Challenging Dexterity of Robot Manipulation: Task and Motion Planning with Control in Piano-Playing Environment

Seojin Hong

4th Year Project Report
Computer Science
School of Informatics
University of Edinburgh

2025

Abstract

Dexterous robotic manipulation remains a challenge in robotics, leading researches to delve into possible improvements in task and motion planning approaches. This study investigates a fundamental challenge of task and motion planning problem by developing a hierarchical motion planning and control framework for robotic piano key manipulation, focusing on improving trajectory accuracy under dynamic constraints. The proposed approach is evaluated across three stages: motion planning (MP) alone, MP with dynamics adaptation, and MP with dynamics integrated with a motion control. The design of this research focuses on the dexterity of high-dimensional robot manipulation by tackling the fundamental challenge, making this project novel. Classical approach in motion planning and control is implemented with adjustment and improvement for its adaptation for the given task, press keys, or playing piano. A direction for further studies is proposed by analyzing and evaluating results of this project, focusing on how this contributes to improvement of dexterity in robotic manipulation.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Seojin Hong*)

Acknowledgements

I would like to express my deepest gratitude to everyone who has supported me throughout the journey of completing this project. This endeavor would not have been possible without the guidance, encouragement, and support of many individuals.

First and foremost, I am deeply grateful to my supervisor, Mohan Sridharan. Their expertise in robotics has been very helpful in developing this project, providing insightful feedback that challenged me to refine my ideas and shape my research. I would also like to express my thanks to Steve Tonneau who have helped settling my dissertation idea with expertise in robotics, and I learned a lot from his Advanced Robotics course.

Finally, I would like to acknowledge my family for their supports in many way that enriched my work without a bad mental breakdown.

Table of Contents

[1] *Modern Robotics: Mechanics, Planning, and Control* (Lynch and Park [2017])
[2] Tunyasuvunakool et al. [2020]

[3]Zakka et al. [2023]

Chapter 1

Introduction

This section delves into the motivation of the dissertation with clear statement of the research problem, objectives and components of the project. Briefly introducing, this project identifies a challenge in task and motion planning with discontinuous dynamics and develop a computational framework with motion planning and control approaches to propose a possible resolution. Piano-playing environment was chosen as the simulation environment to evaluate robustness and performance of the Shadow Hand robot manipulator in a complex environment.

1.1 Motivation

This thesis experiments and analyzes motion planning and control approach of a manipulator for a shadow hand robot playing piano. With its focus on challenges in task and motion planning in robot manipulation, the experiments and analysis are mainly concentrated on motion planning and control components of a robot manipulator whether they can address discontinuity in dynamics during the manipulation and make sure the robot stays collision free during its manipulation.

Despite the industrial focus on the robotic technology, a wide range of challenges in robotics remain unsolved. Billard and Kragic [2019] explains the recent focus of roboticists has been more on dexterous design of human-like robots and enabling wielding a tool in-hand as they require stability of *holding* actions and optimality guaranteed by developing advanced algorithms. This has made the robots to address more complicated environments and tasks. In order to improve robustness of object manipulation, machine learning techniques have been applied to robotics and have enhanced its adaptability with robust control. However, due to limitations in simulators, dexterous manipulation is still considered challenging. Dexterity of robot manipulation is crucial to achieve robust and flexible interaction between humans and robots. This indicates the major challenge that robotics industry face at the moment is to mirror human interaction and attain real-time adaptation capabilities. To achieve this, as mentioned by Billard and Kragic [2019], improving dexterity of planning and control methodology is important, alongside design of the robot itself and efficient structures. Due to this, I have set the hypothesis of this research as tackling motion planning

and control problem of a robot manipulator, as it is crucial to resolve wider range of challenges with robust baseline.

Focusing on the dexterity of the output of the system, identifying and applying proficiently complicated robot manipulation scenario throughout the project was an important starting point. Complexity and dexterity required in human hand kinematics when playing piano has been the inspiration of coming up with manipulation scenario where a dexterous shadow hand model plays the piano. Piano playing is often considered as one of the most complicated movements in human repertoire and understanding the kinematics leads to sensorimotor control of hand movement with high degree of freedom [Furuya et al. [2011]]. A computational organization and representation of such complex behaviors in robot manipulation can provide a stronger baseline of addressing dexterity of human-like robots.

1.2 Problem Statement

Despite significant developments in robotic technology, particularly in industrial applications, there still exists struggle to manage discontinuities in dynamics along robot manipulation. This is often because of limited dexterity in tasks requiring human-like precision, such as piano playing with a dexterous shadow hand. These shortcomings, evident in the inadequacy of simulators and task and motion planning (TAMP) frameworks to handle dynamic irregularities, hinder improvement in adaptability and stability. Inspired by the complex kinematics of human piano playing and the need for improved dexterity highlighted by Billard and Kragic [2019], the hypothesis of this thesis is that developing a computational method to tackle motion planning and control can mitigate the challenge in robotic task and motion planning by enhancing the dexterity of robot manipulation.

1.3 Research Objectives

There are main research questions set for bottom-top hierarchical organization of research in order to accomplish the main objective:

1. Does the proposed motion planning approach generates collision-free trajectory of precise key-strikes in the robot manipulation given finger-key coordination of input note sequences?
2. How does applying dynamics constraints, or a sequence of velocities, affect the execution of predefined trajectory?
3. If dynamics applied to the trajectory cause oscillation in execution, does integrating a motion controller improve trajectory execution accuracy and stability?

Throughout the rest of the paper, the problem formulation and implementation of the computational framework are explained in theoretical, technical, and experimental manner, in order.

1.4 Structure of the Report

Chapter 2 explains the robotics concepts and methodologies in a theoretical manner to ensure understanding of the background knowledge required for the research.

Chapter 3 explains how the theoretical concepts and knowledge explained in the background chapter are implemented in the computational framework and how they are formalized into sub-problems of the thesis. Throughout the research, the implementation and experiment follows the order of the problems related to research objectives in the previous section in order.

Chapter 4 is about experiments conducted to produce data to discuss about the performance of the approaches taken in the project and the accomplishment of the research objectives. The results from the experiments are gathered and analyzed to evaluate what they indicate in robotics and how it relates to the thesis.

Chapter 5 is conclusion where the key findings of the project are explained. Limitations found out throughout the project are discussed so that it can present direction and improvements to be made for further studies. How the research can be extended with application of other technology can lead to further studies or works for more improvement and development of challenges that are not addressed or that are new.

Figure 1.1 show an overview of computational framework in flowchart and this will be explained throughout the paper.

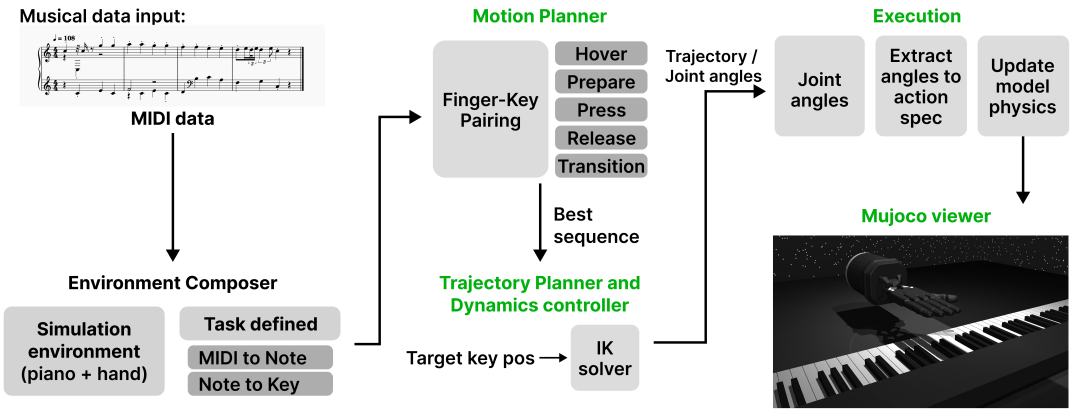


Figure 1.1: This flowchart outlines the computation and planning done for robot manipulation and execution of the joint configurations.

Chapter 2

Background

This chapter introduces a brief overview of research in robotics, task and motion planning, and dynamics control with some theoretical concepts and methods for implementations. This includes human hand kinematics and main components of robot manipulation. Equations for mathematical explanation of the robotics concepts are taken from the book *Modern Robotics: Mechanics, Planning, and Control* (Lynch and Park [2017]).

2.1 Kinematics

2.1.1 Human hand kinematics when playing piano

There are previous studies done about biological kinematics of piano-playing human hand. Furuya et al. [2011] analyzes the pattern of hand joint motions of expert pianists. The key hand kinematics that my research is aiming to mirror are:

- Spatial coordination: Patterns of the hand movement for precise positioning of the hands based on the musical requirement and anatomical differences. This can be represented as continuous motion planning problem, given task and constraints definitions to strictly follow.
- Temporal dynamics: Velocity and acceleration of the hand movement are significantly influenced by timing of pressing a key. Varying timing precision may affect torque control over tone sequence. This can be formulated as motion control problem to handle continuous manipulation of the shadow hand model with dynamics applied.
- Finger independence: Independent motion and force control between striking and non-striking fingers were found to be crucial for different control for key-striking and key-pressing motions of individual fingers.
- Individual variability: Hand kinematics tend to differ between players depending on their biological constraints and style of playing piano. This can be considered as an instance of a dynamic application of the computational framework of this research, mapping discrete tasks into a continuous trajectory of motions.

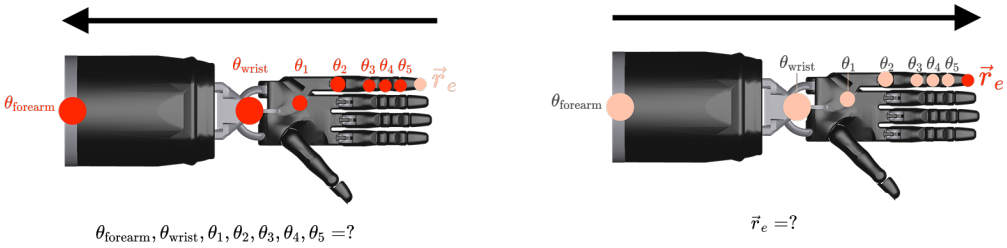
- Task-specific Adaptation: Given different technical and expressive demands of the music being played, hand trajectories and velocities are likely to be modified by the player. This is a representation of complicated and dynamic environment.

Each of the key characteristics of hand kinematics are applied and formulated if necessary in the computational framework in order to mimic the dexterity of human hand as much as possible. These are formulated into structured robotics problem by the implementation of computational framework so that the robot manipulation mirrors human kinematics to proficient extent. This will be explained thoroughly in methodology chapter 3.

2.1.2 Kinematics in robotics

Kinematics is the study of motion without considering the forces that cause it, by considering the configuration or angle of the joint and the position of the end effector. In robotics, understanding kinematics is important to evaluate how robotic manipulators move and position the end-effectors. It is divided into forward kinematics (FK) and inverse kinematics (IK).

Forward kinematics computes the end-effector position and orientation given a set of joint angles, while inverse kinematics determines the joint configurations required to reach a desired end-effector position. These principles are fundamental in robot manipulation, including dexterous task such as piano playing.



(a) Inverse Kinematics is where the joint angles, or configurations, are unknown and to be computed given end effector position and orientation.

(b) For Forward Kinematics, all the joint angles are given and the end effector position and orientation needs to be computed.

Figure 2.1: This clarifies the difference between inverse kinematics and forward kinematics by explaining the goals of the techniques - i.e. inverse kinematics computes the joint configuration and forward kinematics computes the end effector position and orientation

Forward Kinematics (FK)[1]

Forward kinematics computes the mapping of joint configurations to end-effector coordinates. When the robot in manipulation consists of a sequence of links connected

[1]The forward kinematics approach refers to the definitions in *Robot kinematics: Forward and inverse kinematics* (Kucuk and Bingul [2006])

by joints, this forms a kinematic chain where each joint introduces a degree of freedom. Given joint 1 to n, each joint i has:

- A joint variable θ_i
- A transformation from the frame of joint $i-1$ to joint i, defined by the D-H parameters.

Denavit-Hartenberg (D-H) convention assigns a coordinate frame to each joint and defines the transformation between consecutive frames using four parameters:

- a_i: Link length, the distance along x-axis of frame $i-1$ to the origin of frame i.
- α_i: Link twist, the angle between the z-axes of frames $i-1$ and i about the x-axis of frame $i-1$.
- d_i: Link offset, the distance along the z-axis of frame $i-1$ to the origin of frame i.
- θ_i: Joint angle, the angle between the x-axes of frames $i-1$ and i about the z-axis of frame $i-1$.

The transformation from frame $i-1$ to frame i is represented by a 4×4 homogeneous transformation matrix T_i^{i-1}, which combines rotation and translation:

$$T_i^{i-1} = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

This matrix can be broken down into a submatrix and vectors as below:

- Rotation: A 3×3 matrix on the top-left that is the orientation of frame i relative to frame $i-1$.
- Translation: A 3×1 translation vector on the top-right that is the position of the origin of frame i in frame $i-1$.
- $[0,0,0,1]$ for homogeneous coordinates.

The total transformation from the base frame (frame 0) to the end-effector frame (frame n) is the product of all individual transformations along the kinematic chain:

$$T_n^0 = \prod_{i=1}^{n} T_i^{i-1} \tag{2.2}$$

Where:

- T_n^0: The homogeneous transformation matrix from the base frame to the end-effector frame.
- T_i^{i-1}: The transformation matrix for joint i.

This can be written in matrix form:

$$T_n^0 = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ 0 & 1 \end{bmatrix} \tag{2.3}$$

Where **p** and **R** are the position and orientation of the end-effector, respectively.

Inverse Kinematics (IK)

Inverse kinematics computes the joint angles given desired end-effector position. Given the homogeneous transformation matrix T_n^0, which maps the base frame to the end-effector frame, inverse kinematics solves for the joint variables $\theta_1, \theta_2, ..., \theta_n$ such that:

$$T_n^0(\theta_1, \theta_2, ..., \theta_n) = \begin{bmatrix} \mathbf{R}_d & \mathbf{p}_d \\ 0 & 1 \end{bmatrix} \tag{2.4}$$

where:

- $\mathbf{p}_d = [x_d, y_d, z_d]^T$: desired end-effector position
- $\mathbf{R}_d$: desired end-effector orientation.

The inverse kinematics problem can be formulated as solving for the joint variables θ_i given the desired $\mathbf{p}_d$ and $\mathbf{R}_d$.

For complex inverse kinematics that is for more than 6 degrees of freedom (DoFs), numerical approaches can be used for the robustness of the solution. These approaches typically use Jacobian-based iterative method, where Jacobian matrix is used to update joint angles $\Delta\theta$ and end-effector transformation (position and orientation) $\Delta\mathbf{x} = \mathbf{J}\Delta\theta$ where **x** represents the end-effector position and orientation.

- If **J** is invertible,
$$\Delta\theta = \mathbf{J}^{-1}\Delta\mathbf{x} \tag{2.5}$$
- Otherwise,
$$\Delta\theta = \mathbf{J}^{+}\Delta\mathbf{x} \tag{2.6}$$
where $\mathbf{J}^{+} = \mathbf{J}^{\mathbf{T}}(\mathbf{J}\mathbf{J}^{\mathbf{T}})^{-\mathbf{1}}$ is a Moore-Penrose pseudo-inverse (Barata and Hussein [2012].

2.2 Robot Manipulation

Robot manipulation often refers to interaction and manipulation between robot and objects in the environment through a robotic system (Bellicoso et al. [2019]). Integration between high-level discrete planning and low-level continuous motion planning enables robots to perform intricate tasks that require reasoning about both abstract actions and precise movements (Srivastava et al. [2014]). As the robot must adapt to the environment and objects it interacts with along the trajectory, controlling manipulator dynamics is crucial to aid testing control strategies and computational systems in manipulator that could provide useful information in some components of the robot - joints, actuators, and structural stiffness (Brady [1982]). These challenges have led

to the development of advanced frameworks that combine task and motion planning, allowing robots to generate strategies for complex manipulation tasks in unstructured environments. Such frameworks provide a powerful experimental platform for testing and refining computational approaches to robotic manipulation and planning, enabling researchers to explore novel solutions for integrating perception, decision-making, and action execution in real-world scenarios (Bernardo et al. [2023]).

2.2.1 Task and Motion Planning (TAMP)

Task and motion planning (TAMP) combines discrete task planning with continuous motion planning to solve complex robotic problems. Task planning involves high-level reasoning about actions and goals, while motion planning focuses on finding collision-free paths in continuous space (Garrett et al. [2021]). The primary challenge in this problem is integration of the two levels of planning that involve different representations and search spaces (Kim et al. [2022]). This has been more important in robotics as the technology tends to be deployed in more challenging and interaction-rich environments. The integration expansively involves task-level planning with discrete domain representation and motion planning and control with continuous models.

2.2.2 Task Planning

In TAMP, task planning is a discrete part of the problem where appropriate actions are computed with the given information. Classical artificial intelligence planning is often known in robotics as task planning that is used for reasoning about wider range of goals and states which sequences of abstract actions are derived from (Kaelbling and Lozano-Pérez [2012]). Task planning plays an important role in discretizing complex tasks into steps that are adaptable to the action specification of the system. Formulation of task planning problem involves:

- $\mathcal{S}$: Set of discrete states
- $\mathcal{A}$: Set of discrete actions
- $\mathcal{T} : \mathcal{S} \times \mathcal{A}$: Transition function
- $\mathcal{G} \subseteq \mathcal{S}$: Goal states

The object of this problem is to find a sequence of actions $a_1, a_2, ..., a_n$ that transitions from an initial state s_0 to a goal state $s_g \in G$. This formulation captures the high-level task reasoning aspect of TAMP. Garrett et al. [2021] explains that the transition function considers geometric feasibility of motions in addition to symbolic preconditions and effects so that a hybrid state space is generated and add motion feasibility constraints to action transitions.

2.2.3 Motion Planning

Motion planning problem is finding a collision-free path for a robot to move from a starting point to a desired goal point. Motion planning problem for a robot with d degrees of freedom is formulated by information below:

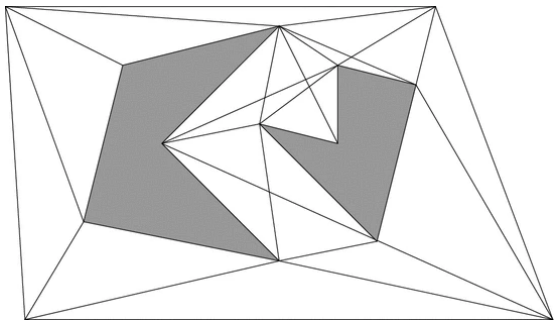


Figure 2.2: Representation of configuration space in *visibility graph* form that was found from roadmap technique in Gasparetto et al. [2015]. Each vertex represents nodes in the graphs and the gray areas are obstacles. The technique has identified configuration space as the not colored areas.

- $\mathcal{Q} \subset \mathbb{R}^d$: configuration space
- $F : \mathcal{Q} \rightarrow \{0,1\}$: constraint
- $q_0 \in \mathcal{Q}$: initial configuration
- $\mathcal{Q}_* \subseteq \mathcal{Q}$: goal set of configurations

The evaluation method of feasibility of the path is to observe whether the configuration is within *feasible* configuration space that is a subset of $\mathcal{Q}$ and satisfies the constraint. Taking the appropriate motion planning approach is important to ensure the framework is compatible with the environment of manipulation and automated planning, or task planning. There are two big categories of motion planning approaches, classical and learning-based motion planning as mentioned by Tamizi et al. [2023]. The goal of this study is that learning-based methods are applied to more scenarios in the recent years compared to the other ones due to their generalization and ability to deal with complex problems. The most commonly used classical motion planning methods are sampling-based methods such as Rapidly-exploring Random Trees (RRT) and Probabilistic roadmaps methods (PRM). These are often considered more valuable than many other approaches for their capacity to solve problem in more complex and limitedly-observable environments (Tamizi et al. [2023]). The configuration space found by roadmap approach can be represented in graph as demonstrated in Figure 2.2

Once motion planner identifies collision-free configuration space in the environment, detailed robot motion manipulation can be computed by trajectory planner. Trajectory planning problem can be framed as controlling the robot manipulation to perform desired motion by generating benchmarking inputs for the controller of the robot (Gasparetto et al. [2015]).

Offline trajectory planning method

In Saravanan et al. [2009], a very understandable explanation of offline and online trajectory planner is provided. Offline path planning involves generation of smooth curve given explored environments, fixed initial and goal positions of the curve, direction of the projection determined, and control points. There exist flexible control points

between fixed control points so that the planner can determine the shape of the curve and ensure those are within feasible configuration space.

Online trajectory planning method

Unlike offline approach, online trajectory planner is more suitable for unknown terrain. In other words, feasible path connecting start and end point is impossible to be generated. The online method instead defines temporary starting and ending points and explores the path line between temporary points determining the curve real-time. This method is considered less expensive for its time cost compared to offline method. However, offline trajectory planners are more likely to have better capacity of generating more complicated curves given less but more descriptive data.

Optimality constraints

According to Gasparetto et al. [2015], planner mostly follows some optimality criterion set, such as minimum execution time, minimum energy, minimum jerk. The terminology ”energy” in trajectory planning approach for robot refers to the effort of the actuators on the robot. This often identify the optimality unit of system involved in a robot manipulation. Specifically for this project, hybrid optimality criteria, time-effort minimum, is proposed to ensure the ”piano key pressing” motion happens at the precise timing generating the appropriate loudness for the dexterous performance of robot. In Gasparetto et al. [2015], identifying trajectory planning method as dynamic or kinematic based on the constraints that the optimization problem takes into account. In the next section, dynamics from wider range of aspects in robotics is explained, and based on this, a reasoning to determine whether trajectory planner of my research is dynamic or kinematics is proposed.

2.3 Execution of Motion with Control [2]

2.3.1 Dynamics

Understanding the dynamics of robotic systems is essential for designing and controlling manipulators, particularly in tasks requiring precise motion and force application, such as playing a piano with a robotic hand. In piano-playing robot manipulation, discontinuity in dynamics represents varying velocity required for each key press motion. Dynamics in robotics refers to study of the forces, torques and motions that govern the behavior of a robotic system, accounting for its mass, inertia, and external interactions. For a high-dimensional robot, open kinematic chains are required which represents serial chains of rigid bodies like the shadow hand model in this project. For the robot with chains of links connected by joints, the dynamics describe how joint torques and external forces produce motion, or conversely, how motion generates forces.

The dynamics of an open chain can be modeled using two primary approaches: the Newton-Euler formulation and the Lagrangian formulation.

The fundamental formulation of motion for an open chain robotic system represents

[2] *Modern Robotics: Mechanics, Planning, and Control* (Lynch and Park [2017])

the relationship between the joint torques τ to the joint positions θ, velocities $\dot{\theta}$, and accelerations $\ddot{\theta}$. The general form of the equation can be written as:

$$\tau = M(\theta)\ddot{\theta} + C(\theta,\dot{\theta})\dot{\theta} + G(\theta) \tag{2.7}$$

where:

- $M(\theta)$ is the mass matrix for capturing the inertia of the system and depends on the joint configuration θ. It describes how the robot's mass distribution affects its resistance to acceleration.
- $C(\theta,\dot{\theta})$ is the Coriolis and centrifugal forces, which arise from the interaction of joint velocities and the changing configuration of the robot. These forces become significant when the robot moves at high speeds.
- $G(\theta)$ represents gravitational forces acting on the links, which depend on the robot's configuration relative to the direction of gravity.

The mass matrix $M(\theta)$ is symmetric and positive definite, meaning it consistently resists acceleration in a physically meaningful way across all configurations. This equation of motion provides a complete description of how torques at the joints produce motion, or conversely, how a desired motion requires specific torques to be applied, and these are what modeling robotic dynamics approaches refer to - forward and inverse dynamics. Recalling the distinction between forward and inverse kinematics, it is easier to distinguish between forward and inverse dynamics.

Forward dynamics computes the motion resulting from given joint torque τ, which is important when simulating robot motion and predicting behaviour in response to applied forces. On the other hand, inverse dynamics determines the required joint torques τ to achieve a desired motion $(\theta,\dot{\theta},\ddot{\theta})$. This is involved in motion control that aims to compute the necessary forces to follow a predefined trajectory accurately. The choice between these approaches depends on the application: motion planning often relies on forward dynamics to verify physically feasible trajectories, while control systems such as PD control use inverse dynamics to generate the required inputs for accurate execution. In piano-playing scenario, forward and inverse dynamics represents the impacts of given torque input on finger motion and motion control by making adjustments to torques accordingly, respectively.

For a piano-playing robotic hand, the dynamics of an open chain are particularly complex due to the high-dimensional nature of the system (multiple fingers, each with several DOF) and the discontinuous dynamics introduced by key presses. Accurate modeling of these dynamics is crucial for planning smooth trajectories and applying appropriate torques to achieve musically precise key presses while avoiding oscillations. Classical control methods, such as computed torque control, use the dynamic model to linearize the system and track desired trajectories, while more advanced methods, like impedance control, adjust the manipulator's response to external forces, ensuring stability during contact tasks. Understanding these fundamental principles of dynamics provides the foundation for addressing the challenges of high-dimensional robot manipulation in my research.

2.3.2 Control

Controlling a robotic system involves using the dynamics model to achieve desired motion or force objectives such as pressing piano keys with the correct timing and force. Motion control leverages the equations of motion to compute the joint torques required to track a desired trajectory or respond to external forces, ensuring stability and precision in the presence of nonlinearities and disturbances. Two fundamental approaches to motion control are feedforward control and feedback control, which can be used individually or in combination to achieve robust performance.

Feedforward control uses the dynamics model to make predictions of the torques required for a desired trajectory $\left(\theta_d, \dot{\theta}_d, \ddot{\theta}_d\right)$:

$$\tau = M(\theta_d)\ddot{\theta}_d + C(\theta_d, \dot{\theta}_d)\dot{\theta}_d + G(\theta_d) \tag{2.8}$$

This approach assumes that the model of the dynamics (i.e., M, C, and G) is accurate and that there are not external disturbances or modeling errors. In the context of a piano-playing robotic hand, feedforward control can be used to pre-compute the torques needed to move a finger along a trajectory that presses a key at a specific time. However, some variations in key resistance may cause inaccuracies.

Feedback control, in contrast, perform real-time correction of errors using the robot's state and measurements. Proportional-derivative (PD) control is a common method that adjusts torques based on the position error $e = \theta_d - \theta$ and velocity error $\dot{e} = \dot{\theta}_d - \dot{\theta}$:

$$\tau_{\text{feedback}} = K_p e + K_d \dot{e} \tag{2.9}$$

where K_p and K_d are the proportional and derivative gain matrices, respectively. The proportional term $K_p e$ applies a corrective torque proportional to the position error, driving the robot toward the desired position, while the derivative term $K_d \dot{e}$ adds damping to reduce oscillations by counteracting the velocity error. For a piano-playing robot, feedback control can correct for deviations caused by external disturbances, such as an unexpected resistance from a piano key.

In practice, feedforward and feedback control are often combined to leverage the strengths of both approaches, resulting in a feedforward-feedback control strategy. This combination uses the feedforward component to predict the nominal torques required for the desired motion and the feedback component to correct errors:

$$\tau = M(\theta)(\ddot{\theta}_d + K_p e + K_d \dot{e}) + C(\theta, \dot{\theta})\dot{\theta} + G(\theta) \tag{2.10}$$

This makes sure a piano-playing robotic hand follows the trajectory while handling disturbances, like varying key resistance, minimising oscillations during contact. For such contact-changing tasks, additional strategies like impedence control can be implemented to manage the force applied, ensuring smooth transitions and stability.

2.4 Related Work

The integration of task and motion planning (TAMP) in robot manipulation has been a critical area of research, particularly for tasks requiring precise and coordinated movements. Recent advancements have increasingly leveraged learning-based approaches to enhance decision-making in TAMP, while other works have explored motion planning and control for music-playing robots, including piano-playing applications. This section review these efforts, focusing on their approaches to TAMP, handling of discontinuous dynamics, and application to music-related tasks, and highlights the gaps that this research addressed by focusing on fundamental problem-solving for smooth motion planning with appropriate dynamics control.

Fundamental Challenge A significant trend in TAMP research is the use of learning-based methods to improve decision-making and adaptability in robotic manipulation. According to Guo et al. [2023], machine learning has been grown for its usage in robotics researches due to enabling real-time problem solving with optimization. The researches are proposing methods to address task and motion planning challenge focusing on the extended scalability of robot technology. In other words, significant focus in robotics industry is recently on wider application of the technology in dynamic and more complex environment. For instance, Xiao et al. [2022] contributed to clearer comparison in classical approach and learning-based approach to the challenge and hypothesize that learning-based methods are more adaptable to complicated and unknown environment. Dong et al. [2023] also reviews classical motion planning methods combined with reinforcement learning methods for some improvements in their performance. However, this research contributes to the improvements in dexterity of robot manipulation by addressing the fundamental challenges in the classic motion planning and control methods and evaluate their robustness with experiments given complicated environment and tasks. According to Liu et al. [2016], despite substantial development in robotics, dexterity of manipulation remains as a major challenge to be solved. Schäffer et al. [2008] introduces approaches to sophisticated design of human-interactive robot and computational framework that can be applied for the robot's manipulation. In contrast, I contribute to the dexterity of robot manipulation by developing a computational framework with more robust extension of classical methods and ensure the manipulation is planned out with less oscillation and error rates from the low-level planning. This leads to proposition of potential direction of further works and researches that could contribute to more development in such challenges in robotics.

Zakka et al. [2023] heavily inspired technical aspect of the research. The study has implemented a framework of dual shadow hands playing a piano sheet with trained dataset. Similar to other previous works mentioned above, the major objective of the research was to apply reinforcement learning planning system in order to get more dynamic robot manipulation. Despite its contribution of providing idea of high-dimensional hand-in manipulation scenario, I ensured the novelty of my paper by focusing on the lower-level planning, or motion planning, and motion control over the execution of the planned trajectory.

High-dimensional hand-in manipulation Many researches done in the past about robot manipulation has common big objective to make the robot technology to be

applied for more complicated environment. As mentioned in the previous paragraphs, advanced approach subjected and proposed from extension of classical methods are often a hotspot for Robotics researches. As the learning-based methods are frequently considered as advanced approach in robot motion planning and control, the experiments are done in high-dimensional robot manipulation. However, many of them do not deal with the dexterity of the classical method which they are extending their study from, and often find out fundamental challenges still exist to be resolved like these papers - Gu et al. [2025], Eze and Crick [2024], Han et al. [2023], and Liu et al. [2021].

On the other hand, there are researches done in addressing challenges in robustness of motion planning and control from more fundamental perspective. Similar to paper Sandakalum and Ang Jr [2022], many prior works were motivated from improvements to make in the redundancy of planning method and controlling motion with dynamics applied in robot manipulation. This is similar to the motivation of this paper. However, the robot manipulation instances for experiments in the papers addressing redundancy of baseline approaches are low-dimensional such as grippers, mobile robots in 2D environments, and some that has computational evaluations without a simulation or robot experiment. [Abdulsaheb and Kadhim [2023], Zhang et al. [2025], Benallegue et al. [2017]]

Sugano and Kato [1987] and Zakka et al. [2023] are the papers surrounding the piano-playing robot manipulation. Sugano and Kato [1987] is a technical low-level approach to structure the human-like hand robot to enable playing piano, while the paper does not address kinematics and dynamics problem. Zakka et al. [2023] is a paper that has been a big motivation of my work as they introduce high-dimensional experimental environment for their computational framework. However, their work contributes to reinforcement learning based approach for AI planning problem of robotics and hardly involve fundamental challenge mitigation.

In contrast, my research topic is centered around engagement of robustness of motion planning and control, or the lower-level approaches, of the robot manipulation. The computational framework that is the major objective of this paper has motion planner for path generation given correspondence of a piano key and finger for a key-strike motion, trajectory planner to output a executable path for the shadow hand model, and motion control to experiment the impact of desired torque applied to the robot while execution and how a motion controller can make sure the trajectory is executed with reduced oscillation with the torque applied. Setting up experiments in piano-playing environment, the redundancy of my computational framework is observed from the simulation of high-dimensional robot manipulation. Due to all this, the computational framework developed from this project makes sure the novelty of the research addressing the planning and control challenge with high-dimensional manipulation.

Chapter 3

Methodology

This section details the implementation of the computation framework for the shadow hand in a piano-playing environment, covering key robotic problem formulation and system implementation.

3.1 Modeling Human Hand

In this research, the objective is to enable the shadow hand model to mimic the kinematics of the hand when playing piano to engage sophisticated control over actions. In this section, I introduce structure of the shadow hand model that was built based on a human hand and how kinematics are computed to control the action of the hand. In the experiments of this research, Shadow Dexterous Hand model from Mujoco Menagerie (Zakka et al. [2022]) is used. Controlling joints of the model requires some extent of understanding of the structure and actuators of the model. This makes it simpler to find corresponding joints in human hand to apply appropriate kinematics and control during motion. As depicted in table 3.1, all fingers except thumb has similar structure where the first two joints are inter-connected in a fixed tendon.

Shadow Hand Structure (Joints)	
FF1 / MF1 / RF1 / LF1	Coupled
FF2 / MF2 / RF2 / LF2	
FF3 / MF3 / RF3 / LF3	
FF4 / MF4 / RF4 / LF4	
LF5	
TH1	
TH2	
TH3	
TH4	
TH5	
WR1	
WR2	

Table 3.1: Shadow Hand Structure (Joints)

A fixed tendon in a robot model means the movement of the coupled joints are strictly linked by a predefined ratio and act as a single unit. This makes sure that the model mimics the biological behaviour of a human hand. The shadow hand model has 22 degrees of freedom (DoFs) in total, meaning there are 22 actuator joints. The joints are grouped based on functional and anatomical segmentation, where each joint corresponds to an actuator (Table 3.2).

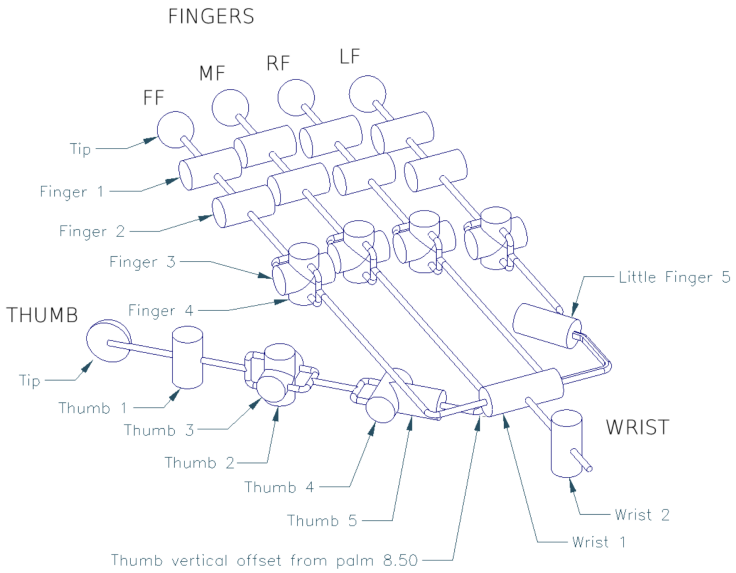


Figure 3.1: Technical drawing of the shadow hand model from Shadow Robot Company [2022]. The bigger the number is in the label of the joint, the close the joint is to the palm of the hand.

The wrist provides two DoFs for overall hand orientation, while the thumb, with its five independent DoFs, enables fine manipulation. Each of the index, middle, and ring fingers has a three-DoF configuration, balancing flexibility and stability, whereas the little finger follows a similar structure with an additional metacarpal joint for enhanced adaptability. The joints named as *middle_distal* are the coupled joints and acts as an actuator that control movements of middle and distal phalanx.

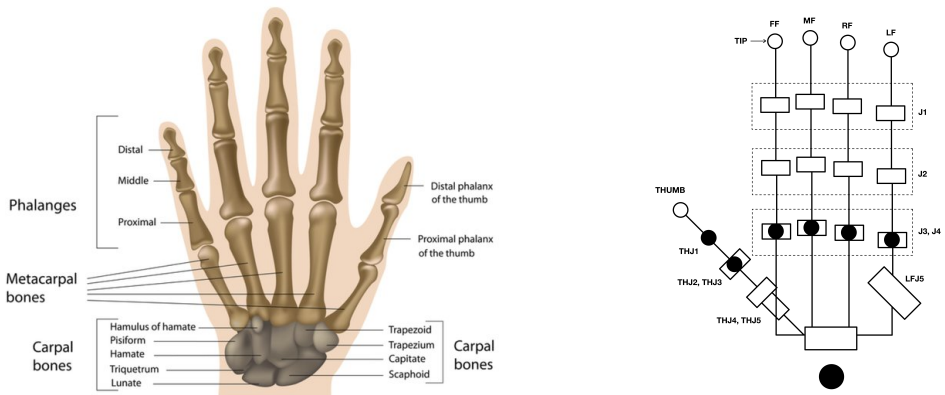
Joint Groups	**Joints**					**DoFs**
	5	**4**	**3**	**2**	**1**	
Wrist	-	-	-	wrist_y	wrist_x	2
Thumb	thbase	thproximal	thhub	thmiddle	thdistal	5
Index (FF) Middle (MF) Ring (RF)	-	knuckle	proximal	middle_distal		3
Little	metacarpal	knuckle	proximal	middle_distal		3

Table 3.2: Joint Groups and Degrees of Freedom (DoFs)

Understanding how the modeling of shadow hand reflects biological structure of a human hand, it is crucial to understand kinematics of human hand when playing piano and how robotic kinematics can be computed to mirror the dexterous hand movement.

3.2 Simulation

The experiment set up of the research includes shadow hand robot and a 88-key piano model. The shadow hand robot is Dexterous Shadow Hand model from Shadow



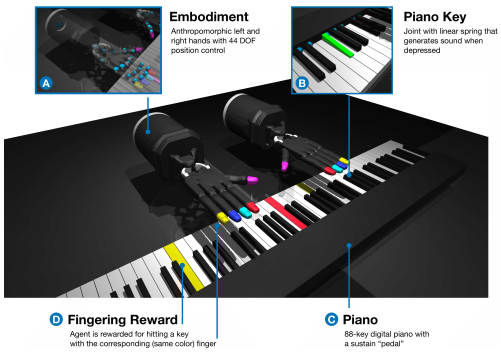
(a) Human hand (Iglovikov et al. [2018]) bone structure. Each phalanx of the hand is a link between the two joints it is connected to other link.

(b) This figure shows how shadow hand model used in the experiment is structured, mimicking human hand structure for its dexterity.

Figure 3.2: This visualize how the human hand structure is modeled into shadow hand in 2D diagram. Fingertips are represented as circle without filling. Joints are represented with rectangles and/or filled circles, which correspond to uniaxial, or hinge-like, joints and multiaxial joints, respectively.

Robot Company [2025]. MuJoCo is used as the physics engine to simulate the robot manipulation. This research is overall using RoboPianist (Zakka et al. [2023]) package which has environment composer and a viewer that extends mujoco viewer. This enables the manipulation run with user input such as MIDI file and control timestep values. The viewer renders the execution of the robot manipulation and reward graphs to observe the performance of the shadow hand robot.

MuJoCo, Multi-Joint dynamics with Contact, physics engine was chosen for this research due to its computational efficiency, accurate contact modeling, and suitability for high-dimensional robotic systems. MuJoCo excels in handling complex articulated systems with soft and precise contact dynamics, making it ideal for simulating interaction of dexterous robotic hands and the piano keyboard object. The inverse and forward dynamics capabilities in MuJoCo allow precise trajectory tracking while incorporating external forces, essential for evaluating motion planning under dynamic constraints. On top of that, MuJoCo provides a more realistic and responsive simulation compared to other engines that rely on more discrete contact models and tend to introduce numerical instability in fine-grained manipulation tasks, due to its computational speed and efficient physics-based trajectory execution. the flexibility of MuJoCo in handling custom torque-based controllers and continuous dynamics modeling further enhances its suitability for this research, where smooth, collision-free motion and dynamic adaptation are critical.



(a) This is how the Robopianist viewer application renders environment with a piano model and shadow hand model interactive with each other. This illustrates the basic set up of the simulation environment on MuJoCo viewer that was implemented by Zakka et al. [2023].

(b) This is reward graph rendered on the viewer application to show time and depth that the key was pressed during manipulation.

Figure 3.3: Key components from RoboPianist viewer application

3.3 Task planning

Given the task to play a sequence of notes on piano, the very first input to the framework is the MIDI file. In order to get the index of the key in the model given provided MIDI file, RoboPianist converts the data into note sequence where each note consists of its corresponding pitch, velocity, starting time, and ending time. The midi data is mapped into note sequences that is appropriate for the `PianoWithOneHand` task class. The MIDI data saved and retrieved from the task represents information of the music sheet to play, including note sequence, tempos(qpm), and total time of the song. The note sequence data is the main focus in this stage of computational framework. Each note in the sequence includes pitch, start time, end time, and velocity values, which is required to be mapped to the corresponding key press motion. The index of the key to press is found by pitch $-$ 21 and the duration of the ”press” state of the finger about the current key that is assigned with is defined by the start time and end time values provided in the input data. This is to make sure the system considers the maneuvering rules and constraints. The Table 3.3 presents the maneuvering framework that was introduced by Yeon [2022] and the constraints defines the motion constraints for shadow hand robot of my project.

Finger-Key(Note) Coordination In order to ensure dexterity and complexity of the given task, there are some rules defined when the framework assign finger to press a key that corresponds to the note read from given MIDI file:

1. If there are less than 2 keys between current and next keys, each finger is assigned to keys in order from thumb(0) to middle finger (2).
2. If there are more than or equal to 2 keys between current and next keys, assign middle finger to the next key.

Finger State	Constraints
Press	• Finger-Key Position • Finger-Key Orientation • Palm-Key Clearance
Prepare & Release	• Z-Clearance of Finger Positions (from Target Keys) • Horizontal Slacks of Finger Positions • Palm-Key Clearance
Transition	• Interpolation of Palm Pose • Palm-Key Clearance • Limiting Maximal Change of Palm Orientation

Table 3.3: Subsequences of finger states that maneuver the computation of inverse kinematics by constraining the movement.

3. Allow thumb to go underneath after middle finger if the next is right next to the key the middle finger is assigned to.

This ultimately regulates the movement of the fingers for the given task and rule 3 gives the dexterity to the robot manipulation as demonstrated in Figure 3.4, providing feasible configuration task space.

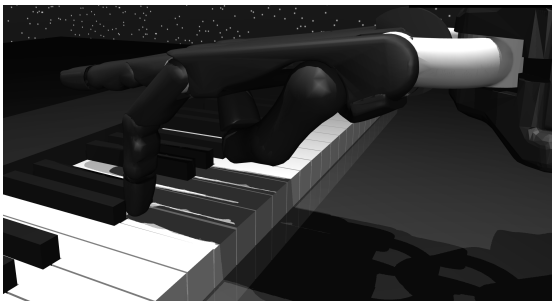


Figure 3.4: Thumb is assigned to press the next note. The current note is assigned to index finger, and having the coordination system allows the thumb to go underneath index finger. This contributes significantly to dexterity of robot manipulation by mimicking human hand kinematics when playing piano.

3.4 Motion planning

Rapidly-exploring Random Tree (RRT) algorithm was used to implement motion planning approach of this project, guiding the robotic hand's fingers to target piano keys, focusing on kinematic positioning. According to Karaman et al. [2011], RRT is computationally efficient and effective in generating possible feasible solutions.

This makes the approach more adaquate to the project as it equips the computational framework with reliable fundamental component. The `PianoWithOneShadowHand` class has been used as the base of the implementation as the task-level planning and definition was implemented by Zakka et al. [2023]. The RRT approach was integrated with the class, beginning by identifying the current joint positions of each finger, derived from the MuJoCo physics engine, and the desired key positions, obtained via the piano's site coordinates adjusted for pressing depth. For each finger assigned to a note (via the `_assign_fingers` method), the RRT motion planning function constructs a collision-free trajectory by iteratively expanding a tree of joint configurations from the starting joint configurations toward a goal configuration that is determined through inverse kinematics. The inverse kinematics technique is using `dm_control`'s `qpos_from_site_pose` function, which is explained more in detail in the next section. The motion planner employs a step size of 0.5 radians and goal bias of 0.2 to balance exploration and convergence, terminating after 1000 iterations or when the goal is reached within a tolerance of twice the step size. Joint limits are enforced to ensure feasibility, and the resulting path is smoothed into discrete segments (10 steps per segment) for execution. Collision avoidance is implemented and called in the motion planner function, ensuring the resulting path is proficiently evaluated for contact constrains in manipulation. This approach was first tested without dynamics, focusing solely on positional accuracy, providing a baseline for subsequent dynamic enhancements.

Mathematically, the motion planning problem is formulated to move a finger's joint configuration from an initial state q_{start} to a goal state q_{goal}, where:

- $q \in \mathbb{R}^n$ represents the joint angles for n joints associated with a finger
- $q_{\text{start}} = [q_1, q_2, ..., q_n]$ is the current joint configuration, extracted from the physics data.
- q_{goal} is computed via inverse kinematics to position the fingertip at the target key's site, adjusted by 5mm for pressing.

The constraint for the joint space is defined by limits:

$$q_{\min,i} \leq q_i \leq q_{\max,i}, \quad i = 1, 2, ..., n \tag{3.1}$$

where $[q_{\min,i}, q_{\max,i}]$ is extracted from the joint range defined in the model itself.

Using this problem formulation, the RRT algorithm implemented as the pseudocode 1. The path computed from the motion planning approach is smoothened and extracted as a Trajectory as demonstrated in pseudocode 2.

3.5 Inverse kinematics[1]

Inverse kinematics (IK) computation technique was used to find desirable joint configurations given a desired fingertip positions, ensuring kinematic feasibility for piano key presses. `dm_control` is one of the important package that is used in the project. For inverse kinematics, the `qpos_from_site_pos` function is used for better compatibility

[1] Tunyasuvunakool et al. [2020]

of the result with the simulation environment that was composed using `dm_control Environment composer`. This function iteratively:

1. Computes the error between the target and current site pose
2. Computes the Jacobian to determine how joint movements affect the erro.
3. Solves for $\Delta\mathbf{q}$ using a pseudo-inverse approach.
4. Applies updates to $\mathbf{q}$ with constraints, ensuring:
    - update step does not exceed the value given for parameter `max_update_norm`
    - process stops if the improvement is too slow (`progress_thresh`)
    - loop terminates when error $||e||$ falls below `tol`

The iteration steps when:

- error $||e||$ smaller than `tol` or
- number of iterations reaches `max_steps` or
- update step is too small relative to the error.

Once the inverse kinematics succeed, it returns $\mathbf{q}$, error norm, number of steps taken. Here, the mathematical explanation of the function is provided:

Problem Formulation

This function finds joint configurations or positions $\mathbf{q}$ that move a specific site, or end-effector, to a desired position, selectively taking desired orientation in 3D space. The given parameters for the problems are target position $\mathbf{p}_{\text{target}} \in \mathbb{R}^3$, target orientation in quaternion $\mathbf{q}_{\text{target}} \in \mathbb{R}^4$, and a set of joint angle $\mathbf{q}$ current pose. The function iteratively update $\mathbf{q}$ to minimize error:

$$e = \begin{cases} \mathbf{p}_{\text{target}} - \mathbf{p}(\mathbf{q}) & \text{transitional error} \\ \text{rotation error}(\mathbf{q}_{\text{target}}, \mathbf{q}(\mathbf{q})) & \text{rotational error} \end{cases} \tag{3.2}$$

where $\mathbf{p}(\mathbf{q})$ and $\mathbf{q}(\mathbf{q})$ are current end-effector position and orientation.

Error Computation

Given $\mathbf{p}_{\text{target}}$ and $\mathbf{q}_{\text{target}}$, function computes the difference between the current and target positions and orientations, respectively. For translational error, $e_{\text{pos}} = \mathbf{p}_{\text{target}} - \mathbf{p}_{\text{current}}$, while rotational error is written as $e_{\text{pos}} = \log(\mathbf{q}_{\text{target}} \cdot \mathbf{q}_{\text{current}}^{-1})$ where log converts a quaternion to a rotation vector. It scales the error by weight.

Jacobian-Based Update

Now the function updates the joint angles $\mathbf{q}$ by leveraging differential kinematics, using the Jacobian matrix $\mathbf{J}$:

$$\mathbf{J} = \frac{\partial \mathbf{x}}{\partial \mathbf{q}} \tag{3.3}$$

where $\mathbf{x} = [\mathbf{p}, \mathbf{q}]$ is end-effector position. Jacobian matrix describes how small changes in joint positions affect the end-effector position and orientation. The method uses:

$$\Delta\mathbf{q} = \mathbf{J}^{+}\mathbf{e} \tag{3.4}$$

where $\mathbf{J}^{+}$ is the pseudo-inverse of $\mathbf{J}$ as the jacobian matrix is not invertible, and this is using *Tikhonov regularization*:

$$\mathbf{J}^{+} = (\mathbf{J}^{T}\mathbf{J} + \lambda I)^{-1}\mathbf{J}^{T} \tag{3.5}$$

where λ is small regularization term.

3.5.1 Collision Avoidance

Collision avoidance is implemented within the system by integrating a collision-checking mechanism into the RRT motion planning algorithm, ensuring that the robotic hand's fingers move safely without colliding with each other or the piano keys. In the `_plan_with_rrt` method, we use MuJoCo's collision detection capabilities to evaluate potential joint configurations during the RRT tree expansion. Specifically, for each sampled joint position, we set the physics state to the candidate configuration, call `mj_forward` to update the system, and check for collisions using data of bodies in contact. If a collision is detected (between fingertips or a wrong piano key), the configuration is rejected, and the RRT algorithm samples a new position. This process ensured that the generated trajectories are collision-free, allowing the fingers to navigate the complex workspace of the piano while maintaining safe and efficient motion during key presses.

3.6 Motion Control

3.6.1 Dynamics Applied

To incorporate realistic motion, dynamics were introduced by augmenting the RRT planner with velocity considerations, reflecting the physical behavior of the robotic hand. In this phase, the framework involves computation of desired joint velocities $\dot{q}_{\text{desired}}$ along side positions, derived as the finite difference between consecutive trajectory points:

$$\dot{q}_{\text{desired}} = (q_{k+1} - q_k)/\Delta t, \quad \text{where } \Delta t = 0.05 \text{ seconds taken as the control timestep} \tag{3.6}$$

Current joint velocities $\Delta q_{\text{current}}$ were retrieved from the physics data of the composed simulation environment, and the control action was extended to included a velocity error term:

$$a = K_p(q_{\text{target}} - q_{\text{current}}) + K_d(\dot{q}_{\text{desired}} - \dot{q}_{\text{current}}) \tag{3.7}$$

where gains K_p and K_d are set to 200 and 10, respectively, after experiments conducted on different values. This approach accounts for inertial and damping effects implicityl through MuJoCo's physics simulation, enhancing trajectory smoothness and realism compared to the kinematics-only baseline, though without explicit dynamic model terms like mass or gravity.

3.6.2 Torque Control

A full motion control with dynamics and Proportional-Derivative (PD) regulation is integrated to the implementation, building on the RRT trajectories and velocity-augmented actions. The velocity-based controller computes control torques τ as:

$$\tau = K_p(q_{\text{target}} - q_{\text{current}}) + K_d(\dot{q}_{\text{desired}} - \dot{q}_{\text{current}}) + G \tag{3.8}$$

where G is the gravity compensation term derived from the physics data. Joint positions and velocities are tracked by extracting `qpos` and `qvel` values, respectively, with K_p and K_d values tuned to balance responsiveness and stability. Torques are clipped to actuator limits and applied to the simulation environment. This method ensured precise key pressing by compensating for gravitational forces and dynamic interactions, as validated through fingertip distance metrics and energy consumption analysis in the simulation environment.

3.7 Evaluation Framework

To assess the effectiveness of the proposed framework or motion planning and control in robotic manipulation, an evaluation framework was developed to quantify the performance of the robotic hand in playing a piano piece within the `dm_control` simulation environment. The evaluation framework focuses on measuring the accuracy, precision, and efficiency of the robotic hand's actions, directly addressing the fundamental challenges of robotic manipulation outlines in the hypothesis - namely, precise interaction with the environment, high-dimensional control, and efficient movement (Section 1.3). This section describes the evaluation metrics, their computation, and their implementation within the framework, using the test case of "Twinkle, Twinkle, Little Star" to illustrate the evaluation process.

Comparing Predefined Trajectories and Executed Trajectores

Comparing the predefined trajectory with the executed trajectory serves as a fundamental evaluation framework for assessing the accuracy and effectiveness of the proposed approach. The predefined and executed trajectories are generated by the motion planning algorithm and observed in the MuJoCo simulation, respectively. This comparison provides insights into trajectory tracking performance, deviation due to dynamic effects, and the impact of control strategies. A precise alignment between those indicates that the motion planning algorithm successfully translates theoretical paths into real-world execution. However, deviations can highlight challenges such as unmodeled dynamics, external forces, joint limitations, or inaccuracies in the control system. By analyzing trajectory deviation graphs, we can quantify errors in terms of position drift, timing inconsistencies, and overshooting at keypress events. Moreover, evaluation on the effect of dynamics, or varying velocities from MIDI input, and PD controller implementation on trajectory execution helps determine the role of feedback mechanisms in mitigating errors. This framework enables an objective assessment of whether the proposed method improves dexterity in robotic manipulation, ensuring both collision-free movement and musically accurate keypress execution. Additionally, predefined trajectories allows

analysis on the positions of fingers at a timestep to evaluate if the system mirrors human hand dexterity proficiently - such as thumb path mapped underneath other fingers.

Key Press Accuracy For each note in the MIDI file, the scheduled timestep t and key index $k = \text{note.pitch} - 21$ are determined. At timestep t, the key's state is compared to the goal state (`self._goal_current[k]`). The key press is correct if the key is pressed (state exceeds the threshold) when the goal state is 1, or unpressed when the goal state is 0. The accuracy computation can be written as:

$$\text{Key Press Accuracy} = \left(\frac{\text{Number of Correct Key Presses}}{\text{Total Number of Notes}} \right) \times 100 \tag{3.9}$$

This directly evaluated the framework's ability to achieve precise interaction with the environment, a core challenge in robotic manipulation. Accurate key presses ensure that the robotic hand plays the correct notes, fulfilling the primary objective of the piano-playing tasks.

Energy Reward By having MIDI file as input, cumulative energy reward and average energy consumption are useful to evaluate the impact of dynamics constraints and motion controller. This framework helps optimize the hand's motion for both energy efficiency and task performance with the computation written as the equation below:

$$R_{\text{energy}} = -\alpha \sum_i \tau_i^2 \tag{3.10}$$

where τ_i are the the torques applied at each timestep, collected from a method called `_update_hand_position`. Difference between the frequency of overshooting over timestep of different version of the system indicates the performance of the system on motion control over the trajectory.

Reward Visualization

In order to provide qualitative insight into the evaluation of the framework's performance, visualization of the reward over time is important, showing how the reward evolves over the episode and identifying timesteps where performance issues occur. The reward at each timestep is collected during the simulation and plotted using a plotting library (e.g. Matplotlib). the x-axis represents the timestep, and the y-axis represents the reward value.

Algorithm 1 RRT Motion Planning Algorithm for a finger

Input:
- q_{start} ← start, or current, joint configuration,
- q_{goal} ← goal or target joint configuration,
- $q_{\min}, q_{\max}$ (joint limits),
- Δ: step size ,
- max_iter: maximum number of iteration allowed,
- goal_bias: threshold to determine whether the sampled configuration reaches, or is close to, the goal configuration.

Output: smooth_path (path of joint configurations)
Initialize more variables:
$T \leftarrow \{(q_{\text{start}}, \text{Null})\}$ ▷ Initialize tree with start node
for $i = 1$ **to** max_iter **do**
  if random$(0,1) <$ goal_bias **then**
    $q_{\text{rand}} \leftarrow q_{\text{goal}}$
  else
    $q_{\text{rand}} \leftarrow \text{Uniform}(q_{\min}, q_{\max})$
  end if
  $q_{\text{near}} \leftarrow \arg\min_{q \in T} \|q - q_{\text{rand}}\|_2$ ▷ Find nearest node
  $d \leftarrow q_{\text{rand}} - q_{\text{near}}$
  $\delta \leftarrow \|d\|_2$
  if $\delta < 10^{-6}$ **then**
    continue
  end if
  $\hat{d} \leftarrow d/\delta$ ▷ Normalize direction
  $q_{\text{new}} \leftarrow q_{\text{near}} + \min(\Delta, \delta) \cdot \hat{d}$
  $q_{\text{new}} \leftarrow \text{clip}(q_{\text{new}}, q_{\min}, q_{\max})$ ▷ Enforce joint limits
  if not is_collision(q_{new}) **then**
    $T \leftarrow T \cup \{(q_{\text{new}}, q_{\text{near}})\}$ ▷ Add to tree
    if $\|q_{\text{new}} - q_{\text{goal}}\|_2 < 2\Delta$ **then**
      $P \leftarrow \text{backtrack}(T, q_{\text{new}}, q_{\text{start}})$ ▷ Extract path
      $Q \leftarrow \text{interpolate}(P, N = 10)$ ▷ Smooth with 10 steps per segment
      return Q
    end if
  end if
end for
return $\emptyset$ ▷ No path found

Algorithm 2 Trajectory Planning with RRT path

Input:
- q_{goal} ← position of the key to press,
- finger index ← the finger assigned to the target key,
- physics ← physics data of the composed environment for the task

Output: smooth_path (path of joint configurations)
q_{start} ← Start joint configuration retrieved from `physics.data.qpos`
`fingertip_position`, `fingertip_xmat` ▷ Get current fingertip position(3,) and orientation(3, 3)
`key_pos` ← position of the site element of the key
`press_pos` ← adjust the key position to z-axis to press the key
`relative_pos` ← convert the key press position to the fingertip's frame
`ik_result`←`qpos_from_site_pose(physics, full_site_name, local_press_pos, None, joint_names)`
if `ik_result.err_norm` > 0.2 **then**
    q_{goal} ← `_project_toward_goal()`
    if q_{goal} None **then**
        return $\emptyset$ ▷ IK fails
    end if
    q_{goal} ← `ik_result`
end if
RRT parameters:
- $q_{\text{min}}, q_{\text{max}}$ (joint limits),
- Δ (step size = 0.5),
- max_iter (1000),
- goal_bias (0.2)

`path` ← `RRT_motion_planner()`
if Path found **then**
    `smooth_path` ← `path`
end if
`smooth_path` ← `original_qpos`
return `smooth_path` ▷ Return smoothened path

Chapter 4

Experiment and Evaluation

The goal of experiment and evaluation is to assess how different levels of motion planning and control contribute to robotic dexterity in task and motion planning problem in robot manipulation. The three main experiments are done in order to ultimately showcase the refinement made on the basic motion planning approach throughout the implementation, anticipating its contribution to dexterous robot manipulation. As tasks require both collision-free trajectory planning and adaptive control to handle dynamic constraints, this study systematically analyses three approaches:

1. **Motion Planning (MP) Only**, which evaluates whether a predefined trajectory ensures precise finger-key coordination
2. **Motion planning with Dynamics (MP + Dynamics)**, which assesses how velocity constraints (derived from musical phrasing) affect motion feasibility.
3. **Motion Planning with Dynamics and PD control (MP + Dynamics + PD controller)** tests whether motion control improves stability and execution accuracy under dynamic constraints.

Each experiment is conducted on a robotic manipulator executing note sequences and chord progressions derived from MIDI files. The system is evaluated using trajectory accuracy, velocity adaptation, collision avoidance, and execution stability. The results will demonstrate how integrating dynamic constraints and control mechanisms impacts dexterity in robotic manipulation.

4.1 Experimental Setup

This section is extending what is explained in section 3.2, but with more detailed and compatible with the experiments and their goals. The experimental setup consists of a robotic manipulator, a motion planning framework, and a task environment where the robot executes key-finger coordination tasks based on musical note sequences. The goal is to evaluate how different levels of motion planning and control influence trajectory accuracy, collision avoidance, and execution stability.

Robotic System The Robot model used for the manipulation is Dexterous Shadow Hand

model (Shadow Robot Company [2025]) that interacts with a piano object model built when they are launched to the MuJoCo simulation viewer. The action in the trajectory is defined for the hand, which is 22 degrees of freedom. The robot is equipped with fingertip site properties that indicates the end-effector of each fingers.

Task Definition The task involves executing predefined note sequences and chord progressions extracted from MIDI files, where each note is associated with a velocity value, indicating the intensity of key presses. These velocity values serve as dynamic constraints in the motion planning process, influencing the planned trajectory and execution timing.

Motion Planning and Control This part of the framework is implemented using RRT algorithm, ensuring collision-free trajectories for the robot's fingers. In the second phase of experiments, these planned trajectories are further refined by incorporating dynamic constraints, where note velocities from the MIDI dataset dictate movement speeds, testing the impact of temporal variations on execution stability. Finally, in the third phase, a Proportional-Derivative (PD) controller is applied to regulate trajectory tracking, compensating for errors introduced by dynamic adjustments and ensuring smoother execution.

The experiments are conducted in a MuJoCo physics engine, with piano keys positioned according to a predefined spatial layout. Each trial involves executing a sequence of key presses using one of the three approaches: motion planning alone, motion planning with velocity adaptation, and motion planning with velocity adaptation plus PD control. Performance is evaluated based on trajectory indicators such as oscillations in the executed trajectory. By systematically analyzing these factors, this study aims to quantify the improvements in robotic dexterity when integrating dynamics constraints and control mechanisms into motion planning.

4.2 Motion Planning (MP Only) Experiment

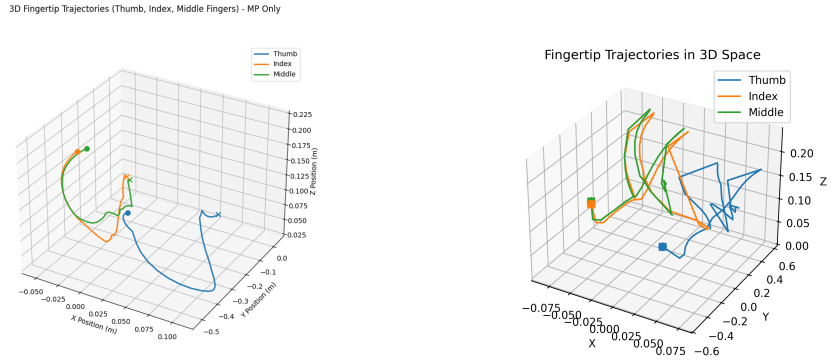
The idea of this experiment is to evaluate how well the robot can generate and execute collision-free trajectories for key-finger coordination. This is corresponding to the research question:

Does the proposed motion planning approach generates collision-free trajectory of precise key-strikes in the robot manipulation given finger-key coordination of input note sequences?

The experiment was done before velocity constraints were considered in the implementation of a motion planning algorithm, ensuring spatial feasibility. In this experiment, trajectory feasibility and motion execution are evaluated.

The Figure 4.1 demonstrates the predefined trajectory and the trajectory extracted from the execution on the robot hand model. There is significant difference between the two figures (4.1a and 4.1b).

The predefined trajectory reflect the independence between motions of each finger throughout the trajectory. Index and middle fingers are then to align with each other



(a) Predefined trajectory (b) Trajectory executed

Figure 4.1: The Figure 4.1a is the predefined trajectory computed from the framework that only has RRT motion planner implemented. The Figure 4.1b is the fingertip positions when the predefined trajectory is executed on the shadow hand.

due to the biological structure of a human hand, but they do not overlap. The thumb trajectory is fairly distinctive from the other two trajectories also due to the biological structure where it has one extra joint to maneuver the finger over different range of action. On the other hand, the executed trajectory shows many spikes which are mostly tend to happen around the timestep when the motion is planned to strike a key.

This is because the predefined trajectory does not consider dynamics that is important to be considered to compute the trajectory that needs to be executed to a physics engine. Therefore, even though the computed trajectory is collision free, the execution time and accuracy is not as good as it was anticipated and makes the robot run into a collision before reaching the key to press. This result confirms that a pure motion planning approach is effective in ensuring precise finger-key coordination while maintaining collision-free within the predefined path, but not in execution. The uniform movement speed does not reflect natural musical phrasing, highlighting the need to incorporate dynamic constraints in the next state.

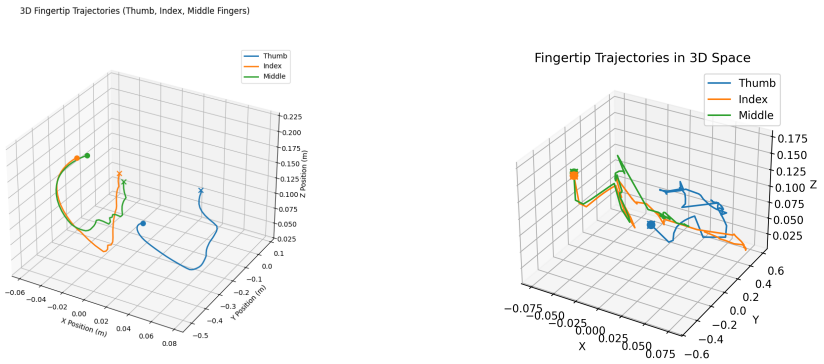
This leads to the next experiment, introducing velocity constraints from the MIDI dataset and make the framework updates dynamics data along the computation of the path. This is expected to allow the robot's motion to adapt to tempo changes and dynamic variations in key presses.

4.3 Motion Planning with Dynamics (MP + Dynamics) Experiment

Given the limitation founds from the previous experiment where the motion planning algorithm does not consider dynamics, the experiment is introduced to address the research question:

How does applying dynamics constraints, or a sequence of velocities, affect the execution of predefined trajectory?

This experiment was designed in order to assess how incorporating velocity constraints from MIDI data affects trajectory stability and accuracy. The implementation of adapting varying note velocities in addition to the motion planning approach was mainly observed for its trajectory deviations, execution consistency, and stability indicators.



(a) Predefined trajectory (b) Trajectory executed

Figure 4.2: The Figure 4.2a is the predefined trajectory computed from the framework that only has dynamics applied to RRT motion planner implemented. The Figure 4.2b is the fingertip positions when the predefined trajectory is executed on the shadow hand.

The predefined trajectories shows more curves throughout the trajectories compared to the ones generated from MP Only version of the system. This is because considering velocities, the computation for each action is considering more constraints and involves more pressing and releasing motion. This indicates improved reflection of the given task as the system knows when to swap states of fingers from press to release more in detail. Observing that the trajectories are still quite far apart from each other, especially thumb trajectory, the dexterous finger movements within human hand should be mimicked better. Comparing these trajectories to their execution recordings, the spikes can still be observed around the timestep when the downward actions are defined. This indicates that the strike and non-strike states getting updated makes the finger spike as system only considers the velocities and do not compute the right torque to apply. This is a problem because in such complicated environment, each action can have different velocities and there can be a fast change in states needs to be made, which cause spike in velocity of the finger movement. Application of velocity constraints does not only cause the spike, but also it makes the system consider the duration of a state that smoothen the trajectories compared to those from MP-only version of the system.

Introducing velocity constraints enhances realism and expressiveness but at the cost of trajectory accuracy and stability. The findings indicate that while the robot adapts to musical dynamics, or velocities, the lack of correction mechanisms leads to drift and instability, particularly in high-speed transitions. As a mitigation of this limitation, the next experiment will introduce motion control strategy to compensate for trajectory deviations and ensure smooth, stable movements.

4.4 Motion planning with Dynamics and PD Control (MP + Dynamics + Controller) Experiment

The version of computational framework ran on this experiment is the final model of the project, and it is evaluated for its trajectory accuracy, reduction in oscillations, and response time. The research question for this phase is:

If dynamics applied to the trajectory cause oscillation in execution, does integrating a motion controller improve trajectory execution accuracy and stability?

This experiment was designed and enabled when the implementation of the computational framework involved minimization of trajectory deviations and refinement in smoothness of execution.

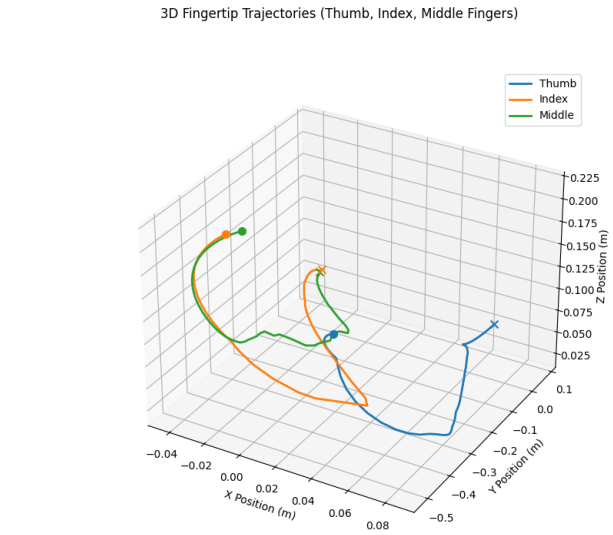


Figure 4.3: Predefined trajectory for motion planner with motion controller that keeps the robot model execute the trajectory smoothly when dynamics is applied.

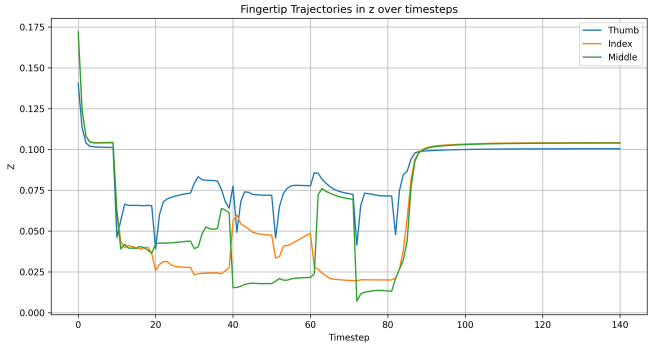


Figure 4.4: This graph shows the movement of each fingertips along the z axis.

From this experiment, trajectory accuracy was observed to be improved even with

applying desired velocities for each key press motion. This demonstrates that the PD controller effectively corrected trajectory deviations, which affected the stability of the execution time of the system. More consistent movement across the given note sequence was observed, which means the oscillations were reduced. On top of that, an important improvement made by the application of controller in the system is that the movement where thumb needs to go under other fingers to reach the next key was enabled. As the active motion control, or error compensation, is added to the manipulator, the hand position gets adjusted appropriately for the thumb-under motion, making the project closer to its contribution to improving dexterity in robot manipulation.

However, as the PD controller introduces additional computational overhead, it requires fine-tuning of gain parameters to prevent excessive correction, or overshooting). The fine-tuned gain parameters in this experiment are : $K_p = 50.0, K_d = 10.0$. This indicates that in different scenarios of robot manipulation, those parameters would need to be fine-tuned properly and this can cause inefficiency in implementations and testings. On top of that, lack of consideration on dynamic interactions, such as stiffness of the piano key and material of the fingertip body, is a limitation of the system, as more advanced control techniques introduced to address this could result in even more dexterous manipulation.

This experiment confirms that incorporating a PD controller was successfully done in the computational framework, significantly improving trajectory accuracy and execution stability when executing velocity modulated motion plans. The controller effectively compensates for speed-induced errors, ensuring precise and natural robot manipulation with active motion corrector, or controller.

4.5 Discussion and Research Implication

In this section, a comparative analysis of the three experimental approaches: Motion planning only, Motion planning with dynamics, and Motion planning with dynamics and PD controller. Overall impact of dynamic constraints and control mechanisms on trajectory accuracy, execution stability, and overall dexterity in robotic manipulation is evaluated through the experiments. In Table 4.1, the trade-offs between different approaches are depicted in organised way, providing some key insights in the project:

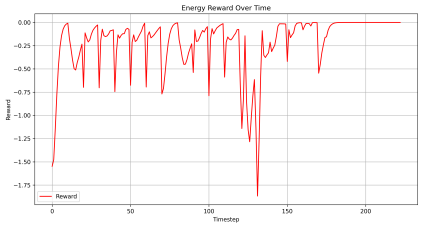
- Motion planning only is best for simple, structured tasks where execution speed is constant.
- Motion planning with dynamics is necessary for realistic movement, but alone it lacks precision.
- Motion planning with dynamics and a motion controller achieves both natural motion and precision, effectively showing that addressing fundamental problem can contribute to the dexterity of robot manipulation.

Summarizing the experimental observations, the proposed motion planning and control framework successfully guides that Shadow Hand to execute precise finger-key coordination based on MIDI sequences. The system demonstrates a high degree of accuracy in key strikes, as evidenced by stable key presses (Figure A.2). Additionally, qualitative assessments from simulation visualizations confirm that the fingers move as intended.

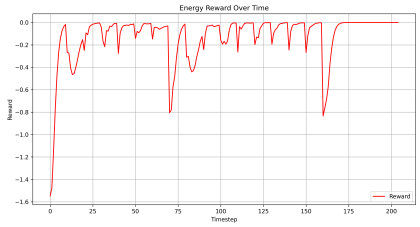
Approach	Strengths	Weaknesses	Best Use Case
Motion Planning Only	Simple, collision-free predefined trajectory	Unrealistic motion, lacks expressiveness, collision in execution	Static or predictable tasks
MP + Dynamics	considers velocity constraints	Accuracy loss (oscillation), unstable timing	Dynamic tasks requiring natural motion
MP + Dynamics + Motion Controller	High accuracy, stable execution, natural movement	Computational cost, requires tuning	Real-time, expressive robotic manipulation

Table 4.1: Trade-offs Between Different Approaches

However, the introduction of musical phrasing and dynamics variation (velocity-based control) introduces challenges, such as oscillatory behavior in the trajectory. The implementation of a proportional-derivative (PD) controller mitigates these issues by smoothing oscillations, as depicted in Figure 4.5a and Figure 4.5b. Comparing those two figures, the energy reward over time has less deviation when the computational framework involves controller, as it compensate the errors in motion, which prevents the fingertip trajectory from spiking up.



(a) Energy reward over timestep illustrated for MP+Dynamics version.



(b) Energy reward over timestep illustrated for MP+Dynamics+Control version.

Figure 4.5: Comparing the energy reward over time for two different version of the system, it can be observed that having controller implemented in the framework reduce deviation of energy overtime.

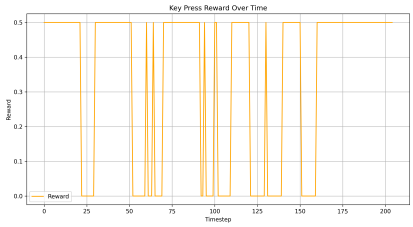
Given the core research questions outlined in Section 1.3, the results affirm that the system achieves precise finger-key coordination and successfully incorporates musical dynamics. However, challenges remain in maintaining collision-free motion during complex chord progressions. This section discusses the contributions of the study in enhancing dexterity in robotic manipulation by analyzing the system's performance across three key aspects:

1. precise key press motion given finger-key coordination,
2. collision-free motion during chord transitions,
3. and the influence of musical phrasing and dynamics on trajectory control.

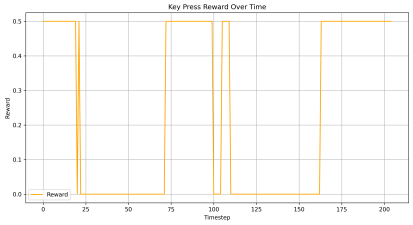
Precise Finger-Key Assignment and Execution The ability to accurately coordinate finger movements with key presses is crucial for dexterous robotic manipulation. The Shadow Hand, with its 23 degrees of freedom, requires sophisticated control mechanisms to achieve such fine-grained motion. The system's effectiveness in this regard can be attributed to:

- Inverse Kinematics (IK) solver - ensures that the fingers reach target key positions while maintaining kinematics feasibility.
- RRT Based Trajectory Planning - generates collision-free paths, adapting to the constraints of the robotic hand.
- Finger assignment and Thumb-Under adjustment - optimizes finger selection to prevent unnecessary reconfiguration during play.

The stable key press reward throughout the experiments confirms that the system achieves precise coordination, thereby contributing to robotic dexterity. Despite the complex task constraints imposed by the piano-playing environment, the framework demonstrates an ability to handle fine motor tasks, which is a core challenge in dexterous robotic manipulation. As illustrated in 4.6a and 4.6b, the system only with motion planning collide with other keys that are not supposed to be pressed. On the other hand, the fully implemented motion planning and control system yields the key press motions with the precise duration without infeasible collision. As the experiment is done in physics engine, the execution of trajectory can collide without motion controller within the system. Therefore, not only motion planning problem but also motion control problem is crucial to make smooth task and motion planning.



(a) Key press reward overtime for MP-only version.



(b) Key press reward overtime for full implemented version (MP + Dynamics + Controller)

Figure 4.6: Comparing the key press reward over time for two different version of the system, it can be observed that having controller implemented in the framework press at the right timing with the right duration.

Collision-Free Motion During Chord Progressions

Collision-free motion is a fundamental requirement in both robotic dexterity and motion planning. The motion planning framework successfully ensures collision-free key presses for simple note sequences, guiding the Shadow Hand to reach target keys at each timestep without unintentional contact. However, when handling more complex MIDI sequences with rapid transitions and dense chords, occasional invalid paths emerge, leading to unintended collisions.

To further enhance dexterity in this context, integrating a real-time collision detection system, such as proximity sensors in a physical setup, could improve response to dynamic constraints. Additionally, augmenting the RRT planner with a refinement step such as post-processing via optimization algorithms may yield smoother and more adaptive solutions. This approach could help manage the increasing complexity of multi-finger interactions in intricate musical pieces. On top of that, incorporating motion controller makes sure the hand or fingers collide into other keys while moving toward the targeted key. In Figure 4.6a, there are many key press events observed as the fingers collide with non-target keys while Figure 4.6b shows key press events with proper duration by having motion controller to compensate errors.

Musical Phrasing and Dynamic on Robot Trajectory

The application of musical phrasing and dynamics variations introduces additional complexity into trajectory planning and execution. The system attempts to capture the expressive nature of musical performance by mapping velocity values from MIDI data to finger movement speeds. However, discontinuous dynamic changes lead to trajectory instabilities, which were not fully mitigated by the control framework.

While the PD controller significantly reduces oscillations, achieving fully stable execution under varying musical dynamics remains a challenge. This highlights a crucial limitation in current motion control strategies for dexterous manipulation. Future improvements could involve adaptive control strategies, such as model-based reinforcement learning or torque optimization, to refine trajectory execution while maintaining expressiveness in robotic movement.

Overall, this study contributes to dexterous robotic manipulation by addressing key challenges in motion planning and control. The integration of trajectory planning, musical dynamics, and feedback control offers a structured approach to enhancing robotic expressiveness, bridging the gap between task and motion planning and physical execution.

Chapter 5

Conclusions and Future Work

This research aimed to enhance dexterity in robotic manipulation by developing a hierarchical motion planning and control framework, evaluated within a piano-playing environment using the Shadow Dexterous Hand. Addressing the fundamental challenge of task and motion planning (TAMP) under dynamic constraints, the study progressed through three experimental stages: Motion Planning (MP) alone, MP with dynamics, and MP with dynamics and a Proportional-Derivative (PD) controller. These stages were designed to answer specific research questions about trajectory generation, dynamic adaptation, and execution stability (Section 1.3). This chapter synthesizes the outcomes, reflects on their implications, and charts a course for future developments, situating the work within the broader context of robotic manipulation.

5.1 Achievements

The framework's development and evaluation yielded significant insights into improving robotic dexterity, directly addressing the hypothesis that a combined motion planning and control approach can mitigate TAMP challenges. The first stage, MP-only demonstrated that the RRT algorithm effectively generated collision-free trajectories for precise finger-key coordination, answering the first research question affirmatively in a planning context. Predefined trajectories showcased spatial accuracy, aligning fingertip with target keys. However, execution revealed collisions and spikes, indicating that kinematics alone insufficiently accounts for physical dynamics - a critical finding for high-dimensional systems.

Incorporating dynamics in a second stage addressed the second research question: how velocity constraints affect trajectory execution. By integrating MIDI-derived velocities, the framework enhanced movement realism, reflecting musical phrasing. Yet, this introduced oscillations and accuracy loss, highlighting a trade-off between expressiveness and stability. This result underscores the necessity of dynamics for naturalistic motion while exposing the need for corrective mechanisms in dynamic environments.

The third stage, integrating a PD controller, tackled the final research question: whether

motion control improves accuracy and stability under dynamic constraints. The results were transformative - executed trajectories smoothed significantly, oscillations reduced, and key press accuracy improved. A standout achievement was enabling the thumb-under maneuver, a hallmark of human-like dexterity, facilitated by active error compensation. Comparative analysis confirmed that the full framework (MP + Dynamics + PD) outperformed simpler approaches, achieving both precision and natural motion.

These achievements validate the hypothesis, demonstrating that integrating planning, dynamics, and control enhances dexterity in robotic manipulation. The framework's ability to handle a high-dimensional task like piano playing - mirroring human kinematics - positions it as a novel contribution to robotics, bridging theoretical planning with practical execution.

5.2 Critical Reflection

While the study achieved its objectives, several limitations temper its scope and scalability, offering a balanced perspective on its contributions. The reliance on the MuJoCo simulation environment, though advantageous for controlled testing, omits real-world factors like key stiffness or friction, potentially skewing execution outcomes. This simulation bias limits direct applicability to physical systems, where hardware variability and external disturbances could cause trajectory deviations observed in earlier stages.

The RRT planner's performance also needs to be investigated more in depth. While effective for simple sequences, it faltered during rapid chord transitions, producing occasional invalid paths. This reflects a broader challenge in sampling-based methods: capturing the full complexity of a 22 DoF configuration space. Similarly, the PD controller's success hinged on manually tuned gains, a process that proved time-consuming and task-specific, reducing adaptability across diverse scenarios.

Additionally, musical expressiveness remains challenge to some extend. Velocity constraints introduced dynamic richness but also instability during discontinuous changes, only partially mitigated by the PD controller. This suggests that current control strategies incompletely address the nuanced demands of artistic tasks, a gap between technical precision and human-like performance. Finally, the evaluation framework prioritized quantitative metrics over qualitative musicality, potentially overlooking subjective aspects critical to the piano-playing context.

These limitations do not undermine the study's success but highlight areas where theoretical developments met practical constraints, guiding the next steps for refinement.

5.3 Future Directions

Building on these reflections, several directions emerge to extend this work and address its shortcomings. First, transitioning to a physical Shadow Hand setup would validate the framework's real-world efficacy. Integrating sensors can enhance collision avoidance and dynamic adaptation, compensating for unmodeled interactions like key resistance.

The step would test the framework's robustness beyond simulation, aligning with industrial robotics goals.

Second, enhancing the motion planner offers significant potential. Augmenting RRT with optimization techniques such as trajectory smoothing using B-Splines or hybrid approaches like combination of sampling and learning-based planners can improve path feasibility for complex chords. Exploring Probabilistic Roadmaps (PRM) or neural motion planners might further reduce computational overhead and enhance adaptability, addressing the high-dimensional challenge.

Improvements on control strategies can make remarkable change in scalability and expressiveness. Adaptive control methods such as Model Predictive Control or impedance control can dynamically adjust gains and incorporate explicit dynamics. This can improve stability across tasks. Reinforcement learning, inspired by Zakka [2024], could optimize torque application for energy efficiency and musicality, refining the thumb-under motion and other dexterous maneuvers.

Finally, expanding the evaluation framework to include qualitative metrics - such as human listener assessments or audio analysis - would enrich performance assessment. This could quantify expressiveness, aligning the framework with human-robot interaction goals.

5.4 Concluding Statement

This thesis has advanced the field of robotic manipulation by demonstrating that a hierarchical framework integrating motion planning, dynamics, and control can significantly enhance dexterity in a high-dimensional, dynamic task. By achieving precise key strikes, adapting to musical phrasing, and stabilizing execution, the study offers robust baseline for TAMP challenges, with implications for both theoretical robotics and practical applications.

The process from MP-only to a fully controlled system reflects a progression toward human-like capabilities, a core aspiration of modern robotics (Billard and Kragic [2019]). While limitations remain, they illuminate a path forward - toward physical deployment, refined planning, and adaptive control. This project not only contributes a novel solution to dexterous manipulation but also inspired a vision of robots that seamlessly blend precision with artistry, fostering a future where technology mirrors and augments human skill.

Bibliography

Jaafar Ahmed Abdulsaheb and Dheyaa Jasim Kadhim. Classical and heuristic approaches for mobile robot path planning: A survey. *Robotics*, 12(4):93, 2023.

João Carlos Alves Barata and Mahir Saleh Hussein. The moore–penrose pseudoinverse: A tutorial review of the theory. *Brazilian Journal of Physics*, 42:146–165, 2012.

C Dario Bellicoso, Koen Krämer, Markus Stäuble, Dhionis Sako, Fabian Jenelten, Marko Bjelonic, and Marco Hutter. Alma-articulated locomotion and manipulation for a torque-controllable robot. In *2019 International conference on robotics and automation (ICRA)*, pages 8477–8483. IEEE, 2019.

Mehdi Benallegue, Jean-Paul Laumond, and Nicolas Mansard. Robot motion planning and control: Is it more than a technological problem? In *Geometric and Numerical Foundations of Movements*, pages 1–10. Springer, 2017.

Rodrigo Bernardo, João MC Sousa, and Paulo JS Gonçalves. A novel framework to improve motion planning of robotic systems through semantic knowledge-based reasoning. *Computers & Industrial Engineering*, 182:109345, 2023.

Aude Billard and Danica Kragic. Trends and challenges in robot manipulation. *Science*, 364(6446):eaat8414, 2019.

Michael Brady. *Robot motion: Planning and control*. MIT press, 1982.

Lu Dong, Zichen He, Chunwei Song, and Changyin Sun. A review of mobile robot motion planning methods: from classical motion planning workflows to reinforcement learning-based architectures. *Journal of Systems Engineering and Electronics*, 34(2): 439–459, 2023.

Chrisantus Eze and Christopher Crick. Learning by watching: A review of video-based learning approaches for robot manipulation. *arXiv preprint arXiv:2402.07127*, 2024.

Shinichi Furuya, Martha Flanders, and John F Soechting. Hand kinematics of piano playing. *Journal of neurophysiology*, 106(6):2849–2864, 2011.

Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4(1):265–293, 2021.

Alessandro Gasparetto, Paolo Boscariol, Albano Lanzutti, and Renato Vidoni. Path planning and trajectory planning algorithms: A general overview. *Motion and*

operation planning of robotic systems: Background and practical approaches, pages 3–27, 2015.

Zhaoyuan Gu, Junheng Li, Wenlan Shen, Wenhao Yu, Zhaoming Xie, Stephen McCrory, Xianyi Cheng, Abdulaziz Shamsah, Robert Griffin, C Karen Liu, et al. Humanoid locomotion and manipulation: Current progress and challenges in control, planning, and learning. *arXiv preprint arXiv:2501.02116*, 2025.

Huihui Guo, Fan Wu, Yunchuan Qin, Ruihui Li, Keqin Li, and Kenli Li. Recent trends in task and motion planning for robotics: A survey. *ACM Computing Surveys*, 55 (13s):1–36, 2023.

Dong Han, Beni Mulyana, Vladimir Stankovic, and Samuel Cheng. A survey on deep reinforcement learning algorithms for robotic manipulation. *Sensors*, 23(7):3762, 2023.

Vladimir I Iglovikov, Alexander Rakhlin, Alexandr A Kalinin, and Alexey A Shvets. Paediatric bone age assessment using deep convolutional neural networks. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: 4th International Workshop, DLMIA 2018, and 8th International Workshop, ML-CDS 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 20, 2018, Proceedings 4*, pages 300–308. Springer, 2018.

Leslie Pack Kaelbling and Tomás Lozano-Pérez. Integrated robot task and motion planning in the now. 2012.

Sertac Karaman, Matthew R Walter, Alejandro Perez, Emilio Frazzoli, and Seth Teller. Anytime motion planning using the rrt. In *2011 IEEE international conference on robotics and automation*, pages 1478–1483. IEEE, 2011.

Beomjoon Kim, Luke Shimanuki, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Representation, learning, and planning algorithms for geometric task and motion planning. *The International Journal of Robotics Research*, 41(2):210–231, 2022.

Serdar Kucuk and Zafer Bingul. *Robot kinematics: Forward and inverse kinematics*. INTECH Open Access Publisher London, UK, 2006.

Hong Liu, Dapeng Yang, Shaowei Fan, and Hegao Cai. On the development of intrinsically-actuated, multisensory dexterous robotic hands. *Robomech Journal*, 3: 1–9, 2016.

Rongrong Liu, Florent Nageotte, Philippe Zanne, Michel de Mathelin, and Birgitta Dresp-Langley. Deep reinforcement learning for the control of robotic manipulation: a focussed mini-review. *Robotics*, 10(1):22, 2021.

Kevin M. Lynch and Frank C. Park. *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, USA, 1st edition, 2017. ISBN 1107156300.

Thushara Sandakalum and Marcelo H Ang Jr. Motion planning for mobile manipulators—a systematic review. *Machines*, 10(2):97, 2022.

R Saravanan, S Ramabalan, and C Balamurugan. Evolutionary multi-criteria trajectory

modeling of industrial robots in the presence of obstacles. *Engineering Applications of Artificial Intelligence*, 22(2):329–342, 2009.

Alin Albu Schäffer, Oliver Eiberger, Markus Grebenstein, Sami Haddadin, Christain Ott, Thomas Wimböck, Sebastian Wolf, and Gerd Hirzinger. Soft robotics, from torque feedback-controlled lightweight robots to intrinsically compliant systems. *IEEE Robotics and Automation Magazine*, 15(3):20–30, 2008.

Shadow Robot Company. Shadow dexterous hand e - technical specification, 2022. URL https://www.shadowrobot.com/wp-content/uploads/2022/03/shadow_dexterous_hand_e_technical_specification.pdf. Accessed: 2025-03-19.

Shadow Robot Company. Dexterous hand series, 2025. URL https://www.shadowrobot.com/dexterous-hand-series/. Accessed: 2025-03-19.

Siddharth Srivastava, Eugene Fang, Lorenzo Riano, Rohan Chitnis, Stuart Russell, and Pieter Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 639–646. IEEE, 2014.

Shigeki Sugano and Ichiro Kato. Wabot-2: Autonomous robot with dexterous finger-arm–finger-arm coordination control in keyboard performance. In *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, volume 4, pages 90–97. IEEE, 1987.

Mehran Ghafarian Tamizi, Marjan Yaghoubi, and Homayoun Najjaran. A review of recent trend in motion planning of industrial robots. *International Journal of Intelligent Robotics and Applications*, 7(2):253–274, 2023.

Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. $dm_control$: $Software and tasks for continuous control. Software Impacts, 6 : 100022, 2020. ISSN 2665 - 9638. doi : .$ URL https://www.sciencedirect.com/science/article/pii/S2665963820300099.

Xuesu Xiao, Bo Liu, Garrett Warnell, and Peter Stone. Motion planning and control for mobile robot navigation using machine learning: a survey. *Autonomous Robots*, 46(5): 569–597, 2022.

Seongho Yeon. Playing piano with a robotic hand, 2022. URL https://github.com/seongho-yeon/playing-piano-with-a-robotic-hand.

Kevin Zakka. Mink: Python inverse kinematics based on MuJoCo, July 2024. URL https://github.com/kevinzakka/mink.

Kevin Zakka, Yuval Tassa, and MuJoCo Menagerie Contributors. MuJoCo Menagerie: A collection of high-quality simulation models for MuJoCo, 2022. URL http://github.com/google-deepmind/mujoco_menagerie.

Kevin Zakka, Philipp Wu, Laura Smith, Nimrod Gileadi, Taylor Howell, Xue Bin Peng, Sumeet Singh, Yuval Tassa, Pete Florence, Andy Zeng, and Pieter Abbeel. Robopianist:

Dexterous piano playing with deep reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2023.

Liding Zhang, Kuanqi Cai, Zewei Sun, Zhenshan Bing, Chaoqun Wang, Luis Figueredo, Sami Haddadin, and Alois Knoll. Motion planning for robotics: A review for sampling-based planners. *Biomimetic Intelligence and Robotics*, page 100207, 2025.

Appendix A

First appendix

A.1 Evaluation Framework for Future Works[1]

Finger Mapping (FM) Accuracy For each key press, the assigned finger is retrieved from `self._keys_current`. The fingertip position of the assigned finger is obtained from the simulation, and the distance to the key's position is computed. The finger mapping is correct if this distance is below the threshold. The accuracy is computed as:

$$\textbf{FM Accuracy} = \left(\frac{\text{Number of Key Presses with Correct Finger Mapping}}{\text{Total Number of Key Presses}} \right) \times 100 \tag{A.1}$$

This allows evaluation on framework's ability to manage high-dimensional control, another fundamental challenge in robotic manipulation. The Shadow Hand has 23 degrees of freedom, and coordinating the correct finger to press the assigned key requires precise control over multiple joints. Correct finger mapping also reflects the effectiveness of the motion planning and control strategies in achieve the planned trajectories.

Wrong Presses The precision of the framework in avoiding unintended interactions with the environment, a critical aspect of robotic manipulation. Assessing wrong presses points out errors in motion planning or control, such as overshooting a trajectory or colliding with adjacent keys. To do this, the indices of keys that should not be pressed are identified at each timestep. The simulation checks if any of these keys are pressed. The total number of wrong presses is accumulated over the episode.

Total Reward At each timestep, the reward is computed as the sum of the three components:

$$\text{Reward}_t = \text{Key Press Reward}_t + \text{Finger Mapping Reward}_t + \text{Energy Reward}_t \tag{A.2}$$

The total reward is the sum over all timestep:

$$\text{Total Reward} = \sum_{t=0}^{T-1} \text{Reward}, \tag{A.3}$$

[1] Zakka et al. [2023]

where T is the number of timestep. This provides a measure of the framework's performance, balancing accuracy (key press and finger mapping rewards) with efficiency (energy penalty). It aligns with the hypothesis by evaluating the framework's ability to achieve precise, high-dimensional control while minimizing energy consumption, a key challenge in robotic manipulation.

A.2 Demo Images

Figure A.1: When motion controller is implemented, the manipulator system enables more dexterous action such as moving thumb underneath the other fingers.

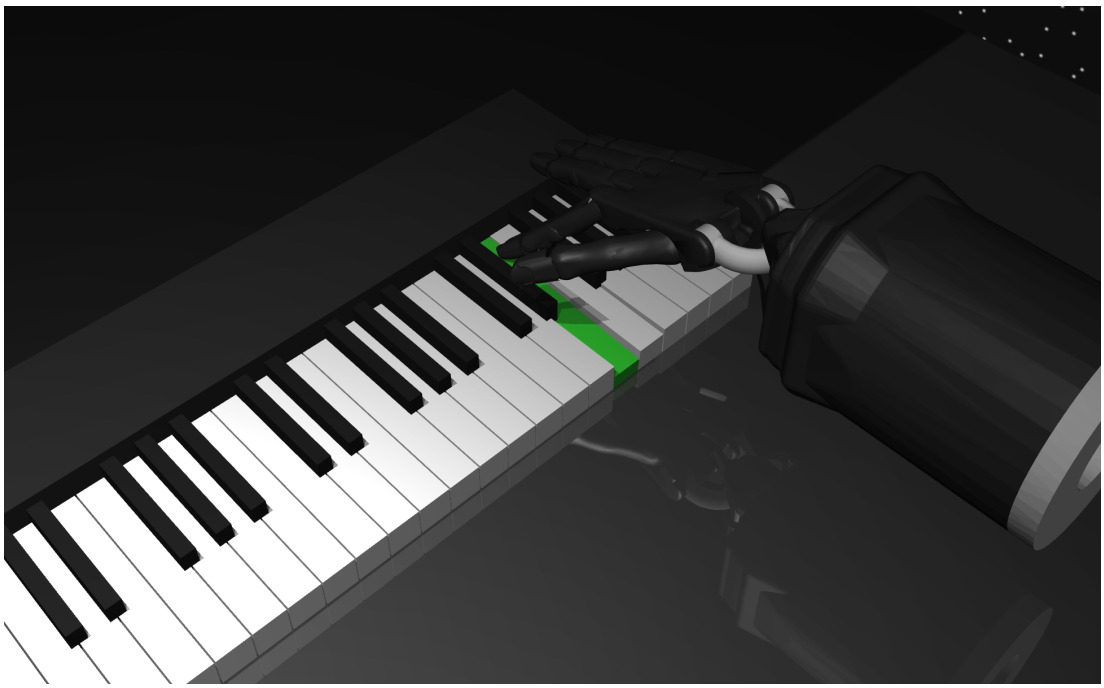


Figure A.2: The hand is pressing the target key.

Figure A.3: When energy reward spikes up due to quick transition required between actions, some infeasible joint configurations can be not applied.