Transfer Learning Methods for Prediction, Replanning, and Adaptations of Autonomous Mobile Robots Under Degraded Conditions

---

A

Doctoral Dissertation

Presented to

the Faculty of the School of Engineering and Applied Sciences

The University of Virginia

---

in partial fulfillment

of the requirements for the degree

Doctor of Philosophy

by

Shijie Gao

August 2024

APPROVAL SHEET

This

Dissertation

is submitted in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

Author: Shijie Gao

This Dissertation has been read and approved by the examing committee:

Advisor: Nicola Bezzo

Advisor:

Committee Member: Zongli Lin

Committee Member: Lu Feng

Committee Member: Tariq Iqbal

Committee Member: Daniel Quinn

Committee Member:

Committee Member:

Accepted for the School of Engineering and Applied Science:

Jennifer L. West, School of Engineering and Applied Science

August 2024

Transfer Learning Methods for Prediction, Replanning, and Adaptations of Autonomous Mobile Robots Under Degraded Conditions

by

Shijie Gao

M.E., Computer Engineering
The University of Virginia, 2023

B.S., Automation
Beijing Institute of Technology, 2017

Abstract

Autonomous mobile robots are becoming increasingly popular and improving the quality of our lives by revolutionizing industries such as transportation, healthcare, logistics, agriculture, emergency response, and manufacturing. This rapid advancement is driven by simulation technologies that enable comprehensive design and testing before deployment. However, the transition from simulation to reality often reveals discrepancies in modeling robots and their environments, creating challenges that can hinder smooth deployment. Additionally, even after successful deployment, robots face numerous challenges and uncertainties, including environmental changes, system aging, unexpected disturbances, and actuator faults. These challenges, which often occur at runtime without prior knowledge, can cause deviations from the systems' intended behavior, leading to unsafe conditions. Considering all these challenges, to enhance the resilience of robotic deployments and operations, it becomes critical to detect the occurrence of dynamic changes and predict the future states of the robot under such changes. Further, quickly adapting the robots' control and planning components to align with the new dynamics is essential to recover the system and resume its intended behavior. Such adaptations not only address changes happening to the system during deployment and operation but can also be used to deal with the well-known transfer-learning problem across different robotic systems for fast and safe deployment.

This dissertation contributes to the existing state of the art in robotics operations against changes in system dynamics and failures. Through the proposed framework, future states of autonomous mobile robots are monitored to proactively update their motion plan and enhance their safety when dynamics may have changed. To this end, first, we introduce a Meta-Learning-based framework that allows the system to predict its future states and state uncertainties after dynamic changes due to unforeseen faults. This framework also monitors these predictions and the surrounding environment, allowing for real-time adjustments of the reference path to ensure safety during operations. Because this proactive path replanning focuses on safety, it may disrupt the continuation of the assigned task. To ensure that a task can resume, further adaptation of the controller and path planner is preferred for better management of the degraded system. To address this challenge, we present a conformal mapping-based transfer learning method. This approach enables the adaptation of control and path planning policies to match a reference system's behavior, bypassing the need for accurate model estimation and effectively compensating for dynamic discrepancies arising from model mismatches. These techniques are validated through extensive simulations and proof-of-concept lab experiment case studies on ground and aerial robotic systems in realistic scenarios.

Acknowledgements

I am profoundly grateful to my advisor, Nicola Bezzo, for his exceptional guidance and unwavering support throughout my studies at the University of Virginia. His dedication to research excellence, deep commitment to student development, enthusiasm for robotics, and motivational advice have fundamentally shaped my journey as a researcher and will continue to inspire my future.

I would like to thank my dissertation committee members: Prof. Zongli Lin, Prof. Lu Feng, Prof. Tariq Iqbal, and Prof. Daniel Quinn for their insightful and constructive feedback about my research as well as their support during the final stages of my PhD studies.

I would like to express my gratitude to current and former members of the AMR Lab – Tony, Esen, Carmelo, Paul, Rahul, Phil, Will, Garrett, Lauren, Jacob, Nick, Patrick, and Beatrice – for being excellent lab mates and friends over the years. We have had many wonderful conversations and times together, which I will always cherish. I would like to thank all of my friends I made in Charlottesville for all the joyful memories we shared. I wish I could list everyone but that would probably be too long.

Lastly, I extend my heartfelt gratitude to my parents, Wenhai and Qin, and my grandparents, Fuwen, Xingping, and Yulian, for their unwavering understanding, support, and unconditional love. They have been my constant pillars of strength through all the highs and lows. Without them, I would not be the person I am today.

Funding: I acknowledge: the Defense Advanced Research Projects Agency (DARPA) for providing financial support for my Ph.D. research under the Assured Autonomy program and CoStar through the Autonomous Building Condition Detection and Evaluation grant.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Figure 1.1: Examples of autonomous mobile robots serving diverse purposes, like: package delivery, transportation, search and rescue, agriculture applications, warehouse logistics, and inspection

Autonomous mobile robots (AMRs), including unmanned ground vehicles (UGV) and unmanned aerial vehicles (UAV), are rapidly gaining popularity and significantly enhancing our daily lives. These robots are designed for various applications across different sectors. For instance, service robots streamline package delivery and are integral in household tasks such as cleaning and providing entertainment. In agriculture, specialized vehicles are deployed for tasks like aerial dusting, harvesting, and crop monitoring, boosting efficiency and productivity. Additionally, field robots are increasingly used in critical roles such as search and rescue, disaster response, and law enforcement operations. The rapid development and deployment of these applications are greatly supported by advances in simulation technologies, which allow researchers to comprehensively design, test, and refine these robots in controlled environments before they are introduced to real-world tasks. This simulation-driven approach accelerates development timelines and enhances the reliability and functionality of robotic systems in practical settings.

However, applications developed in simulators often encounter challenges when deployed in the real world, primarily due to the so-called "Sim-to-Real" gap — the discrepancies between simulated environments and real-world conditions. This gap highlights the limitations of simulations in capturing intricate details, necessitating significant additional effort during real-world deployment. Although researchers strive to improve the robustness of their designs, it can be difficult to anticipate all possibilities during the design phase. This is particularly true when dealing with dynamic changes that cannot be easily modeled or simulated. For example, one of our previous works [26] investigated how airflow dynamics change when a quadrotor is flown close to a surface. The interaction between the propellers and surfaces can generate additional thrust within a range. These effects, known as ground and ceiling effects, impact the vehicle's dynamics and are typically not captured in simulators. Similar changes in dynamics can only be analyzed and characterized by real experiments. On the other hand, due to limited computational resources, developers must carefully decide the level of realism in simulations, balancing the need for detailed simulation against practical development constraints. Consequently, the design of these simulations is often subject to the designers' personal interpretations of the problem they aim to solve, which can significantly influence the development of techniques for robotic applications This subjective element introduces variability in how simulations are constructed and implemented, impacting the overall efficacy and applicability of the resulting technologies. With all this considered, it is impossible to perfectly replicate the real world within a simulation. Therefore, finding effective strategies to seamlessly transition from simulated environments to real-world settings and addressing the "Sim-to-Real" gap are crucial for the swift advancement of robotic applications. Developing robust methodologies to bridge this gap not only accelerates the innovation cycle but also enhances the reliability and performance of robotics in practical applications.

Moreover, once robots are operational in the real world, they continue to face numerous unpredictable challenges that can compromise the effectiveness of previously well-developed techniques. For example, field and service robots may need to drive over different terrains to reach a goal, an agriculture drone may need to deal with wind disturbances during flight, and even worse they may need to deal with failures and changes in dynamics for example due to component aging and deterioration. These scenarios illustrate the broader challenge of model mismatch, which encompasses both dynamic changes and the "Sim-to-Real" gap. Addressing these challenges is crucial for the successful deployment and operation of robotic systems. It necessitates the development of innovative solutions capable of anticipating and mitigating model mismatches. By adapting and

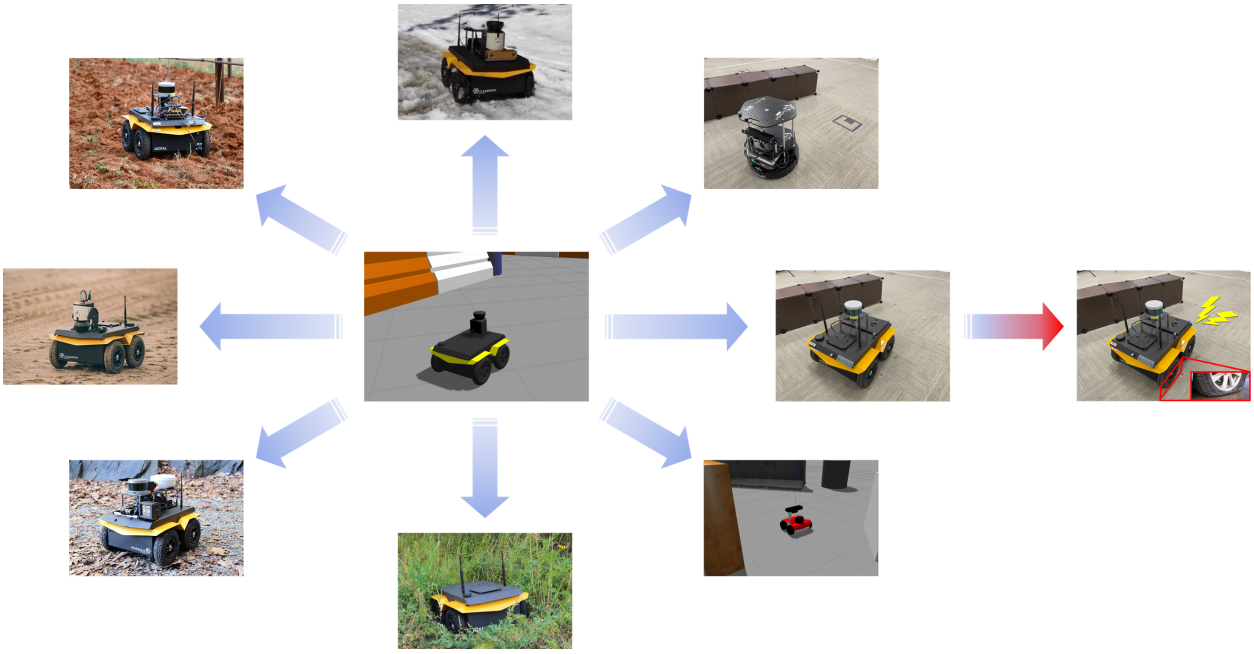


Figure 1.2: Methodologies developed and optimized in simulation often encounter significant challenges upon deployment in real-world settings. These challenges include environmental variations, actuator faults, and dynamic model mismatches, among others.

transferring techniques to handle these discrepancies, we can ensure that robots perform reliably in their intended environments. This continuous improvement cycle is essential to maintain the efficacy and reliability of robotics in real-world applications.

Since the uncertainties that a robot faces at runtime usually occur without a priori knowledge and often cause deviations from its desired behavior, it becomes critical to detect if any dynamical changes happened and to predict where the robot will be when the changes are present. To predict the future state of the vehicle, data-driven approaches have been shown to be powerful tools. However, this problem is challenging for the following reasons: 1) system changes at runtime can be unforeseen, making them impossible to be trained beforehand; 2) it is usually challenging to make a faulty vehicle run long enough to collect sufficient data while simultaneously training a new model at runtime. On the other hand, the dynamic change of a vehicle is often not an immediate alteration, but rather a gradual transition process. Thus, an approach that can quickly adapt the prediction model to fit with the new dynamics by using a few data can be useful for solving these challenges at runtime. At the same time, besides the rapid model adaption, the system should be able to monitor unsafe conditions, provide risk assessments, and replan to maintain safety and liveness constraints.

By predicting future states and adjusting the plan accordingly, the system can avoid dangerous situations. However, changing the originally intended plan may result in a halt on the desired

task. A better approach is to help the system not only detect the system changes but also recover from them and continue the desired task. Considering that dynamic changes due to uncertainties, disturbances, or component faults are gradual transitions rather than dramatic changes, it is efficient to learn the changes and recover the system by directly adapting the knowledge from the original dynamics. Such an adaption problem can be cast as a transfer learning process that adapting the behavior before and after the changes as a ***teacher*** vehicle and a ***learner*** vehicle, respectively. The goal is that for the learner to achieve similar behaviors as the teacher by transferring the control and planning policies. Furthermore, such a framework can also be used in any transferring learning problem in which the target and the source systems are similar (e.g. Sim-to-Real problems, model mismatch problems).

To this end, the objectives of this dissertation are to solve the following challenges:

- How to predict future states and state uncertainties of the system with unforeseen faults at runtime.
- How to proactively monitor the system and replan the desired behaviors to prevent unsafe situations at runtime.
- How to enable the system to learn the operational limits of the degraded system.
- How to recover the system from changes in the dynamics due to model mismatches by transferring knowledge from a known teacher model to a learner system.

In order to accomplish these objectives, this dissertation proposes to leverage a few-shot learning technique based on Meta-Learning for updating the future states predicting model at runtime to account for the unforeseen dynamic changes. A conformal mapping-based transferring framework that adapts the control and planning policies to the degraded system so that it matches the desired behavior while keeping its safety and liveness.

1.1 Overview of Research

The research presented in this dissertation consists of two successive Parts that include: **Part I** states predicting and safe replanning for systems with unforeseen faults, and **Part II** control and path plan policies transferring to mitigate model mismatch problems. A final **Part III** summarizes what we have gained and learned from the first two Parts. Figure 1.3 provides an overview of the research presented in this dissertation.

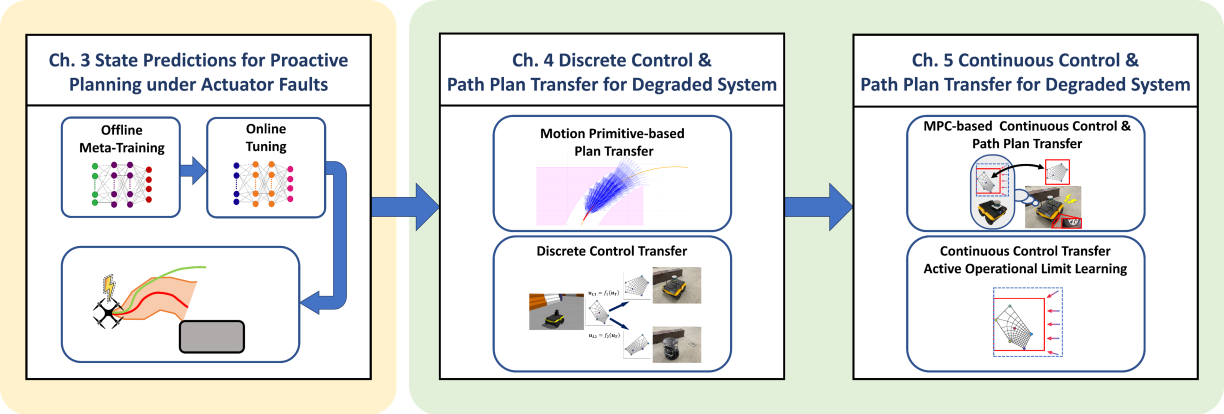


Figure 1.3: Overview of the presented research in this dissertation.

1.2 Dissertation Organization and Contributions

In this section, we detail the structure of this dissertation by summarizing each chapter and highlighting the contributions made within them. Chapter 3 is dedicated to predicting the future states and corresponding uncertainties of autonomous mobile systems when they are experiencing unforeseen actuator faults, and replanning to avoid potential collisions in real-time. Chapter 4 focuses on learning the operational limits of a degraded system and adapting control and motion plan policies from a well-developed system to a degraded one by directly transferring the control input. Chapter 5 further expands on the proposed transferring framework to enable continuous control input transfer. This chapter also enhances the transferring framework by introducing a method for actively refining the learned operational limits of the reference system during the transfer process, which eliminates the need for a separate calibration step prior to transfer. To conclude this dissertation, Chapter 6 provides a summary of our results and discusses potential future works.

Chapter 3: Meta-Learning-based States Prediction and Proactive Planning under System Degradations

In this chapter, we introduce a Meta-Learning-based framework designed for faulty vehicle state prediction and recovery. This framework transfers the knowledge that is Meta-Learned offline from various faulty behaviors to predict future states and state uncertainties at runtime, during which a new fault may arise. Through our proposed framework, the reachable states of a faulty vehicle are predicted and validated. Faulty behaviors are detected when the vehicle deviates from the predicted regions at the designated times, triggering an online update to the Meta-Learning model using a

minimal amount of data gathered during operation to ensure precise predictions with respect to the new dynamics due to faults. Additionally, the framework incorporates a sampling-based proactive replanning method, which generates new waypoints for replanning around the areas where the risk of collisions appears. The path plan is then revised after passing the safety assessment based on the new reachability analysis, ensuring the safe retrieval of the faulty vehicle. Our proposed framework is validated through simulations and real-world experiments consisting of UAVs with different actuator faults navigating through cluttered environments. This chapter is based on the following publications:

- S. Gao, E. Yel, and N. Bezzo, "Meta-Learning-based Proactive Online Planning for UAVs under Degraded Conditions." *in IEEE Robotics and Automation Letters 7, no. 4 (2022): 10320-10327.*.

Chapter 4: Conformal Mapping-Based Transfer Learning for Discrete Control Inputs

In this chapter, we aim to enhance our capability of handling the changed dynamics of AMRs, going beyond fault detection and recovery planning for the faulty system. Our approach extends to transferring the control and path plan policies to better accommodate and manage degraded systems. We assume that the target system, though similar, has reduced capabilities compared to the source system. Our transferring framework observes and contrasts the behaviors of the two systems, establishing a bijection between their command domains through a dedicated calibration stage. This allows us to understand the operational limitation of the reference system and match the geometrical distribution of the control inputs within these domains accordingly. Leveraging Schwarz-Christoffel Mapping, a conformal mapping method, our framework geometrically maps areas on the command domains to derive control inputs that are adapted to the reference system. The planning policy, informed by the operational limits learned for the target system, is further constrained to yield the plan tailored for the source system. This framework can be generalized for addressing model mismatch challenges broadly. Different from existing methods in the literature, our framework is lightweight compared to machine learning methods and directly maps the control input between systems without the need to precisely learn the dynamical model of the target systems. We demonstrate our framework by integrating it with a motion-primitive-based planner, validating its effectiveness through extensive simulations and laboratory experiments with different UGVs for navigation tasks. This chapter is based on the following publication:

- S. Gao, and N. Bezzo, "A Conformal Mapping-based Framework for Robot-to-Robot and Sim-to-Real Transfer Learning." *in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021.*

Chapter 5: Conformal Mapping-based Transfer Learning for Continuous Control Inputs

In this chapter, we complement our Schwarz-Christoffel Mapping-based transfer learning framework by enabling the transfer of the control inputs across a continuous command space. We also enhance the robustness of the framework by accommodating systems with nonnegligible motion noises. Moving beyond the initial calibration period discussed in the previous chapter, we expand the framework by incorporating an active learning component. This integration actively refines the operational limits of the reference system concurrently with the transfer process, particularly for the adaptation of the planner to consider. We integrate our transfer framework with an MPC in a navigation task with UGVs to demonstrate the proposed approach. We validate our enhanced framework through both simulations and laboratory experiments, during which we purposely alter the dynamics of the target system to increase model discrepancies. The UGV seamlessly replicates the desired system's maneuvers, thereby proving the effectiveness of the proposed work. This chapter is based on the following manuscript currently under review:

- S. Gao, and N. Bezzo, "A Conformal Mapping-based Framework for Sim-to-Real Transfer in Autonomous Mobile Robot Operations." *Journal of Intelligent & Robotic Systems(under review), 2024.*

Chapter 6: Conclusions and Future Work

In this chapter, we conclude the dissertation by summarizing the key findings from the studies presented and outlining potential future research directions that build upon these results.

1.3 Summary of Contributions

To summarize, the work presented in this dissertation will contribute to the existing state-of-the-art in autonomous mobile robot operations against failures and changes in systems dynamics by providing:

- A Meta-Learning-based approach that transfers knowledge learned offline about system reachability under various faults for online prediction and validation of future states and uncertainties in systems with unforeseen faults.
- An online monitoring system that validates predicted state, coupled with a sampling-based replanning method for ensuring the safety of degraded systems.
- A novel and generalizable conformal mapping framework based on Schwarz-Christoffel Mapping for transferring control and path planning policies between a teacher and a learner vehicle, with demonstrations on: motion-primitive-based path planning and MPC methods.

Chapter 2

Related Work

In this chapter, we provide an overview of the existing literature on resilient control and safe planning for autonomous mobile robots under uncertainties. Following this, we present an overview of techniques used to transfer control and path planning policies between systems.

2.1 Resilient Control and Safe Planning

For consistent performance under faulty behaviors, control techniques have been widely utilized to adapt the systems' control inputs according to the changes in the system dynamics and to alleviate the effects of faults. For example, for quadrotors with the complete loss of one or multiple actuators, specific controllers can be designed according to the failure to maintain stability and performance [42, 58, 59, 28]. However, these techniques require explicit knowledge of the specific failures and how these changes affect the system's dynamical model to design resilient controllers. When such knowledge is not available, fault identification or adaptive control techniques can be leveraged. In [65], an Extended Kalman Filter (EKF)-based fault identification is used to decide if there are one or multiple rotor failures, and a control allocation is updated based on the failure using a nonlinear Model Predictive Control (MPC). In [24], a self-reconfiguration technique allows the system to decide on its configuration based on the actuator failure and its desired trajectory. Another well-known adaptive control technique called Model Reference Adaptive Control (MRAC) adapts the control variables of the system based on the difference between the observations and reference model output to improve tracking for systems with uncertainties and it has been used to compensate for failures [39, 38]. Recently, machine learning techniques such as Gaussian Processes (GP) [12] and deep neural networks (DNNs) [31] have been leveraged for the adaptive elements in MRAC frameworks. GP-based approaches have also been used to model the effects of changes in

the system model (e.g., due to unknown payload mass) and to provide safe plans [69]. Robustness against faulty systems has also been achieved by using resilient distributed consensus [53, 71], and topology control [70] within the context of multi-agent navigation.

In addition to control approaches, machine learning techniques have also been widely used to improve the performance of UAVs under actuator faults or disturbances. In [54], the authors use MPC with active learning to learn the robot's new model under failure and provide necessary inputs. Reinforcement Learning (RL) techniques are also utilized to adjust the actuator control commands to compensate for component faults [19, 2]. Meta-learning approaches enable systems to speed up their learning process for new tasks with a small number of training samples from new tasks. This property makes meta-learning suitable for learning the models of uncertain systems at runtime for safe planning [36]. One of such techniques–Model-Agnostic Meta-Learning (MAML)–trains the model parameters explicitly to make them easy and fast to fine-tune for a new task [20]. MAML has been leveraged for fault-tolerant operations using MPC and RL in [43, 3]. An algorithm called Fast Adaptation through Meta-Learning Embeddings (FAMLE) is proposed in [32] that meta-learns multiple priors as opposed to a single prior in MAML, and picks the most likely prior to improve online learning efficiency. [55] introduces a concept of meta-active learning in which a Q-function is learned via meta-learning and used to find optimal actions to maximize the probability of staying in the safe region and promote information gain for systems with altered dynamics. In [51], meta-learning is utilized to model the system dynamics under external forces to be used with an adaptive control scheme to improve the tracking performance. All of these approaches assume that the user is given direct access to the controller or the actuator inputs. However, this assumption may not hold, especially when off-the-shelf robotic systems are used.

2.2 Transfer Learning in Robotics

The sim-to-real gap, which arises when transferring from simulation to the real world, primarily exists because the model may not be accurate, or the environmental factors are not represented in the simulation. Although we can always improve the fidelity of the simulation based on real-world observations [10], it is still not possible to perfectly replicate reality. Furthermore, robotic applications developed in such simulators may lose generality when deployed, as pointed out in [63]. Therefore, apart from refining the simulator, it is necessary to develop a more robust controller and planner or other transfer methods to close the sim-to-real gap.

As machine learning techniques have become widely exploited, transfer learning has gained significant attention in robotics toward addressing transferring problem. The core concept of transfer learning is to migrate the knowledge gained from one task to address similar tasks in diverse environments. It takes advantage of existing knowledge [14] and reduces the cost as well as the risk associated with data collection and model training [23, 72, 30]. Researchers have been investigating this topic through various strategies, including domain randomization, domain adaptation, imitation learning, and large fundamental models etc.

Domain randomization approaches parameterize the dynamics and environmental conditions within simulations, seeking to encapsulate real-world complexities through augmented simulation scenarios [60, 47, 62, 61, 10]. Despite the intention to bridge the sim-to-real gap, they often incur high computational costs and can lead to over-generalization, resulting in systems that are capable of handling a broad range of unlikely scenarios but may underperform in typical real-world conditions. Domain adaptation, a well-established technique in machine learning, trains models with samples from a source domain to effectively generalize to a target domain. Proven in computer vision, this method has been particularly useful for bridging the sim-to-real gap in vision-based robotic control problems[6, 18]. Similarly, recent end-to-end approaches[64] and RL-CycleGAN [49] also focus on minimizing the sim-to-real disparity primarily in visual-based control tasks by aligning the visual inputs between reality and simulations. However, these approaches often overlook the underlying model mismatches and are built on the assumption that robotic controls are well-established without sim-to-real discrepancies. This assumption represents a significant oversight—a challenge that our work aims to address.

Another strategy that has been broadly explored is imitation learning, also known as learning from demonstration or behavior cloning, which enables robots to acquire new skills by mimicking expert behaviors. This method shifts from the traditional approach of learning through prolonged repetition of simple tasks by directly deriving policies [56, 33], plan [35, 44], or rewards [4, 21] from an expert. While this technique offers some performance guarantees, it can be adversely affected by suboptimal or inappropriate examples from the expert, and it lacks the ability to handle complex tasks. Additionally, it is sensitive to environmental changes, which can hinder its ability to generalize effectively [50]. Consequently, imitation learning is not ideally suited for addressing sim2real transfer problems, where adaptability across varied real-world conditions is crucial. Recently, large foundation models have been adapted for robotics applications, exemplified by [8, 7]. These works have shown their effectiveness in high-level semantic reasoning using visual

and language inputs. While they enhance human-robot interaction and enable the transfer of high-level plans between different robots and environments, their application is limited to tasks for which the robots have been explicitly trained. Although the potential for transferring control through transformers remains largely unexplored, the extensive data requirements for training such models pose a significant challenge for researchers. Despite the main approaches, researchers have also investigated into other novel methods for closing the sim-to-real gap, including reducing the costly errors by predicting the blind spots in real environments [48] and inflating safety critical regions to reduce the chance of collision [27].

2.3 Conformal Mapping in Robotics

Conformal mapping is a mathematical technique used in complex analysis. A conformal mapping function transfers a complex domain onto another one while preserving angles locally. Researchers have exploited this approach for geometrical problems in robotics. [5] has introduced conformal mapping to aid in correcting the distortion in robotic vision. [34, 57] treat the original path plan as a geometrical pattern and use conformal mapping to adapt the plan to suit the task in specific settings. However, the SCM method proposed in this dissertation is rarely utilized in the robotics field. In [45], the SCM is employed to transform planar motion into continuous linear motion, addressing a coverage control problem for wire-traversing robots. Our recent work [25] first brought the idea of leveraging the SCM method to directly transfer control inputs between two systems. It shows the efficiency and effectiveness of bridging the gap when transferring the path planner and the controller between similar systems. However, [25] relies on motion primitives to achieve motion plan transferring, whose transferring results are discrete and highly depend on the size of the motion primitive library. In this dissertation, we demonstrate the transfer of discrete control and path planning based on motion primitives, followed by a continuous transfer using a receding horizon approach.

Part I

Faulty Behavior Detection and Recovery

Chapter 3

Meta-Learning-based States Prediction and Proactive Planning for Degraded Systems

In this chapter, we present our meta-learning-based approach to predict the future states and associated uncertainties of an unmanned aerial vehicle (UAV)when faced with unforeseen actuator faults. Our method utilizes meta-learning to train a model across various actuator faults. The model makes predictions based on the observed history of the states and the reference path during the corresponding timeframe. We leverage meta-learning for rapidly adapting the model to manage raised new faults during operation. The relearning of the model is triggered whenever the UAV exceeds the predicted states and uncertainties. A runtime monitoring technique is implemented to detect potential collision risks based on the forecasts for safety enhancement. Additionally, a waypoint-sampling-based replanning method is initiated whenever a safety risk is identified. The proposed approach is validated through simulations and real-world experiments in a case study on a faulty UAV path-tracking task, demonstrating the effectiveness of enhancing the safety of autonomous UAV operation. The material covered in this chapter has been accepted for publication in IEEE Robotics and Automation Letters and was presented at the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

- S. Gao, E. Yel, and N. Bezzo, "Meta-Learning-based Proactive Online Planning for UAVs under Degraded Conditions," *in IEEE Robotics and Automation Letters 7, no. 4 (2022): 10320-10327.*.

3.1 Introduction

Autonomous mobile robots like aerial vehicles are rapidly becoming an integral part of our daily lives thanks to their widespread use in different applications from delivery to inspection. When operating

in the real-world, numerous unpredictable challenges (e.g., component faults, external disturbances) can cause performance degradations, potentially leading to unsafe behaviors like collisions. For example, unpredicted wind fluctuations around buildings or bridges or motor failures will put an aerial vehicle at risk of collision while conducting inspection jobs. Since these uncertainties usually occur at runtime without apriori knowledge, it becomes challenging to take them into account during design time.

One way to cope with such unforeseen disturbances is to learn the system model and adapt the controller of the robot at runtime. However, given a well-developed robotic system, it is often the case that the controller is either hard to adapt or is not accessible by the user. Thus, one of the best options is to act on the planner which is by design available to change. By correcting the reference planned trajectory, it is often possible to make a robot with the original controller follow the original desired behavior [68]. This type of approach is effective in improving the performance of a robot dealing with an unforeseen component fault. However, we observe that depending on the fault, the tuning of the adapted planner, and the robot's physical limits, an undesired behavior may still occur, such as a deviation from the desired path, which may lead to potentially unsafe situations.

With these premises, in this chapter, we introduce a novel safety monitoring technique to predict the future states of an autonomous robot under actuator noises and previously unforeseen actuator failures. We employ meta-learning to solve a reachability analysis problem to deal with situations in which even after replanning, the system may still deviate from the desired behavior. In particular, with our proposed framework, future states and associated uncertainties for a faulty autonomous system considering future corrections [68] are predicted at runtime without the need to retrain a learning component. Our framework utilizes these predictions to monitor if the system will violate safety constraints (e.g., a collision with an obstacle). When an unsafe situation is detected, a safe trajectory is then replanned using a sampling-based approach for waypoint selection. Figure 3.1 pictorially shows the problem space of this chapter in which a UAV is tasked to inspect a power plant. Due to unexpected failures, the UAV will deviate and collide with the cooling tower. Our technique proactively monitors and predicts the regions the system may reach over a future horizon and replans the trajectory when these regions intersect with obstacles.

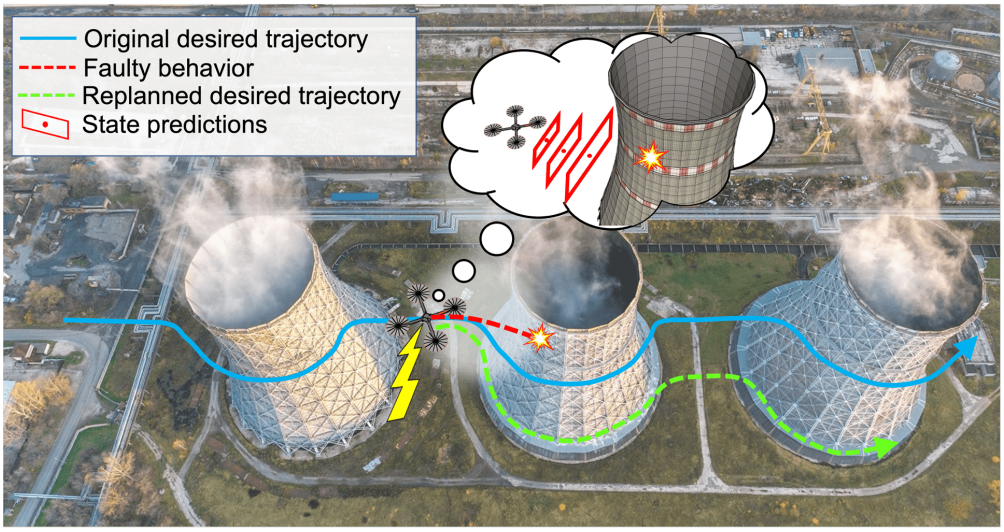


Figure 3.1: Pictorial representation of a UAV experiencing a failure at runtime leading to a potential collision.

3.2 Preliminaries

In this chapter, we assume that once a robot is deployed to perform an operation, we do not have access to its controller or control inputs and we only have access to the high-level planner and reference trajectory generator. This consideration is made in order to increase realism since most robotic systems do not allow manipulations of the controller once a robot is deployed but instead, it is possible to change its goal locations and trajectory waypoints. We assume that the system is already applying corrective counter-measures both at design time and at runtime to alleviate the effects of faults and to follow a desired trajectory closely. However, under some disturbances/failures, its behavior may still become erratic as the corrective actions may not be able to compensate and bring the system back on its desired path.

3.2.1 Notations

In this chapter, we use $\boldsymbol{x}(k)$ to represent the state of the system at time k. $\boldsymbol{p}(k)$ and $\boldsymbol{v}(k)$ represent the position and velocity of the system respectively. The symbol $\tilde{\boldsymbol{x}}$ is used to represent the predicted state, and the symbol $\bar{\boldsymbol{x}}$ is used to represent the mean of sampled states. The notation $x(k:k_N)$ represents an array of values from time k to k_N: $x(k:k_N) = [x(k), x(k+1), \ldots, x(k_N)]^\mathsf{T}$ where $k_N > k$. The notation $x(k:\delta_N:k_N)$ represents an array of values from time k to k_N with $\delta_N \in \mathbb{Z}^+$ increments: $x(k:\delta_N:k_N) = [x(k), x(k+\delta_N), x(k+2\delta_N), \ldots, x(k_N)]^\mathsf{T}$ where $k_N > k$ and $\delta_N > 1$.

3.3 Problem Formulation

The goal is to design a technique that predicts the future states of a system under an unseen fault at runtime, uses these predictions to detect unsafe situations, and proactively replans to improve safety. The system might already be applying corrective actions to compensate for the experienced faults, however, these corrective measures may not be enough to prevent collisions. Formally, we define these problems as follows:

Problem 1: ***Future State Prediction under Failure*****:** An autonomous system with a nominal dynamical model $f(\boldsymbol{x}, \boldsymbol{u})$ as a function of its states $\boldsymbol{x}$ and controller inputs $\boldsymbol{u}$ has the objective of following a predefined desired trajectory $\boldsymbol{x}_\tau$. Under actuator faults and noises, the

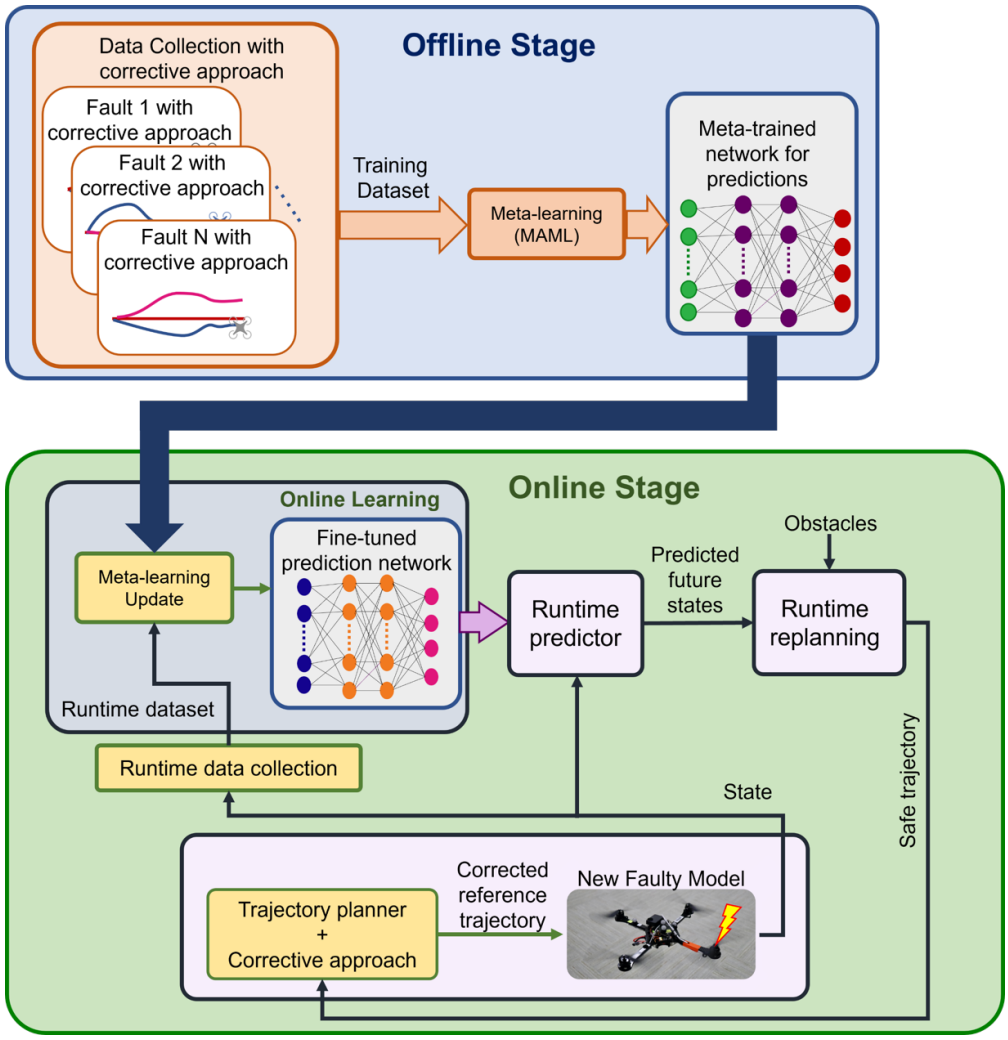


Figure 3.2: Meta-learning-based future state prediction and replanning framework for systems under unknown faults.

system's model changes to $\boldsymbol{x}(k+1) = f'(\boldsymbol{x}(k), \tilde{\boldsymbol{u}}(k))$, where f' is unknown. The system has noisy inputs $\tilde{\boldsymbol{u}}(k)) = \boldsymbol{u}(k) + \boldsymbol{\eta}(k)$ which create state uncertainties, making the exact state prediction of the system challenging. With these premises, our goal is to design a predictor to map the future state predictions ($\tilde{\boldsymbol{x}}$) and state uncertainty predictions ($\tilde{\boldsymbol{\zeta}}$) as a function ($\tilde{h}$) of the history of states and reference trajectory: $[\tilde{\boldsymbol{x}}(k:k+H), \tilde{\boldsymbol{\zeta}}(k:k+H)] = \tilde{h}(\boldsymbol{x}(k-T:k-1), \boldsymbol{x}_\tau)$. H is the state prediction horizon and T is the size of the data history used to make predictions.

Problem 2: *Safe Replanning*: Design an online policy to monitor the safety of the future states of the system and to replan the trajectory when an unsafe situation is detected to ensure that the following safety conditions will be satisfied by the future predicted states:

$$R_{\boldsymbol{p}}|_k^{k+H} \cap \boldsymbol{\mathcal{O}} = \emptyset \tag{3.1}$$

where $\boldsymbol{\mathcal{O}}$ is the set of obstacle positions and $R_{\boldsymbol{p}}|_k^{k+H}$ is the union of future position sets that the system is predicted to reach over a time horizon H with time increments of δ_H:

$$R_{\boldsymbol{p}}|_k^{k+H} = R_{\boldsymbol{p}}(k) \cup R_{\boldsymbol{p}}(k+\delta_H) \cdots \cup R_{\boldsymbol{p}}(k+H) \tag{3.2}$$

The set of positions that the system is predicted to reach at time k is computed as follows:

$$R_{\boldsymbol{p}}(k) = \cup\{\boldsymbol{p} \text{ s.t. } \|\boldsymbol{p} - \tilde{\boldsymbol{p}}(k)\| \leq \tilde{\boldsymbol{\zeta}}_{\boldsymbol{p}}(k)\} \tag{3.3}$$

where $\tilde{\boldsymbol{p}}$ is the predicted position and $\tilde{\boldsymbol{\zeta}}_{\boldsymbol{p}}$ is the predicted position uncertainty.

3.4 Meta-Learning-based Predictions and Proactive Replanning

Our framework consists of offline and online stages as depicted in Figure 3.2. During offline and online stages, the system applies corrective measures to compensate for the faults. During the offline stage, the robot under various faults follows a set of trajectories. A meta-network is trained offline to predict future states and their uncertainties based on the collected training data. At runtime, the robot experiences a new, unforeseen fault. With a few data collected at runtime, the meta-network is fine-tuned to make predictions over a finite horizon about the future states and state uncertainties of the new faulty system considering corrective countermeasures. These predictions are used within a runtime replanning approach to find a safe trajectory if the original desired trajectory is deemed

unsafe with the fault that the system is experiencing.

3.4.1 Offline Training for State and Uncertainty Predictions

To train a model offline for state and uncertainty predictions, we first collect training data using a UAV with various faults following a rich set of trajectories. Then, we use meta-learning to train a network which is easy to fine-tune at runtime using a small number of data [20].

Data Collection

During the offline stage, a dataset is created using the data collected from a UAV with actuator fault from a discrete fault set $\mathcal{F}$ while it is following different trajectories applying fault-tolerant corrective measures. Specifically, we consider a system with faulty dynamics and actuator noise modeled as follows:

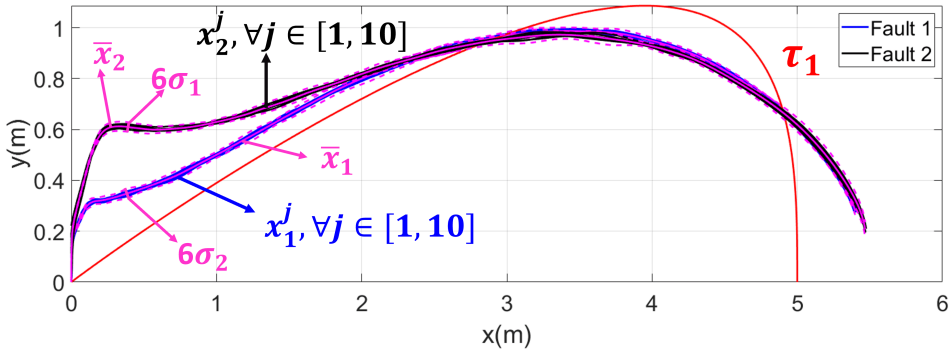
$$\boldsymbol{x}(k+1) = f'(\boldsymbol{x}(k), \boldsymbol{u}(k) + \boldsymbol{\eta}(k)) \tag{3.4}$$

The actuator noise $\boldsymbol{\eta}(k) \sim \mathcal{N}(\boldsymbol{\mu}_\eta, \boldsymbol{\sigma}_\eta)$ is sampled from a normal distribution with mean $\boldsymbol{\mu}_\eta$ and standard deviation $\boldsymbol{\sigma}_\eta$. Each trajectory is run N times and for each sampled run, the mean of the actuator noise is sampled from a normal distribution to capture the behavior of the system under various uncertainties: $\boldsymbol{\mu}_\eta \sim \mathcal{N}(\bar{\boldsymbol{\mu}}, \boldsymbol{\sigma}_\mu)$.

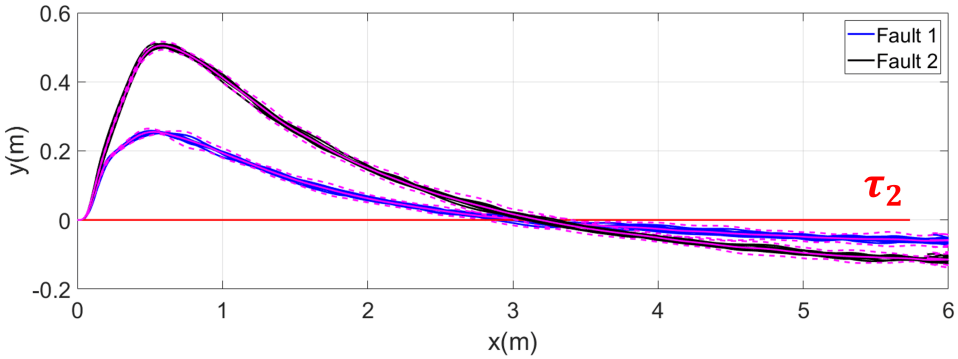
For each trajectory $\tau \subset \mathcal{T}$, we compute the mean and standard deviation of the N sampled paths for each fault $\mathcal{F}_i \subset \mathcal{F}$ and for each discrete sample times $k \in [0, T_\tau]$.

$$\bar{\boldsymbol{x}}_i(k) = \frac{\sum_{j=1}^{N} \boldsymbol{x}_i^j(k)}{N}, \boldsymbol{\sigma}_i(k) = \sqrt{\frac{\sum_{j=1}^{N} |\boldsymbol{x}_i^j(k) - \bar{\boldsymbol{x}}_i(k)|^2}{N-1}}$$
$$\forall k \in [0, T_\tau], \forall i \in \{1, \ldots, |\mathcal{F}|\} \tag{3.5}$$

In Figure 3.3 we show two different sample desired trajectories ($\boldsymbol{\tau_1}$ and $\boldsymbol{\tau_2}$) that are followed by a UAV under two different faults applying the reference update procedure described in [68] to reduce their tracking error. The first faulty UAV has reduced thrust in one actuator: $\mathcal{F}_1 \rightarrow T_1' = T_1 - 0.25\,\mathrm{N}$ and the second faulty system has more degradation on the same actuator: $\mathcal{F}_2 \rightarrow T_1' = T_1 - 0.5\,\mathrm{N}$. Roll and pitch angles of both systems are limited to $|\phi| \leq \frac{\pi}{18}, |\theta| \leq \frac{\pi}{18}$ respectively. Each trajectory is run $N = 10$ times with different actuator noise sampled as explained above. Blue curves in Figure 3.3(a) show these N sampled paths for the first fault and the magenta curve in the middle shows the



(a) Desired trajectory 1 followed by two different faulty systems.



(b) Desired trajectory 2 followed by two different faulty systems.

Figure 3.3: Sample desired trajectories with two different faulty systems.

mean of these samples ($\bar{\boldsymbol{x}}_1$). Dashed magenta curves show the uncertainty around the mean of the samples with $\pm 3\boldsymbol{\sigma}_i$. Figure 3.3(b) shows the paths of the same faulty UAVs following another training trajectory ($\boldsymbol{\tau_2}$). During training, we create 100 training trajectories using minimum-jerk trajectory generation [41] with different final positions, initial and final velocities. Determining the training data size for training an accurate meta-model is an open problem and is beyond the scope of the problem we are focused on. This framework leaves the choice of training data size to the user. However, as for most learning components, it benefits from training data that covers a wide variety of faults.

Meta-network Training

The purpose of meta-learning is to train an easily adaptable model to predict the future positions and position uncertainties of a faulty system. We denote this learning model as h which takes the history of the system's observed states, history of the desired states and future desired states as

inputs and returns the future states and state uncertainties as an output. A training input $\boldsymbol{\chi}_h^i$ and a training output $\boldsymbol{\gamma}_h^i$ are constructed as follows:

$$\boldsymbol{\chi}_h^i(k) = \begin{bmatrix} \boldsymbol{\xi}_{\boldsymbol{x}}^i(k) \\ \boldsymbol{\xi}_{\boldsymbol{\tau}}^i(k) \\ \boldsymbol{\xi}_h^i(k) \end{bmatrix} \qquad \boldsymbol{\gamma}_h^i(k) = \begin{bmatrix} \boldsymbol{\xi}_{\boldsymbol{\gamma}}^i(k) \\ \boldsymbol{\zeta}_{\boldsymbol{\gamma}}^i(k) \end{bmatrix} \tag{3.6}$$

where $\boldsymbol{\xi}_{\boldsymbol{x}}^i(k)$, $\boldsymbol{\xi}_{\boldsymbol{\tau}}^i(k)$ and $\boldsymbol{\xi}_h^i(k)$ represents the vectors related to the history of the system's states, history of the desired states and future desired states respectively and $\boldsymbol{\xi}_{\boldsymbol{\gamma}}^i(k)$ and $\boldsymbol{\zeta}_{\boldsymbol{\gamma}}^i(k)$ are the vectors related to the future state and state uncertainty predictions respectively. For the UAV application considered in this chapter , the input vectors for a quadrotor with the fault $\mathcal{F}_i \subset \mathcal{F}$ are constructed as follows:

$$\begin{aligned}
\boldsymbol{\xi}_{\boldsymbol{x}}^i(k) &= \begin{bmatrix} \bar{x}_i(k-T+1:k) - \bar{x}_i(k-T)\vec{\mathbf{1}} \\ \bar{y}_i(k-T+1:k) - \bar{y}_i(k-T)\vec{\mathbf{1}} \\ \bar{v}_i^x(k-T+1:k) \\ \bar{v}_i^y(k-T+1:k) \end{bmatrix} \\
\boldsymbol{\xi}_{\boldsymbol{\tau}}^i(k) &= \begin{bmatrix} x_\tau(k-T+1:k) - \bar{x}_i(k-T)\vec{\mathbf{1}} \\ y_\tau(k-T+1:k) - \bar{y}_i(k-T)\vec{\mathbf{1}} \\ v_{x,\tau}(k-T+1:k) \\ v_{y,\tau}(k-T+1:k) \end{bmatrix} \\
\boldsymbol{\xi}_h^i(k) &= \begin{bmatrix} x_\tau(k+\delta_H:\delta_H:k+H) - \bar{x}_i(k-T)\vec{\mathbf{1}} \\ y_\tau(k+\delta_H:\delta_H:k+H) - \bar{y}_i(k-T)\vec{\mathbf{1}} \\ v_{x,\tau}(k+\delta_H:\delta_H:k+H) \\ v_{y,\tau}(k+\delta_H:\delta_H:k+H) \end{bmatrix} \\
&\forall \tau \subset \mathcal{T}, \qquad \forall k \in \{1,\ldots,T(\tau)\}
\end{aligned} \tag{3.7}$$

where $\bar{\boldsymbol{p}}_i(k) = [\bar{x}_i(k), \bar{y}_i(k)]$ and $\bar{\boldsymbol{v}}_i(k) = [\bar{v}_i^x(k), \bar{v}_i^y(k)]$ are the position and velocity components of the mean state $\bar{\boldsymbol{x}}_i(k)$ respectively. For this application, we use the positions and velocities in $x-y$ plane as part of the observed and desired states, however, it should be noted that, depending on

the application, higher-order states such as acceleration or jerk values could also be added into the network input. Similarly, the output vectors are constructed as follows:

$$\boldsymbol{\xi}^i_{\boldsymbol{\gamma}}(k) = \begin{bmatrix} \bar{x}_i(k+\delta_H : \delta_H : k+H) - \bar{x}_i(k-T)\vec{\mathbf{1}} \\ \bar{y}_i(k+\delta_H : \delta_H : k+H) - \bar{y}_i(k-T)\vec{\mathbf{1}} \end{bmatrix}$$
$$\boldsymbol{\zeta}^i_{\boldsymbol{\gamma}}(k) = \begin{bmatrix} 3 \cdot \max(\sigma^x_i(k:k+H)) \\ 3 \cdot \max(\sigma^y_i(k:k+H)) \end{bmatrix} \tag{3.8}$$

where $\sigma^x_i(k)$ and $\sigma^y_i(k)$ are the x and y position components of the standard deviation $\boldsymbol{\sigma}_i(k)$ respectively.

The dataset for meta-learning training $\mathcal{D}^H_i$ for fault $\mathcal{F}_i$ contains the training input matrix $\boldsymbol{X}^i_h$ and output matrix $\boldsymbol{Y}^i_h$, with the columns $\boldsymbol{\chi}^i_h$ and $\boldsymbol{\gamma}^i_h$ respectively. The training dataset for meta-learning contains the dataset for each fault: $\mathcal{D}^H_i \subset \mathcal{D}^H$.

The purpose of meta-learning is to learn a model represented by a parameterized function $h_{\boldsymbol{\phi}}$ that maps the model input to the output. We use MAML [20] as a meta-learning algorithm to train the network. During the offline training, the model parameters vector $\boldsymbol{\phi}$ are meta-optimized according to Equation 1 in [20]:

$$\boldsymbol{\phi} \longleftarrow \boldsymbol{\phi} - \beta \nabla_{\boldsymbol{\phi}} \sum_{\mathcal{F}_i \subset \mathcal{F}} \mathcal{L}_{\mathcal{F}_i}(h_{\phi - \alpha \nabla_{\boldsymbol{\phi}} \mathcal{L}_{\mathcal{F}_i}(h_{\boldsymbol{\phi}})}) \tag{3.9}$$

where α is the learning step size, β is the meta step size, and $\mathcal{F}_i \subset \mathcal{F}$ indicates the sample batch of faults among the training data. For more details, we refer readers to [20].

This meta-optimization allows the parameters to be quickly fine-tuned with a few data at runtime. The loss function used during this training is given as follows:

$$\mathcal{L}_{\mathcal{F}_i}(h_{\boldsymbol{\psi}}) = \sum_{\boldsymbol{\chi}^i_h, \boldsymbol{\gamma}^i_h \in \mathcal{D}^H_i} \| h_\psi(\boldsymbol{\chi}^i_h) - \boldsymbol{\gamma}^i_h \|_2^2 \tag{3.10}$$

where $\boldsymbol{\chi}^i_h$ and $\boldsymbol{\gamma}^i_h$ are given in (3.6).

3.4.2 Online Meta-Network Update

At runtime, the UAV may experience a new fault which is not included in the training set and may apply the same corrective measures as during the training stage. While the system moves

under this new fault, we collect K_p consecutive data from its state sensors to update the offline meta-trained model for future state predictions. The inputs and outputs of the online learning data set are constructed in the same way as in (3.6) and (3.7):

$$\boldsymbol{\chi}_h^*(k) = \begin{bmatrix} \boldsymbol{\xi}_{\boldsymbol{x}}^*(k) \\ \boldsymbol{\xi}_{\boldsymbol{\tau}}^*(k) \\ \boldsymbol{\xi}_h^*(k) \end{bmatrix} \qquad \boldsymbol{\gamma}_h^*(k) = \begin{bmatrix} \boldsymbol{\xi}_{\boldsymbol{\gamma}}^*(k) \\ \boldsymbol{\zeta}_{\boldsymbol{\gamma}}^*(k) \end{bmatrix} \tag{3.11}$$

where:

$$\begin{aligned}
\boldsymbol{\xi}_{\boldsymbol{x}}^*(k) &= \begin{bmatrix} x^*(k-T+1:k) - x^*(k-T)\vec{\mathbf{1}} \\ y^*(k-T+1:k) - y^*(k-T)\vec{\mathbf{1}} \\ v_x^*(k-T+1:k) \\ v_y^*(k-T+1:k) \end{bmatrix} \\
\boldsymbol{\xi}_{\boldsymbol{\tau}}^i(k) &= \begin{bmatrix} x_\tau^*(k-T+1:k) - x^*(k-T)\vec{\mathbf{1}} \\ y_\tau^*(k-T+1:k) - y^*(k-T)\vec{\mathbf{1}} \\ v_{x,\tau}^*(k-T+1:k) \\ v_{y,\tau}^*(k-T+1:k) \end{bmatrix} \\
\boldsymbol{\xi}_h^i(k) &= \begin{bmatrix} x_\tau^*(k+\delta_H:\delta_H:k+H) - x^*(k-T)\vec{\mathbf{1}} \\ y_\tau^*(k+\delta_H:\delta_H:k+H) - y^*(k-T)\vec{\mathbf{1}} \\ v_{x,\tau}^*(k+\delta_H:\delta_H:k+H) \\ v_{y,\tau}^*(k+\delta_H:\delta_H:k+H) \end{bmatrix}
\end{aligned} \tag{3.12}$$

for $k \in \{T+1, \ldots, T+K_p\}$ with $\boldsymbol{p}^* = [x^*, y^*]$ and $\boldsymbol{v}^* = [v_x^*, v_x^*]$ the position and velocity of the UAV with an unknown fault at runtime. $\boldsymbol{p}_\tau^* = [x_\tau^*, y_\tau^*]$ and $\boldsymbol{v}_\tau^* = [v_{x,\tau}^*, v_{y,\tau}^*]$ are desired trajectory positions and velocities respectively. The output of the online learning dataset consists of the following vectors:

$$\begin{aligned}
\boldsymbol{\xi}_{\boldsymbol{\gamma}}^i(k) &= \begin{bmatrix} x^*(k+\delta_H:\delta_H:k+H) - x^*(k-T)\vec{\mathbf{1}} \\ y^*(k+\delta_H:\delta_H:k+H) - y^*(k-T)\vec{\mathbf{1}} \end{bmatrix} \\
\boldsymbol{\zeta}_{\boldsymbol{\gamma}}^i(k) &= \begin{bmatrix} \sigma_x & \sigma_y \end{bmatrix}^\mathsf{T}
\end{aligned} \tag{3.13}$$

where σ_x and σ_y are assigned uncertainties in x and y directions respectively and they are initially set to a value larger than the mean of the observed uncertainties during training. By using the data collected at runtime, the meta-learned model parameters $\boldsymbol{\phi}$ are updated to $\boldsymbol{\phi}^*$ using only a few stochastic gradient descent updates. The updated model $h_{\boldsymbol{\phi}^*}$ is then used to make predictions for the future states of the system for the same horizon considered in training. These predictions are used to replan trajectories if unsafe situations are detected, as explained in the next section.

Runtime Validation

After the initial meta-network update at runtime, the runtime inputs are compared to the training inputs to assess if a further meta-network update is necessary or not. The distance between the runtime input and training inputs with training faults is calculated as:

$$d^i_{\mathcal{F}}(k) = \min_{\boldsymbol{\chi}^i_h \in \text{col}(\boldsymbol{X}^i_h)} \|\boldsymbol{\chi}^*_h(k) - \boldsymbol{\chi}^i_h\| \qquad \forall i \in \{1, \cdots, |\mathcal{F}|\}$$
$$\forall k \in \{T + K_p, \ldots, T(\tau)\} \tag{3.14}$$

If the minimum distance between the observed test input and the training inputs is larger than a given threshold, the system re-tunes its meta-trained network:

$$s_H(k) = 1 \qquad \text{if } \min_{i \in \{1,\ldots,|\mathcal{F}|\}} (d^i_{\mathcal{F}}(k)) > \lambda_H \tag{3.15}$$

where s_H is a binary variable that enables re-updating the meta-trained network at runtime using the last K_p runtime training inputs and λ_H is a user-defined threshold.

We also constantly monitor the observed state to check if it is outside of the predicted reachable region. If so, the network is re-tuned:

$$s_H(k) = 1 \qquad \text{if } \boldsymbol{p}(k) \not\subset \tilde{R}_{\boldsymbol{p}}(k) \tag{3.16}$$

where $\tilde{R}_{\boldsymbol{p}}(k)$ is a region where the system is predicted to reach at time k:

$$\tilde{R}_{\boldsymbol{p}}(k) = \cup\{\boldsymbol{p} \text{ s.t. } \|\boldsymbol{p} - \tilde{\boldsymbol{p}}(k)\| \leq \tilde{\boldsymbol{\zeta}}(k)\} \tag{3.17}$$

3.4.3 Runtime Replanning for Safety

After updating the meta-trained network, the future states and state uncertainties of the system are predicted using the fine-tuned network:

$$\begin{bmatrix} \tilde{x}(k+\delta_H : \delta_H : k+H) \\ \tilde{y}(k+\delta_H : \delta_H : k+H) \\ \tilde{\sigma}_x(k+\delta_H : \delta_H : k+H) \\ \tilde{\sigma}_y(k+\delta_H : \delta_H : k+H) \end{bmatrix} = \boldsymbol{h}_\phi^*(\boldsymbol{\chi}_h^*(k)) + \begin{bmatrix} x^*(k-T)\vec{\mathbf{1}} \\ y^*(k-T)\vec{\mathbf{1}} \\ 0 \\ 0 \end{bmatrix}$$
$$\forall k \geq T + K_p \tag{3.18}$$

At runtime, the predicted set based on the updated meta-trained model are used to proactively detect unsafe situations. Given an environment with a set of static obstacles $\boldsymbol{\mathcal{O}}$, the regions that the system is predicted to reach, which are computed as in (3.17), are checked for collision:

$$s^*(k+t) = \begin{cases} 0 & \text{if } \tilde{R}_{\boldsymbol{p}}(k+t) \cap \boldsymbol{\mathcal{O}} \neq \emptyset \\ 1 & \text{otherwise} \end{cases}$$
$$\forall t \in \{\delta_H, 2\delta_H, \ldots, H\} \tag{3.19}$$

At time k, if it is detected that $s^*(k+t) = 0$ (i.e., reachable regions intersect with obstacles for $t \in \{\delta_H, 2\delta_H, \ldots, H\}$), the trajectory is replanned. To this end, here we use a sampling-based replanning method in which a waypoint around the original unsafe desired trajectory point is generated and tested for safety until a safe waypoint is found as outlined in Algorithm 1.

Algorithm 1 Trajectory Replanning

$\tau \leftarrow$ Initialize the desired trajectory
$s^*(k+t) \leftarrow$ Assess safety based on (3.19)
while $s^*(k+t) = 0$ **do**
    $d_s \sim \mathcal{U}_{[0,\bar{d}_s]} \leftarrow$ Sample update distance
    $\psi_s \sim \mathcal{U}_{[-\pi,\pi]} \leftarrow$ Sample update direction
    $\boldsymbol{p}_w = \boldsymbol{p}_\tau(k+t) + d_s[cos(\psi); sin(\psi)] \leftarrow$ Sample a waypoint
    $\tau \leftarrow$ Replan trajectory with $\boldsymbol{p}_w$
    $\tilde{R}_{\boldsymbol{p}}(k) \leftarrow$ Predict the reachable region with τ
    $s^*(k+t) \leftarrow$ Assess the safety of τ based on (3.19)
end while
return τ

The replanning algorithm is applied until the desired trajectory reaches the goal location. It should be noted that as the main objective of this work is not the replanning algorithm itself, a user can consider a different replanning approach based on the application.

3.5 Simulations and Experiments

The proposed meta-learning-based predictive and proactive replanning framework was validated on a faulty UAV infrastructure inspection task. For the following simulations and experiments, we consider a quadrotor UAV that is undergoing a faulty behavior and is equipped with a fault-tolerant corrective method which may or may not be well-tuned to recover the system against all possible failures/disturbances that can occur at runtime. Thus, the vehicle can deviate from its desired trajectory when facing a new fault beyond the corrective method's capabilities. In our simulations and experiments, we chose to use our meta-learning-based trajectory update method presented in [68] as a fault-tolerant corrective method. This technique has shown trajectory tracking improvements under degraded conditions. The corrective method is purposely poorly tuned to show not only the issue more clearly but also the effectiveness of the proposed approach. It is worth noting that we choose to use this method because it does not require access to the controller or controller inputs, however, our predictive and proactive planning framework can be used with other corrective/adaptive techniques as well (e.g., robust and adaptive controllers).

3.5.1 Simulations

In these simulations, the quadrotor is modeled with a 12-dimensional state vector [41] and the fault is modeled as a thrust change on randomly selected motors. Specifically, the fault is simulated by both reducing the thrust on one of the motors and increasing the thrust on the opposite motor of the quadrotor. The details of the faults used during training and testing are shown in Table 3.1. Note that the faults caused by motors 1 and 3 are amplified effects of the fault on the single motor. Considering the quadrotors are symmetric by nature, faults presented on 2 or 4 can be treated similarly. In addition to the faults, we also consider a system with limited roll and pitch angles: $|\phi| \leq \tau_\phi, |\theta| \leq \tau_\theta$ in order to accentuate the issue and for ease of demonstration. For training, we used two different angle limits: $\tau_\phi = \tau_\theta \in \{\frac{\pi}{18}, \frac{\pi}{12}\}$ and for testing we considered a different angle limit: $\tau_\phi = \tau_\theta = \frac{\pi}{16}$.

During the offline stage of collecting training data, the UAV is tasked to follow different minimum-jerk trajectories [41] under different faults. Specifically, the training trajectories are shaped as trapezoidal paths of different lengths and angles. The collected data are used for meta-training the reference trajectory neural network and the state and uncertainty prediction neural network.

Table 3.1: Fault types used during simulations

Training fault	Fault type
$\mathcal{F}_1$	66% thrust on motor 1 and 136% on motor 3
$\mathcal{F}_2$	43% thrust on motor 1 and 160% on motor 3
$\mathcal{F}_3$	21% thrust on motor 1 and 184% on motor 3
Test fault	**Fault type**
${\mathcal{F}_1}^*$	32% thrust on motor 1 and 173% on motor 3

During training, the control loop runs at 40Hz. We use $T = 10$ past data to predict the future states, and we set the future horizon for the predictions $H = 50$ steps which is equivalent to 1.25 s ahead. The state and the uncertainty predictions are given at five future times spaced $\delta^h = 10$ time steps apart. The prediction network contains five hidden layers with 100 nodes and is meta-trained by using a Tensorflow Keras implementation of MAML [20].

During the online testing stage, the UAV is tasked to follow a desired path to inspect a structure while undergoing an unexpected failure. In Figure 3.4 we show a baseline case that is used to compare our framework. Without correction, the UAV collides with the black obstacle (cyan-colored baseline path) while following the desired trajectory (red-colored line). In the same figure, we show also the case in which the UAV only applies corrective reference updates according to [68] (blue line) while trying to follow the desired path without the proposed predictive and proactive replanning technique. The UAV adapts the trained model to deal with the new fault by using prior data obtained during the flight and then uses the adapted model to update the reference trajectory (magenta line in the figure). It should be noted that, as a proof of concept, we used a poorly tuned correction which leads to collisions, to demonstrate the prediction and compensation capabilities of our approach next.

Given the same simulation setup, we validate our prediction and replanning technique while the UAV is applying the same reference trajectory update in [68]. Figure 3.5 shows that the UAV predicts a potential collision within the predicting horizon (1.25 s) and replans its desired path to avoid the obstacle. In the zoomed-in window, we show the instance in which the UAV predicts and detects the collision and keeps predicting and checking the safety of the replanned paths. The

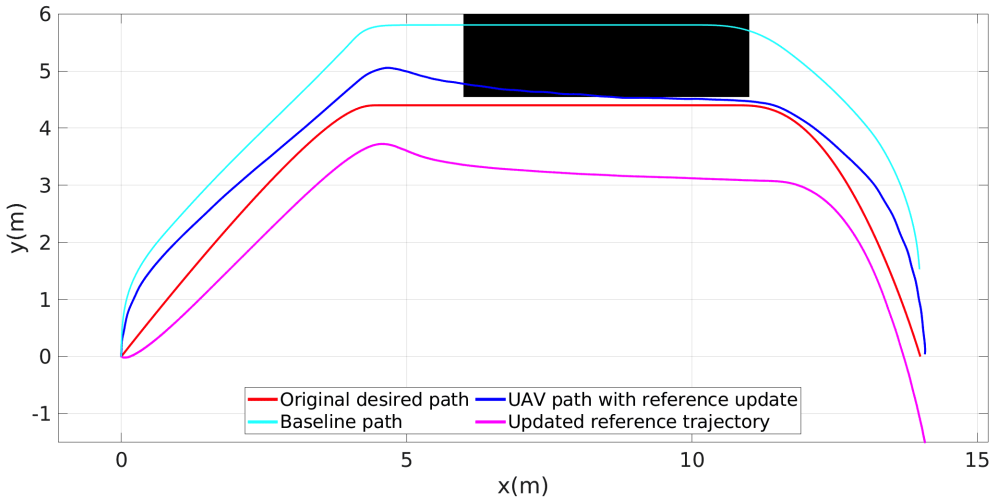


Figure 3.4: UAV path under a test fault without the meta-learning prediction and replanning approach.

proposed replanned desired paths and the predictions for checking the safety are color coded. Given the first proposed replan trajectory which is shown as the dashed brown line, the framework predicts the future positions as well as the uncertainties of the vehicle which are indicated as a series of brown rectangles in the zoomed-in window. The first proposed replan is considered as unsafe since the predicted positions collide with the obstacle and thus it is abandoned. Another replanned trajectory orange-colored is considered but yet deemed unsafe. Finally, a safe replanned desired trajectory (green dashed line) is validated by the predictions of the framework (green rectangles) and the vehicle can overcome the unsafe situation. As the goal of our proposed framework is to recover the faulty vehicle from the unsafe operations, the path replanning prioritize getting the UAV safely to the desired destination over staying close to the initial desired path.

3.5.2 Experiments

The proposed meta-learning-based prediction and proactive re-planning approach was also validated with experiments by using an Asctec Hummingbird quadrotor UAV inside a controlled laboratory space. Similar to the simulations, the UAV uses a baseline PID controller which is designed for the nominal quadrotor (without a fault) to control its position and attitude. To create the faulty behavior on motors 1 and 3, a fault is injected into the system by adding a bias to the commanded pitch angle before it is fed to the attitude controller. A poorly tuned meta-learning-based reference trajectory update approach [68] is used to compensate for the deviation caused by the fault during tracking. The experiments are implemented in ROS and a Vicon motion capture system is used to

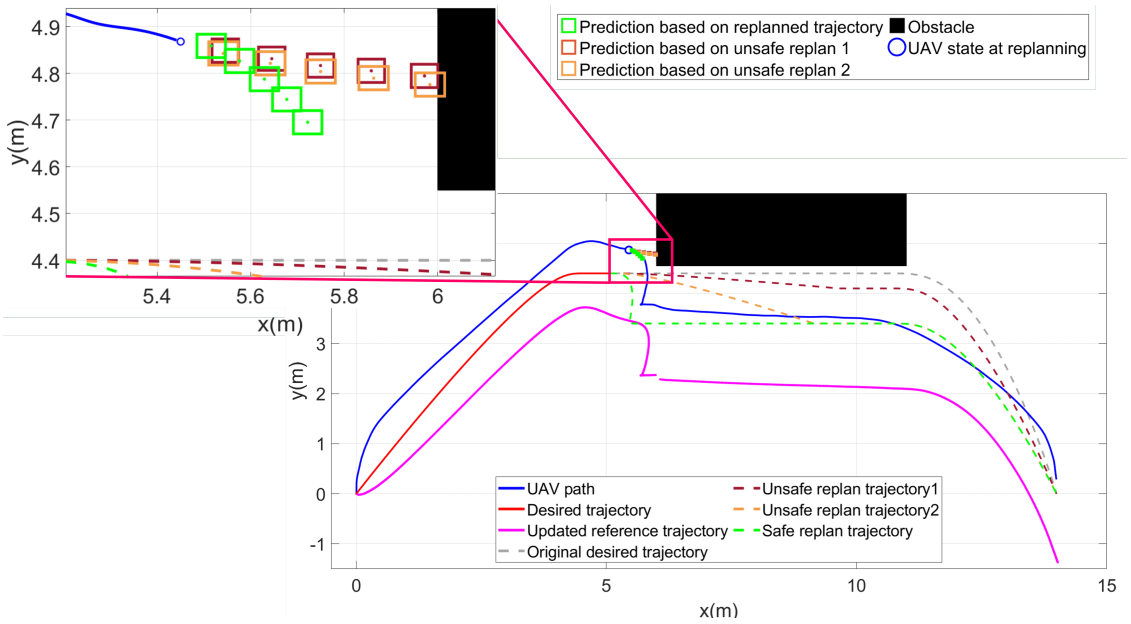


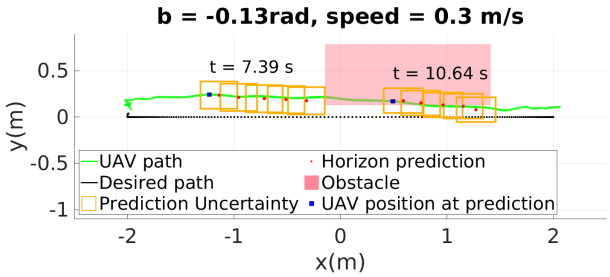
Figure 3.5: UAV path under a test fault with the proposed meta-learning-based prediction and replanning approach.

monitor and track the state of the UAV.

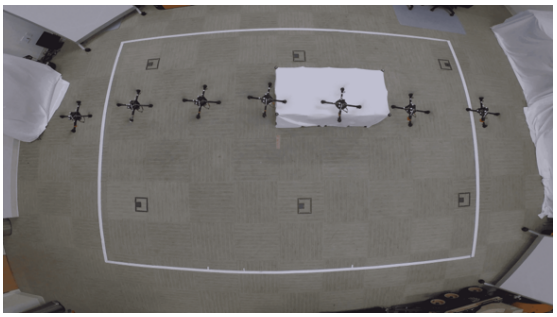
To train a model for future state prediction, we collected training data for the UAV following different trajectories considering various speeds and angled paths. We ran each of the training trajectories with 5 different faults by directly adding biases $b \in \{-0.06, -0.09, -0.12, -0.15, -0.18\}$rad to the pitch command and three runs for each case. As the magnitude of the faults increases, the quadrotor deviates more toward the positive y-direction. A meta-learning model was trained with the collected data to predict the states as well as the uncertainty of the vehicle at different time frames $\{+0.4, +0.8, +1.2, +1.6, +2\}$s in the future. The same architecture of the neural networks in the simulation was used for the experiments.

During the online stage, the vehicle was tasked to follow a desired trajectory at a maximum velocity of 3 m/s while a bias $b = 0.13$rad – which is different from the ones in the training set – was applied to the vehicle. While the UAV was tracking the desired path, it used the offline meta-trained model to predict the positions and uncertainties at 5 different time frames in the future with the maximum predicting horizon of 2 s. Figure 3.6 $\sim$ Figure 3.8 show the results of the UAV taking a desired straight path from the start point to the destination. The predictions are shown as the orange bounding boxes in the figures. Any overlapping area between the predicted boxes and the obstacles is considered a potential collision and thus triggers the re-planning procedure.

Figure 3.6(a) and 3.6(b) show the results from the first experiment in which the re-planning procedure was disabled. To demonstrate the correctness of the prediction and show where the UAV



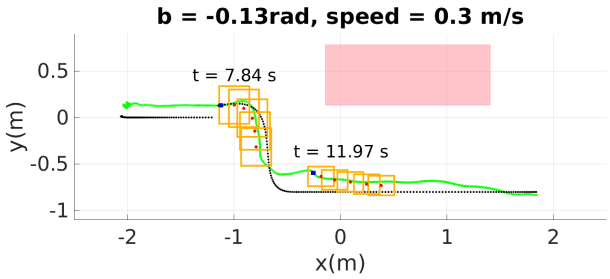
(a) UAV path and examples of predictions.



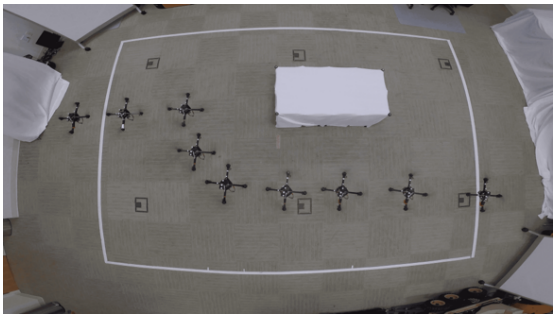
(b) Overlapped screenshots.

Figure 3.6: The UAV with an unknown fault detects a potential collision while replanning is disabled during a flying task from the start point to a destination.

would have reached without re-planning, avoiding damaging the vehicle, we set its height above the obstacle. As a result, the UAV detected a collision at time 7.39 s when it was 0.98 m away from the obstacle and flew through the obstacle because replanning was disabled in this test.



(a) UAV path and examples of predictions.

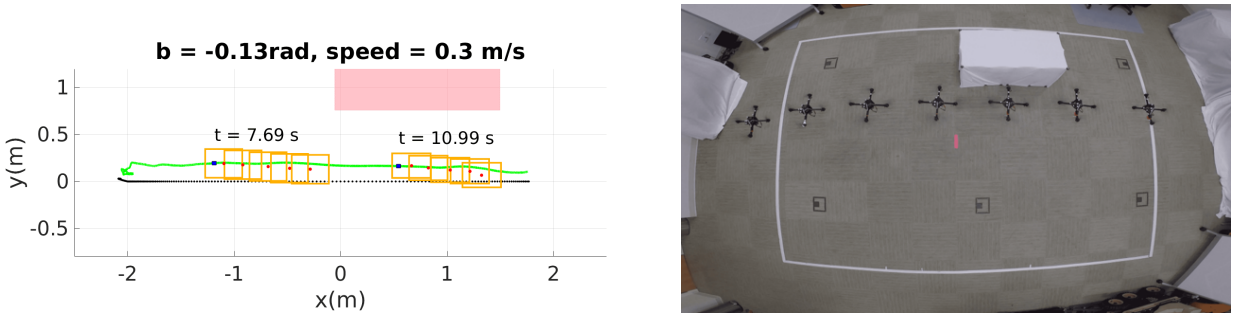


(b) Overlapped screenshots.

Figure 3.7: The UAV with an unknown fault detects a potential collision and avoids the obstacle by replanning during a flying task from the start point to a destination.

In the second set of experiments shown in Figure 3.7(a) and 3.7(b), we enabled the replanning module. While the UAV is flying, at time 7.84 s, the predictions by the meta-learned model indicated a collision risk, which resulted in triggering the replanning behavior. The UAV randomly sampled a waypoint at $(-0.41, 0.8)$m and generated a new desired trajectory to drive around the obstacle. Given the new desired trajectory, the UAV confirmed that the new path is safe within the prediction horizon based on the meta-trained model predictions. We note that as Experiment 1 and Experiment 2 are two separate experiment instances, the UAV detects the collision at different times. In Experiment 2, it took less than 0.05 s to find a safe solution after detecting the collision, which demonstrates that our approach was fast enough to be performed at runtime. It is also noteworthy

that how to pick the waypoints for re-planning is not the contribution of this work; we rather focus on predicting the states of the vehicle with an unknown fault.



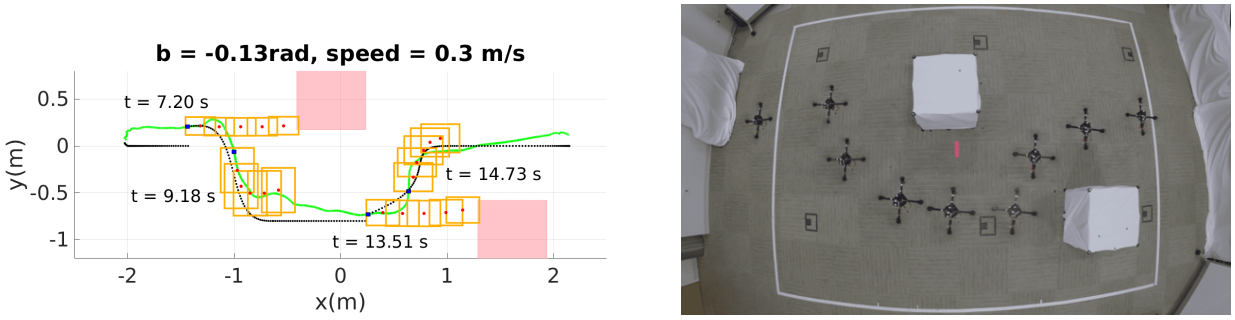
(a) UAV path and examples of predictions.

(b) Overlapped screenshots.

Figure 3.8: Using the proposed approach, the UAV with an unknown fault continuously monitors potential collision risks during flight tasks. No replanning is triggered as the obstacle poses no threat to the UAV's safety.

In the third experiment, the obstacle was set at a different position than the previous two experiments. As shown in Figure 3.8(a) and 3.8(b), the UAV constantly monitored the predictions and did not need to perform re-planning since no potential collision was detected.

We also tested our approach in scenarios with more obstacles. As shown in Figure 3.9, we sampled some examples of the predictions while the UAV navigates through the cluttered environment. At times 7.20 s and 13.51 s the UAV predicted and detected potential collisions. We also showed two other predictions of future states and uncertainties at times 9.18 s and 14.73 s to further demonstrate the output of the proposed framework. Due to the space constraints of our lab, we did not perform tests with more obstacles and failures; however, we believe that the presented results are representative enough to demonstrate the effectiveness of the proposed approach.



(a) UAV path and examples of predictions.

(b) Overlapped screenshots.

Figure 3.9: The UAV detects the collisions and avoids the obstacles multiple times. The legend for this figure is the same of Figure 3.6(a)

3.6 Discussion

In this chapter, we utilized meta-learning to train a highly adaptable network capable of making predictions about the state and state uncertainty of a faulty system. These predictions are used to identify potential collisions within a fixed future horizon. We employed a sampling-based method to proactively replan the system's original desired trajectory. To ensure safety, the replanned path is validated by checking predictions based on this new trajectory to prevent unsafe situations.

Limitations and Future Directions:

In this work, predictions for future states are generated by a meta-trained and fine-tuned network, with the accuracy of these predictions directly dependent on the quality of training. If the training data are insufficient or the network is poorly trained, the predictions may become inaccurate. Our framework includes runtime validation, which involves comparing runtime inputs with training inputs and monitoring prediction accuracy as the system operates. This technique allows for network retuning to enhance prediction accuracy, although it does not offer formal assurance guarantees.

The replanning scheme in our framework relies on sampling waypoints and evaluating their safety, a process that can be time-consuming if a safe waypoint is not identified within a limited number of iterations. To address this challenge, future developments could focus on a learning technique that directly generates safe waypoints. Additionally, while we currently use a fixed horizon for predicting future states, a recursive approach could be developed to enable variable horizon predictions.

Part II

Control and Path Plan Transfer for Degraded Systems

Chapter 4

Conformal Mapping-Based Transfer Learning for Discrete Control Inputs

In this chapter, we introduce a Schwarz-Christoffel Mapping-based transfer framework, designed to migrate control and motion planning policies between two systems. Our framework enables a learner system to adopt the teacher's policies in accordance with its own capabilities, despite not explicitly knowing the closed form of its dynamic model. The proposed framework directly maps the teacher's control inputs to those of the learner, effectively mitigating system differences and enabling the learner to replicate the teacher's desired maneuvers. The primary challenges in this transfer that we aim to address are: 1) determining the inherent relationship between the control inputs of both systems that produce identical maneuvers, and 2) configuring the operational limits of the learner to ensure that the teacher's controller and path planner deliver compatible results. Our framework discovers the geometrical transformations between the command domains of the two systems by establishing a bijective conformal mapping. Given the teacher's commands, the desired learner control input is obtained via a conformal mapping function. The learner's capabilities are assessed through a dedicated calibration stage, which evaluates its responses to a range of test commands within its command limits. We selected a widely adopted path planning method, motion primitive-based path planning, to demonstrate the transfer of path planning policies. The motion plan transfer is achieved by filtering out the primitives that exceed the learner's capabilities. The proposed approach is implemented using the Robot Operating System (ROS), MATLAB ROS Toolbox, and MATLAB Schwarz-Christoffel Mapping Toolbox. The effectiveness of the framework is validated by extensive simulation and real experiments with Clearpath Robotics Jackal and Turtlebot2 UGVs. The material covered in this chapter has been accepted for the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

- S. Gao, and N. Bezzo, "A Conformal Mapping-based Framework for Robot-to-Robot and Sim-to-Real Transfer Learning." *in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021.*

4.1 Introduction

Robotic applications are typically built considering specific systems in mind. For example, popular motion planning methods (e.g., artificial potential field [11], A* [17], probabilistic techniques [40]) and control methods (e.g., MPC, PID [46]) require fine-tuning and knowledge about system model dynamics in order to be fully leveraged and obtain a desired performance on a selected platform. We also note that most technologies are developed through simulations which offer a practical and inexpensive means to create and test the limits and performance of designed algorithms. Researchers usually spend considerable time and resources to create techniques for specific robotic systems and to adapt them to new systems, as well as to compensate for the simulation-reality gap during deployments on actual vehicles. Finally, even when a new technique is developed and deployed on a specific robot, it can still need to be adjusted or adapted over time due to mechanical aging, disturbances, and even failures that deprecate and modify the system's original model. In this chapter, we seek a general framework to transfer and adapt the system's performance. As mentioned above the goal of the proposed work is to:

- Reduce the sim-to-real gap allowing a developer to quickly transfer motion planning and control methods onto a real platform.
- Transfer knowledge designed for a specific robot onto a different robot.
- Compensate for system deterioration/failures by learning quickly the limits and the proper input mapping to continue an operation.

All of the aforementioned problems can be simplified and cast as a *teacher* transferring knowledge to a *learner*.

Specifically, to address these problems, in this chapter, we propose a novel method that leverages a variant of Schwarz–Christoffel mapping (SCM) [16] – a conformal transformation of a simple poly area onto the interior of a rectangle – to transfer a teacher vehicle's control input sequence to a learner vehicle, as depicted in Figure. 4.1. Our proposed method allows the teacher to understand the learner's limitations so that the transferred control input is compatible with the learner's

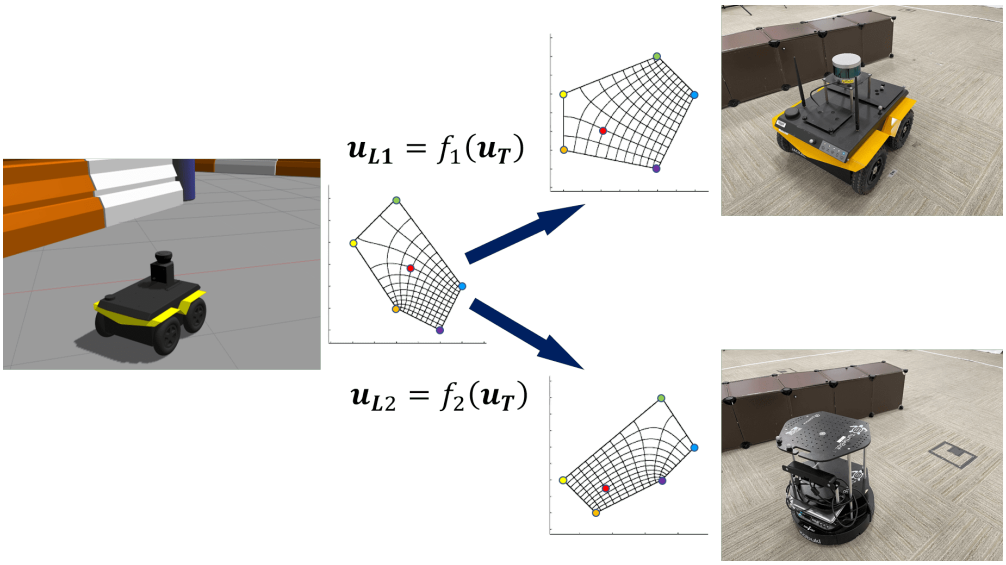


Figure 4.1: Pictorial representation of the proposed work in which motion planning and control policies are transferred from a teacher-simulated vehicle to two vehicles to create the same behavior designed in simulation.

capabilities. Finally, once these limitations are extracted, we propose a mechanism to adapt also the teacher motion planning scheme to create paths compatible with the learner constraints. To deal with this problem, our scheme leverages an optimized finite horizon primitive motion generation.

The main contributions of the work covered in this chapter are twofold: 1) a light-weight transfer framework that leverages SCM theory to directly transfer the control input from teacher to learner so that the learner can leverage the teacher's control policy while its own dynamics remain unknown; and 2) a method for adapting the source system's control and path planning policy to the learner. The method constrains the output of the source system's controller and of the path planner so that the transferred motion plan and control input are guaranteed to be compatible with the target system's dynamics.

The rest of this chapter is organized as follows: in Section 4.2 we formally define the problem we addressed within this chapter. The details of our SCM-based transfer learning framework are presented in Section 4.3. The proposed framework is validated with extensive simulations in Section 4.4 and experiments on real robots in Section 4.5. At last, we conclude this chapter and discuss the limitations as well as the potential future directions in Section 4.6.

4.2 Problem Formulation

The problem addressed in this work can be considered as a transfer learning problem from the teacher system to the learner. The goal is to find a mapping function that allows for a seamless translation of commands from the teacher to the learner, ensuring they result in identical movements (i.e., reaching the same pose and speed). The user possesses comprehensive knowledge of the teacher including its dynamics, finely-tuned controller, and planner. In contrast, the learner is treated as a black box to which we can send control inputs and observe the full states.

4.2.1 Notations

In this work, we use $\boldsymbol{x_T}(t)$ and $\boldsymbol{x_L}(t)$ to represent the state of the teacher and learner systems while $\boldsymbol{u_T}(t)$ and $\boldsymbol{u_L}(t)$ represent the control inputs of the two systems at time t. The symbols $\overline{\boldsymbol{u}}$ and $\underline{\boldsymbol{u}}$ denote the upper and lower bounds of the control inputs, respectively. Formally, we define two problems in this context:

Problem 4.1 ***Teacher-Learner Control Transfer:*** *Given a teacher robot with dynamics* $\boldsymbol{x_T}(t+1)=f_T(\boldsymbol{x_T}(t),\boldsymbol{u_T}(t))$ *and control law* $\boldsymbol{u_T}=g(\boldsymbol{x})$*, find a policy to map* $\boldsymbol{u_T}$ *to a learner input* $\boldsymbol{u_L}$ *such that* $\boldsymbol{x_L}(t+1)=f_L(\boldsymbol{x_L}(t),\boldsymbol{u_L}(t)) \approx f_T(\boldsymbol{x_L}(t),\boldsymbol{u_T}(t))$*, without knowing the closed-form of the* f_L*.*

Problem 4.2 ***Teacher-Learner Motion Planning Adaptation:***

Consider a task to navigate from an initial location to a final goal $\boldsymbol{x}_G$*. Assume that the learner's input space* $\boldsymbol{u_L} \in [\underline{\boldsymbol{u}}_{\boldsymbol{L}}, \overline{\boldsymbol{u}}_{\boldsymbol{L}}] \subset [\underline{\boldsymbol{u}}_{\boldsymbol{T}}, \overline{\boldsymbol{u}}_{\boldsymbol{T}}]$*; design a motion planning policy* $\boldsymbol{\pi}_{\boldsymbol{T}}^{L}$ *for the teacher that considers the limitations of the learner and such that the desired trajectory calculated* τ *can be tracked by the learner, i.e., such that* $\|\boldsymbol{x}_L - \boldsymbol{x}_\tau\| \leq \epsilon$ *where* ϵ *is a maximum allowable deviation threshold.*

We assume that the learner has similar kinematics as the teacher but is less capable. By "similar", we mean that they share the same configuration space $\mathscr{C}_L = \mathscr{C}_T$ and task space $\mathscr{T}_L = \mathscr{T}_T$, and their kinematic models exhibit a fundamental similarity in structure and governing equations. Although differences in parameters or external influences underline their distinctions, a shared mathematical foundation emphasizes their conceptual similarity. This similarity provides a basis for applying methodologies across both systems. For example, both systems might be based on the same dynamic

models but with distinct parameter sets, or they share identical parameters that fail to precisely reflect the actual behavior of the learner system. Regarding "less capability", this implies that the learner's command space is a subset of the teacher's $\boldsymbol{U}_L \subset \boldsymbol{U}_T$. For instance, the learner may not be able to drive as fast or turn as sharply as the teacher. This allows the teacher to perform all the learner's maneuvers but not vice versa. The assumption aligns with our focus on transferring knowledge from the simulated system into a real-world vehicle, as a virtual system can often be designed to surpass the capabilities of its real counterpart in sim-to-real problems As conformal mapping is well-established in two-dimensional spaces but becomes significantly more complex and restricted in higher dimensions, consequently, we focus on the systems whose input spaces are either inherently two-dimensional or can be bijectively represented in two dimensions, denoted as $\boldsymbol{u} \subset \mathbb{R}^2$. Despite this, there is potential for extending our conformal mapping-based approach to higher-dimensional spaces to accommodate the requirements of more complex input spaces.

4.3 Methodology

To address the formalized problem, we propose a conformal-mapping-based transfer learning framework. The block diagram in Figure. 4.2 presents the architectural overview of the proposed framework. The highlighted blocks are the main components for transferring the control inputs. Problem 4.1 is solved by leveraging SCM to conformally map the teacher vehicle's control input to obtain the learner's command. To tackle the Problem 4.2, the equivalent teacher commands, which produce identical motions as those observed from the learner, are used for constraining both the control and planning policies of the teacher. This ensures that the teacher's inputs used for mapping are tailored to accommodate the learner's capability. This section presents the details of the proposed framework, explaining how the two problems are solved.

4.3.1 SCM-based Command Transferring

For the challenges we aim to address, we assume the learner has similar dynamics as the teacher but is less capable. For instance, consider the impact of system aging, where the original, unaged system serves as the teacher and the aged system acts as the learner, unable to achieve the same level of performance. Since the teacher and learner represent the same system at different stages, their dynamics are fundamentally similar. However, a velocity command that once propelled the teacher vehicle effectively may result in diminished speed when applied to the aged learner

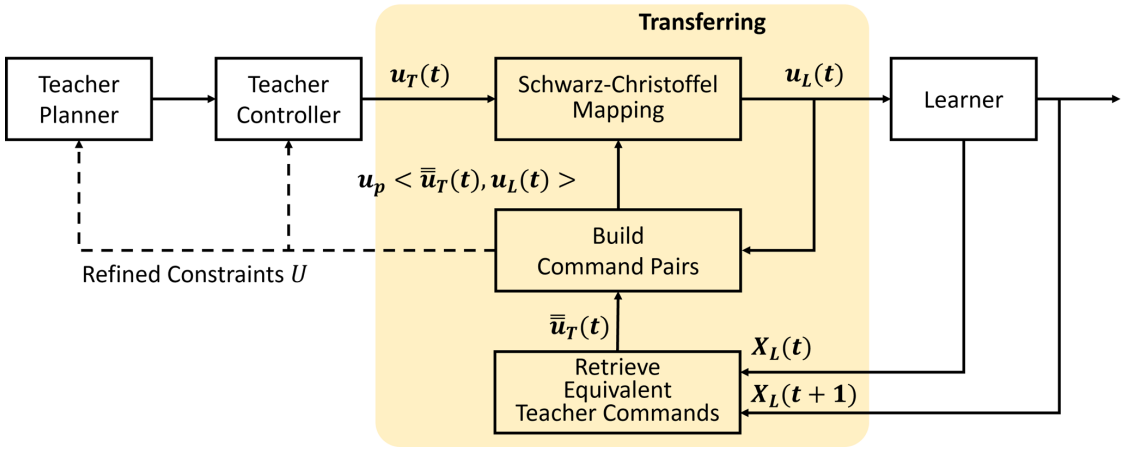


Figure 4.2: The block diagram of the proposed mapping-based transfer learning framework

system. This principle, when applied across all commands within the learner's command spectrum, suggests that the more limited, slower maneuvers now achievable by the learner form a subset of its capabilities prior to aging. By linking the learner's commands back to the teacher's that are capable of mirroring the learner's altered behavior, we can observe that the distribution of the teacher's equivalents is effectively "compressed" compared to the broader range of commands the learner used to execute before experiencing system aging. Inspired by this example case, it is critical to preserve the geometric information of the command domain to grasp the difference between the teacher and the learner. Thus, in this work, we leverage a conformal mapping method, renowned for its ability to preserve local geometries, including angles and shapes, to map the two command domains. The interconnected commands of the teacher and the learner, as highlighted in this example, are instrumental in determining the mapping function that accommodates the geometric alterations within the command domain. We define the two linked commands as command pairs. Formally, a command pair, $\boldsymbol{u_p}$, denotes a pair of a teacher's and a learner's control inputs that result in both systems executing the same maneuver:

$$\begin{aligned} & \boldsymbol{u_p} = \langle \bar{\bar{\boldsymbol{u}}}_{\boldsymbol{T}}, \boldsymbol{u_L} \rangle \\ \text{s.t.} \quad & f_T(\boldsymbol{x}_L(t), \bar{\bar{\boldsymbol{u}}}_{\boldsymbol{T}}(t)) = f_L(\boldsymbol{x}_L(t), \boldsymbol{u_L}(t)) \\ \text{where} \quad & \bar{\bar{u}}_T = f_T^{-1}(\boldsymbol{x_L}(t), \boldsymbol{x_T}(t+1)) \end{aligned} \tag{4.1}$$

An illustrative example of a set of command pairs is color-coded and presented in Figure 4.3. These command pairs are also used for characterizing the learner's dynamics. For example, in Figure 4.3, the learner component of the color-coded command pairs is located on the boundary of

the learner's command space, while the corresponding teacher component forms a convex hull within the teacher's command domain. The interior of the convex hull created by these command pairs effectively captures the dynamics of the learner within the teacher's command domain. Since it is impractical to learn all the command pairs all over the command domain. The proposed transferring framework relies on conformal mapping to find the rest of the command pairs by leveraging a limited number of command pairs.

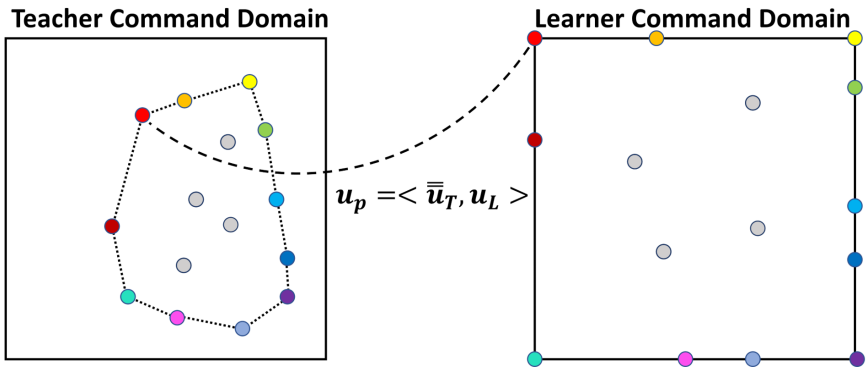


Figure 4.3: Examples of command pairs which are color-coded.

As we showcased at the beginning of this section, when transferring the input between systems, the process can be viewed as a geometric distribution transformation within two systems' command domains. Understanding this geometric alteration in the command space becomes instrumental in facilitating the transfer of control inputs that elicit similar or even identical maneuvers from both systems. Thus, the command pairs offer valuable insights into the result of such transformation. Inspired by this idea, the proposed transferring framework takes three steps to derive the desired learner command from the teacher command. First, the learner uses the teacher's control policy to generate a control input which is the teacher's desired command as if the learner were the teacher. Then, depending on user preference, the learner has the flexibility to choose multiple command pairs from the teacher's side, forming a poly-region that encompasses the teacher's desired command. The corresponding region on the learner's command domain is automatically determined by the learner's commands associated with the same command pairs as the teacher's vertices. At last, the Schwarz-Christoffel Mapping (SCM) is employed to conformally map the region in the teacher's command domain onto the region in the learner's domain. This strategic mapping allows us to pinpoint the precise learner command capable of producing maneuvers akin to those executed by the teacher in response to the desired command. The process is depicted in Figure. 4.4.

For ease of comprehension, we use a toy example to demonstrate the mapping process in this

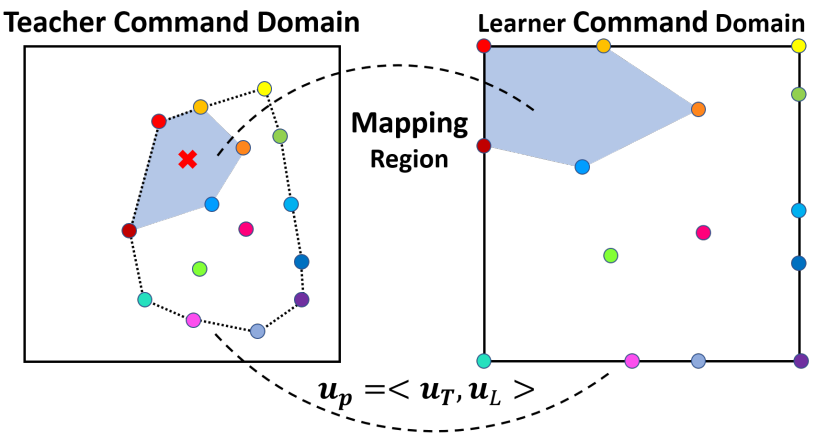


Figure 4.4: SCM maps the two polygon regions which are constructed by the command pairs around the desired command (red cross on the left).

section. In the example, the command domains are put onto the complex domain with the real axis representing the linear velocity and the image axis for the steering angle. In complex analysis, SCM is a technique on the complex domain for conformally mapping the upper-half plane onto the interior of a simple polygon, with the real axis mapped onto the polygon's edges as illustrated in Figure. 4.5. Specifically, let $\mathbb{P}$ be the region in the complex plane with w be a vertex on the boundary of the mapped region and $\alpha_k \pi$ as the interior angle. SCM theory allows us to map $\mathbb{H}^+$ to the interior of the $\mathbb{P}$ while the real axis is mapped to the boundary of the polygon region.

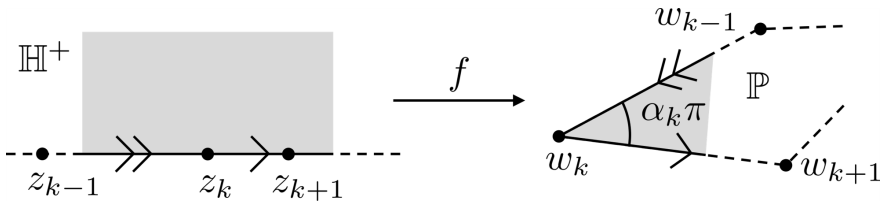


Figure 4.5: An example of SCM mapping the upper half plane to a polygon region.

Let f be a SCM function of the upper half-plane $\mathbb{H}^+$ onto $\mathbb{P}$, and let $z_k = f^{-1}(w_k)$ be the prevertices. The key of the conformal mapping is that the local angle is preserved, thus the fundamental of the Schwarz-Christoffel transformation is that

$$f' = \prod f_k \quad s.t. \quad arg\ f' = \sum arg\ f_k \tag{4.2}$$

where f is analytically continued across the segment (z_k, z_{k+1}) and each $arg\ f_k$ is designed to be a step function so that the resulting $arg\ f'$ is a piece-wise constant which reflects the angle jumps

along the perimeter of the polygon

$$[arg\ f'(z)]_{z^-}^{z^+} = \alpha\pi \tag{4.3}$$

To satisfy these requirements, the SCM function has a form of

$$f'(z) = A\prod_{k=1}^{n-1} f_k(z) \tag{4.4}$$

$$f_k = (z - z_k)^{\alpha_k} \tag{4.5}$$

In the context of this work, we employ a variant of the SCM theory which follows the same form and achieves a mapping from the interior of a polygon to a rectangle. Specifically, within our transfer learning framework, the polygons that are constructed on both sides undergo initial individual mappings to two separate rectangles with unique aspect ratios determined by the shape of the mapping areas. The rationale behind mapping the two regions onto distinct rectangles becomes evident as we delve into the mapping procedure. Subsequently, to connect the two mapped rectangles on both sides, we introduce a unit square as a bridging element. The overall mapping process is outlined in Figure. 4.6. To enhance understanding, we use an example to guide the reader through the application of SCM in the direct transfer of the teacher's control input to the learner.

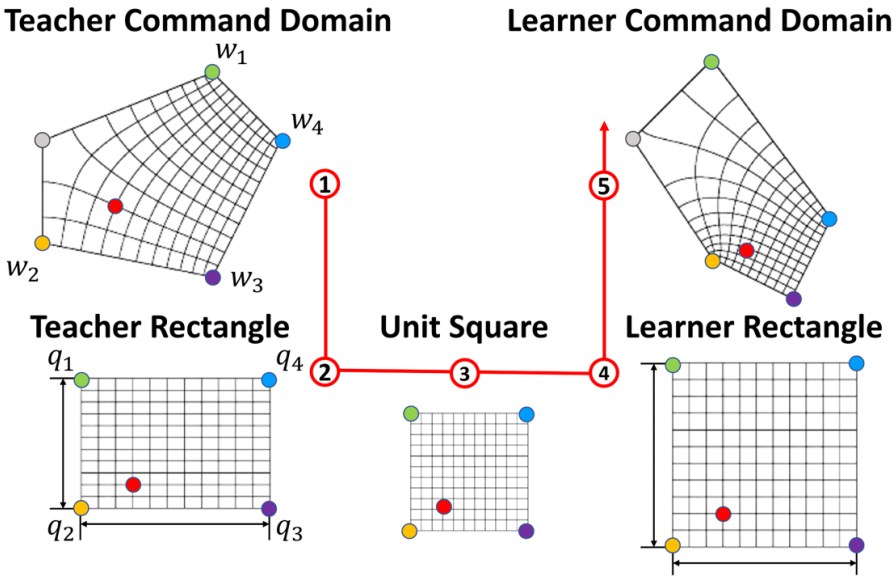


Figure 4.6: The mapping flow of transferring the desired teacher command to the learner. A unit square is used as an intermediate plane to bridge between the rectangles mapped from the polygons on each side.

With the basic conformal mapping concept in mind, in order to map a polygon to the rectangle,

SCM borrows an intermediate bi-infinite strip as a pivotal intermediary as shown in Figure. 4.7. We can treat the polygon as a generalized quadrilateral and select any 4 vertices in counterclockwise order as its four corners. These four vertices will eventually be mapped to the corners of the rectangle. The mapping function from a generalized quadrilateral to a strip is simply the inverse of it.

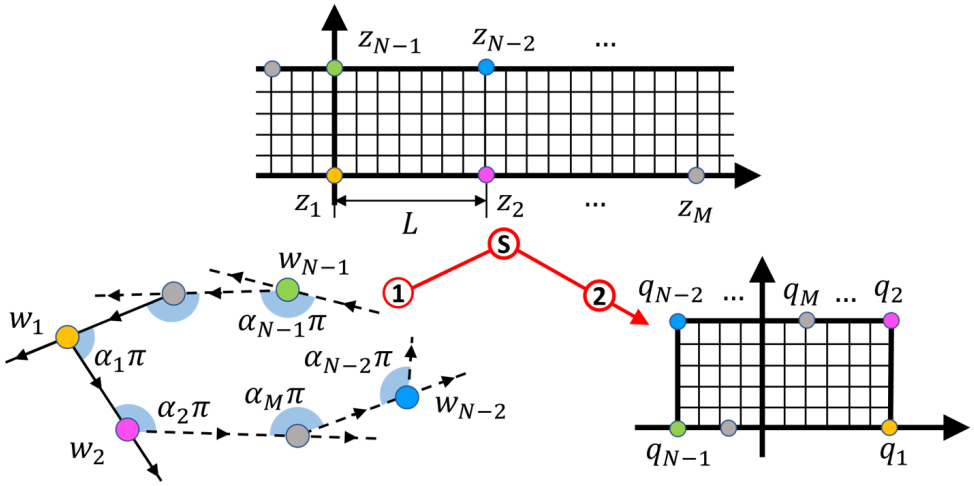


Figure 4.7: The flow of conformal mapping that maps the polygon to the rectangle while using the bi-infinite strip as the intermediate plane.

Consider an irregular polygon τ with vertices denoted as $w_1, ..., w_N$ (where $N \geq 4$), arranged in a counterclockwise manner as illustrated in Figure. 4.7. The interior angles at each vertex, $\alpha_1\pi, ..., \alpha_n\pi$, represent the angles formed by the edges originating from and terminating at that particular vertex. Initially, the vertices $w_1, ..., w_N$ are mapped to the prevertices $z_1, ..., z_N$ on the bi-infinite strip $\mathbb{S}$, and subsequently, they are mapped to the vertices $q_1, ..., q_N$ on the rectangle $\mathbb{Q}$. To compute the mapping function from a strip to a generalized quadrilateral, we introduce the Theorem 4.1.

Theorem 4.1 *Let $\mathbb{P}$ be the interior of a polygon with vertices $w_1, ..., w_N$ arranged in counterclockwise order ($N \geq 4$). $\alpha_1\pi, ..., \alpha_n\pi$ represent the interior angles at each vertex. Let f be any conformal map from a bi-infinite strip $\mathbb{S}$ to the polygon with $f(z_N) = w_N$. Then the Schwarz-Christoffel mapping from the bi-infinite strip $\mathbb{S}$ to $\mathbb{P}$ is:*

$$w = f_{\mathbb{S}}^{\Gamma}(z) = A \int_0^z \prod_{j=0}^{N} f_j(z)dz + C \tag{4.6}$$

where A and C are complex constants that rotate, translate, and scale the polygon and are determined by the shape and location of $\mathbb{P}$. Each factor f_j sends a point on the boundary of the strip to a corner of the polygon while preserving its interior angles. The factor f_j is a piece-wise function which is defined by:

$$f_j(z)=\begin{cases} e^{\frac{1}{2}(\theta_+-\theta_-)z} & j=0, \\ \{-i\cdot\sinh[\frac{\pi}{2}(z-z_j)]\}^{\alpha_j} & 1\leq j\leq M, \\ \{-i\cdot\sinh[-\frac{\pi}{2}(z-z_j)]\}^{\alpha_j} & M+1\leq j\leq N, \end{cases} \tag{4.7}$$

where M is the number of points on the bottom side of the strip. θ_+ and θ_- denote the desired divergence angles at $+\infty$ and $-\infty$, which are $\theta_+=\theta_-=\pi$ in our case. Please refer to [29] for proof.

By leveraging the Jacobi elliptic of the first kind [9, 66], the SCM mapping $f_{\mathbb{Q}}^{\mathbb{S}}$ from the rectangle $\mathbb{Q}$ to the bi-infinite strip $\mathbb{S}$ can be defined by:

$$z=f_{\mathbb{Q}}^{\mathbb{S}}(q)=\frac{1}{\pi}\cdot\ln(\sin(q|m)) \tag{4.8}$$

where q is the point on a regular rectangle and m is the modulus of the Jacobi elliptic that is decided by q. The details of this conformal mapping can be found in [16]. With Eqs. (4.6) and (4.8), a mapping function from the generalized quadrilateral can be obtained. In order to explicitly solve (4.6), there are three parameters z_k that must be specified. For ease of computation, for example, we can fix $z_1=0$, $z_2=L$, $z_{N-1}=i$, and $z_{N-2}=L+i$. The parameter L here is linked to the conformal modulus m.

While the angles of the polygon are computed with (4.6) and (4.7), we need to find where the pre-vertices lie on the boundary of the strip to keep the length for each edge of the polygon. This problem is known as the parameter problem in SCM [16]. Since we already fix $z_1=0$, in (4.6) the translation parameter is set to be $C=0$. Hence, solving (4.6) is equal to solving:

$$w_k=A\int^{z_k}\prod_{j=0}^{N}f_j(z)dz,\quad k=1,2,3,\ldots,N \tag{4.9}$$

In (4.9), the scalar A can be eliminated by the ratio of the adjacent sides length of the polygon:

$$\frac{w_{k+1}-w_k}{w_2-w_1}=\frac{\int_{z_k}^{z_{k+1}}\prod_{j=0}^{N}f_j(z)dz}{\int_{z_1}^{z_2}\prod_{j=0}^{N}f_j(z)dz},\ k=2,3,\ldots,N-2 \tag{4.10}$$

Let

$$I_k = \Big| \int_{z_k}^{z_{k+1}} \prod_{j=0}^{N} f_j(z) dz \Big|, \quad k = 1, 2, \ldots, N-2 \tag{4.11}$$

Then (4.10) can be rewritten as:

$$I_k = I_1 \cdot \frac{w_{k+1} - w_k}{w_2 - w_1}, \quad k = 2, 3, \ldots, N-1 \tag{4.12}$$

To this end, (4.12) leaves us $N-3$ conditions and the unknown parameters of (4.9) are $z_k\,(k = 1, 2, \ldots, N-3)$ which is exactly the number of the side length conditions given by (4.12) . We can get the complex constant A by:

$$A = \frac{w_2 - w_1}{\int_{z_1}^{z_2} \prod_{j=0}^{N} f_j(z) dz}. \tag{4.13}$$

As we get the conformal mapping function $f_{\mathbb{S}}^{\Gamma}$ from the strip to the generalized quadrilateral, we can compute $L = z_2 - z_1 = {f_{\mathbb{S}}^{\Gamma}}^{-1}(w_2) - 0$. Considering (4.8) which maps the rectangle to the strip, the SCM function that maps the interior and the boundary of the generalized quadrilateral to the rectangle with a unique aspect ratio can be obtained by:

$$q = f_{SCM}(w) = {f_{\mathbb{Q}}^{\mathbb{S}}}^{-1}({f_{\mathbb{S}}^{\Gamma}}^{-1}(w)). \tag{4.14}$$

As the shape of the rectangle $\mathbb{Q}$ depends on the parameter L, the aspect ratio of the rectangle is determined after L is computed. This explains why we map the two polygons from the teacher and the learner command domains to two different rectangles. Since the dynamics of the teacher and learner are different, the shape of the polygons from the teacher and the learner cannot be identical and neither are the mapped rectangles. A unit square is borrowed to bridge between the two mapped rectangles resulting in a complete mapping process from teacher to the learner, such that any teacher command that falls in the teacher's mapping area is connected to an image on the learner side.

4.3.2 Primitive Path Planning

As the vehicle learns the mapping function, it is also important to know the limitations of the learner so that the teacher's policy can generate the command to plan the motions that are compatible with the learner. This means that we want to find where the command boundary of the learner lies within

the teacher command domain. This can be achieved by getting the command pair $\boldsymbol{u_p}=\langle\overline{\overline{\boldsymbol{u}}}_T, \boldsymbol{u_L}\rangle$ when $\boldsymbol{u_L} \in \{\underline{\boldsymbol{u}}_{\boldsymbol{L}} \cup \overline{\boldsymbol{u}}_{\boldsymbol{L}}\}$. As shown in Figure. 4.4, the teacher's control inputs from these command pairs can build a multi-dimensional convex hull that separates the interior of the convex hull from the rest of the command area. From the teacher's perspective, the boundary of the convex hull indicates the limitations of the learner. Any of the teacher's commands from the interior of the convex hull can be matched with the learner's command, enabling the two vehicles to produce a similar motion with their own commands. However, to obtain better mapping performance, it is recommended to consider additional command pairs inside of the polygon.

We use a trajectory-tracking case study to validate our approach. The teacher uses a search-based path planning method to compose a sequence of motion primitives that allows it to drive along the desired path P within certain bounds. The teacher's input sequence associated with these primitives will be the desired commands for mapping.

A motion primitive results from feeding a known sequence of control inputs to the vehicle. To build one primitive $p=[\boldsymbol{x_T}_1, \boldsymbol{x_T}_2, \ldots, \boldsymbol{x_T}_t]$, we feed the teacher a sequence of the same control input for a certain amount of time and record its state sequence. Following the same procedure, a library of primitives can be built with different teacher's commands. In Figure 4.8, we show 5 different motion primitives that resulted from 5 different teacher's commands. The one-to-one primitives and the corresponding commands are color-coded. The command pairs are shown as the gray points and the white region indicates the capability of the learner.

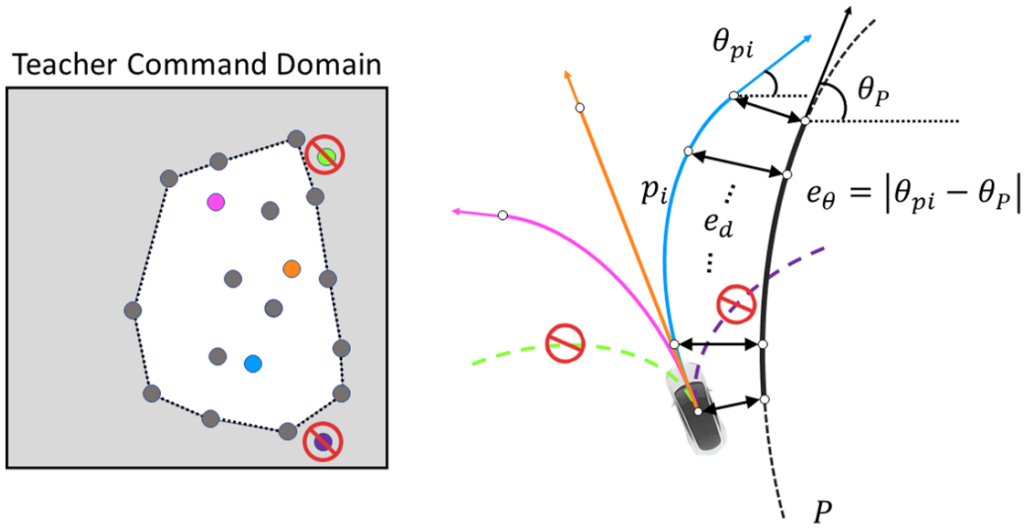


Figure 4.8: The teacher commands and the corresponding motion primitives are shown on the left while a path planning scenario is shown on the right.

We want to point out that: 1) To better adapt to the capability of the learner, only the command that falls inside of the convex hull should be considered. 2) The learner can leverage the teacher's motion planner as soon as the convex hull is built. 3) The convex hull does not need to capture the entire command domain of the learner, it just provides a boundary that makes sure the learner is operating within the known capability.

As the path planner searches primitives from the library to use, it evaluates the difference between each of the primitives and the corresponding segment on the desired path. As shown in (4.15) and in Figure. 4.8, the difference is measured by considering both the dynamic time warping (DTW) distance e_d and the heading difference e_θ at the end of the primitive:

$$\begin{aligned} \delta_i &= k_d \cdot e_d + k_\theta \cdot e_\theta \\ &= k_d \cdot DTW(P, p_i) + k_\theta \cdot |(\theta_P - \theta_{p_i})|, \\ p_i^* &= \min_{p_1,\ldots,p_i} \delta_i. \end{aligned} \tag{4.15}$$

The two types of differences are weighted by two user-defined gains ($k_d \geq 0$, $k_\theta \geq 0$). A large k_d will force the vehicle to remain close to the trajectory while a large k_t will give the primitives that are parallel to the trajectory a better chance to be chosen. Using these metrics, the planner searches through all the primitives in the library and selects the one with the least difference as the optimal local path plan p_i^*. The teacher's control input $\boldsymbol{u}_{\boldsymbol{T}}^*$, which is associated to p_i^*, is the command that will be mapped to the learner.

After a command sequence is executed, the learner will evaluate the situation and use the planner to generate a new local path and corresponding command sequence. The learner will continue to repeat this planning procedure until it arrives to the destination.

Since the learner has differing dynamics from the teacher, as the learner executes the command sequence to follow the composed path, it may deviate from it. When the learner is in an open area, such deviation is not critical because the command sequence only lasts a short period of time and it can always be corrected by the planner at the next planning step. However, such deviation can compromise the safety of the learner when it maneuvers in a cluttered environment. To provide safety guarantees to the system, we introduce an event-triggered mechanism to monitor the learner at runtime. The runtime monitor measures the distance between the learner and the planned path $d_{\hat{e}}$. The re-planning procedure is triggered when $d_{\hat{e}} > \epsilon$. The smaller that the threshold ϵ is, the more conservative the learner behaves. As we discussed, the learner does not need to constantly re-plan if

the deviation happens in an open area. Thus, the threshold ϵ should be dynamically changed to reflect how crowded the surroundings are. In our work, the threshold is defined as:

$$\epsilon = \begin{cases} \eta * \min(||p - o_i||) & i = 1, 2, \ldots, N_o, \\ \infty & i = \varnothing, \end{cases} \tag{4.16}$$

where N_o is the number of obstacles in the learner's field of view, o_i is the position of obstacle i, and η is a constant.

4.4 Simulations

For the simulations, we validate our transferring framework through a case where the vehicle suffers from compromised dynamics. The kinematics for the vehicle is given by the following bicycle model:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} + \Delta t \begin{bmatrix} v_k \cos\theta_k \\ v_k \sin\theta_k \\ \gamma_k \end{bmatrix}, \quad \boldsymbol{u}_k = \begin{bmatrix} v_k \\ \gamma_k \end{bmatrix} \tag{4.17}$$

where v and ω denote the linear and angular velocities respectively. The teacher, represents the original vehicle with full capabilities, while the learner refers to the vehicle after being compromised. The command ranges for both the teacher and the learner in this study case are detailed in Table 4.1. For ease of implementation, we consider the system to have small inertia thus the acceleration periods are ignored (e.g., an electric vehicle). A Gaussian noise of $G \sim \mathcal{N}(0, 0.1)$ is added to the learner's position to simulate measurement errors. The learner is asked to follow a "S"-shaped trajectory through a cluttered environment.

Table 4.1: Parameters for Motion Primitive Transferring Simulations and Experiments

		$\underline{v}$ (m/s)	$\overline{v}$ (m/s)	$\underline{\omega}$ (rad/s)	$\overline{\omega}$ (rad/s)
Simulation	Teacher	0	3	$-\pi/3$	$\pi/3$
	Learner	0	1	$-\pi/8$	$\pi/8$
Experiment	Teacher	0	1.6	-1.2	1.2
	Learner	0	1	-1	1

We use the normalized command domain for better visualization, and the domain is normalized based on the corresponding ranges in Table 4.1. As shown in Figure 4.9, a 5×5 grid of command

pairs is collected beforehand. The boundary of the command pairs on the teacher's command domain marks the limitation of the learner. Figure 4.10 shows all of the teacher's motion primitives alongside their corresponding commands, each generated by driving the teacher with a specific control input for $1s$. Out of the 121 motion primitives, 35 are retained for path planning after excluding those that exceed the learner's capabilities. For the path planner, we set the planning horizon to s=2 and the threshold to trigger re-planning as η=0.5.

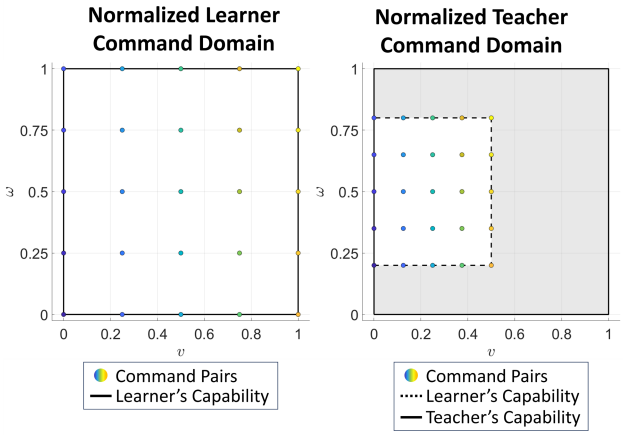


Figure 4.9: Command pairs for the simulation. The command pairs are one-to-one color-coded across the two command domains.

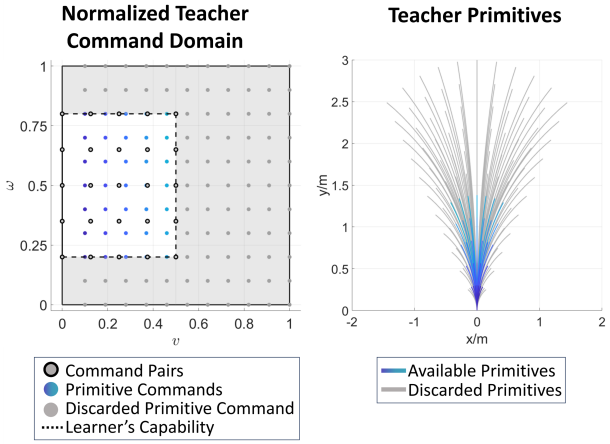


Figure 4.10: Example of learner's capability assessment and transfer of path planner. The preimitives within the learner's limit are preserved for path planning.

Figure 4.11 shows two snapshots from the simulation, illustrating that the learner closely follows the desired trajectory. For the teacher's path planner, we set the planning horizon at 2, meaning the local path consists of two primitives. The learner adopts a more conservative plan when obstacles are present within its field of view (FOV) as η is lower than the open space.

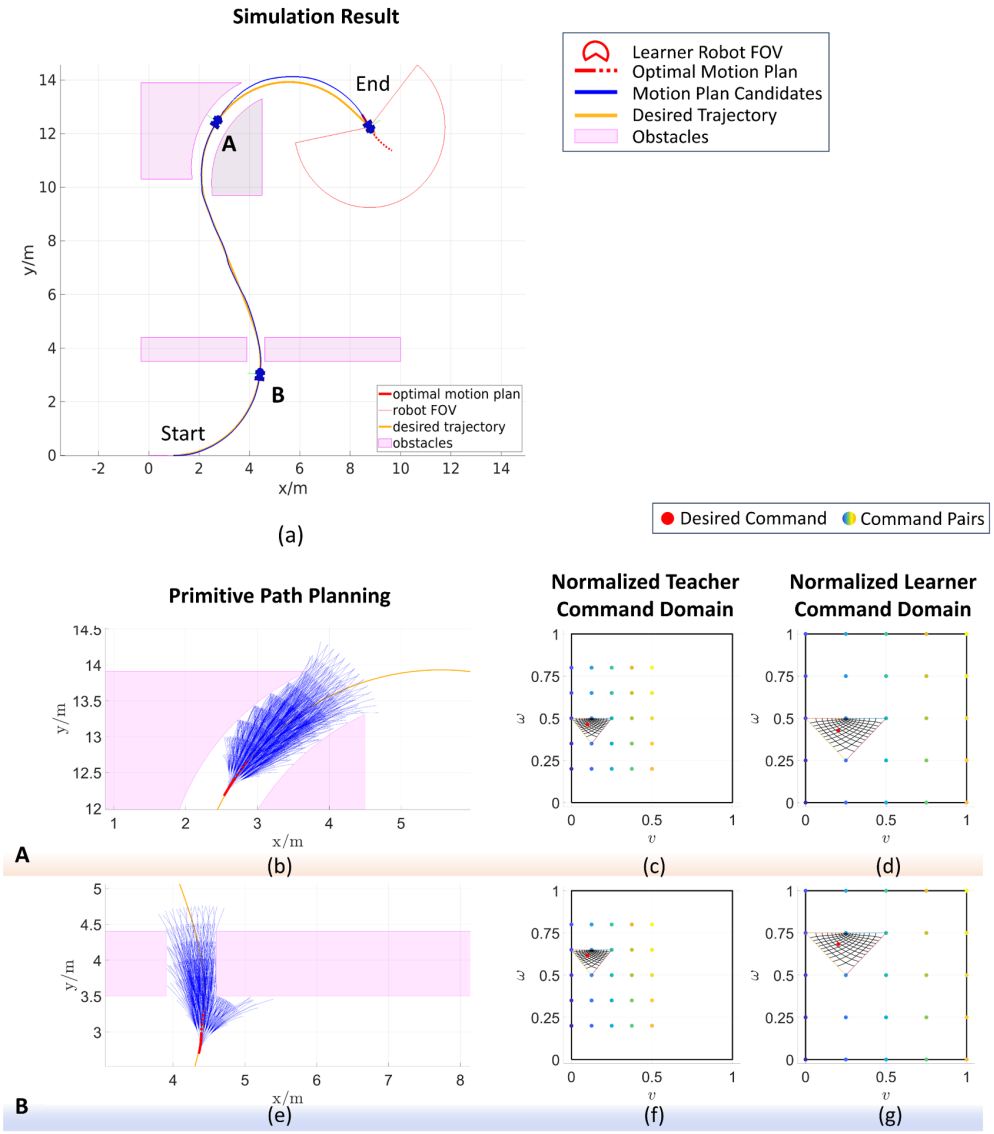


Figure 4.11: (a)The simulation result of learner following a "S"-shaped path. The local path planning and the SCM mapping results for the robot at position '**A**' are shown in (b), (c), (d), and the results at position '**B**' are shown in (e), (f), and (g).

In Figure 4.12, we show the result of the baseline learner for the same task. The baseline learner directly applies the teacher's commands, unaware of any necessary adjustments for its dynamics. As expected, the learner failed because it used commands not adapted to its new dynamics.

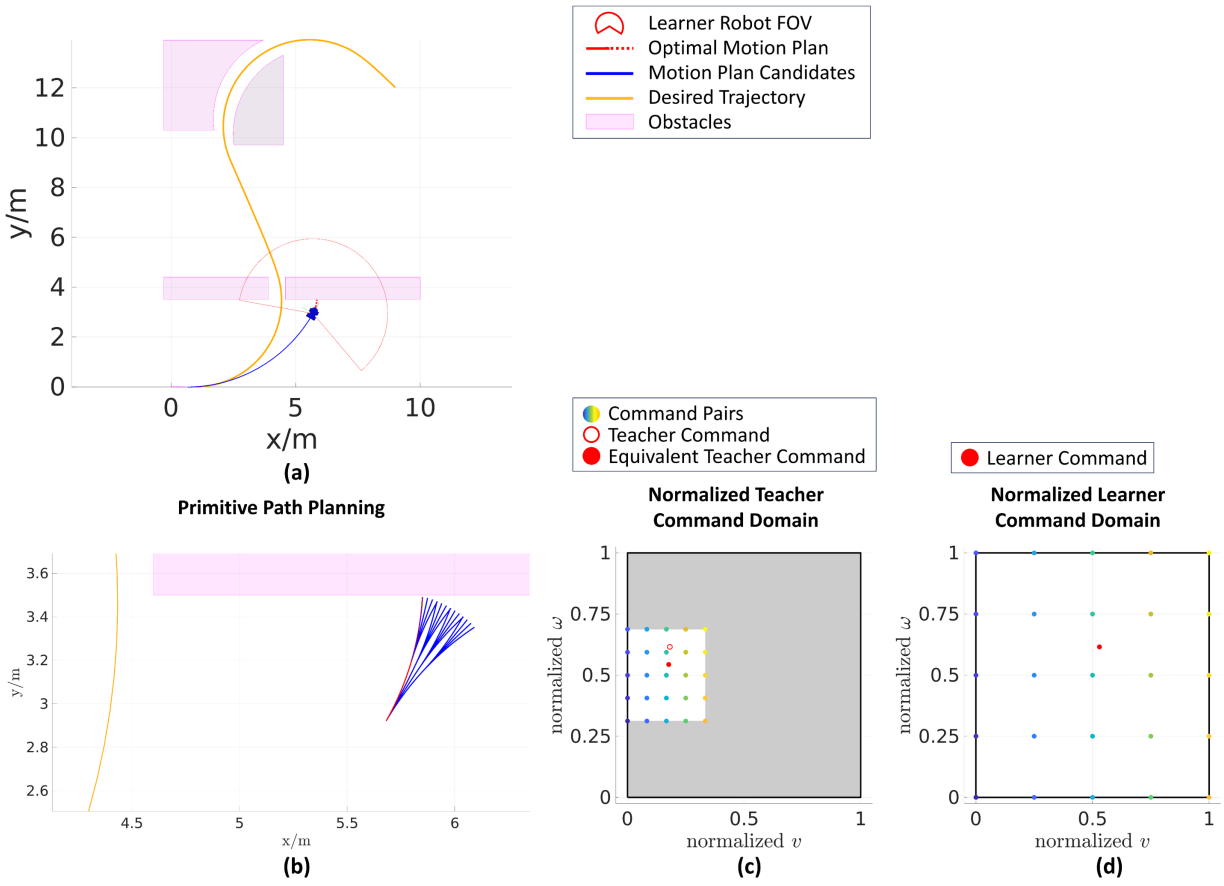


Figure 4.12: Simulation results for baseline learner. (b),(c), and (d) show the primitive path plan, the normalized teacher command domain, and the normalized learner command domain at the moment in (a).

4.5 Experiments

Our proposed transfer learning approach was validated by a set of experiments in which we transferred the planning and control knowledge of a simulated teacher into two real learner vehicles. The real experiment setup is similar to the simulated experiments and we used the same simulated teacher vehicle as the previous section. The commend ranges for the vehicles are listed in Table 4.1. The method is implemented using the MATLAB Schwarz-Christoffel toolbox [15] and vehicle control is managed via the MATLAB ROS Toolbox in conjunction with the Robot Operating System (ROS).

The experiments are conducted indoors, with the vehicles' states tracked by a VICON motion capture system. The architecture of the experiment setup is depicted in Figure 4.13.

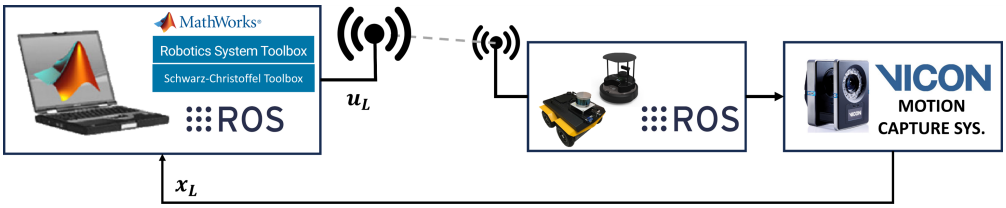


Figure 4.13: Architecture of the experimental setup.

As shown in Figure 4.14, Clearpath Jackal UGV is the learner vehicle for tracking an "S"-shaped path through a gate in the first experiment. The vehicle's capabilities are characterized by collecting command pairs through specific commands executed throughout $1s$. The command pairs and the teacher's primitives used for planning the learner's path are displayed in Figure 4.15. The Jackal's capability is indicated within the white area. The gray points on the dashed boundary are the commands that were tested on the Jackal for extracting the limitations. The blue-colored commands on the left create the primitives on the right and are used for mapping to the real UGV. To assess the robustness of our proposed approach, the learner's initial heading is set with a $\frac{\pi}{4}$ offset from the desired orientation. During the tracking task, the maximum distance recorded between the desired path and the actual trajectory was $0.1905m$, while the maximum deviation between the actual trajectory and the local motion plan was $0.0293m$. Given the vehicle's initial misalignment with the desired path and its dimensions of approximately $0.5m \times 0.43m \times 0.25m$, this deviation is considered negligible. For comparison, the baseline experiment was conducted without the SCM component and the results are shown in Figure 4.16.

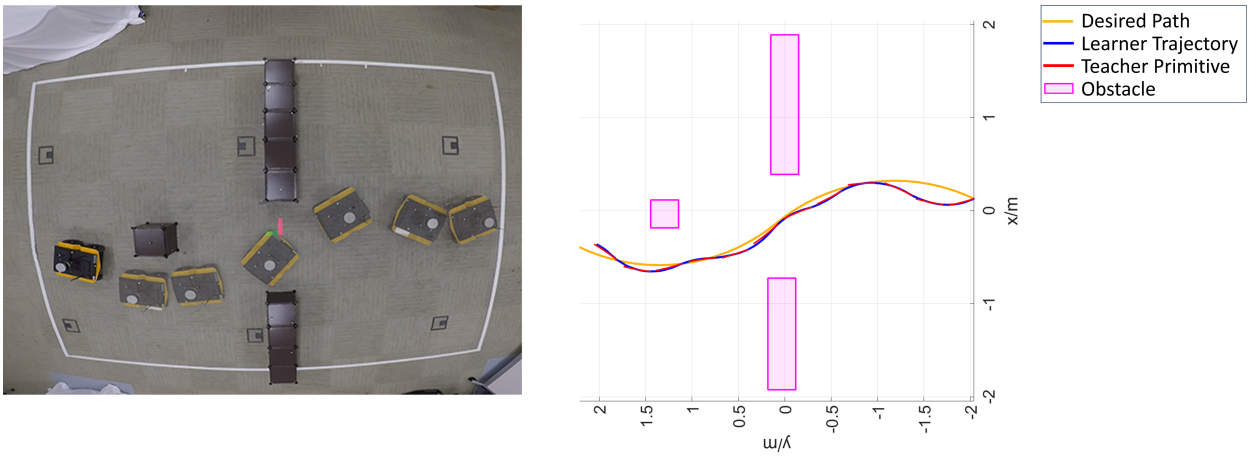


Figure 4.14: Jackal experiment with SCM.

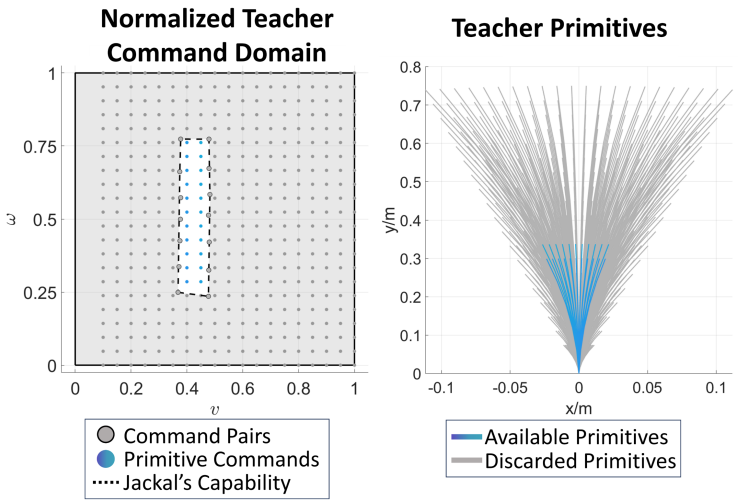


Figure 4.15: The Jackal operational limit calibrated on the simulated teacher command domain and the motion primitives used for planning the path.

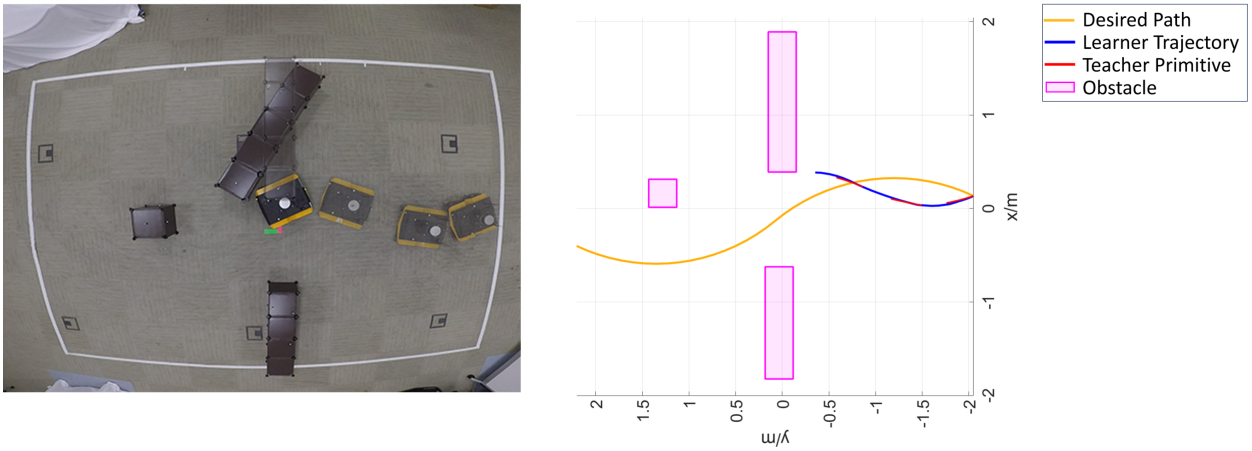


Figure 4.16: Jackal experiment results for directly applying the teacher's commands.

To demonstrate the generalizability of our proposed framework, we conducted an additional experiment using Turtlebot2 as the learner with the same learner configuration listed in Table 4.1. The results indicate that, with our proposed approach, Turtlebot2 successfully adapted the teacher's controller and path planner to follow the desired path with a maximum deviation of 0.1381 m. The tracking error between the learner's trajectory and the planned primitive remained within 0.0978 m, demonstrating close alignment as illustrated by the nearly overlapping path plans and final trajectory in Figure 4.17. The command pairs, Turtlebot2's capability, and primitives used for planning the path that is compatible with the limit of Turtlebot2 are depicted in Figure 4.18. We observed that the Turtlebot2 demonstrates a smoother trajectory compared to the Jackal experiment,

albeit with larger deviations. This smoother trajectory can be attributed to the Turtlebot2's lower maximum speed relative to the Jackal, as the comparison shown in Figure 4.18. Consequently, the Turtlebot2 tends to overshoot less from the desired "S"-shaped path. The relatively larger deviation arises because the selection of motion primitives aligned with the desired path is prioritized over minimizing deviations from that path. A closer tracking can be achieved by tuning k_d and k_θ. We use the same weight parameters for both UGV experiments to remain consistent.

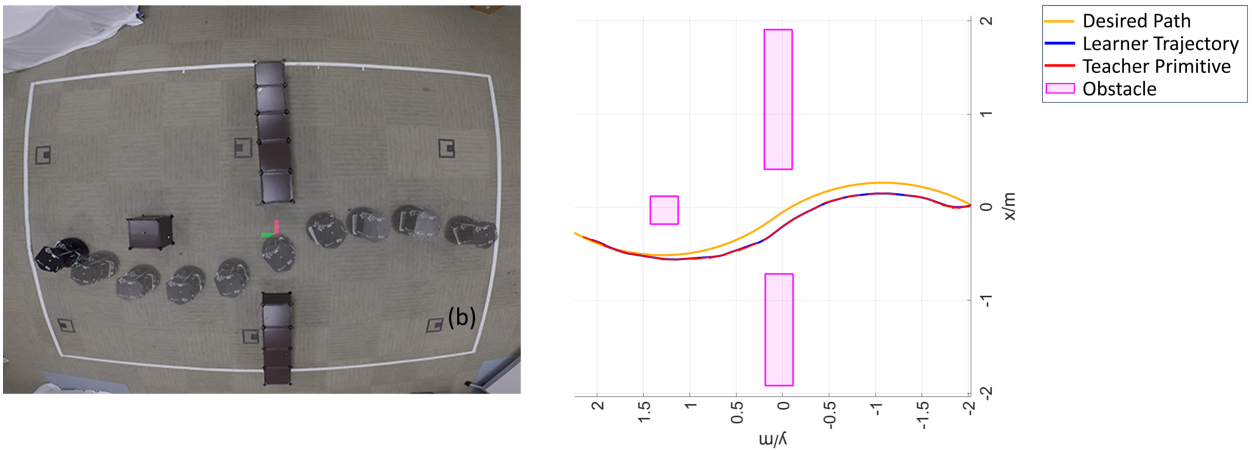


Figure 4.17: Turtlebot2 experiment with SCM.

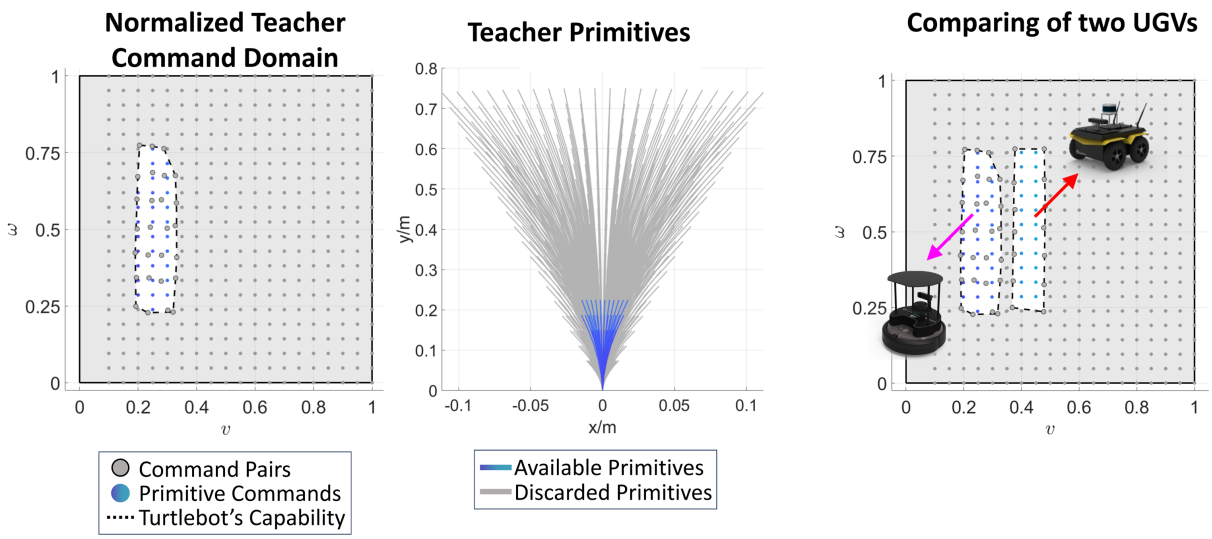


Figure 4.18: The Turtlebot2 operational limit calibrated on the simulated teacher command domain and the motion primitives used for planning the path.

4.6 Discussion

In this chapter, we proposed a novel lightweight transfer learning framework based on conformal mapping. We illustrated in detail how to explicitly solve the SCM problem and utilize the SCM theory in our application. The SCM is used to directly map the control input from the teacher to the learner without explicitly knowing the dynamical model of the learner. The framework transfers not only the control policy but also adapts the teacher's motion planning policy to make it compatible with the learner. We integrated a motion primitive-based planning method in the proposed framework to show that the learner can safely adapt the control and motion planning policy to suit its own dynamics.

Limitations:

One of the primary limitations of transferring path plans based on motion primitives is that the desired teacher commands are tied to these primitives. Given that the motion primitive library contains only a limited selection of primitives, and it is impractical to construct a comprehensive set of all possible motion primitives, the range of available teacher commands is restricted to a discrete set. This limitation presents an opportunity for enhancement by enabling the generation and transfer of commands within a continuous command space. Furthermore, during each control loop, the vehicle receives a sequence of commands associated with the optimal motion primitive. The vehicle maps each command in the sequence and only begins to generate the next sequence after the current one has been fully mapped and executed, or when a replan is triggered. The length of the command sequence, which depends on the motion primitives built beforehand, can hinder the system's ability to respond promptly to dynamic conditions.

Future Directions:

The transfer framework presented in this chapter has demonstrated its effectiveness in transferring control and path planning policies between two similar systems. Despite its success, we have identified several limitations. Next, we aim to enhance the framework by enabling the transfer of control inputs across a continuous command space. Additionally, the current framework requires a dedicated calibration stage to configure the learner's operational limits within the teacher's command domain prior to the transfer stage. To improve this, we plan to develop a learning approach that allows for the active refinement of the learner's capabilities without the need for a dedicated calibration stage, thereby enabling spontaneous configuration and transfer of knowledge.

Chapter 5

Conformal Mapping-based Transfer Learning for Continuous Control Inputs

In this chapter, we enhance our Schwarz-Christoffel Mapping-based transfer framework to address the limitations identified in the previous chapter. We demonstrate how to transfer control inputs over a continuous command domain and actively refine the learner's capabilities by integrating an MPC with our framework. In this dissertation, we refer to the variations in system responses due to process noise and environmental disturbances as "motion noise" when a command is applied to a system. One advantage of the motion primitive-based control and motion planning method introduced earlier is that it allows for the averaging out of motion noise. This is achieved through the repetitive execution of the same control input via a selected motion primitive, facilitating the retrieval of equivalent teacher commands for this learner command when constructing command pairs. However, with the MPC, as each control step could yield a different control input across the entire continuous command space, motion noise presents significant challenges in retrieving the equivalent teacher command for forming the command pairs. To enhance the robustness of our proposed framework, we refine the definition of command pairs to mitigate inconsistencies caused by motion noise, where the same learner command may correspond to different equivalent teacher commands. Furthermore, we design a method to strategically shrink the command domain to actively approximate the learner's capabilities within the teacher command domain. This allows us to eliminate the dedicated calibration stage before the transferring stage, enabling us to commence transferring without prior knowledge of the learner's capabilities. The material covered in this chapter has been submitted for review in the Journal of Intelligent & Robotic Systems (JINT).

- S. Gao, and N. Bezzo, "A Conformal Mapping-based Framework for Sim-to-Real Transfer in Autonomous Mobile Robot Operations." *Journal of Intelligent & Robotic Systems(under review), 2024.*

5.1 Introduction

In recent years, the advancements in simulation technologies have led to a significant surge in robotic research and applications [67, 13]. Simulations provide a low-cost, virtual proving ground for designing and controlling robots, allowing for rapid prototyping and testing without associated risks [1]. However, despite the flawless performance of behaviors and algorithms in simulated settings, the "reality gap" between simulated and real environments and inherent discrepancies in robotic models often lead to performance discounts or even failures when directly applying them in the real world. Despite considerable time and resources devoted to creating techniques within simulations, researchers usually still face formidable challenges when applying these methods to specific platforms in the real world. Thus, closing this gap is essential for advancing the practical deployment of robotic systems in diverse fields such as soft robots [37], agriculture [52], and service industries.

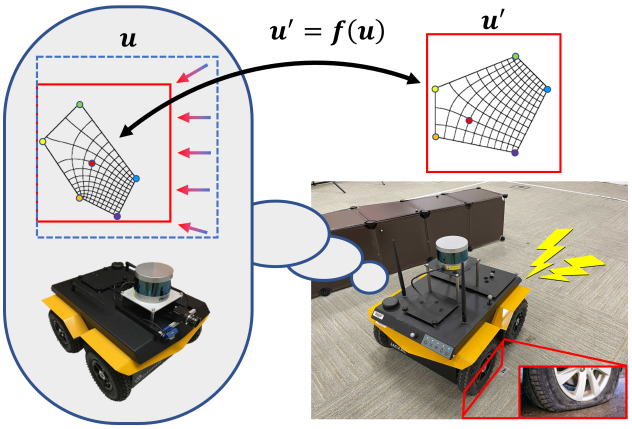


Figure 5.1: Pictorial representation of a robot suffering from a flat tire and actuator fault after being deployed into a real-world environment. By quickly understanding the dynamic differences between the current situation and the ideal model, the robot is able to find a proper mapping function that maps the desired command to adapt to the new dynamics of the vehicle.

Moreover, understanding how to effectively bridge this gap contributes to solving broader domain-transfer problems. For instance, consider the impact of system aging. Although systems before and after aging are different stages of the same system with fundamentally similar dynamics, a velocity command that once effectively propelled the vehicle might result in diminished speed due to wear and tear. This type of gap, while distinct from the sim-to-real discrepancy, belongs to the broader category of model mismatches. Similar problems are not limited to mechanical aging, but can also be found in deployment environment changes, undergoing external disturbances, and

even failures that deprecate and modify the system's original model. In this chapter, we seek a general framework to transfer and adapt the system's performance from one vehicle to another on a continuous command domain.

All of the aforementioned problems can be simplified and cast as a *teacher* transferring the control and motion planning policies to a *learner* similar to the previous chapter. In this chapter, we focus on achieving the control transferring on the continuous command domain and eliminating the dedicated calibration stage. To address this transferring problem we refine our lightweight, conformal mapping-based transfer learning framework. The proposed framework maps directly the control inputs of teachers to learner systems, avoiding learning their dynamic models. The framework also learns and considers the learner's ability, so that the transferred motion plan is achievable by mapped control inputs. We aim to provide a robust solution to the pervasive problem of model mismatch in robotics, enhancing both the efficacy and efficiency of deploying robotic applications. Overall, the contribution of the work captured in this chapter is twofold:

1. **Control Inputs Transferring**: We present a lightweight transfer framework, utilizing SCM theory for the direct transfer of control inputs from the teacher to the learner. This innovative approach allows the learner to adopt the teacher's control policies without the need to understand its dynamics.

2. **Model Predictive Control Transfer**: We incorporate a Model Predictive Controller (MPC) as a unified teacher controller and path planner within the proposed transfer framework. The framework imposes constraints on the MPC to ensure that the optimized control input remains within the learner's operational limits. These inputs are then mapped to the learner, allowing it to mirror the teacher's movements in a continuous control space.

In addition to the contributions mentioned above, we validate the proposed sim-to-real migration framework with extensive simulations and real-world experiments. To our knowledge, the work covered in this chapter is pioneering in leveraging the conformal mapping method for transferring control and motion planning policies between robotic systems. Distinct from existing literature, our work bridges the sim-to-real gap by focusing exclusively on the domain of control inputs, rather than attempting to learn specific model discrepancies. By directly translating control inputs, we can achieve a lighter-weight seamless transfer that avoids the need for extensive data collection and training to learn the precise model mismatch or to customize the controller and the planner.

5.2 Continuous Space Transferring

One of the primary limitations of motion primitive-based path planning transfer is that the commands available for mapping are constrained by the number of motion primitives in the library. Once the motion primitive is selected through an exhaustive search, the learner must map and execute the associated command sequence, either until completion of the current sequence or until an event-triggered re-planning intervenes. This results in the learner having only a limited number of discrete commands to choose from. Additionally, the fixed length of the command sequence may hinder the learner's ability to quickly adjust to deviations. Furthermore, significant effort is required to obtain the motion primitives and calibrate the learner's capability in advance. To address these limitations, we introduce a case study that incorporates a receding horizon controller within our proposed transfer framework. Specifically, we utilize Model Predictive Control (MPC) as a unified method for both control and planning for the teacher. Additionally, we demonstrate how to dynamically learn and adjust the learner's capabilities from scratch without separating it from the transfer process.

5.2.1 Model Predictive Controller

The simulated teacher uses the same model as introduced in (4.17). The teacher exploits an MPC that tracks the desired path while avoiding the obstacles. The cost function and the optimal control problem are formalized as (5.1) and (5.2).

$$\ell(\boldsymbol{x}, \boldsymbol{u}) = \|\boldsymbol{x} - \boldsymbol{x}^{ref}\|_Q^2 + \|\boldsymbol{u} - \boldsymbol{u}^{ref}\|_R^2 \tag{5.1}$$

$$\begin{aligned} \min_{u} \quad & J_N(\boldsymbol{x}_0, \boldsymbol{u}) = \sum_{k=0}^{N-1} \ell(\boldsymbol{x}_k, \boldsymbol{u}_k) \\ s.t. \quad & \boldsymbol{x}_{k+1} = f_T(\boldsymbol{x}_k, \boldsymbol{u}_k) \\ & \|\boldsymbol{x}_k - \boldsymbol{x}_{\mathcal{O}i}\| > r_i, \forall i \in [0, M] \\ & \boldsymbol{x}_k \in X, \forall k \in [0, N] \\ & \boldsymbol{u}_k \in U, \forall k \in [0, N-1] \\ & \boldsymbol{u}' \leq \epsilon \end{aligned} \tag{5.2}$$

where N and M denote the predicting horizon and the number of obstacles respectively. $\boldsymbol{x}_{\mathcal{O}}$ and r denote the position and the radius of the obstacle. We constrain the changing rate of the input as we ignore the acceleration period out of simplicity. The reference states $\boldsymbol{x}^{ref}$ are sampled along the desired path, beginning at the point closest to the teacher. Each subsequent point is spaced at the maximum distance the teacher can traverse along the path within one timestep, optimizing the time needed for completing the task. During the early stages of the transfer process, users have the flexibility to adjust the spacing between these reference states. This allows the teacher to perform a variety of maneuvers, which aids in populating the command pairs across the teacher's command domain. Whenever the command pairs are updated, leading to a refinement in the configured learner's capability on the teacher's command domain, the constraints on U are revised to align with the updated operational boundaries of the learner. This adjustment ensures that the optimized control inputs consistently remain within the defined capabilities of the learner.

5.2.2 Refinement of Command Pairs

When constructing command pairs, the process of identifying equivalent teacher commands is relatively straightforward in scenarios with subtle motion noise, as the same learner commands typically result in almost identical motions and, consequently, nearly identical equivalent teacher commands. However, when motion noises are non-negligible, variations in motions occur even with identical commands. Such discrepancies lead to inconsistent equivalent teacher commands for the same learner command, posing challenges in retrieving the correct equivalent teacher command.

To address this issue, we refine the definition of command pairs by first partitioning the normalized learner command domain into grid cells. Subsequently, the previously defined command pairs are grouped into clusters based on whether their learner components are within the same cell. Each refined command pair is then derived by averaging those within the same cluster. To mitigate motion noise, a user-defined minimum cluster size k_{min} is introduced for constructing a command pair. While increasing k_{min} enhances the precision of the command pairs, it extends the time required for construction. Figure 5.2 demonstrate the refined command pairs.

5.2.3 Control Transfer

If the learner's capabilities have already been characterized, transferring the teacher's command to the learner resembles the previous case, except the available teacher commands are within the continuous space constrained by the learner's boundaries. In scenarios with few or no pre-learned

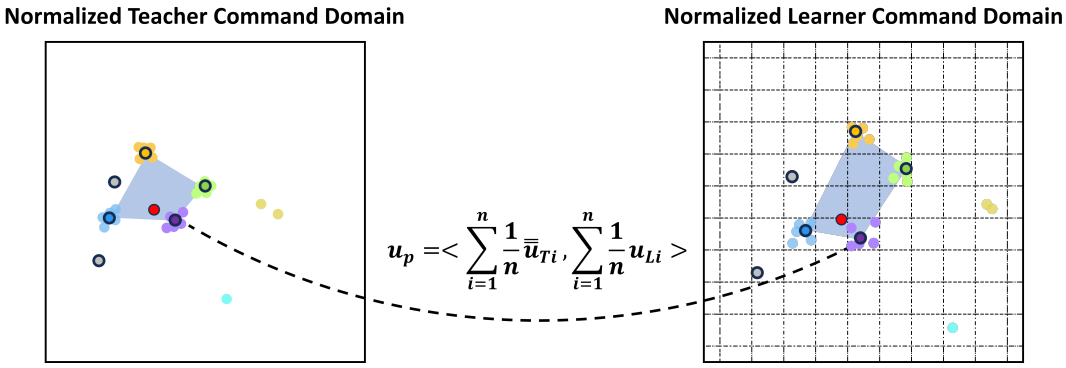


Figure 5.2: An example of the refined command pairs and the corresponding clusters. Command pairs are color-coded and shown in circles with black edges.

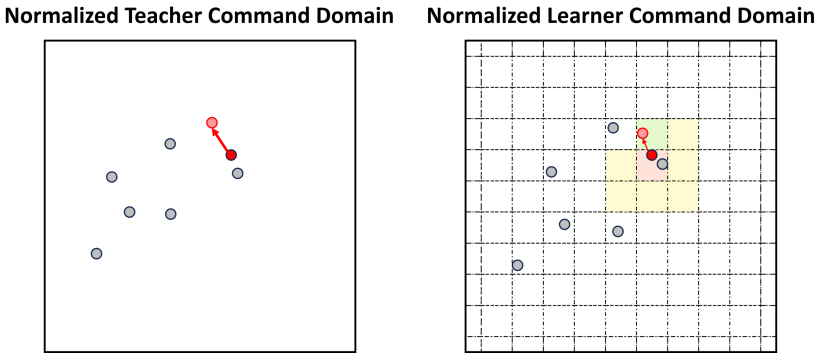


Figure 5.3: An example of perturbing the desired learner command to one of the adjacent empty command cells.

command pairs, to facilitate accurate control transfer, it is desirable to construct as many command pairs as possible in a timely manner. For learners with non-negligible motion noise, this process can be time-consuming because it involves collecting more linked learner-teacher commands to create each pair. To address this, without losing generality, we offer strategies to accelerate the construction of command pairs to compensate for the increased effort required in collecting more commands.

Our comprehensive framework with these enhancements is shown in Figure 5.4. We emphasize the components that help in scenarios where existing command pairs are insufficient. When existing command pairs cannot form a polygon area that includes the desired command, the cell containing the desired command may or may not already have a configured command pair. If no command pair is configured in the cell, the learner directly uses the teacher's command, as it has not yet learned the local geometric differences across the command domains between the systems. Conversely, if a command pair is already configured in the cell, learners are encouraged to explore neighboring

cells by slightly perturbing the desired command into adjacent, unconfigured cells. Figure 5.3 demonstrates this technique, where the desired command is perturbed to expedite the collection of command pairs. In the example, existing command pairs are represented as gray dots. From the 8 adjacent unconfigured cells (yellow), one is selected (green) for random sampling of a new learner command. Note that the extent to which the desired command is perturbed is a trade-off between learning more command pairs and deviating more from the desired behavior. The degree of perturbation is ultimately determined by user preference.

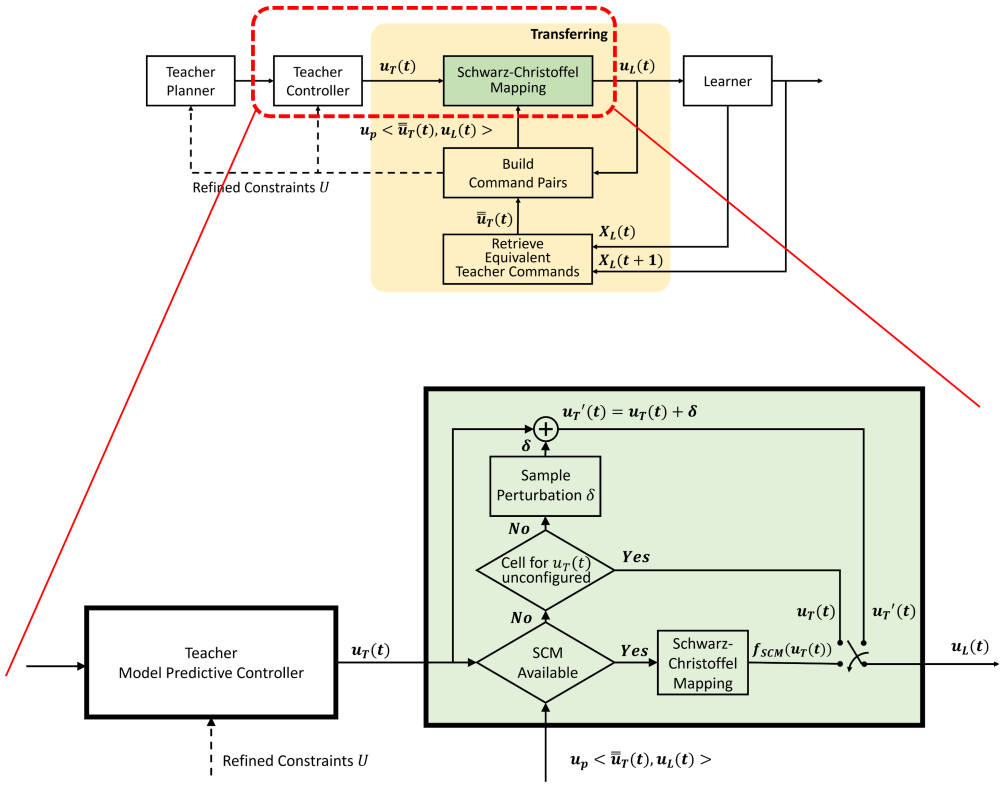


Figure 5.4: The block diagram of the proposed SCM-based learning framework without pre-learned learner's capabilities.

5.2.4 Path Planning Transfer

The motion limit of the learner is associated with the commands residing on the boundary of the learner's command domain. To characterize the learner's capability within the teacher's command domain, we leverage command pairs with the learner component on its boundaries. The equivalent teacher commands from these pairs form a convex hull that effectively indicates the learner's capabilities. Commands within the interior of the convex hull correspond to feasible motions for the learner. Figure 5.2 demonstrates this method. Establishing such a convex hull provides evidence for

imposing constraints on the teacher's controller and the path planner, ensuring that the commands generated remain within the hull, thus making the path achievable and able to be accurately followed by the learner. If the command pairs employed for constructing the convex hull are not confined to the learner's boundary, it suggests the transfer is not fully exploiting the learner's potential, as the corresponding area of the convex hull does not encompass the entire learner's command domain.

Conservatively, a straightforward but conservative approach is to assess the learner's capability before transferring, isolating it from the transfer process through a dedicated calibration stage. During this calibration phase, a series of extreme learner commands are imparted to the learner to assess its motion limits. Upon the conclusion of the calibration stage, the characterized learner's capability is finalized and is used for imposing the constraints on the teacher's controller and planner.

Alternatively, another approach is to dynamically adapt the boundary of the learner's capability whenever it is needed along with the transferring process. When a new command pair, which behaves as an outlier in comparison to all existing command pairs, is introduced, it triggers a recalculation of the boundary in the teacher's command domain. On the one hand, if the newly established command pair resides on the boundary of the learner's command domain, the equivalent teacher command marks the maximum or minimum boundary of allowable teacher commands. Subsequently, all other existing normalized learner command pairs are also re-scaled to account for the adjustments in the command range. Figure 5.5(a) presents an example for this case, where the yellow dots indicate the newly added command pair. On the other hand, if the new command pair outlier does not align with the boundary of the learner's command domain, the range of permissible teacher commands undergoes proportional reduction. This proportional adjustment can be computed using (5.3) and is visually demonstrated in Figure 5.5(b).

$$\overline{\overline{u}}_{T_{norm}|max} = \frac{\overline{\overline{u}}_{T_{norm}}}{u_{L_{norm}}} \tag{5.3}$$

5.3 Simulation Results

The initial command ranges for the teacher and the learner are outlined in Table 5.1. During the transferring process, the teacher's command range corresponds to the range of the refined learner's capability. We intentionally unbalanced the turning abilities towards both sides to increase the discrepancies between the two systems.

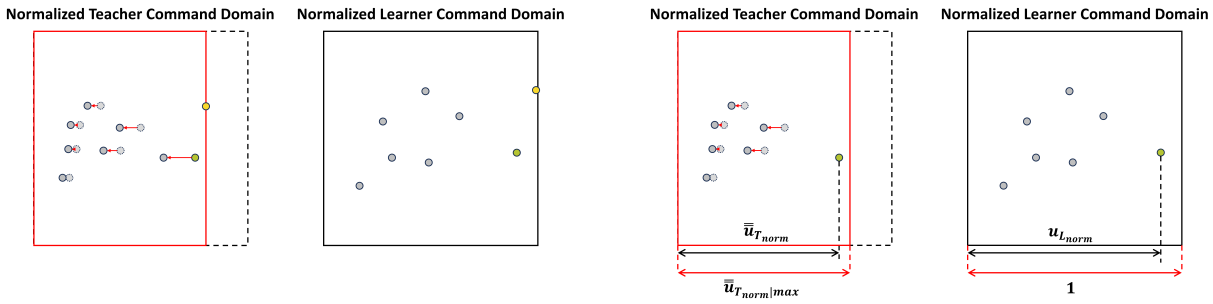


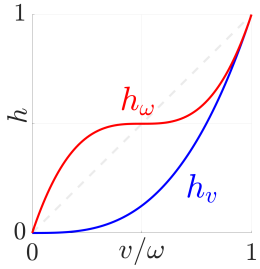
Figure 5.5: Examples of characterizing learner's capabilities on the teacher command domain. (a) the new command pair directly marks the boundary of allowable teacher command; (b) proportionally shrinks the teacher's command space to approximate the learner's capability as the learner portion of the command pair is an outlier but not on the boundary.

Table 5.1: Parameters for MPC Transferring Simulations and Experiments

		$\underline{v}\ (m/s)$	$\overline{v}\ (m/s)$	$\underline{\omega}\ (rad/s)$	$\overline{\omega}\ (rad/s)$
Simulation	Teacher	0.05	0.6	$-\pi/4$	$\pi/4$
	Learner	0.05	0.3	$-\pi/16$	$\pi/12$
Experiment	Teacher	0.05	0.6	$-\pi/4$	$\pi/4$
	Learner	0.05	0.2	$-\pi/8$	$\pi/8$

The kinematic model of the simulated learner is similar to that of the teacher. However, to further demonstrate the effectiveness of the proposed approach, we introduced a nonlinear transformation within the learner's model. This nonlinearity, which leads to progressively more aggressive vehicle behavior as input values increase, modifies the learner command by three steps: 1) min-max normalizing with respect to the current command ranges; 2) applying a nonlinear function; 3) rescaling them to their original range. The specific formulation of the learner's model is described in (5.4).The learner's command domain is divided into an 11×11 grid for grouping and constructing command pairs, with a minimum cluster size of ($k_{min} = 5$) set for the simulation.

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} + \Delta t \begin{bmatrix} d(h_v(n(v_k)))\cos(\theta_k) \\ d(h_v(n(v_k)))\sin(\theta_k) \\ d(h_\omega(n(\omega_k))) \end{bmatrix}, \boldsymbol{u}_k = \begin{bmatrix} v_k \\ \omega_k \end{bmatrix} \tag{5.4}$$



where, $h_v(v) = v^3$

$h_\omega(\omega) = 4 \cdot (\omega - 0.5)^3 + 0.5$

$n(\times) = (\times - \underline{\times}_L)/(\overline{\times}_L - \underline{\times}_L)$

$d(\times) = \times \cdot (\overline{\times}_L - \underline{\times}_L) + \underline{\times}_L$

Figure 5.6 illustrates the results of following a desired path without any pre-existing command pairs. In this setup, we compare the performance of a learner using our proposed transfer framework against an ideal teacher and another learner that directly applies the teacher's commands as the baseline, serving as the baseline. All three simulations start from the same initial pose, intentionally misaligned with the desired path. (a) shows that using our approach, the learner's trajectory closely follows that of the ideal teacher, except during an initial adjustment period while the framework adapts to the learner's capabilities. The command pairs built during the tracking task are depicted in figures (b) through (e). The learned learner's capabilities are marked by a red rectangle on the teacher's command domain which closely approximates the properties set in Table 5.1. The significant nonlinearity between the two systems is evident when comparing the distribution of command pairs across the command domains of both systems ((b) and (c)).

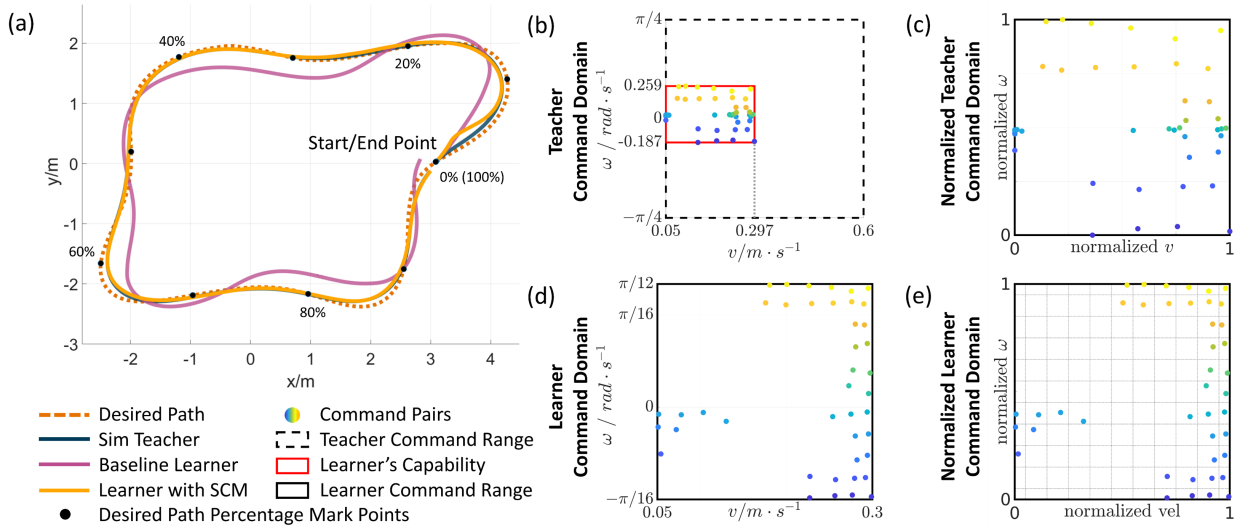


Figure 5.6: Simulation results of SCM-based transfer with MPC. (a) trajectories comparison; (b) teacher command domain; (c) normalized teacher command domain; (d) learner command domain; (e) normalized learner command domain.

Figure 5.7 presents snapshots at different stages of the simulation. The first column compares the trajectories and tracking progress at specified times. Row (b) highlights how the learner perturbed the original desired command to expedite the construction of command pairs in an unconfigured cell. This perturbation resulted in introducing a new command pair, leading to proportionally shrink the learner's minimum angular velocity capability, as indicated by the red rectangle on the teacher's command domain. Row (c) displays an example of using SCM to derive the learner's command, where the equivalent teacher command overlaps with the desired teacher command, demonstrating

the effectiveness of our transfer framework between the teacher and the learner. Finally, row (d) shows the final frame of the learner that uses the proposed framework.

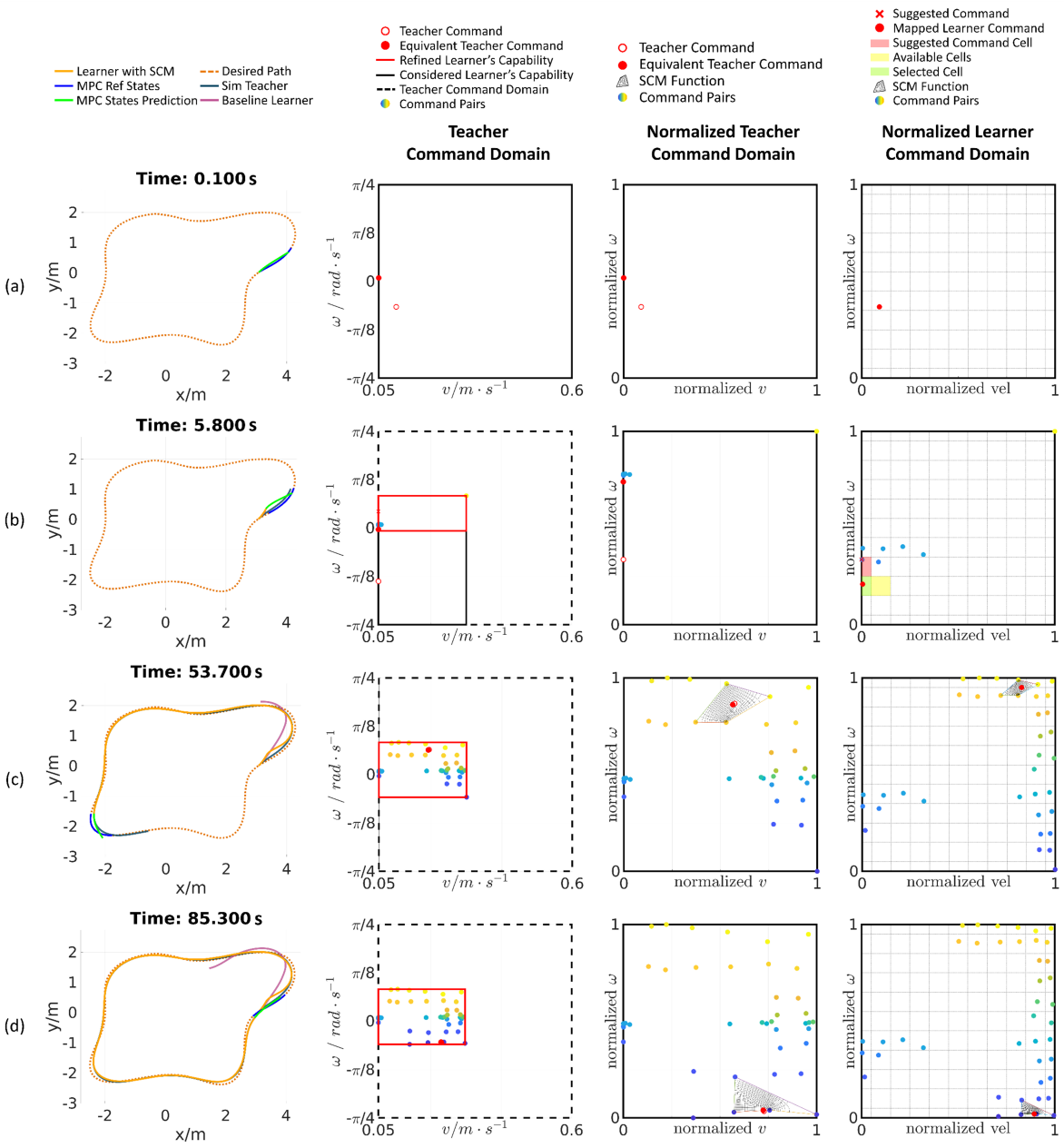


Figure 5.7: Examples from simulations demonstrating the use of SCM to transfer an MPC policy from a teacher to a learner at various time frames. Each column: 1) compares the trajectories between the simulated teacher, baseline learner, and our learner with SCM; 2) teacher command domain and configured learner's capability; 3) normalized teacher command domain within the learner's boundary; 4) normalized learner's command domain.

Note that when the learner using our proposed approach completes the task, the learner directly applying the teacher's commands is still in the early stages. Figure 5.8(b) shows a comparison of the time taken to complete the same path-tracking task; the learner with our transfer framework finishes shortly after the ideal teacher. Figure 5.8(c) compares the deviation from the desired path throughout the tracking progress. For ease of comparison, we present Figure 5.8(a) as a reference for task progress. The errors from the learner with our approach closely align with those of the ideal teacher. Larger deviations occur only when the path requires commands from an unconfigured area of the command domain. Once the learner constructs new command pairs in that area, the deviation significantly decreases, effectively aligning with the teacher's performance. In construct, the baseline learner shows dramatic differences compared with the ideal teacher.

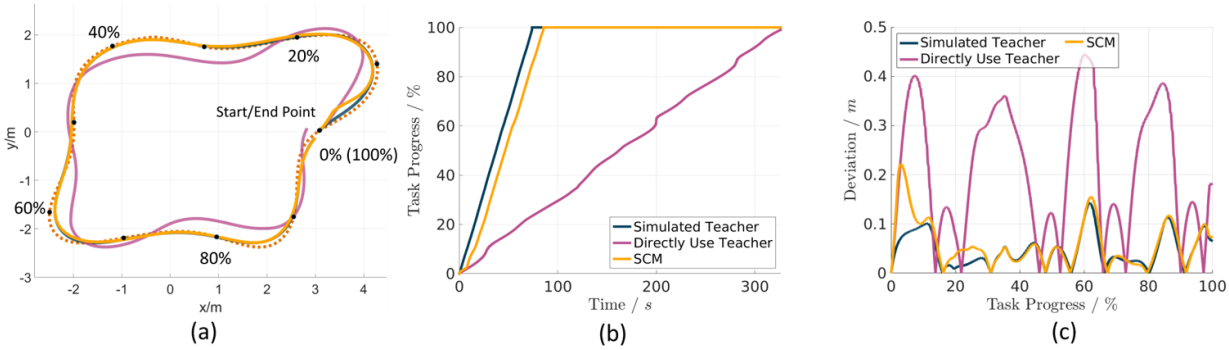


Figure 5.8: (a) shows a comparison of the progression over time among the proposed approach, the baseline, and the ideal teacher; (b) compares the deviation from the desired path in relation to the task's progression.

To demonstrate the effectiveness of our transfer framework, based on the command pairs and characterized learner limits from the simulation result, we did an exhaustive test for teacher commands within the learner's capability. The results are compared between the ideal simulated teacher, the baseline learner, and the learner with SCM. Given the same teacher command, all three systems start at the same initial pose and drive for $0.1s$. We measure the position errors as well as the orientation errors between the teacher and the two learners. The results are depicted in Figure 5.9. (a) categorizes the normalized teacher command space based on the method that the learner with SCM employed: SCM mapping, direct application of teacher commands, or perturbation for exploring unconfigured spaces. (b) and (d) present the position errors, while (c) and (e) show the orientation errors. The results reveal minimal heading and position errors in areas using SCM mapping (corresponding to the yellow area in (a)), suggesting maneuvers nearly identical to the teacher's. In areas with sparser command pairs, leveraging distant command pairs for SCM resulted

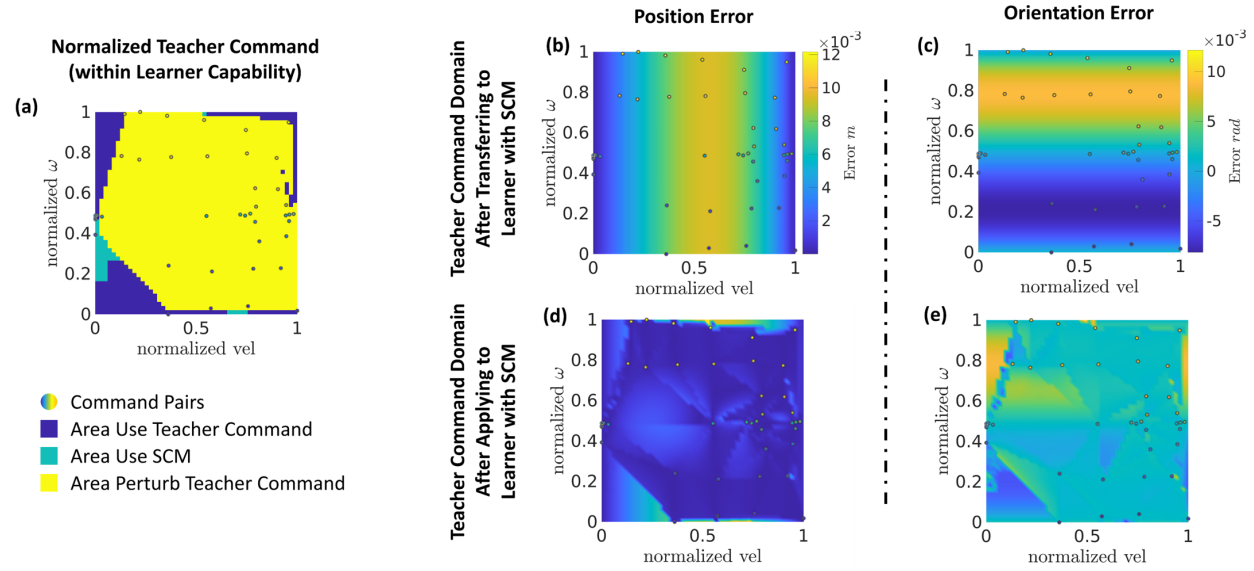
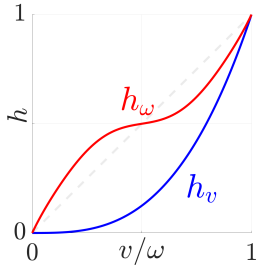


Figure 5.9: Results from extensive testing of SCM-based command transfer: (a) Normalized teacher command domain (min-max normalization within the learner's limits); (b) Position errors between the teacher and the baseline learner; (c) Orientation errors between the teacher and the baseline learner; (d) Position errors between the teacher and the SCM-enhanced learner; (e) Orientation errors between the teacher and the SCM-enhanced learner, all given the teacher commands and drive the systems for 0.1s.

in slightly higher errors as expected due to a lack of local geometrical information. Conversely, the baseline learner struggled to match the teacher's behaviors.

5.4 Experiment Results

The SCM-based transfer framework with MPC is also validated with real vehicles. The hardware setup is similar to the previous study case in 4.5. The parameters used for the simulated teacher and the learner vehicle, Clearpath Jackal, are listed in Table 5.1. The learner's command domain is divided into 11×11 grid cells with $k_{min} = 20$. The nonlinear transformation functions used to further alter the Jackal's kinematics are,



$$
\begin{aligned}
h_v(v) &= v^3 \\
h_\omega(\omega) &= 31.42\omega^7 - 109.91\omega^6 - 144.65\omega^5 - 86.98\omega^4 \\
&\quad + 24.77\omega^3 - 5.08\omega^2 + 2.12\omega
\end{aligned} \tag{5.5}
$$

Due to limited indoor space, a set of command pairs characterizing the Jackal's capabilities was pre-constructed. Figure 5.10 shows the results of a path-tracking experiment with the Jackal, with snapshots provided in Figure 5.11. The Jackal starts positioned away from the desired path. In Figure 5.10(d), the pre-constructed command pairs are circled, with those updated during the experiment highlighted in blue. Most commands sent to the learner are derived through conformal mapping, while a few low-velocity commands are directly applied or perturbed from the teacher's commands due to the absence of a suitable polygon for SCM application.

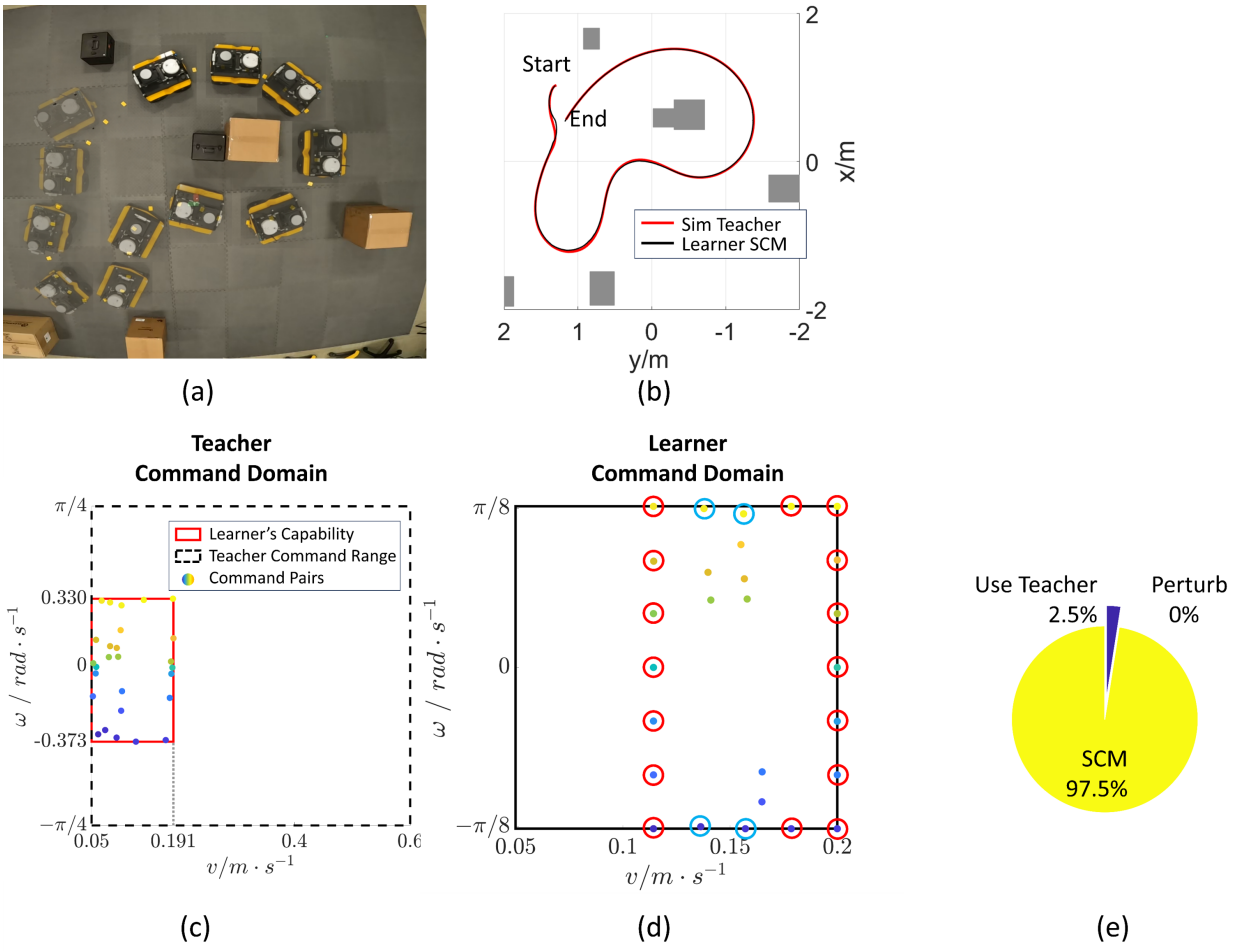


Figure 5.10: (a) snapshots of the experiment; (b) compares the trajectory between the ideal teacher in simulation and the learner with the proposed approach; (c) and (d) present the color-coded command pairs on the teacher's and the learner's command domain respectively after following the desired path;(e) summarizes the method used for obtaining the final learner's command.

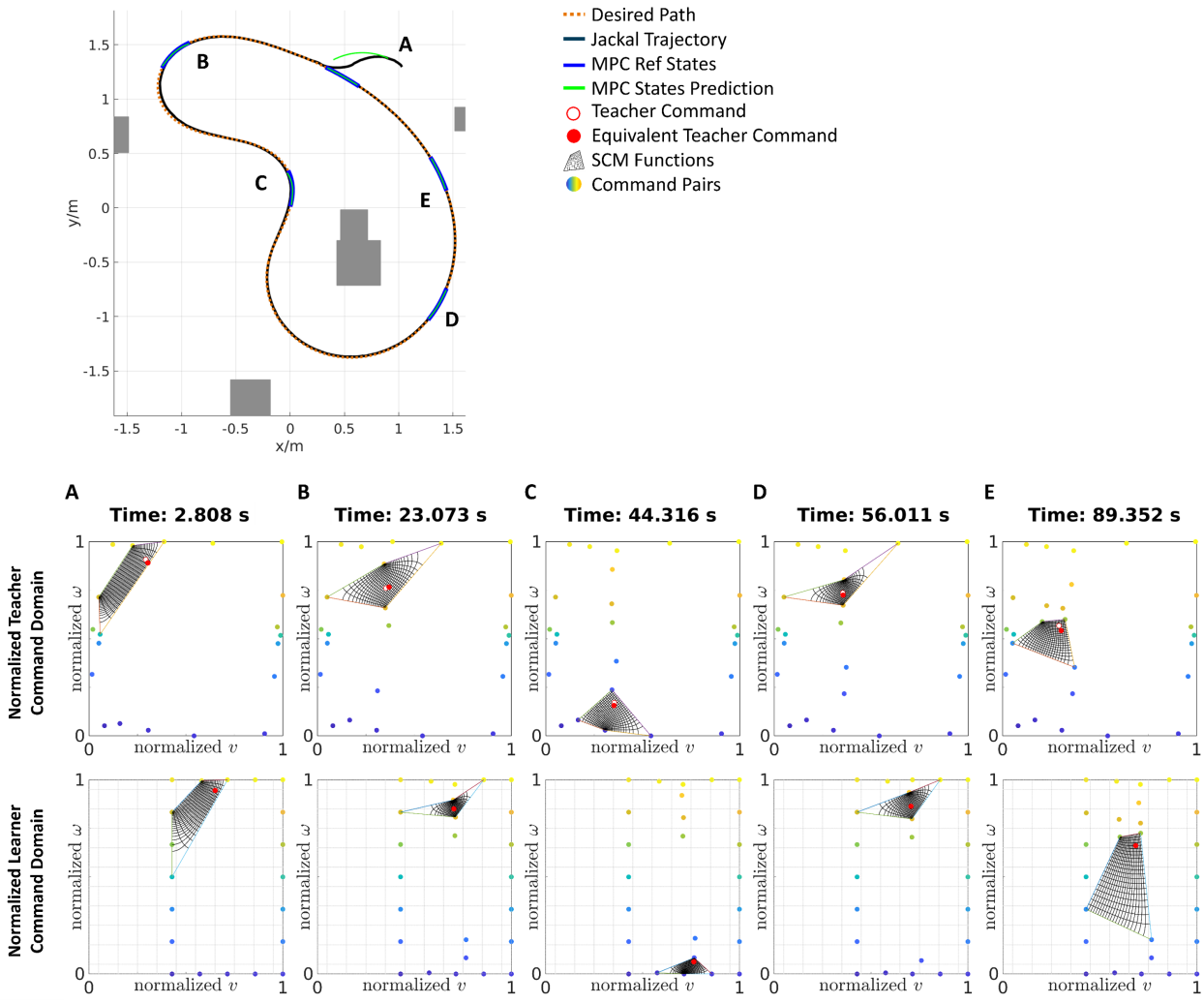


Figure 5.11: Examples from experiment demonstrating the use of SCM to transfer MPC from a simulated teacher to a Clearpath Jackal UGV at different time frames.

The effectiveness of the proposed transferring framework is further validated by comparison with a baseline Jackal for the same task, where teacher commands are directly applied, as shown in Figure 5.12. The baseline Jackal moved much slower, failed to follow the path and ultimately collided with an obstacle. In a separate test, the transfer framework was deactivated at various points during the task, turning the Jackal into a baseline learner. Figure 5.13 shows the Jackal's trajectory after the framework is turned off, and in all five cases, the Jackal collides with obstacles.

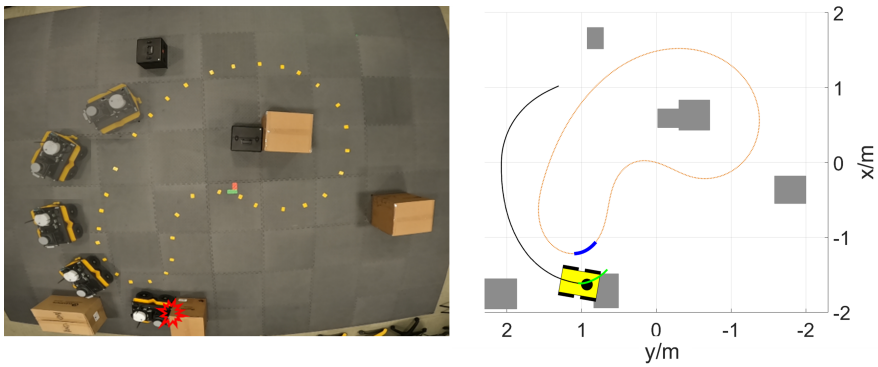


Figure 5.12: The result of directly applying the simulated teacher's commands to the Jackal

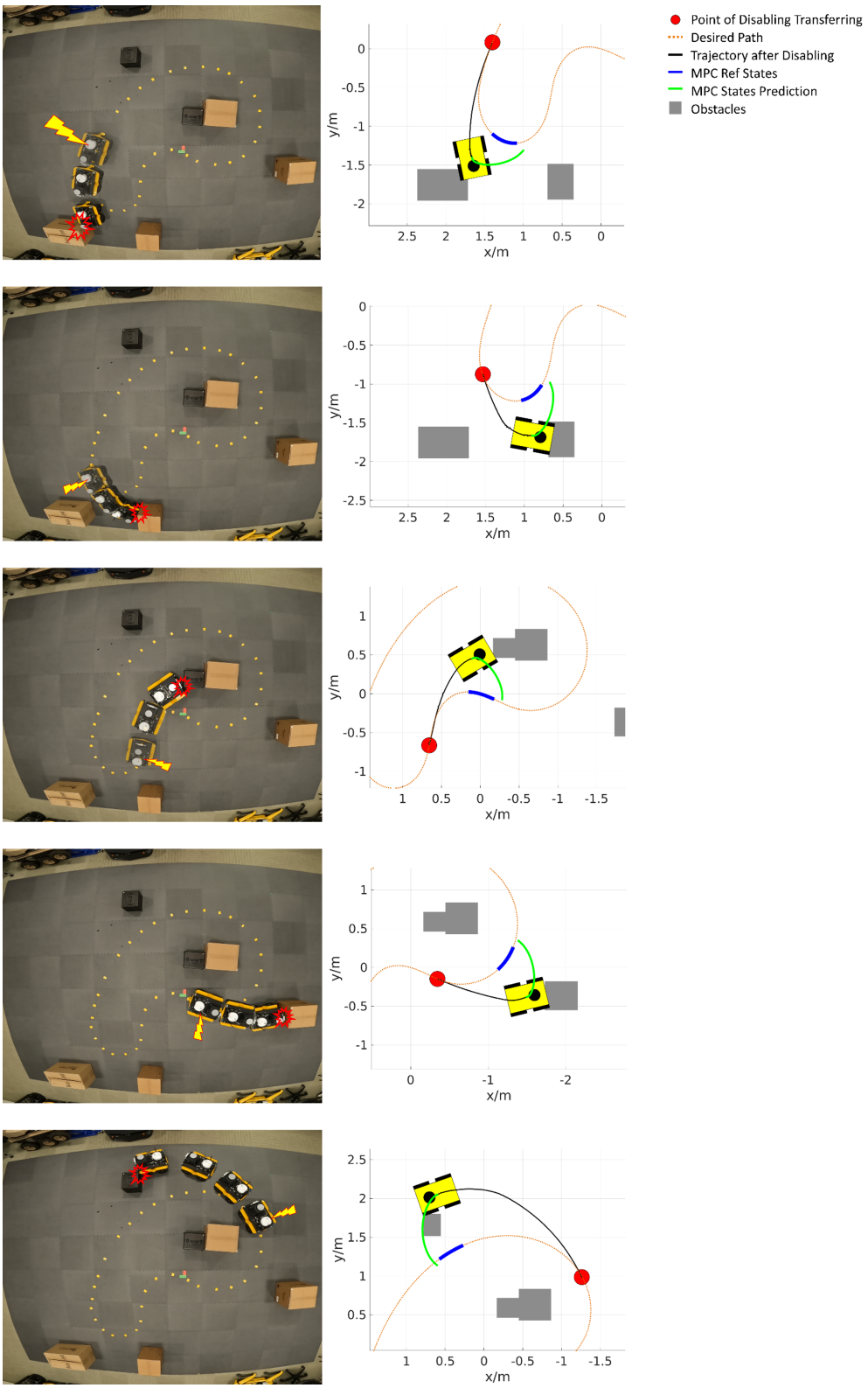


Figure 5.13: (a) presents the result of deactivating the proposed approach and directly applying the teacher's command at different points during the task; (b)-(f) show the learner's trajectory, the reference states, and the predicted states from the teacher's MPC controller at the moment the learner crashes into obstacles.

5.5 Discussion

In this chapter, we delved deeper into our lightweight transfer learning framework designed to effectively mitigate the sim-to-real gap and tackle model mismatches. Our focus has particularly been on enhancing continuous control input transfer, handling motion noise, and actively learning operational limits. We detailed the improved transfer process through the integration of MPC with our framework. We addressed the issue of inconsistent equivalent teacher commands caused by motion noise by refining the definition of command pairs through clustering and filtering the effects of motion noise. Additionally, the framework closely approximates the learner's capability by strategically reducing the limits of the original teacher command domain. Our framework facilitates seamless behavior transfer between a known teacher system and an a-priori unknown learner system, eliminating the need for prior calibration.

Limitations: The proposed transfer learning approach, based on conformal mapping, effectively mitigates discrepancies between systems by analyzing the geometrical transformation of their command domains. Given that Schwarz-Christoffel Mapping (SCM) operates within the 2-dimensional complex domain, our current framework is best suited for systems whose input spaces are inherently 2-dimensional or can be condensed into 2 dimensions. The success of our approach hinges on the availability of a precise equivalent teacher command. The absence of a rich set of accurate command pairs could potentially undermine the effectiveness of our proposed method.

Future Directions: We have demonstrated the effectiveness of our conformal mapping-based transfer framework in narrowing the gap between simulation and real-world applications. Moving forward, we aim to apply our proposed framework to facilitate transfers between heterogeneous robotic systems and to tackle scenarios where the learner robots possess capabilities surpassing those of the teacher systems. Additionally, we are interested in utilizing Schwarz-Christoffel Mapping (SCM) to transition control and planning policies from low-dimensional representations to high-fidelity tasks. Another promising direction that can be explored is the use of multidimensional conformal mapping to enable transfers within higher-order command domains between systems, such as in aerial vehicles.

Part III

Epilogue

Chapter 6

Conclusions and Future Work

In this chapter, we conclude the dissertation by summarizing our achievements and insights, followed by an in-depth discussion of the frameworks we have developed. We also explore potential future directions that could extend and build upon the progress we have made thus far.

6.1 Conclusions

Our research presented in this dissertation focuses on enhancing the resilience of autonomous mobile robots by strengthening various components within their operational frameworks. We have first introduced a Meta-Learning method that is trained offline and fine-tuned online, which adaptively predicts the state and uncertainty of a faulty system. This technique improves system resilience by monitoring predictions to assess collision risks and identifying safe, sampling-based replanned paths. Although enhancing safety, these interventions can disrupt ongoing tasks. To maintain operations in degraded systems, we have proposed a novel, lightweight transfer learning framework based on conformal mapping. This framework efficiently transfers control and planning policies, helping autonomous mobile robots adjust to issues like system aging, model discrepancies, and environmental changes. It achieves the direct transfer of control inputs without requiring an accurate dynamic model of the system. Initially, we applied this framework using a widely adopted motion primitive-based control and path planning method for discrete command space transfer as a proof of concept. Subsequently, we extended it to transfer Model Predictive Control over a continuous command space, without needing prior detailed system knowledge, thus enhancing the framework's utility. This approach demonstrates that even degraded systems can remain operational and continue their tasks with reduced capabilities.

Throughout this dissertation, we have conducted extensive simulations and employed state-of-the-art experiments with ground and aerial robots. These implementations underscore the practicality of our frameworks in addressing real-world robotic challenges.

6.2 Discussion and Possible Future Work

In this dissertation, the Meta-Learning-based prediction technique we propose capitalizes on the rapid adaptability of meta-learning. This approach efficiently leverages knowledge from offline samples of system faults, enabling quick adaptation to new dynamics with minimal runtime data. However, the accuracy of these predictions is significantly dependent on the quality of training. Insufficient or poor-quality training data can result in inaccurate predictions.

Regarding the conformal mapping-based transfer learning framework the conformal mapping-based transfer learning framework, despite the limitations and issues outlined in Section 5.5, we aim to delve deeper into the design aspects of the SCM-based transfer framework. As SCM has a rich mathematical theory behind it, there are choices of the designs of the approach that have mathematical reasons behind it to optimize the performance of calculating the mapping function. We pick out a few points that are worth discussing here while more details can be found in the original SCM papers [16, 29]:

1. **Using bi-infinite strip in rectangle SCM.** "Crowding" is one of the greatest challenges for numerically computing conformal mapping functions. The high ratio of an elongated shape can lead to a situation where the prevertices are spaced exponentially close on the real axis becoming indistinguishable. A bi-infinite strip can significantly ease the effort of solving numerical SCM solutions over the elongated shape. Even for a less elongated shape, this can speed up the computing process. As mathematically solving the SCM is not the main contribution of this work, we direct readers to [16, 29] for a qualitative analysis of how using bi-infinite strip eases rectangle SCM computation.

2. **The reason for choosing rectangle SCM.** Although this section focuses on using a rectangle SCM with polygons having at least four vertices ($N \geq 4$), our framework is also compatible with triangle SCM ($N = 3$). With triangle SCM, triangles from both command domains can be mapped directly to the same set of prevertices on a disk or an upper half-plane, simplifying the process by eliminating the need to connect both ends with a unit disk. However,

rectangle SCM is preferred in practice because it can incorporate more command pairs without as much concern for SCM crowding issues.

3. **The distribution density of the command pairs.** The proposed novel approach directly transfers the control input by geometrically mapping across the command domains. The nature of this approach depends on accurate command pairs and benefits from selecting pairs that are geometrically close to the desired command. As the closer command pairs can better capture the local geometry information and further reflect more similar motions of the learner system. Thus, having the mapping area well covered by the command pairs is advantageous.

This dissertation has demonstrated the importance of effectively monitoring and adapting techniques for autonomous mobile robots, which significantly enhances robotic operations. Recently, the emergence of multimodal robotic applications has begun to revolutionize robotic applications, greatly benefiting long-duration tasks and advancing robotic reasoning capabilities. However, the challenges of system faults and model discrepancies cannot be overlooked for providing the backbone of these advanced robotic technologies. As robot designs are getting increasingly complex and robots with more capabilities are constantly evolving, ensuring resilient operation becomes even more crucial for complex robotic systems. In our future work, we are looking into leveraging multi-dimensional conformal mapping to transfer from a higher-order system to a lower-order system, such as from an aerial vehicle to a ground vehicle. We plan also to extend our framework to deal with learners that have more capabilities than the teacher, through which we want to continue contributing to enhancing the resiliency of robotic applications. Leveraging foundation models for robotic transfer learning presents another promising direction. These models, which learn general representations from large data sets before being fine-tuned for specific tasks, offer substantial potential to enhance robotics by streamlining the transfer of knowledge across different systems. This capability could lead to the development of more adaptable and intelligent robots that can generalize across various tasks, accelerating advancements in the field as discussed in [22].

Bibliography

[1] Afsoon Afzal, Deborah S Katz, Claire Le Goues, and Christopher S Timperley. "A study on the challenges of using robotics simulators for testing". In: *arXiv preprint arXiv:2004.07368* (2020).

[2] S. R. Ahmadzadeh, P. Kormushev, and D. G. Caldwell. "Multi-objective reinforcement learning for AUV thruster failure recovery". In: *2014 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*. 2014, pp. 1–8.

[3] Ibrahim Ahmed, Marcos Quiñones-Grueiro, and Gautam Biswas. "Complementary Meta-Reinforcement Learning for Fault-Adaptive Control". In: *Annual Conference of the PHM Society*. Vol. 12. 1. 2020, pp. 8–8.

[4] Andrea Bajcsy, Dylan P Losey, Marcia K O'malley, and Anca D Dragan. "Learning robot objectives from physical human interaction". In: *Conference on robot learning*. PMLR. 2017, pp. 217–226.

[5] Eduardo Bayro-Corrochano, Leo Reyes-Lozano, and Julio Zamora-Esquivel. "Conformal geometric algebra for robotic vision". In: *Journal of Mathematical Imaging and Vision* 24 (2006), pp. 55–81.

[6] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, et al. "Using simulation and domain adaptation to improve efficiency of deep robotic grasping". In: *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2018, pp. 4243–4250.

[7] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. "Rt-2: Vision-language-action models transfer web knowledge to robotic control". In: *arXiv preprint arXiv:2307.15818* (2023).

[8] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. "Rt-1: Robotics transformer for real-world control at scale". In: *arXiv preprint arXiv:2212.06817* (2022).

[9] Paul F. Byrd and Morris D. Friedman. *Handbook of elliptic integrals for engineers and scientists*. Springer-Verlag, 1971. ISBN: 0387053182.

[10] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. "Closing the sim-to-real loop: Adapting simulation randomization with real world experience". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 8973–8979.

[11] H. Chiang, N. Malone, K. Lesser, M. Oishi, and L. Tapia. "Path-guided artificial potential fields with stochastic reachable sets for motion planning in highly dynamic environments". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 2347–2354. DOI: 10.1109/ICRA.2015.7139511.

[12] Girish Chowdhary, Hassan A. Kingravi, Jonathan P. How, and Patricio A. Vela. "A Bayesian nonparametric approach to adaptive control using Gaussian Processes". In: *IEEE Conference on Decision and Control*. 2013, pp. 874–879. DOI: 10.1109/CDC.2013.6759992.

[13] Jack Collins, Shelvin Chand, Anthony Vanderkop, and David Howard. "A review of physics simulators for robotic applications". In: *IEEE Access* 9 (2021), pp. 51416–51431.

[14] Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. "Learning modular neural network policies for multi-task and multi-robot transfer". In: *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2017, pp. 2169–2176.

[15] Tobin A Driscoll. "Algorithm 843: improvements to the Schwarz-Christoffel toolbox for MATLAB". In: *ACM Transactions on Mathematical Software (TOMS)* 31.2 (2005), pp. 239–251.

[16] Tobin A Driscoll and Lloyd N Trefethen. *Schwarz-christoffel mapping*. Vol. 8. Cambridge University Press, 2002.

[17] František Duchoň, Andrej Babinec, Martin Kajan, Peter Beňo, Martin Florek, Tomáš Fico, and Ladislav Jurišica. "Path planning with modified a star algorithm for a mobile robot". In: *Procedia Engineering* 96 (2014), pp. 59–69.

[18] Kuan Fang, Yunfei Bai, Stefan Hinterstoisser, Silvio Savarese, and Mrinal Kalakrishnan. "Multi-task domain adaptation for deep learning of instance grasping from simulation". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 3516–3523.

[19] Fan Fei, Z. Tu, D. Xu, and X. Deng. "Learn-to-Recover: Retrofitting UAVs with Reinforcement Learning-Assisted Flight Control Under Cyber-Physical Attacks". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)* (2020), pp. 7358–7364.

[20] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks". In: *International conference on machine learning*. PMLR. 2017, pp. 1126–1135.

[21] Chelsea Finn, Sergey Levine, and Pieter Abbeel. "Guided cost learning: Deep inverse optimal control via policy optimization". In: *International conference on machine learning*. PMLR. 2016, pp. 49–58.

[22] Roya Firoozi, Johnathan Tucker, Stephen Tian, Anirudha Majumdar, Jiankai Sun, Weiyu Liu, Yuke Zhu, Shuran Song, Ashish Kapoor, Karol Hausman, et al. "Foundation models in robotics: Applications, challenges, and the future". In: *arXiv preprint arXiv:2312.07843* (2023).

[23] Daniel J Fremont, Edward Kim, Yash Vardhan Pant, Sanjit A Seshia, Atul Acharya, Xantha Bruso, Paul Wells, Steve Lemke, Qiang Lu, and Shalin Mehta. "Formal scenario-based testing of autonomous vehicles: From simulation to the real world". In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2020, pp. 1–8.

[24] N. Gandhi, D. Saldaña, V. Kumar, and L. T. X. Phan. "Self-Reconfiguration in Response to Faults in Modular Aerial Systems". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 2522–2529.

[25] Shijie Gao and Nicola Bezzo. "A conformal mapping-based framework for robot-to-robot and sim-to-real transfer learning". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 1289–1295.

[26] Shijie Gao, Carmelo Di Franco, Darius Carter, Daniel Quinn, and Nicola Bezzo. "Exploiting ground and ceiling effects on autonomous UAV motion planning". In: *2019 international conference on unmanned aircraft systems (ICUAS)*. IEEE. 2019, pp. 768–777.

[27] Shromona Ghosh, Somil Bansal, Alberto Sangiovanni-Vincentelli, Sanjit A Seshia, and Claire Tomlin. "A new simulation metric to determine safe environments and controllers for systems with unknown dynamics". In: *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*. 2019, pp. 185–196.

[28] Zhiwei Hou, Peng Lu, and Zhangjie Tu. "Nonsingular terminal sliding mode control for a quadrotor UAV with a total rotor failure". In: *Aerospace Science and Technology* 98 (2020), p. 105716. ISSN: 1270-9638. DOI: https://doi.org/10.1016/j.ast.2020.105716.

[29] Louis H. Howell and Lloyd N. Trefethen. "A Modified Schwarz–Christoffel Transformation for Elongated Regions". In: *SIAM Journal on Scientific and Statistical Computing* 11.5 (1990), pp. 928–949. DOI: 10.1137/0911054.

[30] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. "Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 12627–12637.

[31] Girish Joshi and Girish Chowdhary. "Deep Model Reference Adaptive Control". In: *IEEE Conference on Decision and Control*. 2019, pp. 4601–4608. DOI: 10.1109/CDC40024.2019.9029173.

[32] Rituraj Kaushik, Timothée Anne, and Jean-Baptiste Mouret. "Fast Online Adaptation in Robotics through Meta-Learning Embeddings of Simulated Priors". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 5269–5276. DOI: 10.1109/IROS45743.2020.9341462.

[33] Petar Kormushev, Sylvain Calinon, and Darwin G Caldwell. "Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input". In: *Advanced Robotics* 25.5 (2011), pp. 581–603.

[34] Amirreza Kosari and Seyyed Iman Kassaei. "Using conformal mapping in developing a novel optimal obstacle-avoidance trajectory-planning for a flying robot". In: *2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)*. IEEE. 2017, pp. 3080–3084.

[35] Dana Kulić, Christian Ott, Dongheui Lee, Junichi Ishikawa, and Yoshihiko Nakamura. "Incremental learning of full body motion primitives and their sequencing through human motion observation". In: *The International Journal of Robotics Research* 31.3 (2012), pp. 330–345.

[36] Thomas Lew, Apoorva Sharma, James Harrison, Andrew Bylard, and Marco Pavone. "Safe Active Dynamics Learning and Control: A Sequential Exploration–Exploitation Framework". In: *IEEE Transactions on Robotics* (2022), pp. 1–20. DOI: 10.1109/TRO.2022.3154715.

[37] Hod Lipson. "Challenges and opportunities for design, simulation, and fabrication of soft robots". In: *Soft Robotics* 1.1 (2014), pp. 21–27.

[38] Yu Liu and Gang Tao. "Multivariable MRAC for aircraft with abrupt damages". In: *American Control Conference*. 2008, pp. 2981–2986. DOI: 10.1109/ACC.2008.4586949.

[39] Yu Liu, Gang Tao, and Suresh M. Joshi. "Modeling and Model Reference Adaptive Control of Aircraft with Asymmetric Damage". In: *Journal of Guidance, Control, and Dynamics* 33.5 (2010), pp. 1500–1517. DOI: 10.2514/1.47996. eprint: https://doi.org/10.2514/1.47996. URL: https://doi.org/10.2514/1.47996.

[40] J. D. Marble and K. E. Bekris. "Asymptotically Near-Optimal Planning With Probabilistic Roadmap Spanners". In: *IEEE Transactions on Robotics* 29.2 (2013), pp. 432–444. DOI: 10.1109/TRO.2012.2234312.

[41] D. Mellinger and V. Kumar. "Minimum snap trajectory generation and control for quadrotors". In: *IEEE International Conference on Robotics and Automation*. May 2011, pp. 2520–2525.

[42] M. W. Mueller and R. D'Andrea. "Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 45–52.

[43] Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. "Learning to Adapt in Dynamic, Real-World Environments through Meta-Reinforcement Learning". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=HyztsoC5Y7.

[44] Scott Niekum, Sarah Osentoski, George Konidaris, and Andrew G Barto. "Learning and generalization of complex tasks from unstructured demonstrations". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 5239–5246.

[45] Gennaro Notomista and Magnus Egerstedt. "Coverage control for wire-traversing robots". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 5042–5047.

[46] Lluis Pacheco and Ningsu Luo. "Testing PID and MPC Performance for Mobile Robot Local Path-Following". In: *International Journal of Advanced Robotic Systems* 12.11 (2015), p. 155. DOI: 10.5772/61312.

[47] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. "Sim-to-real transfer of robotic control with dynamics randomization". In: *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2018, pp. 3803–3810.

[48] Ramya Ramakrishnan, Ece Kamar, Debadeepta Dey, Eric Horvitz, and Julie Shah. "Blind spot detection for safe sim-to-real transfer". In: *Journal of Artificial Intelligence Research* 67 (2020), pp. 191–234.

[49] Kanishka Rao, Chris Harris, Alex Irpan, Sergey Levine, Julian Ibarz, and Mohi Khansari. "Rl-cyclegan: Reinforcement learning aware simulation-to-real". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11157–11166.

[50] Harish Ravichandar, Athanasios S Polydoros, Sonia Chernova, and Aude Billard. "Recent advances in robot learning from demonstration". In: *Annual review of control, robotics, and autonomous systems* 3 (2020), pp. 297–330.

[51] Spencer M. Richards, Navid Azizan, Jean-Jacques Slotine, and Marco Pavone. "Adaptive-Control-Oriented Meta-Learning for Nonlinear Systems". In: *Proceedings of Robotics: Science and Systems*. Virtual, July 2021. DOI: 10.15607/RSS.2021.XVII.056.

[52] Carlo Rizzardo, Sunny Katyara, Miguel Fernandes, and Fei Chen. "The importance and the limitations of sim2real for robotic manipulation in precision agriculture". In: *arXiv preprint arXiv:2008.03983* (2020).

[53] Kelsey Saulnier, David Saldaña, Amanda Prorok, George J. Pappas, and Vijay Kumar. "Resilient Flocking for Mobile Robot Teams". In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 1039–1046. DOI: 10.1109/LRA.2017.2655142.

[54] M. L. Schrum and M. C. Gombolay. "When Your Robot Breaks: Active Learning During Plant Failure". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 438–445.

[55] Mariah L Schrum, Mark Connolly, Eric Cole, Mihir Ghetiya, Robert Gross, and Matthew C Gombolay. "Meta-active Learning in Probabilistically-Safe Optimization". In: *arXiv preprint arXiv:2007.03742* (2020).

[56] Yonadav Shavit, Nadia Figueroa, Seyed Sina Mirrazavi Salehian, and Aude Billard. "Learning augmented joint-space task-oriented dynamical systems: A linear parameter varying and synergetic control approach". In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 2718–2725.

[57] Daeun Song and Young J Kim. "Distortion-free robotic surface-drawing using conformal mapping". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 627–633.

[58] S. Sun, L. Sijbers, X. Wang, and C. de Visser. "High-Speed Flight of Quadrotor Despite Loss of Single Rotor". In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3201–3207.

[59] S. Sun, X. Wang, Q. Chu, and C. d. Visser. "Incremental Nonlinear Fault-Tolerant Control of a Quadrotor With Complete Loss of Two Opposing Rotors". In: *IEEE Transactions on Robotics* (2020), pp. 1–15.

[60] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. "Sim-to-real: Learning agile locomotion for quadruped robots". In: *arXiv preprint arXiv:1804.10332* (2018).

[61] Josh Tobin, Lukas Biewald, Rocky Duan, Marcin Andrychowicz, Ankur Handa, Vikash Kumar, Bob McGrew, Alex Ray, Jonas Schneider, Peter Welinder, et al. "Domain randomization and generative models for robotic grasping". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 3482–3489.

[62] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. "Domain randomization for transferring deep neural networks from simulation to the real world". In: *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2017, pp. 23–30.

[63] Joanne Truong, Max Rudolph, Naoki Harrison Yokoyama, Sonia Chernova, Dhruv Batra, and Akshara Rai. "Rethinking sim2real: Lower fidelity simulation leads to higher sim2real transfer in navigation". In: *Conference on Robot Learning*. PMLR. 2023, pp. 859–870.

[64] Joanne Truong, April Zitkovich, Sonia Chernova, Dhruv Batra, Tingnan Zhang, Jie Tan, and Wenhao Yu. "IndoorSim-to-OutdoorReal: Learning to Navigate Outdoors without any Outdoor Experience". In: *IEEE Robotics and Automation Letters*. 2024.

[65] D. Tzoumanikas, Q. Yan, and S. Leutenegger. "Nonlinear MPC with Motor Failure Identification and Recovery for Safe and Aggressive Multicopter Flight". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 8538–8544.

[66] Yufeng Wang, Anzhao Ji, and Jianbin Cui. "Numerical Solution of Schwarz-Christoffel Transformation From Rectangles to Arbitrary Polygonal Domains". In: *Applied Mathematics & Mechanics (1000-0887)* 40.1 (2019).

[67] Mengyuan Yan, Qingyun Sun, Iuri Frosio, Stephen Tyree, and Jan Kautz. "How to close sim-real gap? transfer with segmentation!" In: *arXiv preprint arXiv:2005.07695* (2020).

[68] Esen Yel and Nicola Bezzo. "A meta-learning-based trajectory tracking framework for uavs under degraded conditions". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 6884–6890.

[69] Esen Yel and Nicola Bezzo. "Gp-based runtime planning, learning, and recovery for safe UAV operations under unforeseen disturbances". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 2173–2180.

[70] Fei Zhang and Weidong Chen. "Self-healing for mobile robot networks with motion synchronization". In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2007, pp. 3107–3112. DOI: 10.1109/IROS.2007.4399565.

[71] Haotian Zhang and Shreyas Sundaram. "Robustness of information diffusion algorithms to locally bounded adversaries". In: *2012 American Control Conference (ACC)*. 2012, pp. 5855–5861. DOI: 10.1109/ACC.2012.6315661.

[72] Jesse Zhang, Brian Cheung, Chelsea Finn, Sergey Levine, and Dinesh Jayaraman. "Cautious adaptation for reinforcement learning in safety-critical settings". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 11055–11065.