{S}_{1:N}^\top \\ \mathbf{s}_{1:N}^{new\top} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \mathbf{K}(\mathbf{R},\mathbf{R}) & \mathbf{K}(\mathbf{R},\mathbf{r}^{new}) \\ \mathbf{K}(\mathbf{r}^{new},\mathbf{R}) & \mathbf{K}(\mathbf{r}^{new},\mathbf{r}^{new}) \end{bmatrix}\right), \tag{3.5}$$

where $\mathbf{K}$ indicates the matrix of Gaussian kernel [59]. $\mathbf{s}_{1:N}^{new} \triangleq [\mathbf{s}_1^{new\top}, \mathbf{s}_2^{new\top}, \cdots, \mathbf{s}_N^{new\top}]^\top \in \mathbb{R}^{L\times 1}$ is the column vector consisting of the new style parameters. $\mathbf{r}^{new}$ is the newly updated environmental parameters calculated from the actual environment. The vector of the style parameters $\mathbf{s}_{1:N}^{new}$ is obtained as

$$\mathbf{s}_{1:N}^{new} = (\mathbf{K}(\mathbf{r}^{new},\mathbf{R})\mathbf{K}(\mathbf{R},\mathbf{R})^{-1}\mathbf{S}_{1:N}^\top)^\top. \tag{3.6}$$

The offline work includes the calculation of the multiple trajectories for demonstrations and $\mathbf{K}(\mathbf{R},\mathbf{R}) \in \mathbb{R}^{M\times M}$ for the mapping function, which demands a relatively high computational load. During the online phase, the new style parameters $\mathbf{s}_{1:N}^{new}$ is obtained immediately from (3.6) with the calculation of kernel matrix $\mathbf{K}(\mathbf{r}^{new},\mathbf{R}) \in \mathbb{R}^{1\times M}$ and the current environmental parameters. Since a set of weights is already learned for basis function from (3.4), the final attractor function is obtained by simply multiplying new style

Offline demos	Offline learning	Online reproduction
$\mathcal{O}(M)$	$\mathcal{O}(M^2L)$ or $\mathcal{O}(M^3)$	$\mathcal{O}(ML)$

Table 3.1: Complexity in the proposed PDMPs.

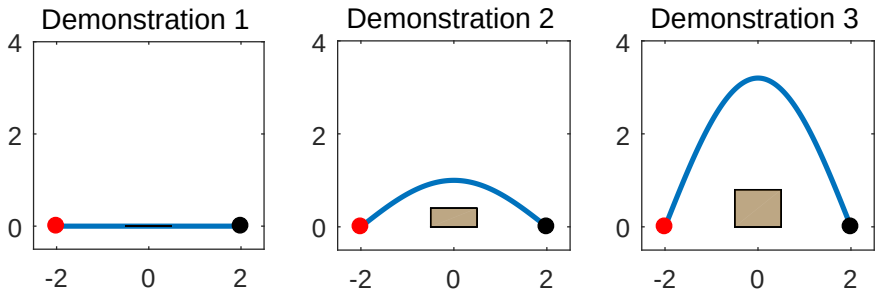


Figure 3.3: Three demonstrations are provided for two-dimensional hurdling motion.

parameters and the represented basis function. Hence, once the combined offline framework is built, the motion can be generated or modified quickly in the online phase. For both offline and online process, the complexity of GPR is analyzed in Table. 3.1. Be noticed that the complexity order is reduced from $\mathcal{O}(M^3)$ to $\mathcal{O}(ML)$ for online computation. Moreover, M scores relatively small number compared to other GPR problems, so our formulation is not dominantly affected by the complexity issue of the GPR. The problem of increasing M for complex missions will be addressed in Chapter 4 of the paper.

3.3 Simulation Results

3.3.1 Two-dimensional Hurdling Motion

Consider a simple generalization problem of two-dimensional hurdling motion. In this scenario, for a simple description, only the height of the hurdle is supposed to vary from zero to any value. The environmental parameter is the height of the hurdle. In order to

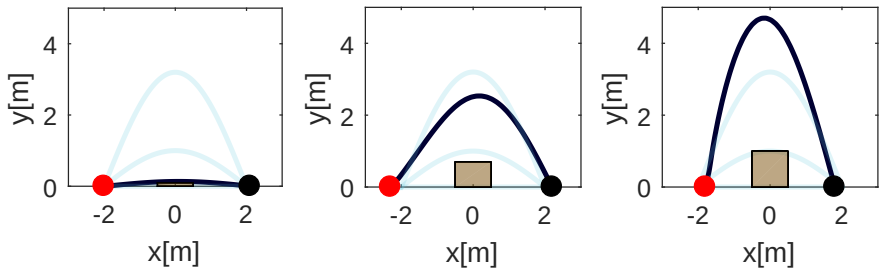


Figure 3.4: The simulation results for two-dimensional hurdling motion. Each red and black point indicates the start and goal states, respectively. The dark blue line shows the computed trajectory of the robot. Light blue lines show the demonstrations in corresponding PDMPs.

generalize the movement of the robot avoiding the hurdle, three different demonstrations are provided in Fig. 3.3. The generalization results are listed in Fig. 3.4. With only three demonstrations, the avoidance of any height of hurdle is possible.

3.3.2 Cooperative Aerial Transportation

In order to show that this proposed framework can serve as a real-time optimal motion planner, this section provides simulation results for cooperative aerial transportation scenario in obstacle environments. From that this framework generalizes the required behavioral style, the computed movements will sufficiently reflects the desired property of demonstrations. Thus, both properties of quickness and optimality are satisfied by providing optimal demonstrations to PDMPs using an optimal motion planner. Here, RRT* is used in the way that it achieves asymptotic optimality without the worry of converging local minima, eternally. Here, RRT* runs in off-line so the issue of slow convergence diminishes.

Ideally, it is desirable to configure the demonstrations by randomly changing the environmental parameters and extracting the characteristic data values observed among them. In the example scenarios in this dissertation, the demonstrations using trigonometric func-

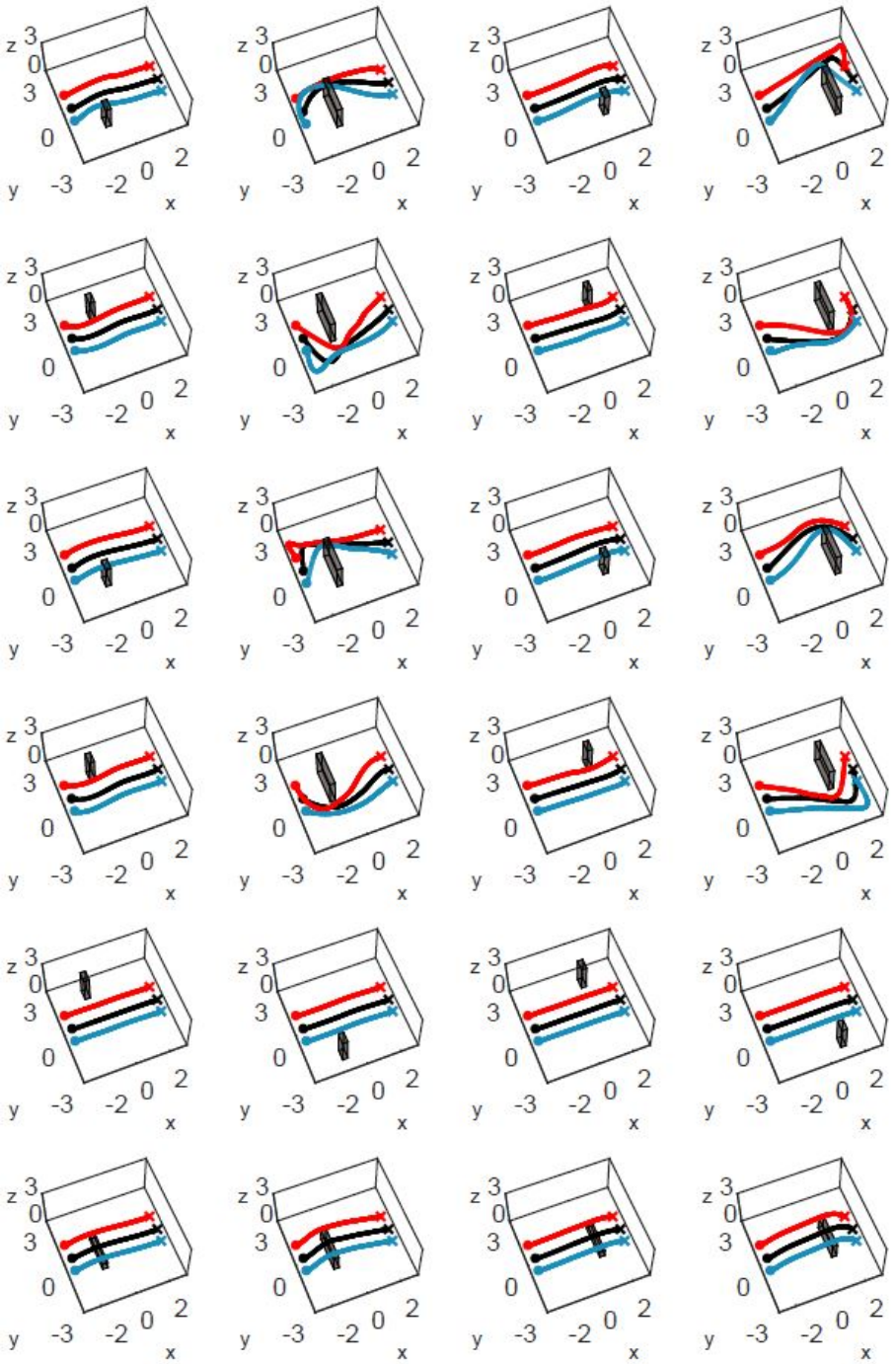


Figure 3.5: 24 Demonstrations using RRT*. In each environment, an obstacle having different dimension and location is located. Each black, red and blue line shows the trajectory of body positions of object, left and right aerial manipulator, respectively. The first two rows and the next two rows demonstrate negative and positive yawing motion for avoidance in y direction.

		Avoidance direction			
		Positive y	Negative y	Positive z	None
Yawing during the avoidance	Positive	Case 1	Case 2		
	Negative	Case 3	Case 4		
	zero			Case 5	Case 6

Table 3.2: The distinctive avoidance skills against ground obstacles in cooperative aerial manipulation.

tions, RRT* and iLQR methods were used as data for PDMPs. Each RRT* and iLQR is used to generate the trajectory for cooperative aerial transportation in sections 2.2.2 and 3.3.2 and hang-dry mission in section 6.2, respectively. However, getting demonstrations in mobile manipulators is not practical for random trial. (Be noticed that the method of obtaining the demonstrations is not strictly part of the PDMPs algorithm. PDMPs only generalize the movements from demonstrations set that previously obtained) When the generalization goal is specified, it is possible to specify distinctive demonstration settings and save the effort of obtaining all the demonstrations. From the RRT* results in Fig. 3.5, six distinct cases are observed for essential avoiding skills as listed in Table. 3.2. In order to represent the magnitude of specific motion, at least two demonstrations are required per each case. The distance from the obstacle is also considered by setting it at two different values of x coordinate, which leads to the total number of demonstration set for mimicking general environments as 24 ($= 6 \times 2 \times 2$). In Fig. 3.5, 24 distinctive results from RRT* are listed. 24 RRT* demonstrations (Fig. 3.5) and basis functions $\mathbf{b}_n$ for $n = 1, 2, ..., N$ are computed. Here, the demonstrations in the second row from bottom show similar movements but are included to training demonstrations. Since PDMPs learn not only individual movements in demonstrations but also corresponding environmental parameters, those demonstrations are treated as individual data. The explicit mapping function was obtained as described in section 3.2. Based on the above works, the motion trajectory is computed during the on-line phase. In the particular setting reported here, $k = 8$ for the environmental parameters including the closest x, y and z positions of obstacle edges. In a new environment, the information of obstacles is sensed in real-time and used to form a set of environmental

parameters. From the environmental parameters, the style parameters are obtained and used to generate motion skills based on attractor dynamics with $f_n = \mathbf{s}_n^T \mathbf{b}_n$.

In order to guarantee that the produced motion is collision-free to newly placed obstacle, algorithm runs in 2000 sets of environments containing obstacles with new dimension and location and all of these results are checked as collision-free. From the results, it is confirmed that avoidance is successfully performed from the obstacles located in space $\mathcal{C} = \{[x, y, z]^\top \in \mathcal{C} | -1.7 \leq x \leq 1.7, -2 \leq y \leq 2, 0 \leq z \leq 2\}$, quantitatively. In Figs. 3.7a – 3.7d, the examples are presented for a single and twin obstacle(s) environments. The black and grey lines indicate the produced trajectory and 24 demonstrations of the center of object. Each red and blue line is the body trajectory of left and right aerial manipulator, respectively. For the cases of twin obstacles in Figs. 3.7c and 3.7d, style parameter is updated right after the first obstacle is avoided and second obstacle is observed. If there are enough space to converge on a modified path to avoid second obstacle (with updated style parameter), the path is collision-free. Once the combined offline framework of RRT*-PDMPs is built, the motion can be generated or modified quickly in the online phase. The effectiveness of this learning-based motion planning, RRT*-PDMPs is compared with RRT* algorithm in Fig. 3.6 and Table. 3.3.

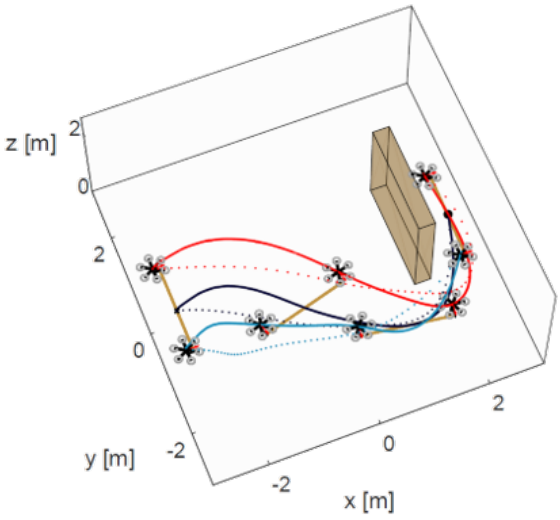
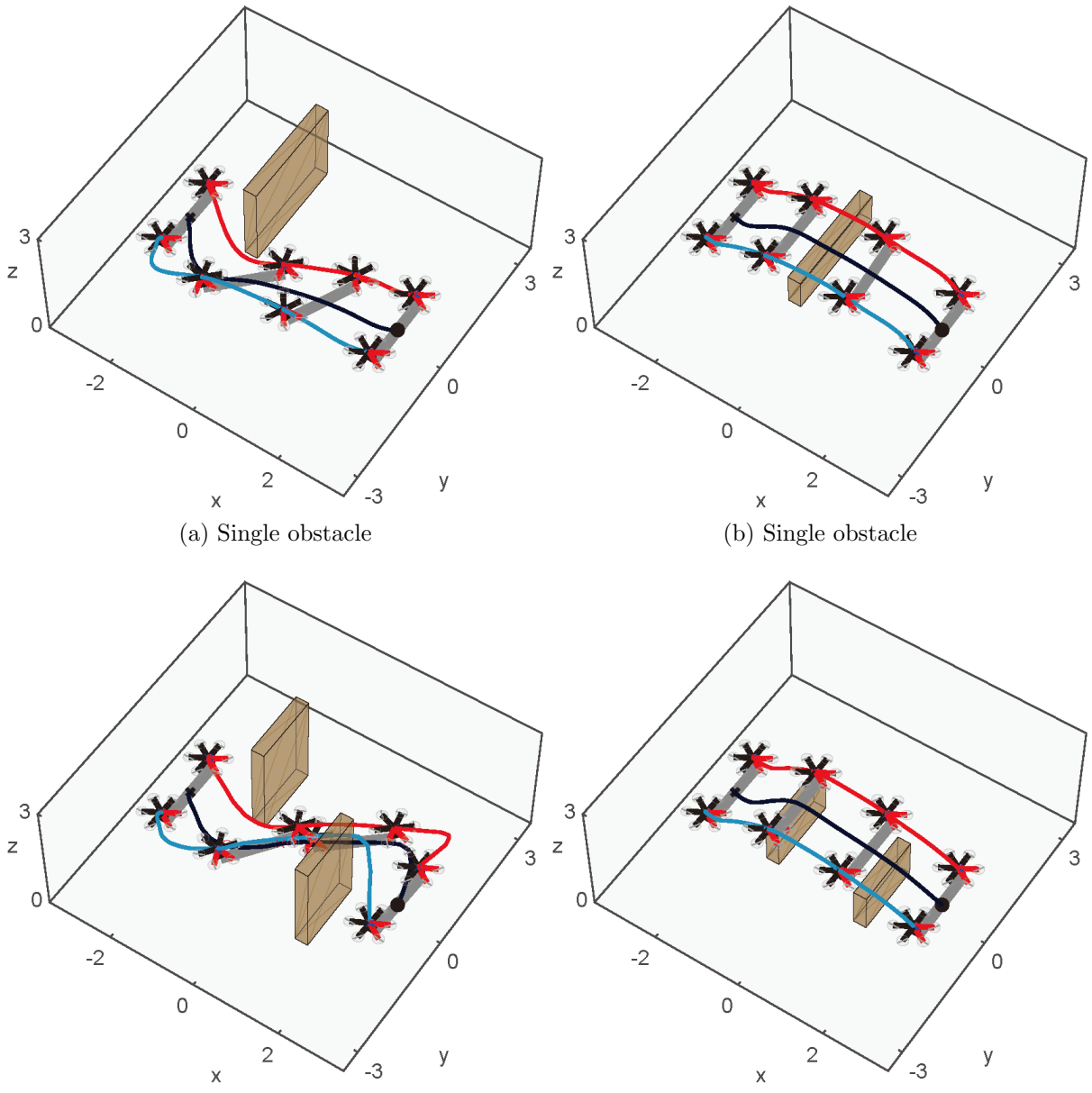


Figure 3.6: Comparison of the efficiency and computational time with sampling-based planning (RRT*) and learning-based planning (RRT*-PDMPs). The red and blue lines are the body trajectories of aerial manipulators, respectively. The black line indicates the produced trajectory of the center of object. Each dashed and solid line indicates the trajectory of aerial manipulator from RRT* and RRT*-PDMPs, respectively.

	RRT* only (dashed)		RRT*-PDMPs (solid)
	First trajectory	Optimized trajectory (30,000 nodes)	Online phase only
Computational time	121.23 s	1,830.53 s	**4.21 ms**
Travel distance	21.16	**13.16**	**13.94**

Table 3.3: Performance comparison between RRT* and RRT*-PDMPs (See Fig.3.6).



(a) Single obstacle

(b) Single obstacle

(c) Twin obstacles

(d) Twin obstacles

Figure 3.7: Simulation results of RRT*-PDMPs.

4

Motion Generalization with Safety Guarantee

Suppose that there are unsafe values of state variables. For example, some robot states result in exceeding the actuation limits and break the system stability. In robot manipulation, internal interference occurs between the robot arm links in the particular state variables. In order to address the safety of motion avoiding these types of unsafe states, this section addresses the additional process to be applied to the proposed framework in section 3. This process can be viewed as a way of managing the demonstrations in advance.

4.1 Safety Criterion in Style Parameter

Once the PDMPs framework is built on multiple demonstrations, the online behavior of the PDMPs is determined by calculating the style parameters for the current environment. Therefore, the safety of online motion, i.e., the motion computed during the online process, is verified by checking in advance whether the style parameter produces an unsafe behavior of the robot. Here, the safety criterion in the style parameter domain reflects the new behavior computed through the generalization process, as well as the individual motion safety of the demonstration set.

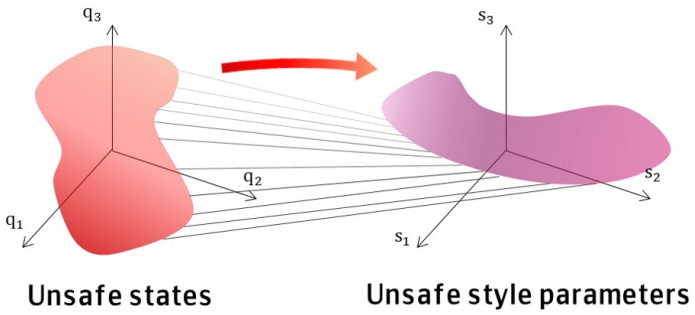


Figure 4.1: The set of unsafe states is mapped into the region of unsafe style parameters.

In the following, the optimization problem is described to obtain the boundary of the safe value of style parameters. Let $\mathbf{q}_{us}$ denote an unsafe state variable. The objective in this section is to obtain the style parameter which calculates the closest movement to the given $\mathbf{q}_{us}$. Here, the closest movement indicates the movement corresponding to the value of state variables very close to the unsafe range such as dangerous position, maximum velocity, and so on. Via optimization, the set of unsafe states is mapped to the region of unsafe style parameters (See Fig.4.1).

Consider the vector $\bar{\mathbf{q}}^t = [\mathbf{q}^{t\top}\ \dot{\mathbf{q}}^{t\top}]^\top$ that consists of the state vector at the t-the time step, $\mathbf{q}^t$, and the time derivatives of $\mathbf{q}^t$. From 4.1, attractor function for all state variables is represented as

$$\mathbf{f}(\chi, \mathbf{s}; \mathbf{w}) = \mathbf{B}(\chi; \mathbf{w})^\top \mathbf{s}, \tag{4.1}$$

where $\mathbf{B}$ is the basis matrix defined as,

$$\mathbf{B}_{target} = \begin{bmatrix} \mathbf{B}_{1,target} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_{2,target} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{B}_{N,target} \end{bmatrix}. \tag{4.2}$$

By applying the dynamics of PDMPs in 3.1, the dynamics of $\bar{\mathbf{q}}^t$ can be represented as

$$\dot{\bar{\mathbf{q}}}^t = \mathbf{A}\bar{\mathbf{q}}^t + \boldsymbol{\beta}^t\mathbf{s} + \mathbf{D}^t, \tag{4.3}$$

where,

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}_N & \mathbf{I}_N \\ -\alpha_1\mathbf{I}_N/\nu^2 & -\alpha_2\mathbf{I}_N/\nu \end{bmatrix},$$

$$\boldsymbol{\beta}^t = \begin{bmatrix} \mathbf{0}_N \\ \alpha_1\mathbf{B}^t(\chi^t;\mathbf{w}^t)/\nu^2 \end{bmatrix}, \tag{4.4}$$

$$\mathbf{D}^t = \begin{bmatrix} \mathbf{0}_N \\ \alpha_1(\mathbf{g} - \mathbf{g}\chi^t + \mathbf{q}_0)/\nu^2 \end{bmatrix}. \tag{4.5}$$

Here, each $\mathbf{0}_N$ and $\mathbf{I}_N$ represents $N \times N$ zero and identity matrix, respectively. The superscript $t = 1, \cdots, T$ indicates that the variables are concerned with the variables at the t-th time step.

Now, define the vector $\tilde{\mathbf{q}} = [\bar{\mathbf{q}}^{0\top} \cdots \bar{\mathbf{q}}^{T\top}]^\top$. Then, $\tilde{\mathbf{q}}$ can be simplified to

$$\tilde{\mathbf{q}} = \boldsymbol{\Phi}\mathbf{s} + \boldsymbol{\Psi} \tag{4.6}$$

where the t-th elements of each $\mathbf{\Phi}$ and $\mathbf{\Psi}$ matrix for $t = 1, \cdots, T$ are represented as

$$\Phi^t = \sum_{i=1}^{t} (\mathbf{A}\mathrm{dt} + \mathbf{I}_N)^{i-1} \boldsymbol{\beta}^{t-i} \mathrm{dt}, \tag{4.7}$$

$$\Psi^t = (\mathbf{A}\mathrm{dt} + \mathbf{I}_N)^t \bar{\mathbf{q}}_0 + \sum_{i=1}^{t} (\mathbf{A}\mathrm{dt} + \mathbf{I}_N)^{i-1} \mathbf{D}^{t-i} \mathrm{dt}. \tag{4.8}$$

In order to compute only the position difference to unsafe states rather than time derivatives, set the matrix $\mathbf{\Theta}^t = [\mathbf{I}_N \ \mathbf{0}_N; \mathbf{0}_N \ \mathbf{0}_N]$ and $\mathbf{\Theta} = diag([\mathbf{\Theta}^1 \cdots \mathbf{\Theta}^T])$. Then, the objective function can be formulated as

$$J = \frac{1}{2} \mathbf{s}^\top \mathbf{H} \mathbf{s} + \mathbf{h}^\top \mathbf{s} \tag{4.9}$$

where, $\mathbf{H} = 2\mathbf{\Phi}^\top \mathbf{\Theta}^\top \mathbf{\Theta} \mathbf{\Phi}$ and $\mathbf{h} = 2\mathbf{\Phi}^\top \mathbf{\Theta}^\top (\mathbf{\Psi} - \tilde{\mathbf{q}}_{us})$.

For all the values of the unsafe states, the corresponding optimal movements and their style parameters are computed. Here, optimal movements indicate the nearest motion. Those style parameters consist of the safety criterion for current PDMPs. Here, any optimization methods, including quadratic programming (QP), can be employed.

4.2 Demonstration Management

After the safety criterion of the style parameters is computed via optimization formulation, the demonstrations causing unsafe motion are eliminated based on this criterion. The overview of the proposed process is shown with the red arrows in Fig. 4.2.

The detail process is described in the following. First, there can be specific unsafe conditions that may have been unconsidered when obtaining the previous demonstrations in PDMPs (e.g. new environmental settings). Then, the safety criterion for style parameters is extracted as described in section 4.1. After the criterion is obtained, it is identified whether each given demonstration is safe or not. When the value of style parameter of a demonstration is outside the safety region, the corresponding demonstration is removed from the

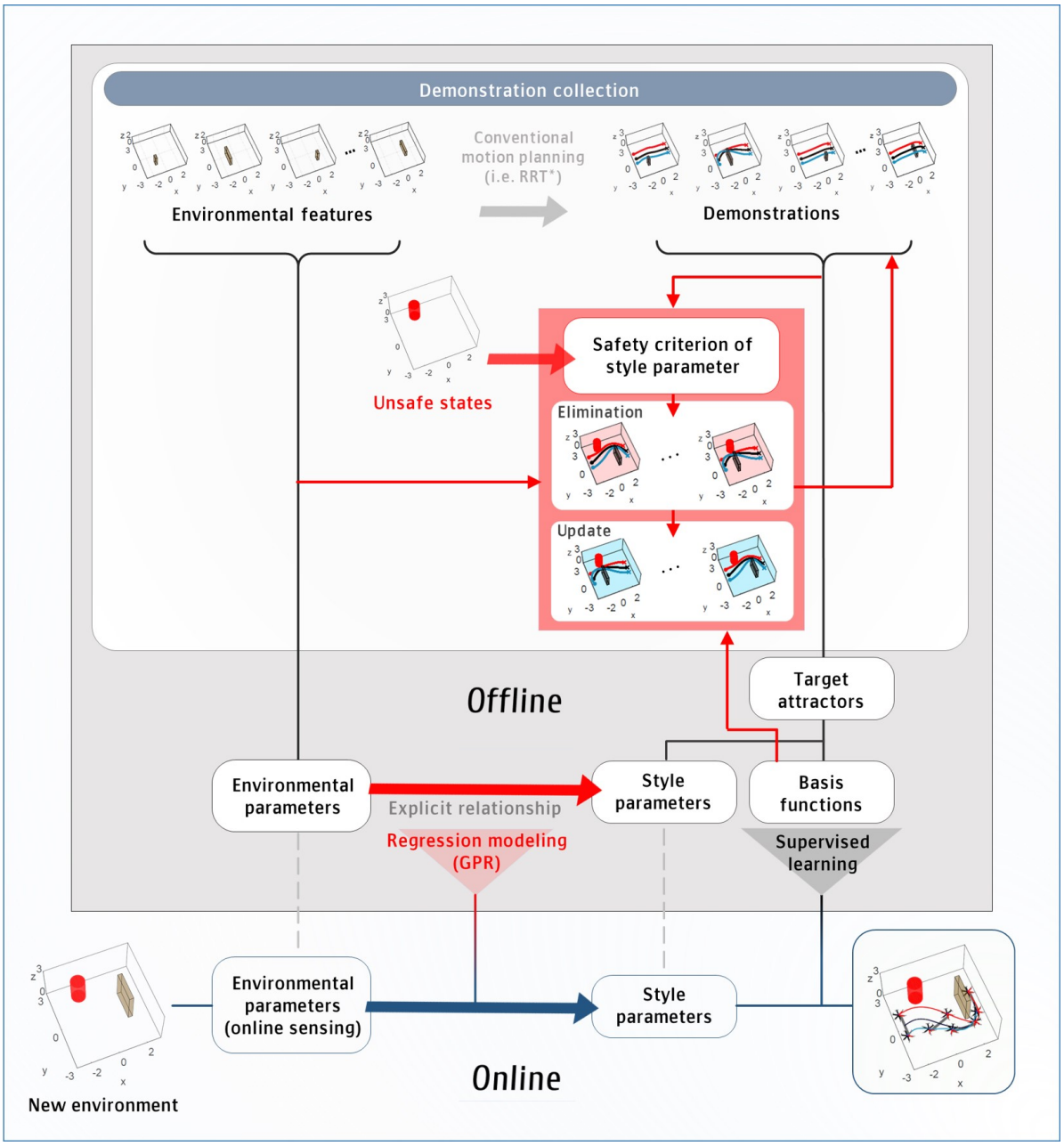
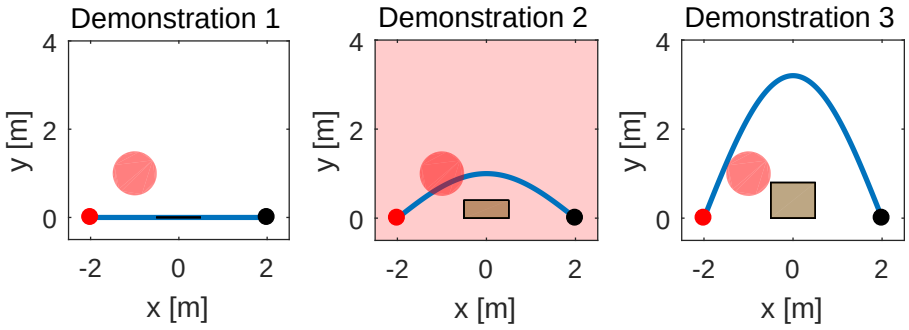


Figure 4.2: An overview of the proposed PDMPs framework with the safety guaranteeing process.

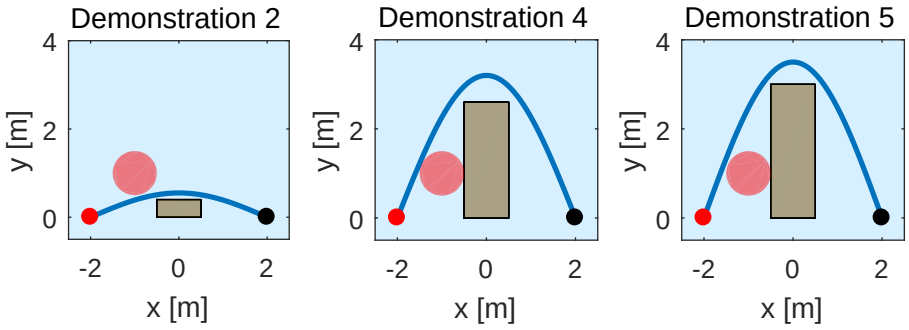
demonstration set. Instead, a new safe demonstration is provided for the same environmental setting. Moreover, to enhance the performance of representation near the unsafe set, additional demonstrations are computed from the style parameters at the boundary of safe region. It is possible that, although there is no unsafe demonstration in the dataset an unsafe motion is computed online for some untested environmental settings, because the motion computed via the generalization process in PDMPs. In such cases, trace back the style parameters that are at the boundary of the safe region and provide additional demonstrations. Here, new demonstrations are provided as follows:

1. Calculate the values of style parameters for all the possible range of environmental parameters.
2. Trace back the environments where some demonstrations caused unsafe motion online for the boundary of the unsafe style parameter values.
3. Provide new demonstrations for the particular environments in Step 2.
4. Check the safety of online motion and repeat Steps 1–3 until there remains no unsafe value of style parameters. (This step determines how many demonstrations are needed.)

After removing the unsafe demonstrations, in step 3, the demonstration set is complemented with new demonstrations. This is intended to maintain generalization performance, especially if the demonstrations removed are dominant data. Although the decrease in the number of demonstrations does not mean that PDMPs are not working, generalization performance is usually degraded. If the removed demonstrations include the data that played a dominant role in the generalization process, performance would degrade further. (See the generalization performance with insufficient data in the examples of driving multiple tracks in Section 5.5.) Thus, providing new demonstrations is important as well as elimination of unsafe demonstrations.



(a)

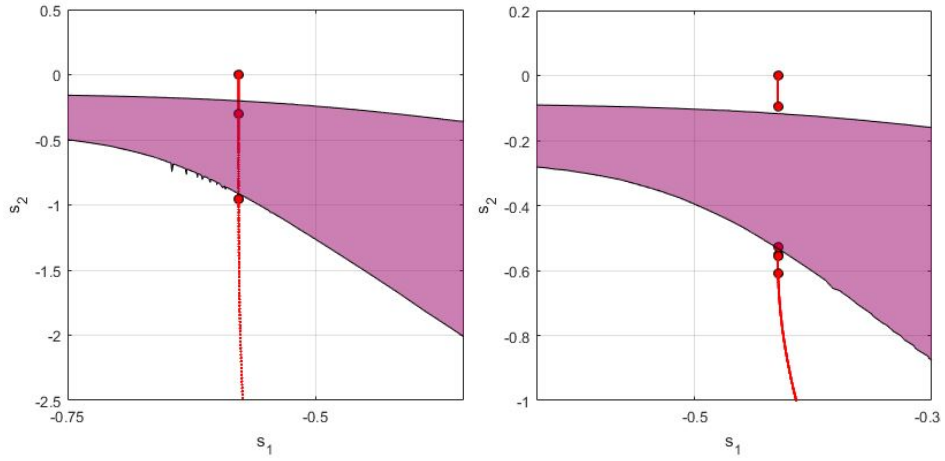


(b)

Figure 4.3: The demonstrations are given for the hurdling motion scenario where each x and y denotes forward and height, respectively. Each red and black point indicates the start and goal states, respectively. The red colored circle shows the unsafe region in state space. The blue line shows the trajectory of the robot. (a) Three demonstrations which are previously given. Based on the safety criterion, the second demonstration is concluded to (marked with red background) invade the given unsafe region. (b) Three newly provided demonstrations.

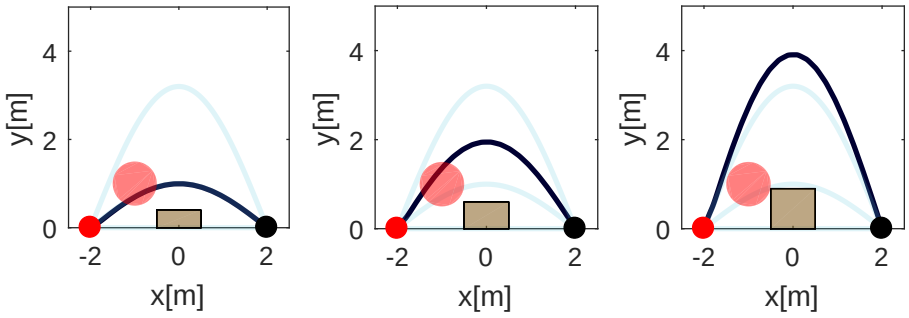
4.3 Simulation Validation

This section includes the simulation results for two different scenarios, similar to section 3.3. The additional setting here is a static obstacle intuitively representing unsafe states. The objectives in both scenarios are to avoid unsafe states occupying this new obstacle while moving to the goal state.

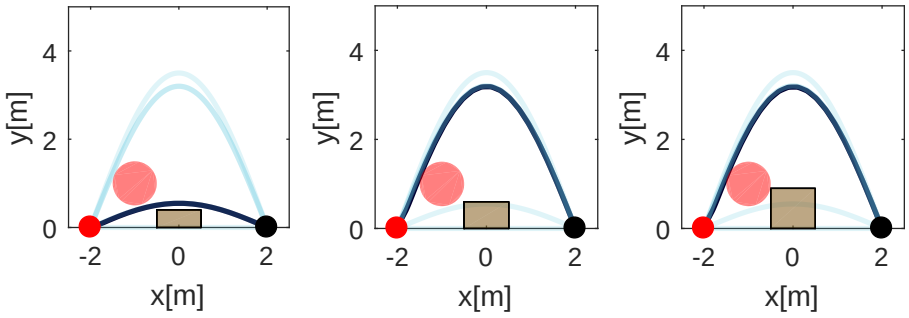


(a) (b)

Figure 4.4: The safety criteria for style parameters in the hurdling motion scenario (a) with the previous demonstrations and (b) with the updated demonstrations. The unsafe regions are colored in purple. Each red point indicates the style parameter obtained from the demonstration data. The red line shows the set of style parameters computed from the GPR function with all the possible obstacle configurations. In (a), there exists an unsafe demonstration, whereas it has been properly removed in (b).

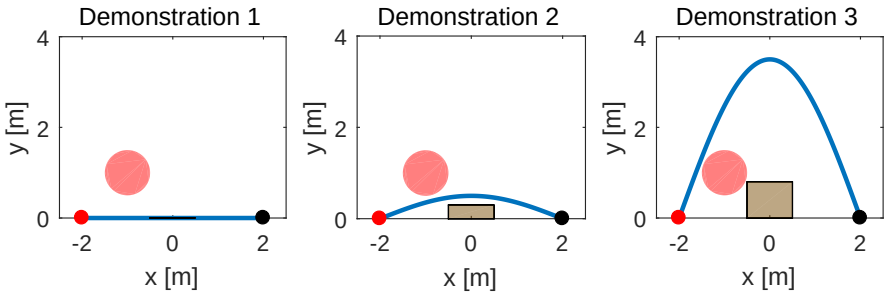


(a)

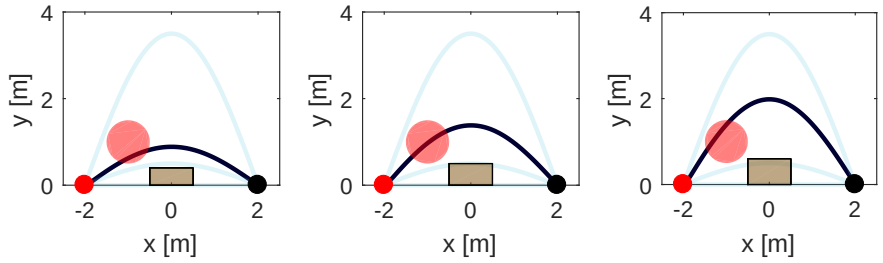


(b)

Figure 4.5: The simulation results for the hurdling motion scenario in the framework of PDMPs (a) without and (b) with the proposed process. Each red and black point indicates the start and goal states. The dark blue line shows the trajectory of the robot. Light blue lines show the demonstrations in corresponding PDMPs.



(a)



(b)

Figure 4.6: The simulation results to elucidate the issue that the resulting motion can be unsafe even though only safe demonstrations are included. (a) Provided demonstrations. (b) Resulting online motion.

4.3.1 Two-dimensional Hurdling Motion

Consider a robot avoiding a stationary ground obstacle. In this scenario, for simple description, only the height of the obstacle is supposed to vary from zero to any value.

First, in order to generalize the movement of the robot avoiding the obstacle, three different demonstrations are provided. If a new unsafe state is specified, the algorithm extracts the unsafe region of style parameters and removes the unsafe demonstration(s). Then, new demonstrations are provided to replace the unsafe motion and enhance the description of the motion near the unsafe region. In Figs. 4.3a and 4.3b, demonstrations are listed, which are previously provided and newly given according to the proposed process, respectively. The background of an unsafe demonstration is colored in red while the backgrounds of newly given demonstrations are marked with blue. The unsafe regions of style parameters are computed as the purple area in Fig. 4.4. Fig. 4.4a is computed from the PDMPs with the previous demonstrations while Fig. 4.4b shows the unsafe region of style parameter with updated demonstrations. As shown in Fig. 4.4a, it is obvious that the unsafe states are included in the second demonstration which is eliminated from the demonstration set. After providing new demonstrations, the values of style parameters are in the safe region for all the possible environmental settings (i.e. obstacle configuration) as shown in Fig. 4.4b. In Fig. 4.5, the online paths with various environmental settings in the PDMPs without and with the proposed process. The trajectories in Fig. 4.5a have been obtained from the original demonstrations only (i.e. Fig. 4.3a), whereas the trajectories in Fig. 4.5b have been computed from the demonstration set excluding unsafe one for the new environment and including additional ones (i.e. five demonstrations as in Figs. 4.3a and 4.3b) From the results, it is shown that the PDMPs successfully consider the unsafe states with the proposed process.

Even when PDMPs are constructed with safe demonstrations only, unsafe motion can still be produced. For the hurdling motion scenario, the demonstrations in Fig. 4.6a can be provided instead of the existing one. Resulting online motions in Fig. 4.6b are not safe

	Demonstration construction		
	w/ the proposed process		w/o the proposed process
	Safety criterion	RRT* demos (required #)	RRT* demos(required #)
Computational time	23.16 s	9,602,83 s (8)	736,533.12 s (28)
Total	**9,626.00 s**		736,533.12 s

Table 4.1: The computational times spent in obtaining demonstrations. Here, the demonstrations can be provided by techniques other than RRT* as before [1, 2]. Be noticed that demonstrations is obtained in the offline phase of PDMPs and the online motion generation is completed in a short time to be used in real-time.

even they are based on the PDMPs with safe demonstrations. By providing the sufficient demonstrations with the proposed process, the safety of motion is guaranteed as in Fig. 4.3b of the manuscript.

4.3.2 Cooperative Aerial Transportation

As described before, the online path is computed by using PDMPs even for the high-dimensional and complex dynamic system such as cooperative aerial manipulators. In section 3.3, PDMPs with the proposed generalization process is applied for cooperative aerial transportation. By providing optimal movements for demonstrations using RRT*, aerial manipulators adapts to the environments with the various size and location of obstacles. Now, a new cylinder-shaped obstacle is added as shown in Fig. 4.7 (compare it to Fig. 3.5). The black line represents x_o, y_o, z_o which are the position of the center of the common object. The unsafe set of states is specified with respect to x_o, y_o, z_o and yaw angle considering that not only the center position of the common object but the yaw angle in the cooperative pose should be considered to check the collision with the obstacle. The safety criterion of style parameters is computed as shown in Fig. 4.8a. Then, four demonstrations intrude the unsafe state. By replacing them with new demonstrations as listed in Fig. 4.7b, the proposed process set the demonstrations which are safe as shown in Fig. 4.8b. In Fig. 4.9, PDMPs successfully avoid the unsafe states with the proposed process. The efficiency of the proposed process is also confirmed as shown in Fig. 4.1.

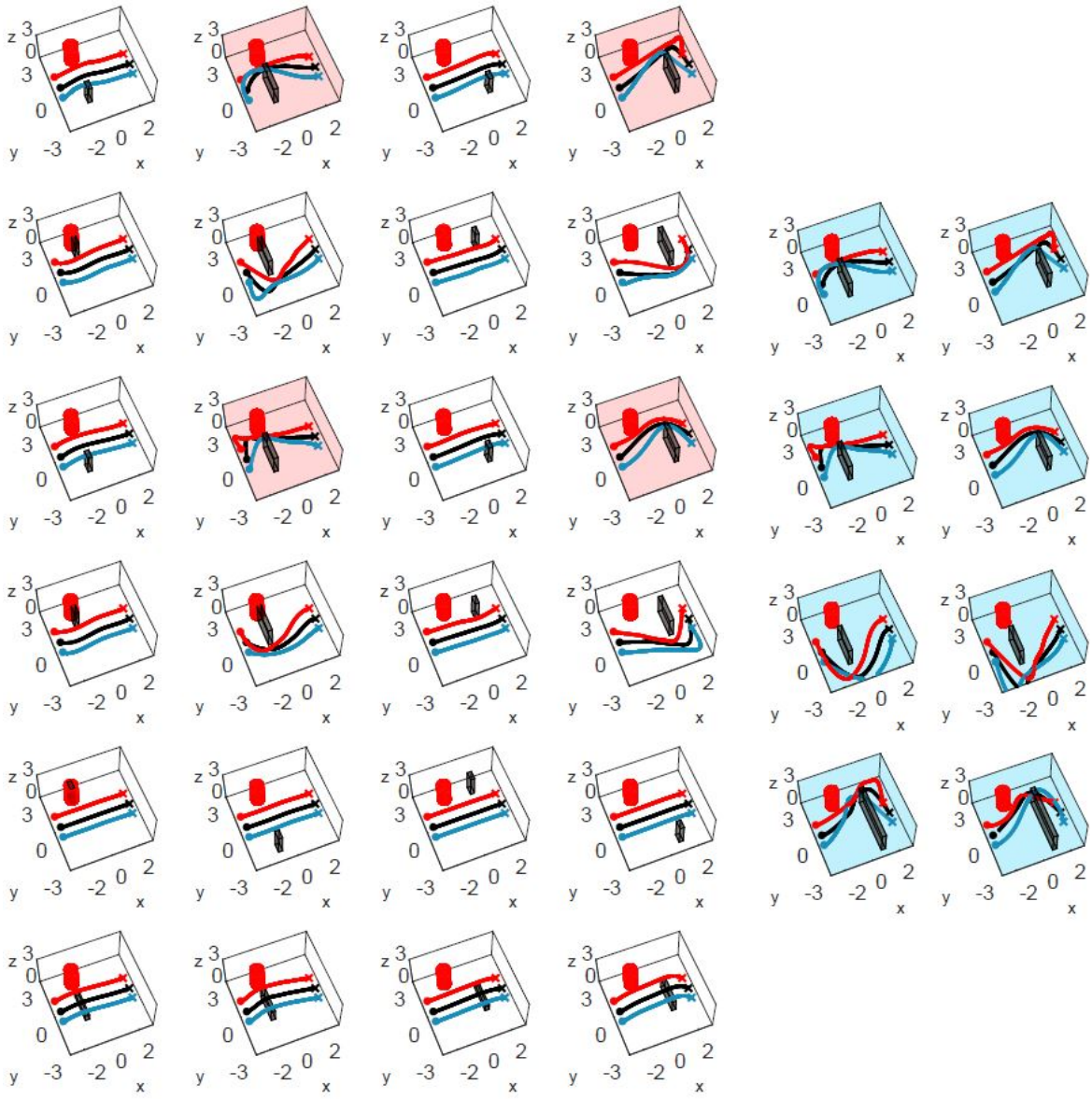
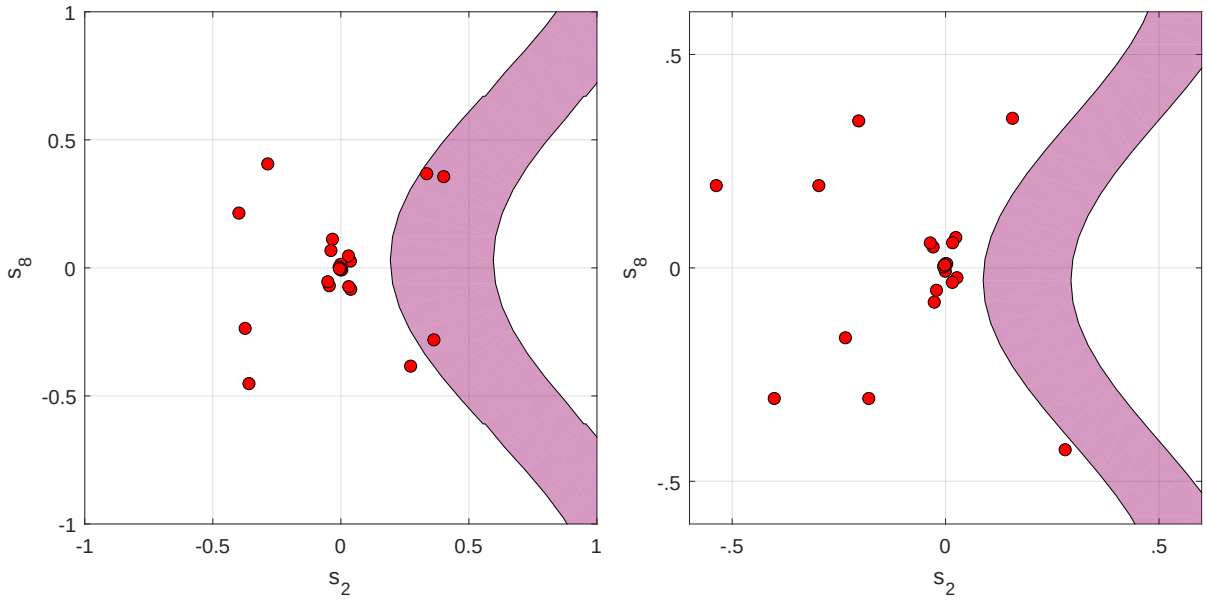


Figure 4.7: The demonstrations provided for the cooperative aerial transportation. Each red and blue line indicates the body trajectories of left and right aerial manipulator, respectively. The black line shows the trajectory of the center of the common object. (a) The firstly given demonstrations did not consider a cylinder-shaped obstacle marked with a red cylinder. With the safety criterion of the style parameter, unsafe demonstrations are marked with red background. (b) The additional demonstrations which are newly given. The first four demonstrations are computed from the same environments as the unsafe demonstrations in (a). Others are the safe demonstrations near the boundary of safety criterion.



(a) (b)

Figure 4.8: The safety criterion for style parameters in the cooperative aerial transportation scenario (a) with the previous demonstrations and (b) with the updated demonstrations. Each red point indicates the style parameter obtained from the demonstration data. The unsafe regions are colored in purple. In (a), there exist unsafe demonstrations in the demonstration, whereas they have been properly removed in (b).

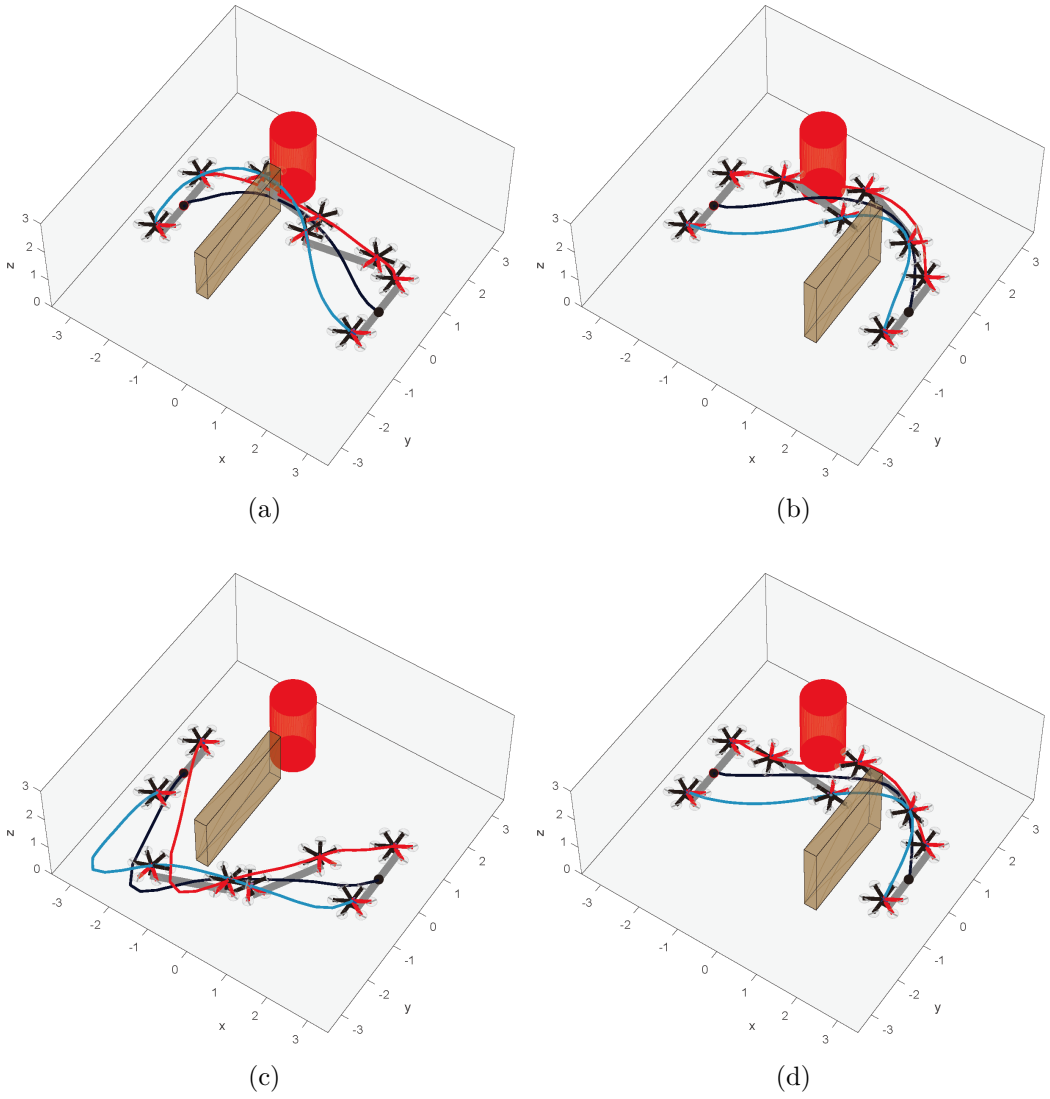


Figure 4.9: The simulation results for the cooperative aerial transportation scenario in the framework of PDMPs (a), (b) without and (c), (d) with the proposed process. Each red and blue line indicates the body trajectories of left and right aerial manipulator, respectively. The black line shows the trajectory of the center of the common object. In (a) and (b), aerial manipulators collide with the cylinder-shaped obstacle following the trajectory learned from the original demonstration set. On the other hand, safe motions are computed in (c) and (d) in the same environments as (a) and (b).

5

Motion Generalization for Complex Missions

This section presents the extension approach, seg-PDMPs, using the proposed PDMPs frameworks. This extension segments the trajectory of entire mission into unit movements and set multiple PDMPs. This is useful to reduce the number of demonstrations where the mission consists of multiple unit sub-tasks that should be generalized independently. For example, hang-dry mission (Fig. 1.1) has multiple sub-tasks (e.g., transporting, stretching, grasping) which can take various styles depending on the situation.

5.1 Overall Structure of Seg-PDMPs

The overview of the framework is shown in Fig. 5.1. First, the demonstrations for a complex mission are segmented into unit movements to represent multiple sub-tasks. Then, multiple PDMPs are formed independently in correlated sub-tasks. A period of performing sub-task is called as a phase in seg-PDMPs. The phase-decision process determines which sub-task and the associated PDMPs should be executed online, allowing multiple PDMPs to be configured within an integrated framework. GPR is applied to obtain reasonable execution time and regional goal configuration in each phase from the actual environment.

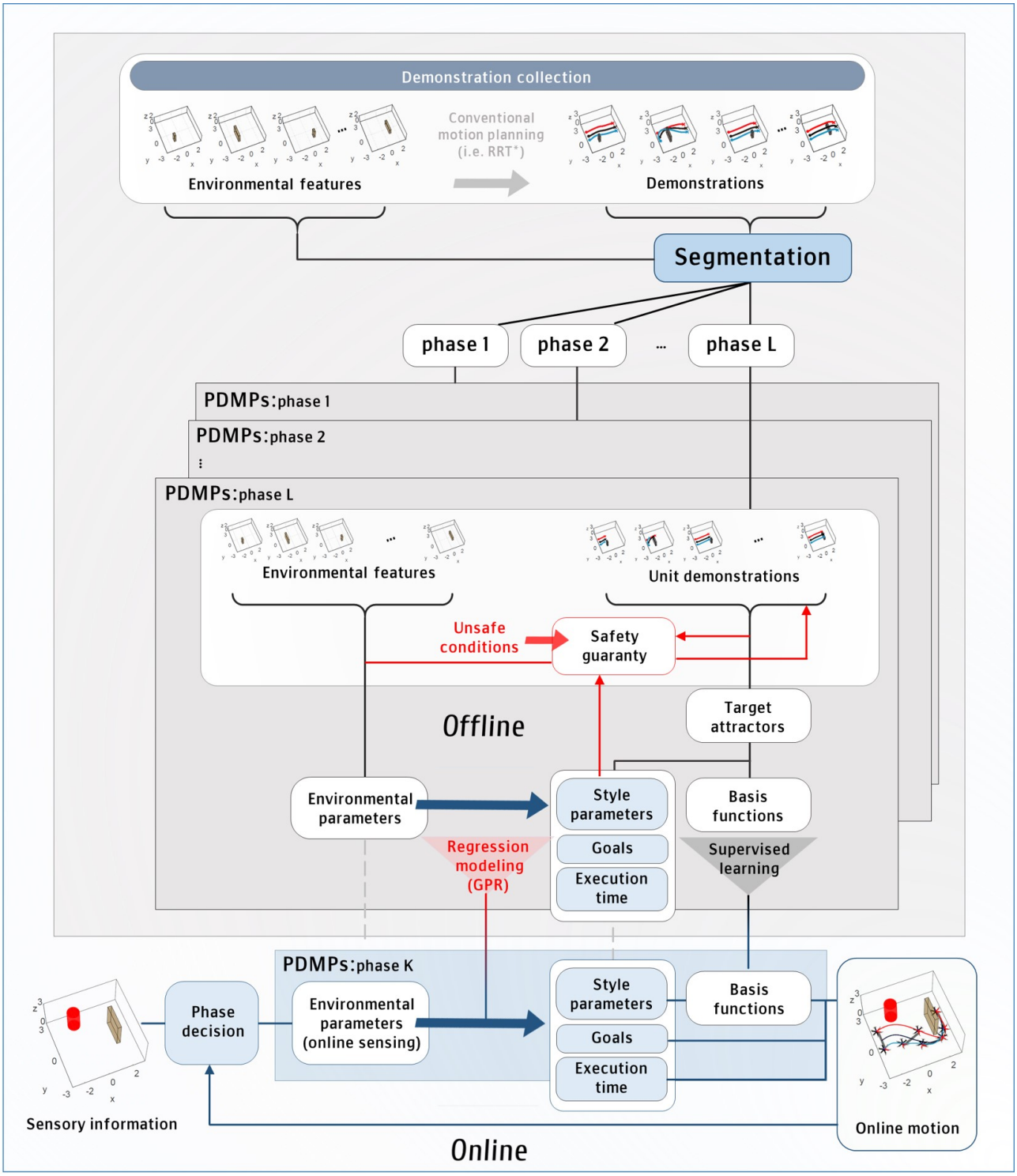


Figure 5.1: The overview of the proposed seg-PDMPs framework.

	PDMPs		Seg-PDMPs
	fixed sequencing	flexible sequencing	flexible sequencing
No. of demos	$M_1 \times M_2 \times \cdots M_{L^{ST}}$	$M_1 \times M_2 \times \cdots M_{L^{ST}} \times L^{ST}!$	$\text{Max}(M_1, M_2, \cdots, M_{L^{ST}})$

Table 5.1: Required number of demos for a complex mission.

5.2 Motion Segments

As mentioned before, M is the number of demonstrations and increasing M is not desirable from the perspective of the complexity of the algorithm (see TABLE. 3.1). If M gets significantly increased, the offline process requires a large amount of computational load in addition to much efforts for demonstration acquisition. Here, the number of required demonstrations is reduced by subdividing the recorded demonstrations rather than using them directly. Let L^{ST} be the number of sub-tasks and M_i for $i = 1, \cdots, L^{ST}$ denote the number of required demonstrations to generalize the styles of each i-th sub-task, respectively. In TABLE 5.1, the number of required demonstrations is listed with PDMPs and seg-PDMPs. Without segmentation, the sequential sub-tasks and their different styles are viewed in one PDMP framework. Thus, the number of required demonstrations for PDMPs will be $M_1 \times M_2 \times \cdots \times M_{L^{ST}}$ when their sequencing order is specified. If the sequencing order is not fixed, the number of possible sequences is multiplied to the number of required demonstrations. In contrast, seg-PDMPs only need fewer demonstrations where the styles of each sub-task are configured at least once. That is, the maximum value from M_1 to $M_{L^{ST}}$ is the number of required demonstrations in seg-PDMPs.

Instead of using segmented demonstrations, it is possible to provide demonstrations for each sub-task independently. However, using segmented demonstrations is preferred because it gives the values of execution time and regional goal state of sub-tasks when the robots perform sub-tasks sequentially. There are various segmenting algorithms such as Hidden Markov Models (HMMs) [60], transition state clustering [60], and zero crossing point(ZCP) or velocity(ZCV) [61]. This dissertation employ simple segmenting methods that focusing on motion segmentation like zero-crossing point or velocity.

5.3 Phase-decision Process

Each of multiple PDMPs generalizes the behavior of the corresponding sub-task in a phase. Seg-PDMPs autonomously determine which sub-task should be performed depending on the situation. This is called the phase-decision process and proceeds as follows.

Let $p = 1, 2, \cdots, L^{ST}$ denote a phase parameter. There are two different ways to specify a phase parameter according to the actual situation. One is using environmental information, and the other is using a segmentation criterion already used in the previous section. In the first approach, the clustering problem can be formulated to compute p from environmental information. The environmental parameters are used as samples and phase parameters are utilized as labels. Then, a clustering algorithm such as supervised k-means clustering is applied can be applied. In the second approach, the segmentation criterion can be used (e.g., ZCV or ZCP). Since the segmentation is already done with this criterion, the second approach gives more direct and accurate results for determining the phase.

Once the phase is determined, the corresponding PDMPs compute the generalized motion for the current sub-task. Whenever the robot completes the current sub-task, the phase-decision process deploys another PDMPs to be executed next. Until the robot reaches the goal or performs the final sub-task of the mission, multiple PDMPs will be configured in this way. The advantage of this architecture is that the robots can flexibly sequence sub-tasks depending on the situation. In order words, any mission with the structure of recombined sub-tasks is possible to be generalized.

5.4 Seg-PDMPs for Single Phase

The PDMPs framework in a single phase proceeds quite similarly to the general PDMPs except for two GPR processes. One process is to calculate the regional goal configuration, and the other is to allocate execution time in the current phase. The regional goals of sub-tasks are the terminal states of segmented demonstrations. The execution times are

the time lengths of segmented demonstrations. Naturally, the segmented demonstrations have different time lengths and goal states depending on the environmental settings and phase, and the GPR function is useful to express a nonlinear relationship between them as the style parameters are computed. Let $\mathrm{t}_{i,p}$ denote the time length for i-th demonstration in p-th phase, where $i = 1, \cdots, M_p$ and $p = 1, \cdots, L$. M_p is the number of demonstrations in p-th phase. By defining the time matrix as $\mathbf{T}_p^\top = [\mathrm{t}_{1,p}, \cdots, \mathrm{t}_{M_p,p}]^\top$, GPR function is formulated to

$$\begin{bmatrix} \mathbf{T}_p^\top \\ \mathrm{t}_p^{\mathrm{new}\top} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \mathbf{K}(\mathbf{R},\mathbf{R}) & \mathbf{K}(\mathbf{R},\mathbf{r}^{\mathrm{new}}) \\ \mathbf{K}(\mathbf{r}^{\mathrm{new}},\mathbf{R}) & \mathbf{K}(\mathbf{r}^{\mathrm{new}},\mathbf{r}^{\mathrm{new}}) \end{bmatrix}\right). \tag{5.1}$$

The Gaussian formulations for regional regional goals are similarly obtained by substituting $\mathrm{t}_{i,p}$, $\mathbf{T}_{i,p}$ for $\mathbf{g}$, $\mathbf{G}_{i,p}$, where $\mathbf{g}_{i,p}$, where $\mathbf{g}_{i,p}$ is the terminal state for $i = 1, \cdots, M_p$ and $\mathbf{G}_p^\top = [\mathbf{g}_{1,p}, \cdots, \mathbf{g}_{M_p,p}]^\top$. Finally, the execution time and regional goal are calculated as,

$$\mathrm{t}_p{}^{new} = (\mathbf{K}(\mathbf{r}^{\mathrm{new}},\mathbf{R})\mathbf{K}(\mathbf{R},\mathbf{r}^{\mathrm{new}})^{-1}\mathbf{T}_p^\top). \tag{5.2}$$

$$\mathbf{g}_p{}^{new} = (\mathbf{K}(\mathbf{r}^{\mathrm{new}},\mathbf{R})\mathbf{K}(\mathbf{R},\mathbf{r}^{\mathrm{new}})^{-1}\mathbf{G}_p^\top)^\top. \tag{5.3}$$

5.5 Simulation Results

Here, simulation results are provided for the mission of driving various tracks to compare the enhanced performance of seg-PDMPs with previous PDMP approaches.In this mission, a mobile vehicle drives P-, S-, and U-shaped tracks in sequence (Fig. 5.2). As mentioned earlier, general DMP can only generalize initial and terminal offset, while parametric skill enables PDMPs and seg-PDMPs to generalize the motion style of various demonstrations. Four demonstrations are provided on each track (two different curvatures $\times$ two different directions, $M_i = 4$ for $i = 1, 2, 3$). All four settings of each course should be considered for generalization in PDMPs or seg-PDMPs. When applying PDMPs for the entire mis-

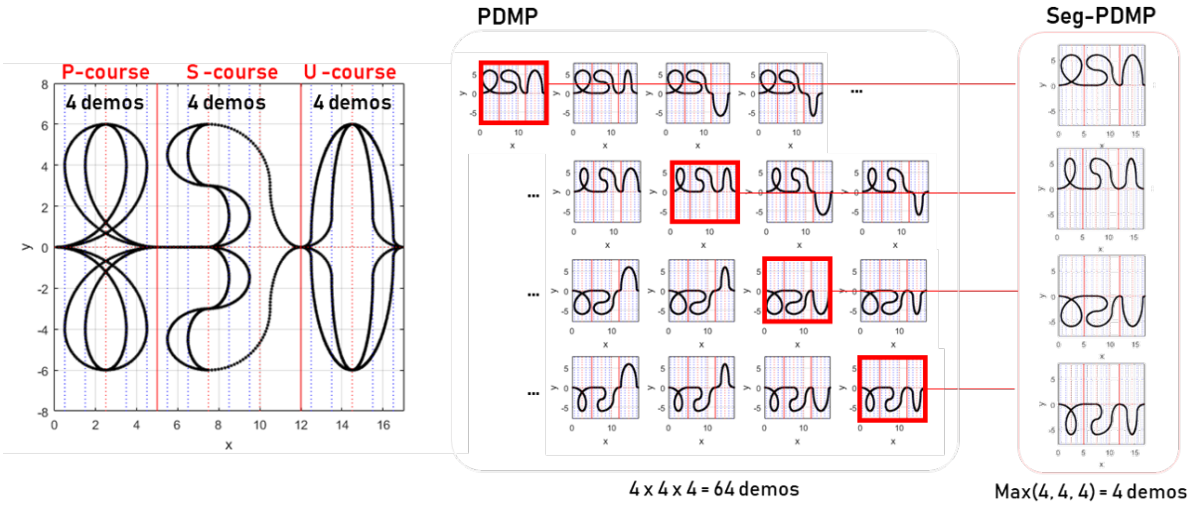
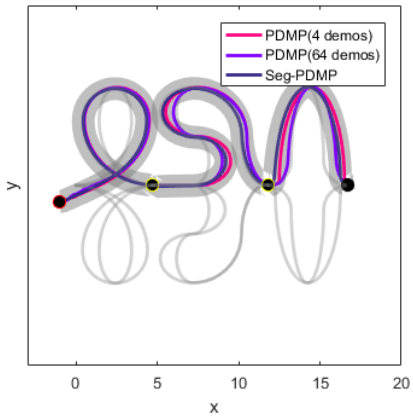


Figure 5.2: Required demonstrations for full generalization of driving multiple tracks scenario. In each demonstration, a mobile robot drives the course consisting of sequential P-, S-, and U-shaped courses. Each course may have various curvature and direction. Therefore, each course requires at least 4 demonstrations (2 directions $\times$ 2 curvatures).

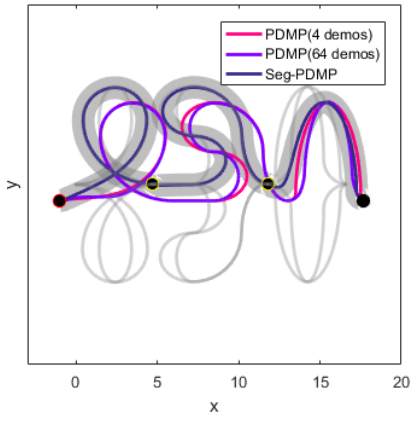
sion rather than individual tracks, the numbers of demonstrations are $M = 64$ for fixed sequencing and $M = 1536$ for flexible sequencing, respectively. On the other hand, seg-PDMPs require $M = 4$. In the following, the performances are compared between PDMPs with 64 demonstrations and the performance of PDMPs and seg-PDMPs with the same four demonstrations.

5.5.1 Initial/terminal Offsets

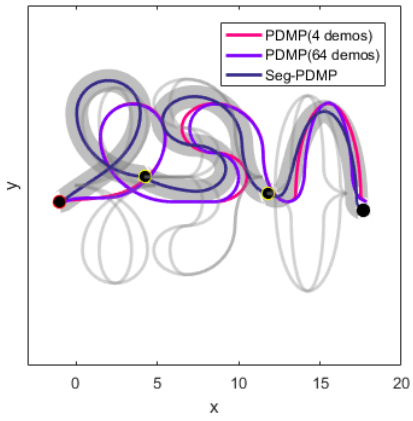
In the online processes of both DMPs and PDMPs, the overall representation of the reference trajectory is modified to deal with the offsets in initial and terminal configurations. Here, the reference trajectory indicates an expected trajectory without variations. The mission conditions, such as via-points or specific movements considered in the reference trajectory, may not be satisfied due to the offsets. For instance, in extending or shortening the reference trajectory, a new trajectory may deviate from via-points. This problem can be reduced to some extent in seg-PDMPs. Since the seg-PDMPs deal with sub-tasks indi-



(a)

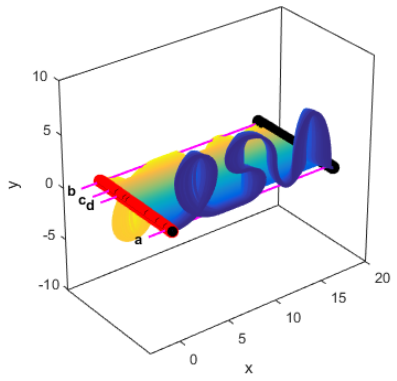


(b)

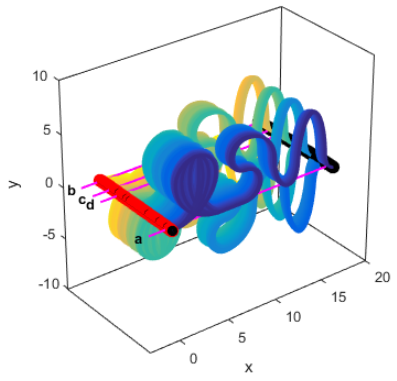


(c)

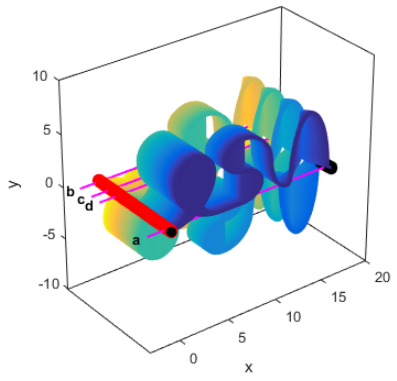
Figure 5.3: Simulation results for driving multiple tracks scenario with different initial/final offsets and via-points. Thick gray line shows the differed environmental settings including the initial, final, and via-point offset. Light gray lines are the demonstrations.



(a)



(b)



(c)

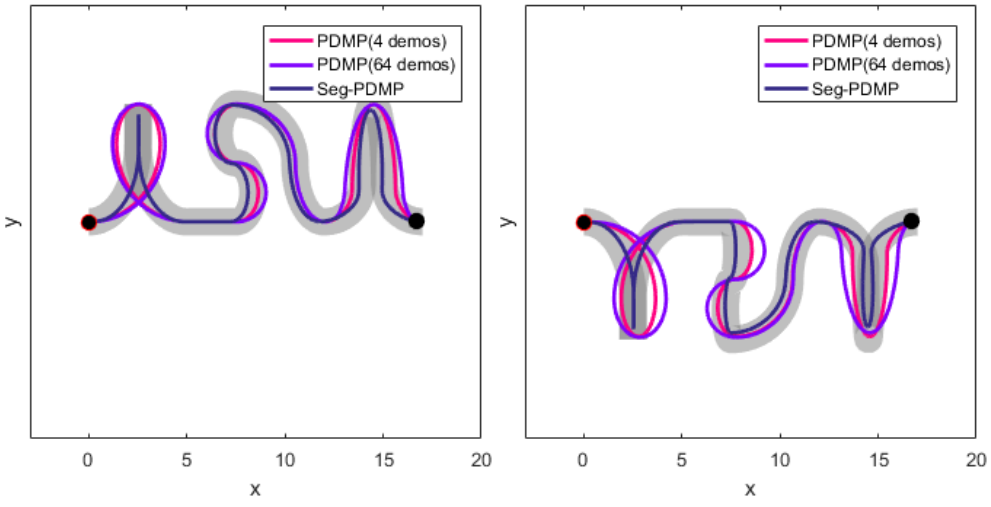
Figure 5.4: Generalization results of (a) PDMPs with four demonstrations, (b) PDMPs with 64 demonstrations, and (c) seg-PDMPs for all possible environmental parameter values (or possible environments). The results of each environment is represented along z-axis. In Fig. 5.5, the results for 'a', 'b', 'c', and 'd' environments are listed in detail.

vidually, maneuvers are modified where the current phase is directly involved in the offsets. Also, for via-point conditions given as intersection points between sub-tasks, the via-point can be satisfied by setting them to the regional goal state of the preceding phase. To afford via-points here, the GPR process to obtain the regional goal in Section 5.4 is ignored.

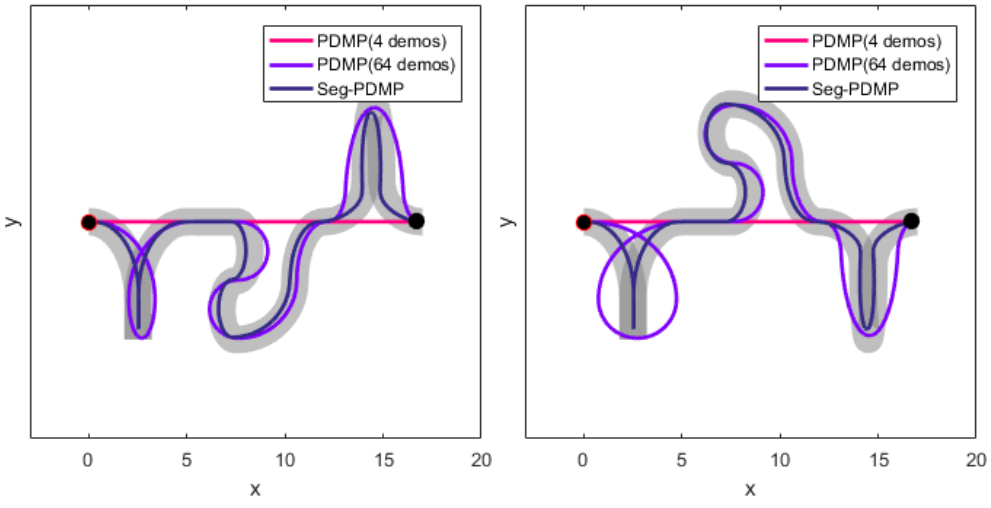
In Fig. 5.3, the representation results are provided for the driving mission using seg-PDMPs and two PDMPs with four and 64 demonstrations. Initial offsets are given for all cases and terminal offsets are given in Figs. 5.3b-5.3c. In Fig. 5.3a, only P-track in seg-PDMPs is affected by the initial offset while S- and U-tracks are additionally affected in PDMPs. Figs. 5.3b-5.3c show that the via-point conditions cannot be satisfied due to offset in some cases in PDMPs. In contrast, seg-PDMPs satisfy via-points by considering them as the regional goals of sub-tasks.

5.5.2 Style Generalization

With the parametric skills, motion styles as well as offsets can be generalized in PDMPs and seg-PDMPs. For the driving mission, the generalization performances on various tracks with different curvatures and directions are compared in Fig. 5.4. Simulation results for all the possible track settings are displayed along the z-axis where different curvature and direction values are set. Each of Fig. 5.4a, Fig. 5.4b, and Fig. 5.4c respectively shows the generalization results of PDMPs with four and 64 demonstrations and seg-PDMPs with four demonstrations. In Figs. 5.5c-5.5b, the generalization results for four different environments are presented. When the same set of demonstrations are provided to PDMPs and seg-PDMPs ($M = 4$), seg-PDMPs show better generalization performance. It is expected since at least 64 demonstrations need to be provided to PDMPs for full generalization. Even 64 demonstrations are provided for PDMPs, the generalization performance is poor regarding the S- and U-shaped tracks which are located much farther than the P-shaped track from starting point. Compared with PDMP results, seg-PDMPs keep the generalization performance for both S- and U-shaped tracks. This is because when extracting the attractor basis by taking SVD in PDMPs, the values related to P-track are relatively dom-



(a) (b)



(c) (d)

Figure 5.5: Reproduction results from two PDMPs and seg-PDMPs for (a) 'a', (b) 'b', (c) 'c', and (d) 'd' environment in Fig. 5.4. Each pink, purple, and navy line indicates the result of PDMPs with four and 64 demonstrations, and seg-PDMPs with four demonstrations, respectively. Thick grey line shows the differed environmental settings. Red and black circles are initial and goal configurations.

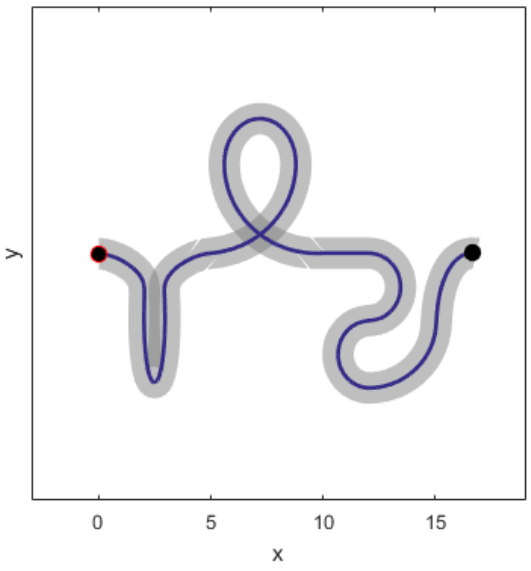
	DMP	PDMPs		Seg-PDMPs
of demonstrations	-	4	64	4
Initial/final offset	Task constraints may not be satisfied by stretching or shrinking to overcom offset			-
Generalization	unable	unable	able	able
Re-combination	unable	unable		able

Table 5.2: Algorithm summary of single DMP, PDMPs, and seg-PDMPs.

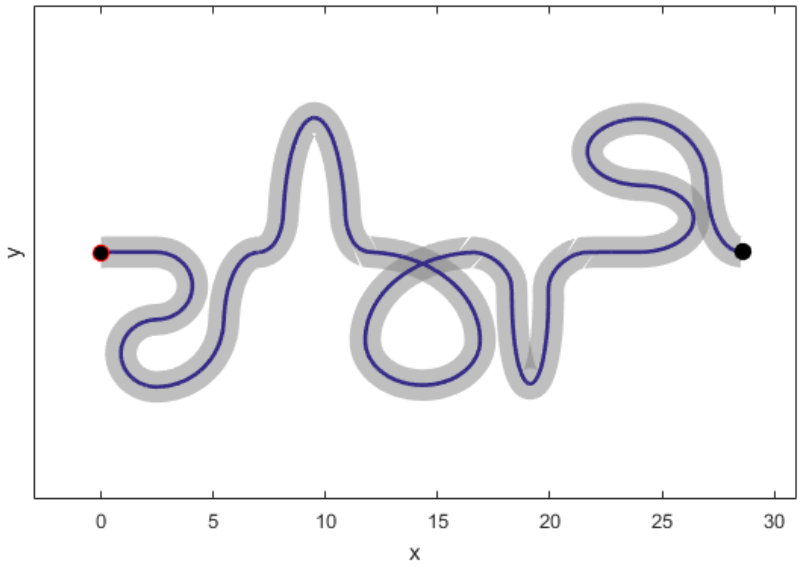
inant compared to other tracks. The PDMPs with four demonstrations show invalid results in some environments as shown in Figs. 5.5c-5.5d.

5.5.3 Recombination

Unlike the general PDMPs, seg-PDMPs has the structure where multiple PDMPs are configured on each sub-task individually and called autonomously according to the situations. Therefore, depending on the situation, the robot can recall the sub-task already done, so the overall mission composition may vary. Figs. 5.6a-5.6b show the results of different combinations of driving course compared to the demonstrations. Table 5.2 summarizes the previous simulation results.



(a)



(b)

Figure 5.6: Course reconstruction results using seg-PDMPs algorithm. In each simulation, the different settings are given from the learned demonstrations including (a) the course order and initial position, and (b) the course order, initial position, and final position, and the number of unit courses. Thick grey line shows the differed environmental settings.

6

Experimental Validation and Results

In this section, the proposed PDMPs algorithm and seg-PDMPs are validated with experiments of mobile manipulators with two types of vehicles. The first two scenarios regarding cooperative aerial transportation demonstrate the excellence of the proposed technique in terms of quick computation, generations of optimum movement, and safety assurance for the movement. The last two scenario deal with two mobile hang-dry missions using ground vehicles and show how the seg-PDMPs are applied to perform complex missions. The experimental setups and control architectures are presented in section 2.2 for both types of mobile manipulators. This part focuses on checking the reliability of the resulted motions and completeness of the missions in real environments.

6.1 Cooperative Aerial Transportation

The scenarios for aerial transportation have already been validated in sections 3.3 and 4.3 through the simulation results. Here, experiments verify the aerial manipulators also complete the transportation mission through the proposed PDMPs with real-time feedback. Since robot manipulation is affected by various kinds of perturbations during practical

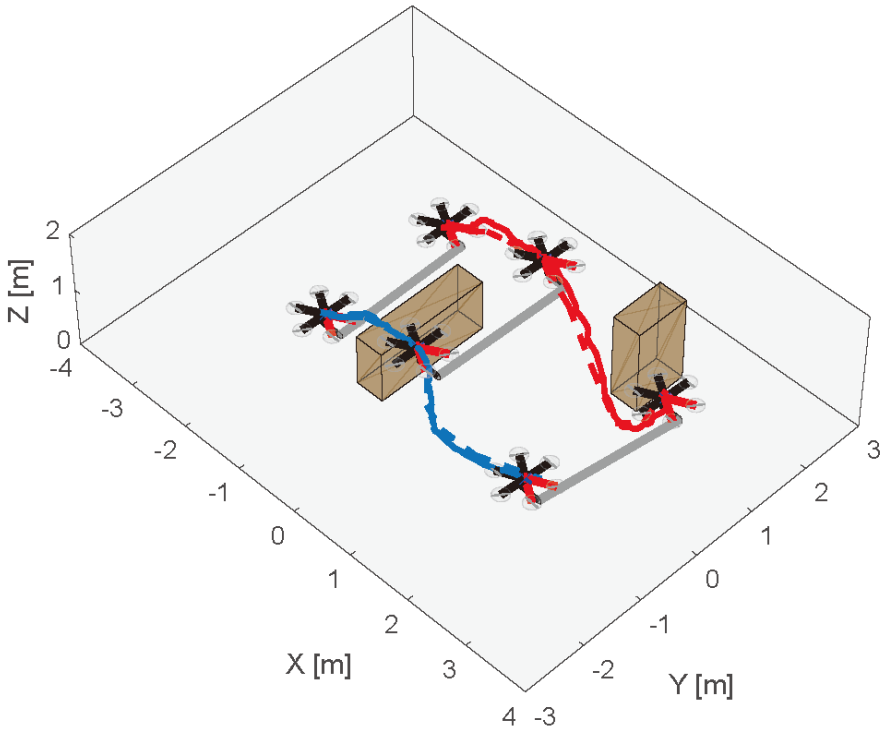
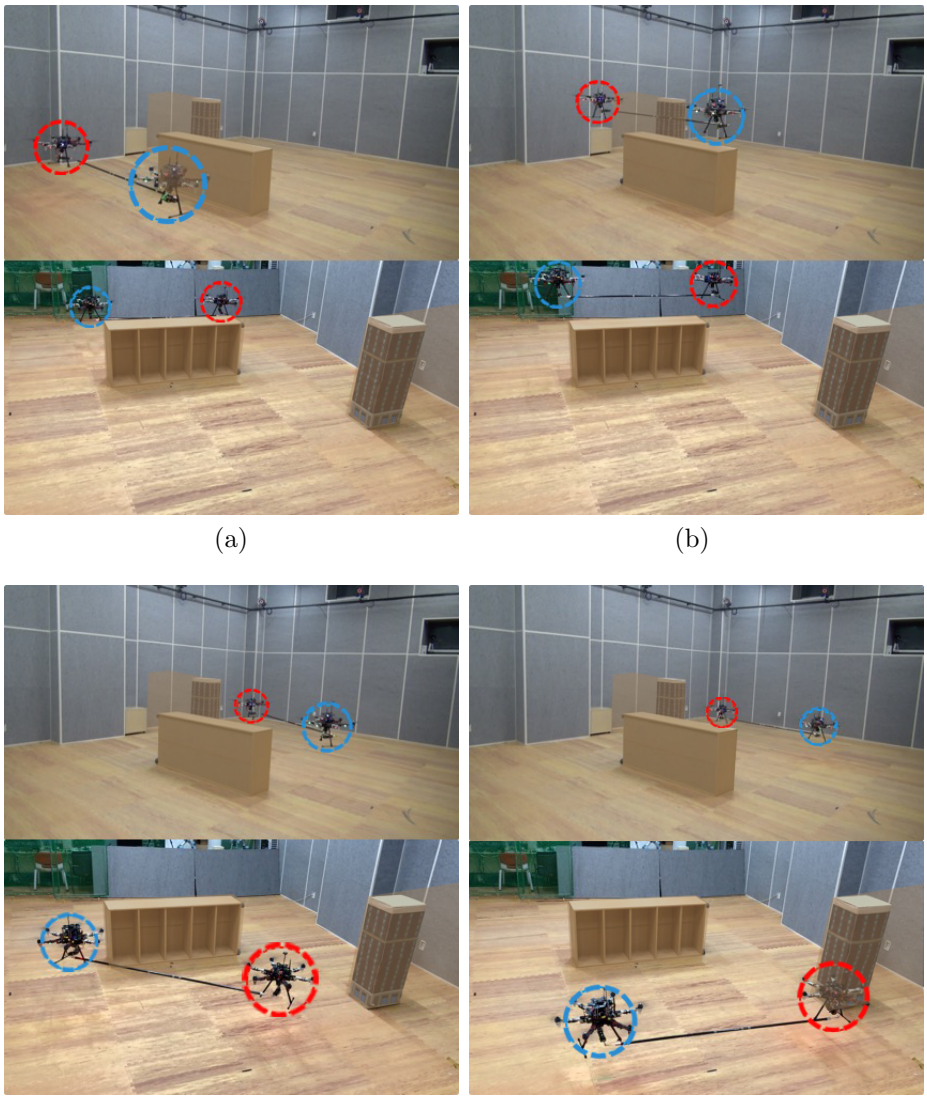


Figure 6.1: Resulting trajectories recorded from experiments for the first scenario. Desired trajectories are depicted in dashed lines, and actual trajectories are depicted in solid lines. Each red and blue line shows the body trajectory of the left and right aerial manipulator, respectively.

operation, this experimental validations show the effectiveness of PDMPs robustness to the perturbations.

As described above, the scenario settings are same as sections 3.3 and 4.3. The experimental results are reported in Figs. 6.1-6.6. The first scenario regards section 3.3 while the second scenario is related to section 4.3. The resulting trajectories from aerial manipulators for are shown in Figs. 6.1 and 6.4. Figs. 6.2 and 6.5 show the snapshots of the experiments. The time histories of the state variables are listed in Figs. 6.3 and 6.6. In the second scenario, the additional cylindrical obstacle is located near the starting point, which is never considered in the first scenario. As described in section 4.3, demonstrations are updated by applying the proposed safety guaranteeing process in section 4, and the aerial manipulators successfully avoid the obstacle.



(a) (b)

(c) (d)

Figure 6.2: Snapshots of the experiment for the first scenario. (a) starting operation, (b) avoiding first obstacle, (c) avoiding newly faced obstacle, and (d) terminating operation. Each red and red blue indicates left and right aerial manipulators, respectively.

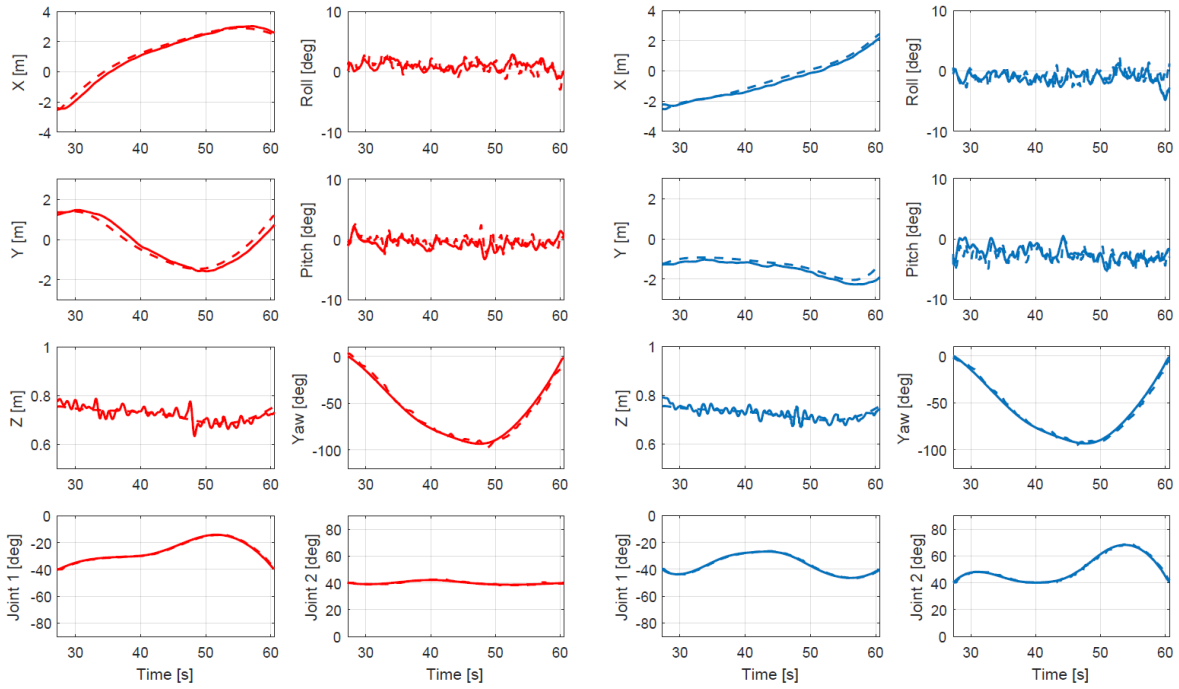


Figure 6.3: Time histories of the state variables of the aerial manipulators for the first scenario. (a) Position, attitude, and joint angles of the left aerial manipulator, and (b) the right manipulator. Desired trajectories are shown in dashed lines, and actual trajectories in solid lines.

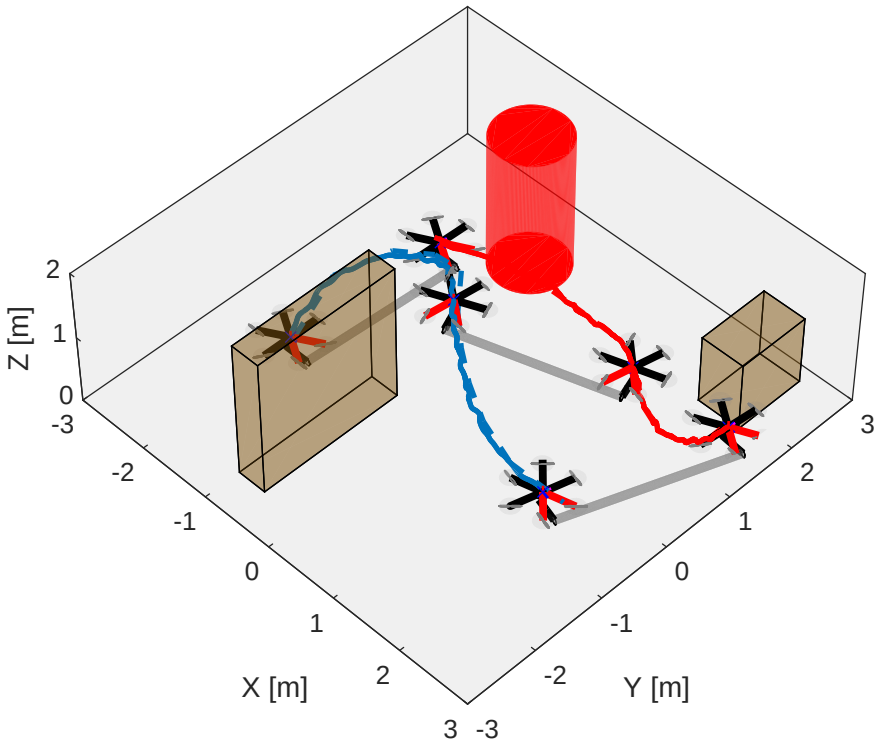
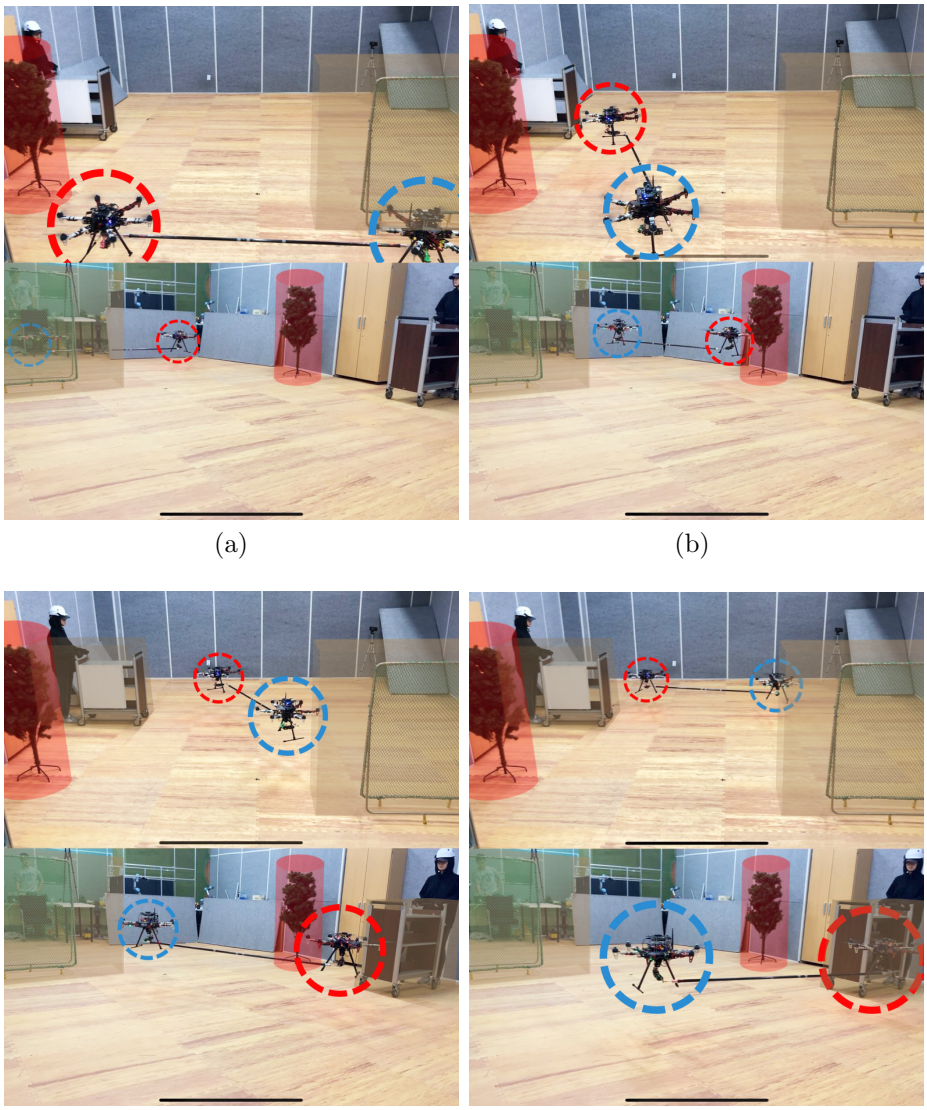


Figure 6.4: Resulting trajectories recorded from experiments for the second scenario. The cylindrical obstacle is added from the first scenario and other obstacle settings are also changed. Desired trajectories are depicted in dashed lines, and actual trajectories are depicted in solid lines. Each red and blue line shows the body trajectory of the left and right aerial manipulator, respectively.



(a) (b)

(c) (d)

Figure 6.5: Snapshots of the experiment for the second scenario. Snapshots at (a) starting operation, (b) avoiding first obstacle, (c) avoiding newly faced obstacle, and (d) terminating operation. Each red and red blue indicates left and right aerial manipulators, respectively.

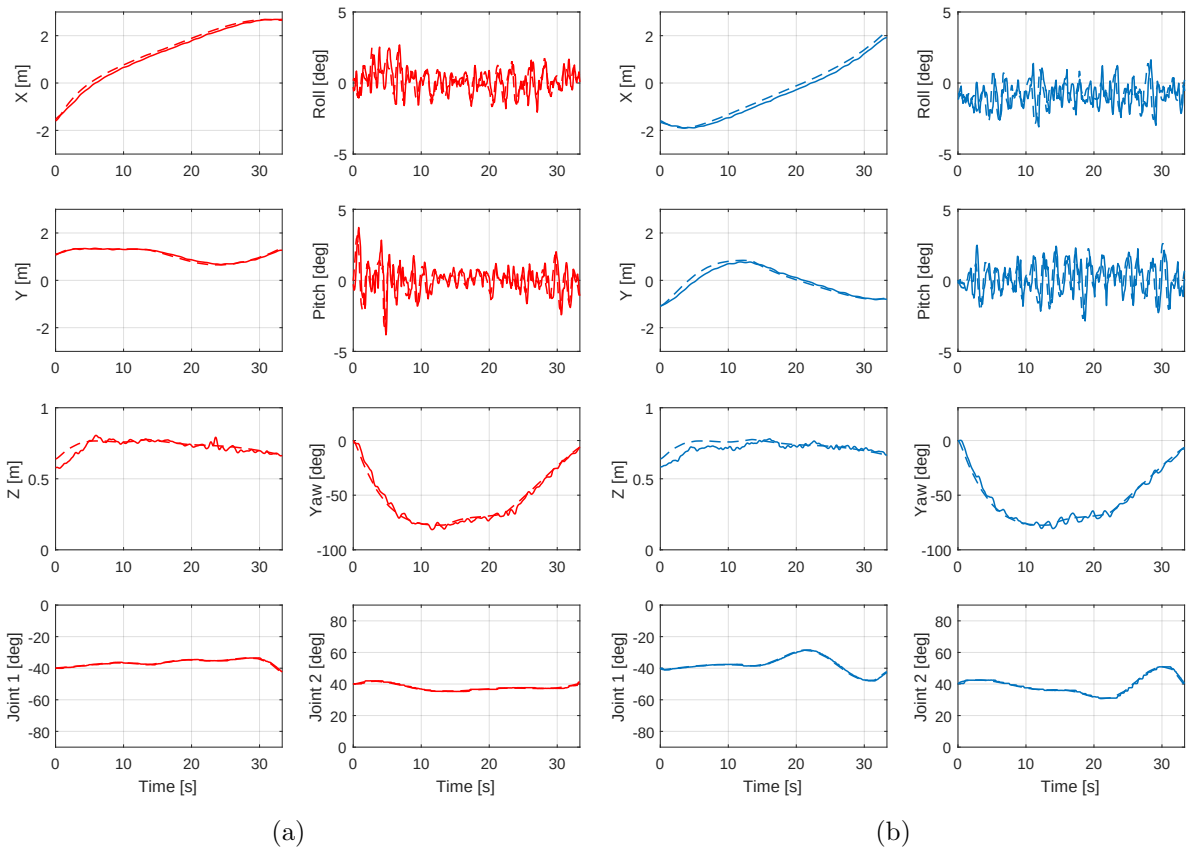


Figure 6.6: Time histories of the state variables of the aerial manipulators for the second scenario. (a) Position, attitude, and joint angles of the left aerial manipulator, and (b) the right manipulator. Desired trajectories are shown in dashed lines, and actual trajectories in solid lines.

6.2 Cooperative Mobile Hang-dry Mission

The experiments on hang-dry mission regard seg-PDMPs, which is the extension PDMPs algorithm in section 5. Two mobile manipulators are employed where each manipulator consisting of the two-wheeled mobile robots and four-DoF robotic arms. The detailed system descriptions and hardware setups are already presented in sections 2.2 and 2.3. Hang-dry missions for cooperative mobile manipulation system is also shown in Fig. 1.1. In the manipulation scenario, first, two mobile manipulators start to move towards a pile of clothes from different positions. Until they grasp each tip of a given object (cloth), they maneuver independently. The transportation process begins after both robots grab the cloth. In the process of transportation, the robots are constrained to each other by holding the cloth cooperatively, and sometimes they encounter obstacles to avoid. When they get near hanger, they stretch out the cloth and move to the each opposite side of the hanger. Finally, they pull down both ends of the cloth and finish the task. Thus, during the hang-dry mission, the robots perform sub-tasks including motions like grasping, stretching and releasing.

6.2.1 Demonstrations

There are various ways to provide demonstrations and they are affected by factors like complexity of the system or task [28]. While most of the work for a few demonstrations have made use of human demonstrators, the simulated planners is used to repeatably compute the demonstrations with a different environment. Moreover, objective of this work deals with complex motion skills which are typically difficult to control, the simulated planner would benefit various environments. The computational time to generate demonstrations does not need to be a real-time scale as the process of organizing a demonstration set is an offline process in PDMPs. Therefore, in order to generalize the efficient task execution, the trajectories are obtained from optimal motion planner. As the optimization technique, iterative linear quadratic regulator (iLQR) is applied for hang-dry mission. Unlike RRT* in cooperative aerial transportation scenario, iLQR can formulate via-points for hang-dry

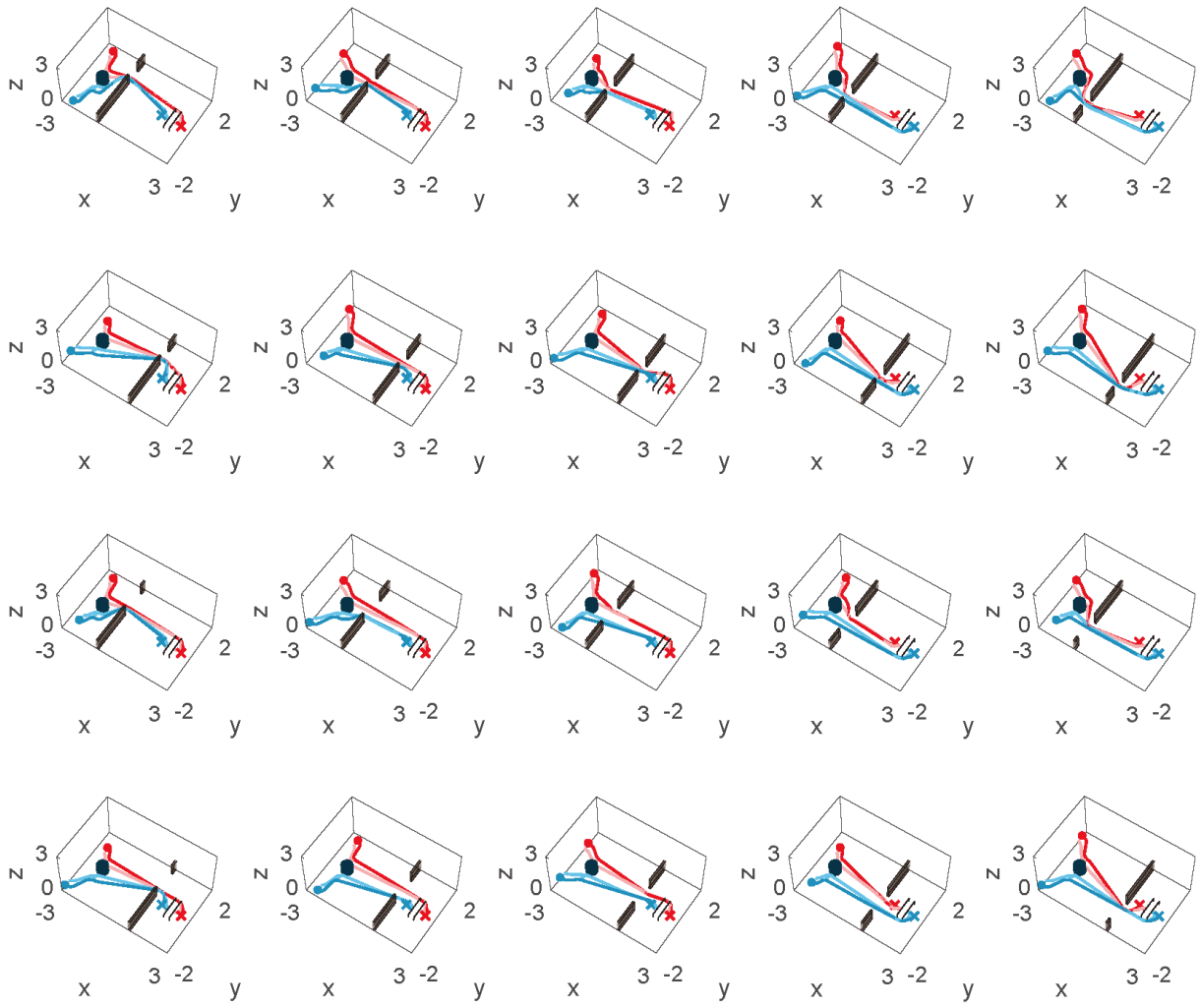


Figure 6.7: Twenty demonstrations computed from iLQR. The performance index of the trajectories are the travel length through the operation. The starting positions of robots and the locations of freeway against obstacles are different in each environment. Each red and blue color indicates that the trajectories are concerned with the left and the right mobile manipulator, respectively. The solid lines are the body trajectories of robots while light colored lines are end-effector trajectories.

M_1	M_2	M_3	M_4	M_5	M_6
16	2	20	20	4	4

Table 6.1: The number of required demonstrations in sub-tasks for hang-dry mission.

mission. As a result, twenty different environmental settings are selected to fully describe general environment for each sub-tasks. Hang-dry mission consists of six sub-tasks and the corresponding number of required demonstrations in each task are in Table. 6.1. Be notices that if PDMPs are applied instead of the seg-PDMPs, the total number of required demonstrations will be $16 \times 2 \times 20 \times 20 \times 4 \times 4 = 204,800$ here. In those environments, environmental settings are various with different starting positions, locations and dimensions of obstacles, heights of the first position of cloth or hanger. Fig. 6.7 shows the demonstrations of iLQR. The segmented demonstrations corresponding to each phase is shown in Fig. 6.8.

6.2.2 Simulation Validation

In order to show the seg-PDMPs allow repeated sub-tasks, two different environments are set for simulation by giving sequential one and three obstacles while the demonstrations only include one obstacle. Here, the obstacles are never given as demonstration settings before. The simulation results are shown in Fig. 6.9. Each green and yellow area shows starting and goal points in the entire task execution, respectively. The red lines show the trajectories of the left mobile manipulator while the blue lines show the trajectories of the right one. The phase bars are shown at the bottom of the figures indicating which phase is operated along the time. In the seg-PDMPs, repeatable tasks such as obstacle avoidance ($p = 3, 4$) can be automatically computed if the agents determine the task should be operated again based on the current situation. Thus, more than one obstacle can be deployed in the online seg-PDMPs. All the simulations also have different starting points and obstacle positions compared to demonstrations. From the results, the required sub-tasks are exactly executed in sequence at proper moments during the task.

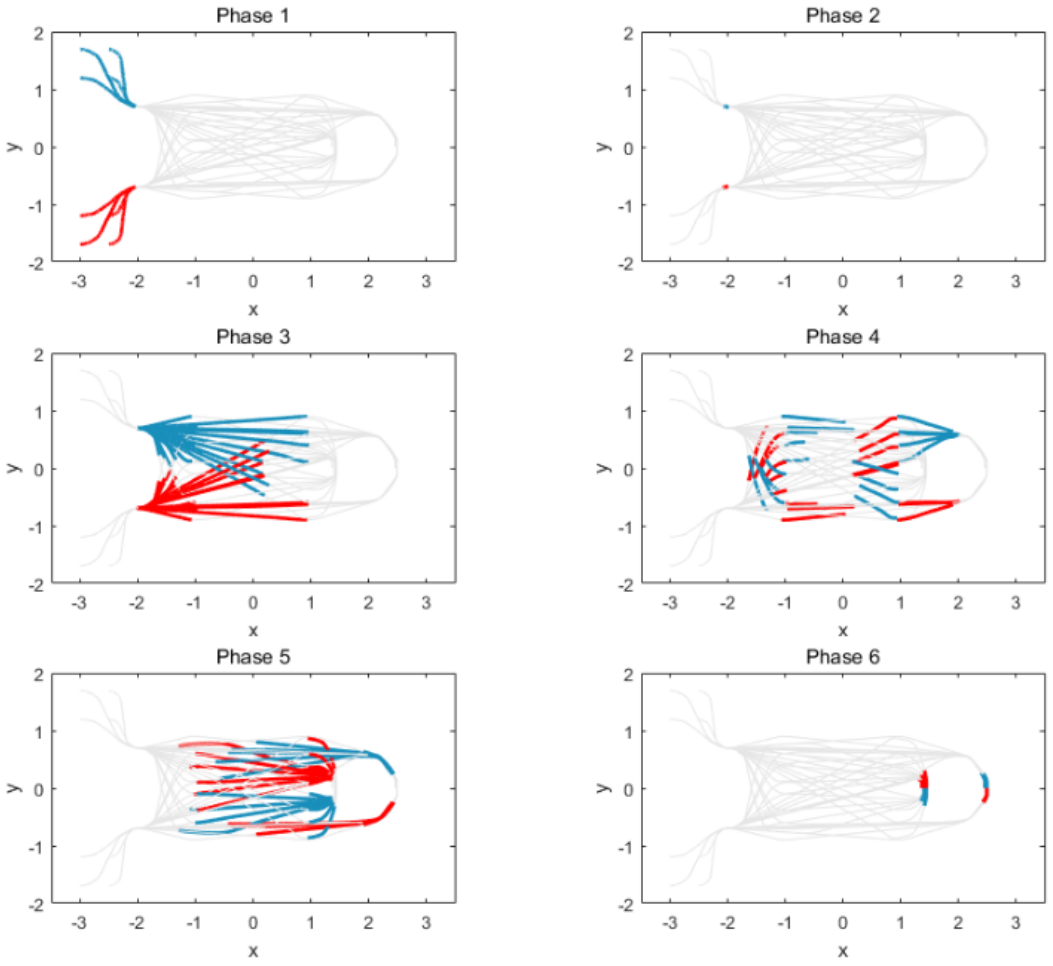
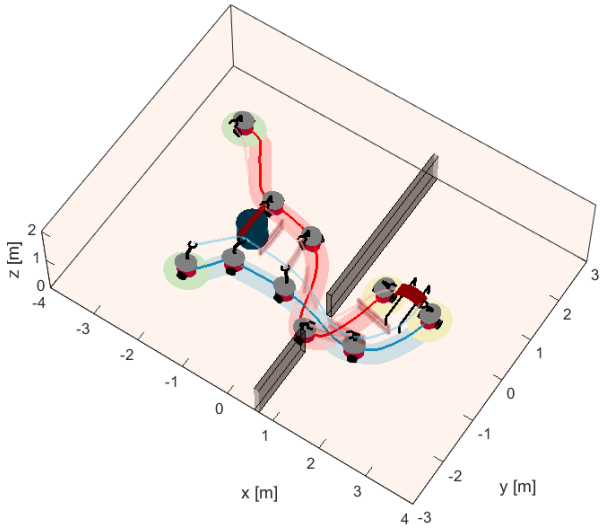
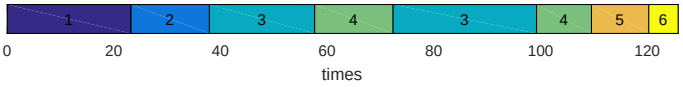


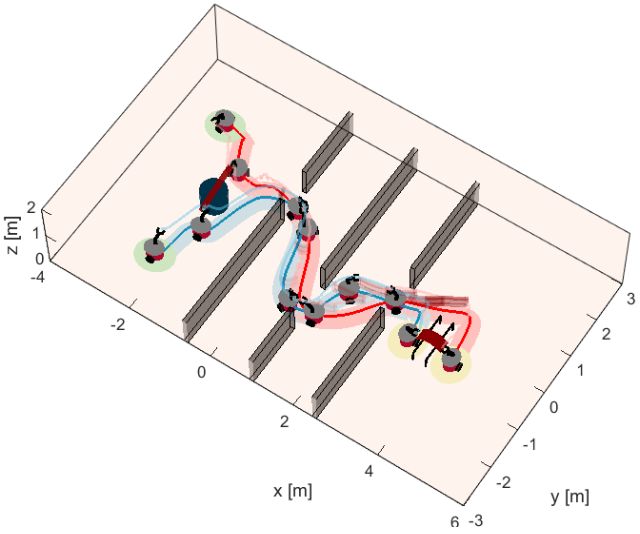
Figure 6.8: Six phases consist of the hang-dry mission in Fig. 6.7. Each red and blue line is the trajectories of the left and the right mobile manipulator, respectively. The grey lines show the rest trajectories in each phase.



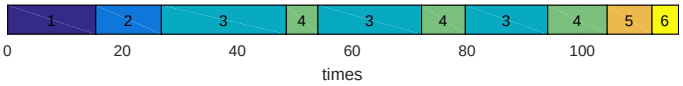
(a)



(b)



(c)



(d)

Figure 6.9: Computed trajectories of two mobile manipulators with differed environmental settings from demonstrations. (a,b) Single obstacle is considered and (c,d) three obstacles sequentially appear during transportation.

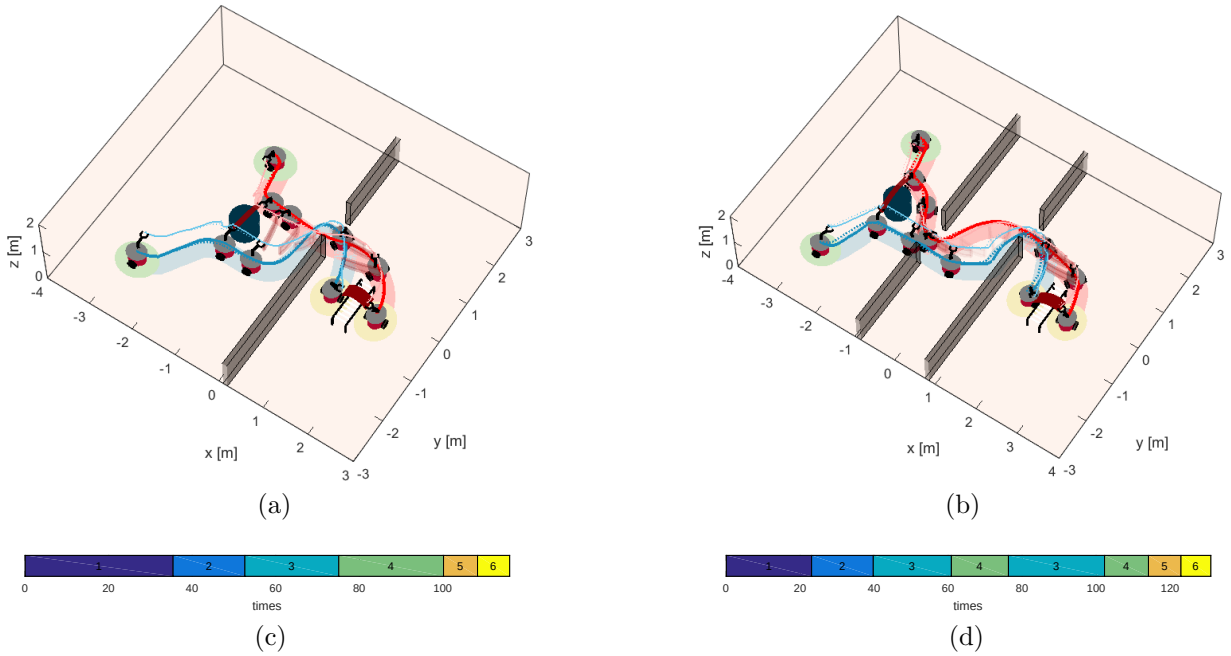
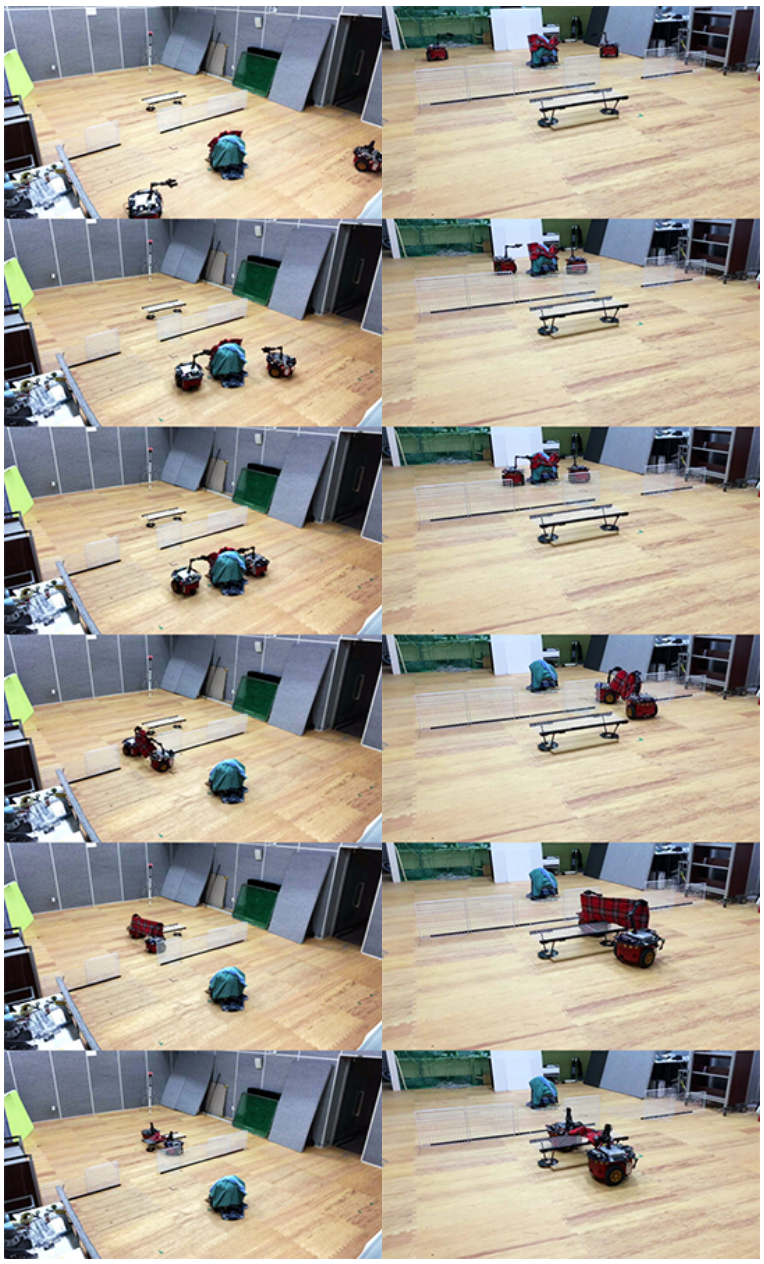
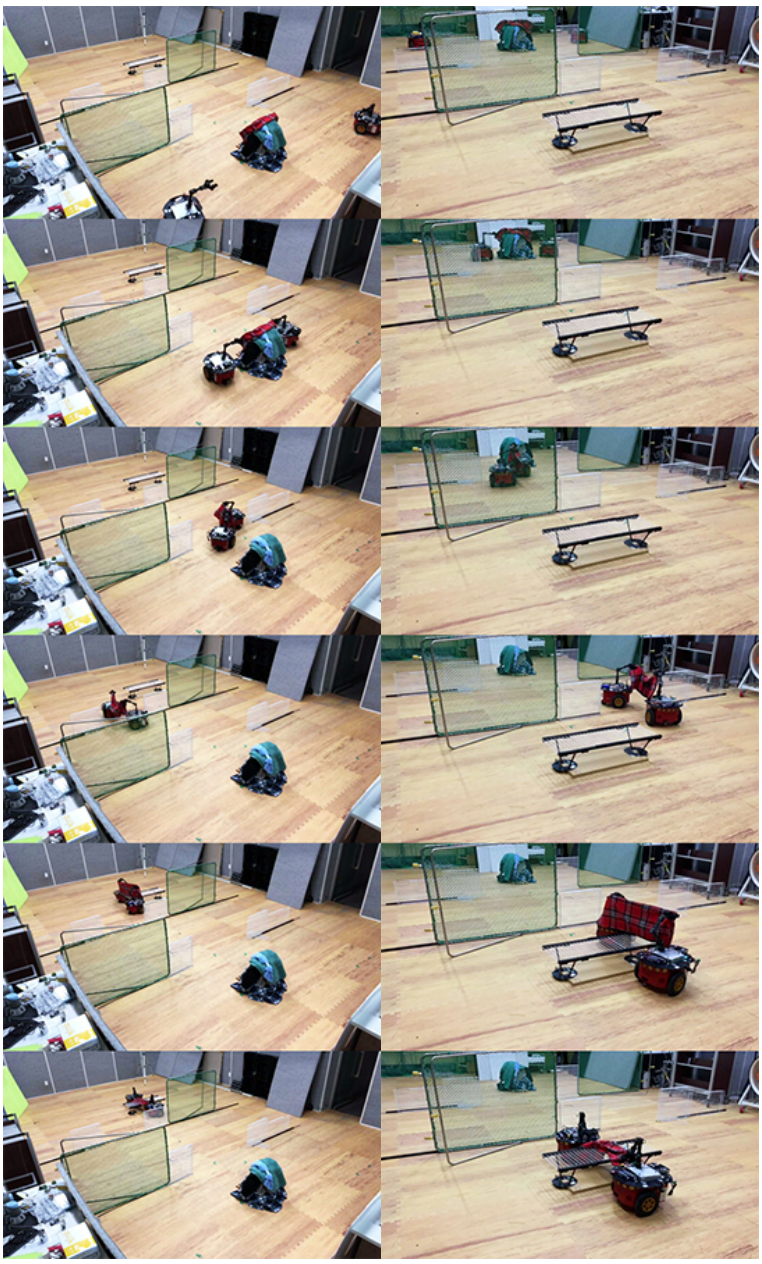


Figure 6.10: Resulting trajectory recorded from the (a,c) first and (b,d) second experiments. Two mobile manipulators move from the starting point (green circled area) to the hanger (yellow circled area). Each red and blue line shows the trajectory of the left and right mobile manipulator, respectively. The dark lines of each color denote the trajectories of the body position of mobile manipulators while the light lines represent the position of end-effectors. Each light red and light blue area shows the actual trajectories with the left and right platform dimension, respectively. The solid line indicates the recorded trajectories and dashed line indicates the desired trajectories computed from the proposed seg-PDMPs.



(a) (b)

Figure 6.11: Snapshots during the first experiment taken from two different perspectives (a) and (b). The robots start moving towards the pile of clothes (0 sec). The second robot first reaches the pile and grabs the fabric (red checkered blanket) located on top of the pile (20 sec). After that, the first robot also arrives in the pile and grabs the opposite end of the fabric (30 sec). The robots avoid obstacle (70 sec) while cooperatively holding the fabric (70 sec). When they reach the hanger, they stretch the fabric (90 sec) and hang it on the hanger (116 sec).



(a) (b)

Figure 6.12: Snapshots during the second experiment, taken from two perspectives (a) and (b). The robots start moving towards the pile of clothes (0 sec). Both robots reach the pile simultaneously and grabbing the each tip of the fabric (red checkered blanket) at the top of the pile (20 sec), respectively. In the second experiment, there are two obstacles. The robots avoid the first obstacle (70 sec) and the second obstacle (90 sec) while cooperatively holding the fabric. When they reach the hanger, they stretch the fabric (110 sec) and hang it on the hanger (131 sec).

<table>
<tr><td rowspan="2"></td><td rowspan="2">Phase</td><td rowspan="2">N</td><td rowspan="2" colspan="2"># of demos M</td><td rowspan="2">L</td><td rowspan="2" colspan="2">Time length</td><td colspan="4">Computational time</td></tr>
<tr><td>Getting demos (offline)</td><td colspan="2">Learning (offline)</td><td colspan="2">Reproduction (online)</td></tr>
<tr><td>Cooperative aerial transportation</td><td>Single</td><td>7</td><td colspan="2">24</td><td>25</td><td colspan="2">1001</td><td>602,342 s (RRT*)</td><td colspan="2">2.63 ms</td><td colspan="2">3.91 ms</td></tr>
<tr><td rowspan="6">Cooperative hang-dry mission</td><td>1</td><td rowspan="6">14</td><td rowspan="6">20</td><td>16</td><td>36</td><td rowspan="6">1855</td><td>230</td><td rowspan="6">156.24 s (iLQR)</td><td rowspan="6">44.57 ms</td><td>5.88 ms</td><td rowspan="6">16.55 ms</td><td>1.04 ms</td></tr>
<tr><td>2</td><td>2</td><td>4</td><td>50</td><td>1.25 ms</td><td>0.54 ms</td></tr>
<tr><td>3</td><td>20</td><td>58</td><td>750</td><td>12.29 ms</td><td>4.96 ms</td></tr>
<tr><td>4</td><td>20</td><td>79</td><td>525</td><td>12.50 ms</td><td>5.25 ms</td></tr>
<tr><td>5</td><td>4</td><td>53</td><td>125</td><td>9.17 ms</td><td>3.16 ms</td></tr>
<tr><td>6</td><td>4</td><td>43</td><td>175</td><td>4.47 ms</td><td>1.60 ms</td></tr>
</table>

Table 6.2: Detailed computational time.

6.2.3 Experimental Results

This section shows two online experiments involving one or two obstacles which are previously unknown. Similar to the simulations Fig. 6.9, the different positions of the obstacles and the robots are given from the prescribed demonstrations. During the task, the ground robots are controlled to follow calculated trajectories. From the VICON system, current environmental information and the robot states are transferred to VICON computer. This information is also used for seg-PDMPs to calculate trajectories. The calculated trajectories are transmitted from ground station to the on-board computer via wireless communication.

The experimental results are reported in Figs. 6.10-6.12. Each Fig. 6.10a and Fig. 6.10b shows the resulting trajectories recorded from the first and second experiments, respectively. Figs. 6.10c and 6.10d demonstrate the phase state along the time. Since there are two obstacles in the second scenario, the phases $p = 3, 4$ involving an obstacle avoidance task are repeated again during the task. Figs. 6.13-6.14 demonstrates the time histories of the state variables of the two agents. Fig. 6.13 shows the first experiment and Fig. 6.14 indicates the second experiment. Each Figs. 6.11-6.12 shows the snapshots during the first and second experiments, respectively. In both environments, the agents approach the pile of clothes and pick up the cloth(pink colored cloth) whenever they arrive at the pile. During the task, they encounter the obstacles once or twice. Based on the proposed framework, they generate the motion which can be compatible with the current environment. Be noticed that both environments have different environmental settings such as positions of starting points or

obstacles to show that the proposed PDMPs framework can allow any environments which are never given previously as one of the demonstration set.

The detailed computational times for experiments including 6.1 and 6.2 are listed in Table. 6.2

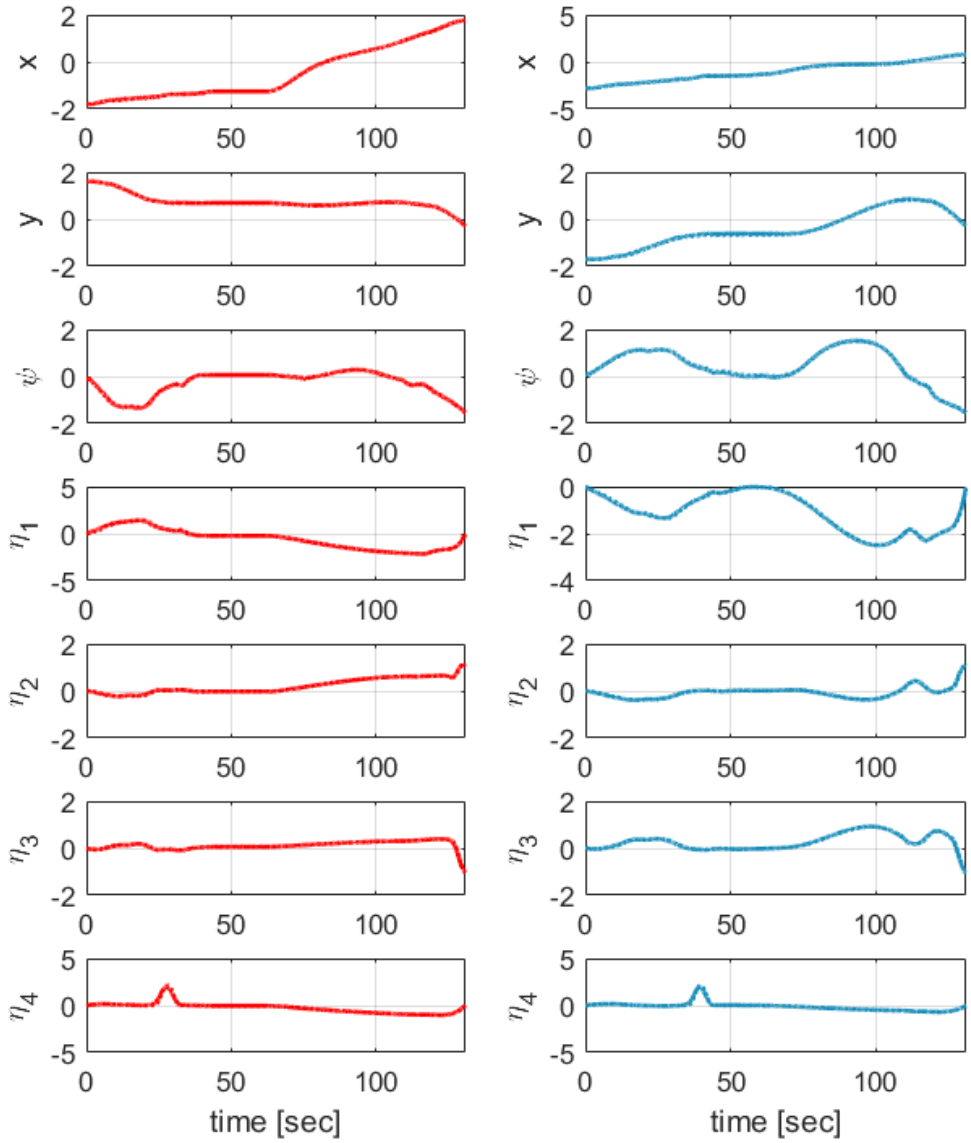


Figure 6.13: Time histories of the state variables of the mobile manipulators for the first experiment. Desired trajectories are shown in dashed lines, and actual trajectories in solid lines. Each red and blue line indicates the left and the right mobile manipulator, respectively.

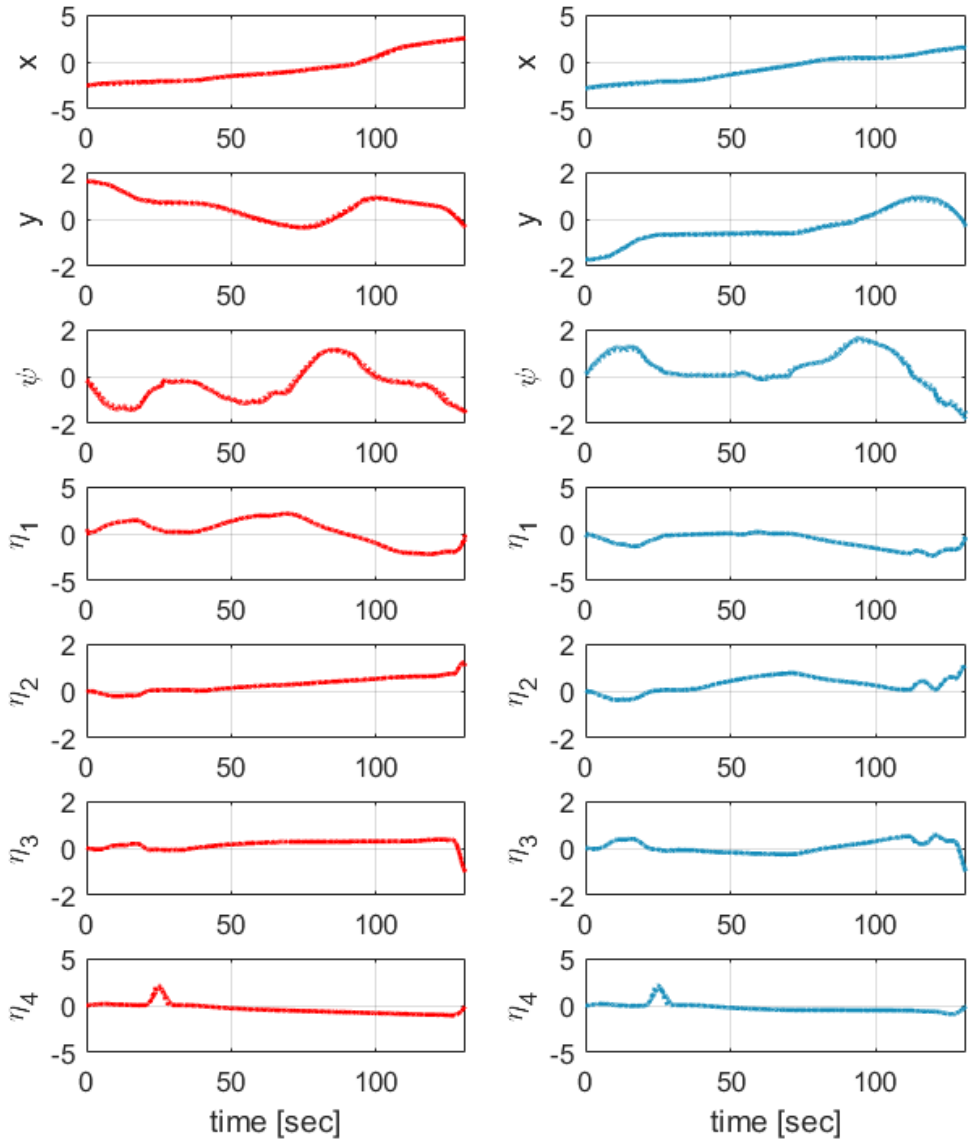


Figure 6.14: Time histories of the state variables of the mobile manipulators for the second experiment. Desired trajectories are shown in dashed lines, and actual trajectories in solid lines. Each red and blue line indicates the left and the right mobile manipulator, respectively.

7

Conclusions

Dynamic movement primitives (DMPs) are well known as a promising method to specify the robot movement particularly for robot manipulations. They guarantee quick computations and robust representation against the perturbations. Parametric DMPs (PDMPs) take advantages of these DMPs and furthermore, they provide parameterized representation from multiple demonstrations. The new representation of motion is possible in PDMPs by selecting the parameters. In this dissertation, the relevant research of PDMPs were addressed regarding (i) generalization of motion, (ii) safety guarantee, and (iii) managing the number of demonstrations.

- First topic was the generalization process in PDMPs. By learning the relationship between known environments and corresponding motions using Gaussian process regression(GPR), the proposed framework can serve as a real-time and optimal planner by learning multiple optimal movements. This dissertation provided the simulation results for aerial transportation scenarios within obstacle environments. By providing the optimal demonstrations of RRT*, the algorithm computed the motion emulating optimal-avoidance for a new environment. This indicates that the trade-off issue be-

tween the motion optimality and the computational time in the conventional motion planning problem is effectively relieved by using the proposed framework.

- Second, the safety of motion was addressed. This work suggested the process for calculating the safety criterion of the PDMP internal parameter value. The safety criterion reflected the new behavior computed through the generalization process, as well as the individual motion safety of the demonstration set. The demonstrations causing unsafe behavior were identified and removed through this safety criterion. Further, demolished demonstrations were replaced by proven demonstrations upon the safety criterion. Thus, the representation performance kept the previous level. This work also was utilized to reuse demonstrations when the static environmental settings have changed where all demonstrations should have been replaced.

- Third, this dissertation also presented an extension approach, seg-PDMP, which is beneficial for reducing the number of required demonstrations within the proposed framework. Especially when a single assignment consists of multiple unit sub-tasks and requires numerous demonstrations to generalize them, this approach showed great performance in reducing the number of demonstrations. Through this work, the whole trajectories were segmented into multiple sub-tasks representing unit motions. Multiple PDMPs were formed independently for the correlated-segments called phases. The phase-decision process allowed multiple PDMPs to be configured within an integrated framework. Gaussian process regression (GPR) was applied to obtain execution time and regional goal configuration within each phase from environmental variables.

- Finally, the proposed algorithm and its extension were validated in the experiments of two types of mobile manipulators. The first two scenarios showed the effectiveness of the proposed framework in terms of quick computation, generations of optimum movement, and safety assurance for the movement. The last scenario dealt with two mobile manipulations using ground vehicles and showed how the proposed extension of the proposed algorithm is applied to perform complex assignments. From this

experimental validation, the superiority of this research is evaluated by reducing over 200,000 number of demonstrations into 20 demonstrations.

In general, manipulation points can be assigned during operation and must be corrected in real time. In Seg-PDMPs, via-points can be considered unlike in PDMPs, but there is a limitation that they can be considered only as regional goals of each unit sub-task. Therefore, the possible future works of this study will be the development of a framework that can flexibly consider via-points.

References

[1] H. Kim, H. Lee, S. Choi, Y.-k. Noh, and H. J. Kim, "Motion planning with movement primitives for cooperative aerial transportation in obstacle environment," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2328–2334.

[2] H. Kim, H. Seo, C. Y. Son, H. Lee, S. Kim, and H. J. Kim, "Cooperation in the air : A learning-based approach for the efficient motion planning of aerial manipulators," *IEEE Robotics & Automation Magazine*, vol. 25, no. 4, pp. 76–85, 2018.

[3] D. Mellinger, M. Shomin, N. Michael, and V. Kumar, "Cooperative grasping and transport using multiple quadrotors," in *Distributed autonomous robotic systems*. Springer, 2013, pp. 545–558.

[4] K. Sreenath and V. Kumar, "Dynamics, control and planning for cooperative manipulation of payloads suspended by cables from multiple quadrotor robots," in *Proceedings of Robotics: Science and Systems*, 2013.

[5] C. Y. Son, D. Jang, H. Seo, T. Kim, H. Lee, and H. J. Kim, "Real-time optimal planning and model predictive control of a multi-rotor with a suspended load," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5665–5671.

[6] C. Y. Son, H. Seo, D. Jang, and H. J. Kim, "Real-time optimal trajectory generation and control of a multi-rotor with a suspended load for obstacle avoidance," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1915–1922, 2020.

[7] F. Caccavale, G. Giglio, G. Muscio, and F. Pierri, "Cooperative impedance control for multiple uavs with a robotic arm," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2015, pp. 2366–2371.

[8] H. Lee, H. Kim, and H. J. Kim, "Planning and control for collision-free cooperative aerial transportation," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 1, pp. 189–201, 2018.

[9] H. Yang and D. Lee, "Hierarchical cooperative control framework of multiple quadrotor-manipulator systems," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4656–4662.

[10] S. Kim, S. Choi, and H. J. Kim, "Aerial manipulation using a quadrotor with a two dof robotic arm," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 4990–4995.

[11] G. Heredia, A. Jimenez-Cano, I. Sanchez, D. Llorente, V. Vega, J. Braga, J. Acosta, and A. Ollero, "Control of a multirotor outdoor aerial manipulator," in *Intelligent Robots and Systems (IROS), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 3417–3422.

[12] V. Lippiello, J. Cacace, A. Santamaria-Navarro, J. Andrade-Cetto, M. A. Trujillo, Y. R. Esteves, and A. Viguria, "Hybrid visual servoing with hierarchical task composition for aerial manipulation," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 259–266, 2016.

[13] S. Kim, H. Seo, J. Shin, and H. J. Kim, "Cooperative aerial manipulation using multirotors with multi-dof robotic arms," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 2, pp. 702–713, 2018.

[14] S. Kim, S. Choi, H. Kim, J. Shin, H. Shim, and H. J. Kim, "Robust control of an equipment-added multirotor using disturbance observer," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 4, pp. 1524–1531, 2017.

[15] H. Seo, S. Kim, and H. J. Kim, "Locally optimal trajectory planning for aerial manipulation in constrained environments," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 1719–1724.

[16] G. Tartaglione, E. D'Amato, M. Ariola, P. S. Rossi, and T. A. Johansen, "Model predictive control for a multi-body slung-load system," *Robotics and Autonomous Systems*, vol. 92, pp. 1–11, 2017.

[17] G. Garimella and M. Kobilarov, "Towards model-predictive control for aerial pick-and-place," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4692–4697.

[18] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[19] O. Brock and L. E. Kavraki, "Decomposition-based motion planning: A framework for real-time motion planning in high-dimensional configuration spaces," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, vol. 2. IEEE, 2001, pp. 1469–1474.

[20] H. Najjaran and A. Goldenberg, "Real-time motion planning of an autonomous mobile manipulator using a fuzzy adaptive kalman filter," *Robotics and Autonomous Systems*, vol. 55, no. 2, pp. 96–106, 2007.

[21] H. Kim, H. Seo, J. Kim, and H. J. Kim, "Sampling-based motion planning for aerial pick-and-place," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 7402–7408.

[22] S. Calinon, F. D'halluin, D. G. Caldwell, and A. G. Billard, "Handling of multiple constraints and motion alternatives in a robot programming by demonstration framework," in *2009 9th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2009, pp. 582–588.

[23] D. Kulić, C. Ott, D. Lee, J. Ishikawa, and Y. Nakamura, "Incremental learning of full body motion primitives and their sequencing through human motion observation," *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 330–345, 2012.

[24] M. Saveriano, S.-i. An, and D. Lee, "Incremental kinesthetic teaching of end-effector and null-space motion primitives," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 3570–3575.

[25] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[26] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto, "Robot learning from demonstration by constructing skill trees," *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 360–375, 2012.

[27] C. G. Atkeson and S. Schaal, "Robot learning from demonstration," in *ICML*, vol. 97. Citeseer, 1997, pp. 12–20.

[28] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[29] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 763–768.

[30] B. Huang, M. Ye, Y. Hu, A. Vandini, S.-L. Lee, and G.-Z. Yang, "A multirobot cooperation framework for sewing personalized stent grafts," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 4, pp. 1776–1785, 2017.

[31] T. Matsubara, S.-H. Hyon, and J. Morimoto, "Learning parametric dynamic movement primitives from multiple demonstrations," *Neural Networks*, vol. 24, no. 5, pp. 493–500, 2011.

[32] D. Bruno, S. Calinon, and D. G. Caldwell, "Learning autonomous behaviours for the body of a flexible surgical robot," *Autonomous Robots*, vol. 41, no. 2, pp. 333–347, 2017.

[33] C. E. Reiley, E. Plaku, and G. D. Hager, "Motion generation of robotic surgical tasks: Learning from expert demonstrations," in *2010 Annual international conference of the IEEE engineering in medicine and biology*. IEEE, 2010, pp. 967–970.

[34] H. Kim, H. Seo, S. Choi, C. J. Tomlin, and H. J. Kim, "Incorporating safety into parametric dynamic movement primitives," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2260–2267, 2019.

[35] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 1168–1175.

[36] S. Lee, H. Seo, S. Choi, H. Kim, and H. J. Kim, "Smooth trajectory generation for soft catching a flying object with an aerial vehicle," in *2017 11th Asian Control Conference (ASCC)*. IEEE, 2017, pp. 790–794.

[37] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *Ieee access*, vol. 2, pp. 56–77, 2014.

[38] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., 1998.

[39] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Survey: Robot programming by demonstration," *Handbook of robotics*, vol. 59, no. BOOK_CHAP, 2008.

[40] G. Goretkin, A. Perez, R. Platt, and G. Konidaris, "Optimal sampling-based planning for linear-quadratic kinodynamic systems," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 2429–2436.

[41] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Learning movement primitives," in *Robotics research. the eleventh international symposium*. Springer, 2005, pp. 561–572.

[42] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," Tech. Rep., 2002.

[43] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural computation*, vol. 25, no. 2, pp. 328–373, 2013.

[44] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *2002 IEEE International Conference on Robotics and Automation*. IEEE, 2002, pp. 1398–1403.

[45] D.-H. Park, H. Hoffmann, P. Pastor, and S. Schaal, "Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields," in *Humanoids 2008-8th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2008, pp. 91–98.

[46] Y. Zhou, J. Gao, and T. Asfour, "Learning via-point movement primitives with inter- and extrapolation capabilities," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4301–4308.

[47] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," in *Advances in neural information processing systems*, 2013, pp. 2616–2624.

[48] T. Matsubara, S.-H. Hyon, and J. Morimoto, "Learning stylistic dynamic movement primitives from multiple demonstrations," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 1277–1283.

[49] S. Miller, M. Fritz, T. Darrell, and P. Abbeel, "Parametrized shape models for clothing," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 4861–4868.

[50] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-specific generalization of discrete and periodic dynamic movement primitives," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 800–815, 2010.

[51] A. Pervez and D. Lee, "Learning task-parameterized dynamic movement primitives using mixture of gmms," *Intelligent Service Robotics*, vol. 11, no. 1, pp. 61–78, 2018.

[52] M. Brand and A. Hertzmann, "Style machines," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 2000, pp. 183–192.

[53] H. Lee, H. Kim, and H. J. Kim, "Planning and control for collision-free cooperative aerial transportation," *IEEE Transactions on Automation Science and Engineering*, accepted for publication.

[54] F. Abi-Farraj, T. Osa, N. P. J. Peters, G. Neumann, and P. R. Giordano, "A learning-based shared control architecture for interactive task execution," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 329–335.

[55] G. Maeda, M. Ewerton, T. Osa, B. Busch, and J. Peters, "Active incremental learning of robot movement primitives," in *Conference on Robot Learning (CORL)*, 2017.

[56] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," in *Advances in neural information processing systems*, 2003, pp. 1547–1554.

[57] R. M. Murray, Z. Li, S. S. Sastry, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[58] I. I. San Juan, C. Sloth, A. Kramberger, H. G. Petersen, E. H. Østergård, and T. R. Savarimuthu, "Towards reversible dynamic movement primitives," in *Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019.

[59] C. E. Rasmussen, "Gaussian processes for machine learning," 2006.

[60] S. Krishnan, A. Garg, S. Patil, C. Lea, G. Hager, P. Abbeel, and K. Goldberg, "Transition state clustering: Unsupervised surgical trajectory segmentation for robot learning," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1595–1618, 2017.

[61] J. Barbič, A. Safonova, J.-Y. Pan, C. Faloutsos, J. K. Hodgins, and N. S. Pollard, "Segmenting motion capture data into distinct behaviors," in *Proceedings of Graphics Interface 2004*. Citeseer, 2004, pp. 185–194.

국 문 초 록

시연 학습 기법(Learning from demonstrations, LfD)은 로봇이 특정 동작을 수행할 수 있도록 하는 유망한 동작 생성 기법이다. 로봇 조작기가 인간 사회에서 다양한 업무를 대체해 감에 따라, 다양한 임무를 수행하는 로봇의 동작을 생성하기 위해 LfD 알고리즘들은 널리 연구되고, 사용되고 있다.

본 논문은 LfD 기법 중 모션 프리머티브 기반의 동작 재생성 알고리즘인 Parametric dynamic movement primitives(PDMP)에 기초한 알고리즘을 제시하며, 이를 통해 다양한 임무를 수행하는 모바일 조작기의 궤적을 생성한다. 기존의 동작 재생성 알고리즘과 달리, 이 연구는 제공된 시연에서 표현된 동작을 단순히 재생성하는 것에 그치지 않고, 새로운 환경에 맞게 일반화 하는 과정을 포함한다. 이 논문에서 제시하는 일반화 과정은 PDMPs의 내부 파라미터 값인 스타일 파라미터와 환경 변수 사이의 비선형 관계를 가우스 회귀 기법(Gaussian process regression, GPR)을 이용하여 수식적으로 표현한다. 제안된 기법은 또한 최적 시연를 학습하는 방식을 통해 강력한 최적 실시간 경로 계획 기법으로도 응용될 수 있다.

본 논문에서는 또한 로봇의 구동 안전성도 고려한다. 기존 연구들에서 다루어진 시연 관리 기술이 로봇의 구동 효율성을 개선하는 방향으로 제시된 것과 달리, 이 연구는 강한 구속조건으로 로봇의 구동 안전성을 확보하는 시연 관리 기술을 통해 안정성을 고려하는 새로운 방식을 제시한다. 제안된 방식은 스타일 파라미터 값 상에서 안전성 기준을 계산하며, 이 안전 기준을 통해 시연을 제거하는 일련의 작업을 수행한다. 또한, 제거된 시위를 안전 기준에 따라 입증된 시위로 대체하여 일반화 성능을 저하시키지 않도록 시위를 관리한다. 이를 통해 다수의 시연 각각 개별 동작 안전성 뿐 아니라 온라인 동작의 안전성까지 고려할 수 있으며, 실시간 로봇 조작기 운용시 안전성이 확보될 수 있다. 제안된 안정성을 고려한 시연 관리 기술은 또한 환경의 정적 설정이 변경되어 모든 시연을 교체해야 할 수 있는 상황에서 사용할 수 있는 시연들을 판별하고, 효율적으로 재사용하는 데 응용할 수 있다.

또한 본 논문은 복잡한 임무에서 적용될 수 있는 PDMPs의 확장 기법인 seg-PDMPs를

제시한다. 이 접근방식은 복잡한 임무가 일반적으로 복수개의 간단한 하위 작업으로 구성된다고 가정한다. 기존 PDMPs와 달리 seg-PDMPs는 전체 궤적을 하위 작업을 나타내는 여러 개의 단위 동작으로 분할하고, 각 단위동작에 대해 여러개의 PDMPs를 구성한다. 각 단위 동작 별로 생성된 PDMPs는 통합된 프레임워크내에서 단계 결정 프로세스를 통해 자동적으로 호출된다. 각 단계 별로 단위 동작을 수행하기 위한 시간 및 하위 목표점은 가우스 공정 회귀(GPR)를 이용한 환경변수와의의 관계식을 통해 얻는다. 결과적으로, 이 연구는 전체적으로 요구되는 시연의 수를 효과적으로 줄일 뿐 아니라, 각 단위동작의 표현 성능을 개선한다.

제안된 알고리즘은 협동 모바일 로봇 조작기 실험을 통하여 검증된다. 세 가지의 시나리오가 본 논문에서 다루어지며, 항공 운송과 관련된 첫 두 가지 시나리오는 PDMPs 기법이 로봇 조작기에서 빠른 적응성, 임무 효율성과 안전성 모두 만족하는 것을 입증한다. 마지막 시나리오는 지상 차량을 이용한 두 개의 로봇 조작기에 대한 실험으로 복잡한 임무 수행을 하기 위해 확장된 기법인 seg-PDMPs가 효과적으로 변화하는 환경에서 일반화된 동작을 생성함을 검증한다.

주요어: 동작 재생성 알고리즘, 모바일 매니퓰레이터, 동작 생성 기법, 시연 학습 기법

학 번: 2014-21897