Use of Slip Prediction for Learning Grasp-Stability Policies in Robotic-Grasp Simulation

by

Lukas Stracovsky

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Mechanical and Mechatronics Engineering

Waterloo, Ontario, Canada, 2023

Author's Declaration

Abstract

The purpose of prosthetic hands is to restore a portion of dexterity lost through upper limb amputation. However, a key capability of human grasping that is missing from most currently available prosthetic hands is the ability to adapt grasp forces in response to slip or disturbances without visual information. Current prosthetic hands do not have the integrated tactile sensors or control policies to support adaptive grasp stabilization or manipulation. Research on slip detection and classification has been providing a pathway towards integrating tactile sensors on robotic and prosthetic hands; however, current literature focuses on specific sensors and simple graspers. Policies that use slip prediction to adapt grasp forces are still largely unexplored.

Rigid-body simulations have recently emerged as a useful tool for training control policies due to improvements in machine learning techniques. Simulations allow large amounts of interactive data to be generated for training. However, since simulations only approximate reality, policies trained in simulation may not be transferable to physical systems. Several grasp policies with impressive dexterity have been trained in simulation and transferred successfully to physical systems. However, these grasp policies used visual data as policy inputs instead of tactile data. This research investigates if rigid-body simulations can use slip prediction as the primary input for training grasp stabilization policies.

Since current slip detection and prediction literature is based on specific tactile sensors and grasper setups, testing slip-reactive grasp policies is difficult, especially with an anthropomorphic hand. As an alternative to implementing a system-specific policy, real human grasp poses and motion-trajectories were used to test if the trained policy could replicate known human grasp stability. To acquire the human grasp data, grasp and motion trajectories from a human motion-capture dataset were adapted into a simulation. Since motion-capture only includes grasp and object pose data, grasp forces had to be inferred through a combination of analytical and iterative methods. Simulation contacts are also just approximate models; therefore, slip in the simulation was characterized for detection and prediction. The stability of the converted grasps was tested by simulating the grasp manipulation episodes with no control policy. Viable grasps were expected to maintain stability until the manipulation trajectory caused grasp degradation or loss. The initial grasps maintained stability for an average of 27.7% of the grasp episode durations, though with a wide standard deviation of 35%. The large standard deviation is due to episodes with high hand acceleration trajectories, as well as grasp objects with varying grasping difficulty.

Policy training using the imported grasps and trajectories was performed using reinforcement learning, specifically proximal-policy optimization. Policies were trained with and without slip prediction inputs, using different reward functions: a reward proportional to the duration of grasp stability, and a reward that also added a grasp-force magnitude penalty. A multi-layer perceptron was used as the policy function approximator. The policies without slip-prediction inputs did not converge, while the policy with slip inputs and the grasp-force penalty-reward function converged on a poorly performing policy. On average, episodes tested with the policy that used a grasp-force-penalty had a 0.11 s reduction in grasp stability duration compared to the initial grasp duration results. However, episodes that did have improved stability under the learned policy improved on average by 0.38 s, significantly higher than the average stability loss. Moreover, the change in stability duration under the trained policy negatively correlated with the initial stability duration (Pearson -0.69, p-value 9.79e-11). These results suggest that slip predictions contribute to learned grasp policies, and that reward shaping is critical to the grasp-stability task. Ultimately, the trained policies did not perform better than the baseline no-policy grasp stability, suggesting that the slip predictions were not sufficient to train reasonable grasp policies in simulation.

Acknowledgements

I would like to thank Professor Kofman, my supervisor, who patiently helped and guided me through my degree. I would also like to thank my co-supervisor Professor Jeon for always pointing out what I should focus on in my research. Thanks to Professor Khamesee and Professor Kwon for being committee readers and providing comments on how to improve my thesis. I am grateful to my lab-mate Nathan Ng for always letting me ask him about things I've forgotten. And finally, and most importantly, special thanks to Daniel and my family, especially Roy, who I owe immeasurably.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Human Grasp-Dexterity Reliance on Tactile Feedback

The human hand can grasp a wide range of objects with high dexterity. In addition, humans are able to stabilize and manipulate objects with little conscious attention and minimal prior knowledge of object parameters such as mass and surface conditions. The ability for high dexterity without prior knowledge of the object shape is partly due to a complex system of afferent nerves that react to spatial and temporal features across multiple types of tactile feedback [1] [2] [3]. These afferents appear to measure direct values such as force, pressure, and contact location in addition to surface features such as friction and curvature. For example, in grasping, humans perform force adjustments to exceed the minimum slip forces of grasps with a constant safety factor ranging between 10-40% [2] [4]. Since minimum slip forces are a function of object mass, shape, and surface conditions, it is likely that human tactile feedback includes cues that distinguish these features and provide force and pressure feedback for error correction. The timescales for tactile feedback can be small. Frictional grip force responses in adults are typically approximately 100 ms [2] [5] [6], with anticipatory force changes based on proprioception and predicted disturbances that can reduce the reaction time further [2] [6]. In contrast to normal grasp dexterity, human grasping with inhibited tactile feedback is characteristically clumsy and loses all the force adaptations related to surface conditions [4]. Subjects with such limitations have difficulty with detailed tasks or low-friction surfaces, often letting objects slip and overcompensating for slip thresholds with significantly more force than necessary [1] [4]. This indicates that tactile feedback and slip-force management are core components of human dexterous manipulation.

1.2 Prosthetic Hand Areas of Improvement

In 2005, it was estimated that approximately 541,000 individuals in the United States had some form of upper limb amputation [7]. Most of these (61%) are transcarpal (below the wrist), while the next largest categories are transradial (12%, below the elbow) and trans-humeral (16%, above the elbow) [7]. Each of these amputations involves a significant loss of dexterous capability, which is not fully restored by currently commercially available prostheses.

A meta-analysis of upper-limb prosthesis-user surveys lists desired features related to grasp dexterity across all prosthesis types. The broadest desired feature was the ability to perform detailed daily task actions such as cutting meat and tying shoelaces. More specific desires included providing users with sensory feedback, applying high grasp forces, controlling grasp strength, performing actions without visual attention, performing stable grasps by avoiding slippage, changing object position or orientation, improving small object precision, and moving finger independently [7]. A direct analysis of prosthesis usage in daily life through egocentric (head-mounted) video [8] revealed that users demonstrated a higher proportion of non-prehensile (non-grasping) manipulations with their prostheses compared to their intact hand, suggesting that grasping with the prosthetic hand was often too limited to be used. A specific observed behavior was the use of the prosthetic hand for holding unnecessary objects, while the intact hand performed manipulations [8], indicating that the prosthesis was only trusted to handle static grasps. The directly observed behavior combined with the broader need priorities indicated by prosthesis users demonstrates that current prostheses do not have sufficient dexterity and control. These areas of improvement could be addressed in part, though tactile-sensor integration. However, integration of tactile sensors and tactile control-loops into prosthetic hands has so far been limited. A review of prosthetic-hand control [1] found that of the 12 reviewed prosthetic hands, half of them did not have any tactile sensors, and only one commercially available hand included a slip recognition mechanism.

1.3 Need for Automation in Tactile Integration in Prosthetic Hands

A prosthetic hand with integrated tactile sensors contains a control loop. Tactile sensors measure information such as contact location, force, vibration, or surface conditions from the grasp, and finger forces and position adjustments act based on the sensor information to maintain the grasp. Force and position adjustments could be directly made by the user, but current direct control methods are limited to a small number of control channels. Body-powered mechanical prostheses map hand manipulations onto shoulder movements [7] or other body flexions; however, the number of available mechanical mappings for other parts of the body fundamentally limit the degrees of freedom (DOF) of these mechanical prostheses. Myoelectric devices sense muscle or neural activation using electrodes, ultrasound, or other interfaces [7]. Most commercially available myoelectric devices are single channel, single direction devices, typically with on/off trigger control [1]. While there is extensive research on multi-electrode patterns for multi-channel control, such setups still have consistency issues due to posture or muscle shift, fatigue, sensor crosstalk, and sweat [1] [7]. The small number of channels available via direct control methods limits the degree of grasp dexterity when directly controlling a prosthetic hand.

Automating portions of the grasp process could allow increased prosthetic hand capability, while reducing or eliminating the need for direct control channels. So far, automation has seen success with user task intent decoding and grasp selection for prosthetic hands. Such methods used for both intent decoding and grasp selection included visual identification and classification of objects using convolutional neural networks [9], and myoelectric signal feedback for error correction or action confirmation [10] [11]. Although grasp selection and grasp approach have been demonstrated, these only cover the initial stages of the grasping process. By contrast, grasp stability automation for prosthetic hands has seen less development. Grasp stability is highly sensitive to the amount of observable environmental information. In controlled environments, where there is accurate information from sensors in the environment and prior information about the grasped objects, grasp stability is a solved problem that has been demonstrated in industrial robotics [5]. Prosthetic hands operate in a much more restricted environment where all sensors must be placed on the grasper itself, and prior object information is not available. Methods used in controlled environments [12] are thus not possible in prosthetics, requiring the development of new control policies tailored for this specific application and environment.

1.4 Learning Grasp-Stability Policies in Simulation

A specific method of learning grasp-stability policies that has shown great promise is machine learning in simulation. Both simulation environments and machine learning models have improved drastically in recent years, allowing sophisticated learned models that have demonstrated capable grasp-policies that transfer to real-world setups [13] [14] [15]. These models benefited from an end-to-end training pipeline that iterates over simulated states much faster than a physical setup, with flexibility over the physical parameters, such as surface friction or softness, and direct access to environmental information such as contact location, forces, or relative surface motion.

The ability to adjust the observable information of the environment that is fed into the grasp policy is crucial. Current literature on tactile grasp stability exhibits a high degree of variance in type of tactile information used. Many real-world studies use contact, force, and pressure information to improve grasp stability and correction [16] [17] or perform grasp readjustment [18] [19]. In addition, a growing number of research studies focus on classifying slip or overall grasp stability from specific tactile sensors [17] [20] - [24] for use with grasp stability or manipulation. However, only a few studies handle grasp-force adjustment in relation to slip management [14]. Most slip research is performed on simple graspers such as parallel-jaw grippers [16][17][21][22] or 3-jaw grippers [20], preempting the development of grasp management policies. Most grasp-policy simulation studies on anthropomorphic hands still use visual information or contact force information as inputs to their grasp policies [13] [25] - [29], which does not reflect the kinds of tactile information used in real-world tactile grasp studies.

The primary drawback of simulation is that it is only an approximation of reality, making simulation-to-real transfer and vice-versa not guaranteed. A critical part of any robotic simulation is the methods used to ensure that a policy trained in simulation generalizes in a way that includes real-world environments and deals with simulation simplifications. Some grasp simulations start from synthetic grasps or manually selected neutral starting points (such as an object cupped in a hand) and use methods during training such as domain randomization to prompt learned policies to generalize enough for transfer to real grasps [13] [14] [25]. An alternate method for reducing simulation divergence might be to incorporate real grasps into the simulation from the start by training on actual grasps that have been transferred into the simulation [26] [27]. The ability of the simulation and learned policy to stabilize known viable grasps would serve as a measure of how well rigid-body simulations can use slip prediction to train grasp stabilization policies.

1.5 Motivation and Research Goals

In summary, tactile feedback is critical to stabilizing grasps and manipulating objects; however, tactile sensors are not currently integrated in prosthetic hands. It may be possible to extend grasp robustness and capability by automating parts of the grasping process such as grasp stabilization using tactile data. Simulations have proved to be viable environments to train grasp policies with good stabilization and manipulation capabilities that are transferable to real grasps. However, many of these simulations start from synthetic or manually chosen grasps, and then rely on visual information, which is not available on a prosthetic hand. Some simulations use force and pressure data as inputs, while many real-world grasp studies focus on slip classification and prediction for tactile feedback. This thesis research creates an environment in simulation for grasp-stabilization policy-learning that uses real human grasps and manipulation trajectories and provides slip prediction in simulation as potential policy input. Finally, a simple machine-learning structure is used in the environment to attempt to learn a grasp stabilization policy.

The objectives of this research thesis are to:

1. Create an environment in a rigid-body simulation that incorporates real human grasps and manipulation trajectories.
    a. Convert human motion-captured grasp poses with objects, into simulation.
    b. Adjust grasp states to mimic inferred forces and initial states from motion-capture dynamics.
    c. Recreate grasp hand-trajectories found in motion-captured data.
2. Characterize contact slip in rigid-body simulation to replicate slip prediction for simulated grasps.
3. Determine through the created grasp environment, if a simple multilayer-perceptron network can replicate real grasp stability using slip prediction inputs.

Chapter 2 Background

2.1 Modeling Contacts and Grasps

Mathematically modeling contacts and grasps is a fundamental step in describing and predicting contact and grasp behavior. Although mathematical representations do not capture all real contact qualities, they form the basis of simulator contact models and can be used to approximate the required forces for a grasp.

A common way to quantify contacts in a grasp is to consider all contacts between the grasped object and hand as points with individual contact forces exerted along all cartesian and torque axes (Eq. 2.1) [30] [31]. In this convention the *z*-axis is defined as the axis colinear with the surface normal at the point of contact, pointed inwards towards the grasped object.

$$F_{c_i} = [F_x \quad F_y \quad F_z \quad \tau_x \quad \tau_y \quad \tau_z]^T \tag{2.1}$$

where F_{c_i} is the contact wrench, the vector containing all forces and torques exerted by the i'th contact in the local reference frame. F_x, F_y are the cartesian forces of the contact that are tangential to the surface normal, F_z is the normal force, and τ_x, τ_y, τ_z are the torques about each cartesian axis.

In the idealized case of frictionless contact, the contact wrench F_{c_i}, could be trivially expressed as the normal force f_{c_i} mapped from a scalar onto the wrench vector. A matrix created from the three force and three torque unit components using a simple vector of ones and zeros (the wrench basis) performs the mapping. The basis in Eq. 2.2 only includes contacts that exert pure normal force without any tangential or torsional forces.

$$F_{c_i} = [0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0]^T f_{c_i}, \tag{2.2}$$

where f_{c_i} is the contact normal force subject to: $f_{c_i} \geq 0$.

A more complex soft-finger frictional contact representation incorporates the coulomb friction model $|F_{friction}| \leq \mu F_{normal}$ to represent a contact with tangential friction as well as resistive torque about the contact normal. The friction model defines the maximum potential resistive contact forces as less than the multiple of the normal force with a constant friction coefficient. The tangential and torsional friction forces use independent friction coefficients μ and γ. Under this contact representation the contact wrench F_{c_i} can be expressed as:

$$F_{c_i} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} f_{c_i},\ f_{c_i} \in K, \tag{2.3}$$

where $K = \left\{ f \in \mathbb{R}^4 : \sqrt{f_1^2 + f_2^2} \leq u f_3,\ f_3 \geq 0, |f_4| \leq \gamma f_3 \right\}$ represents the friction cone constraints created by applying the coulomb friction model to each force and torque component. f_{c_i} is the force vector that includes the tangential forces f_1 and f_2, the z-axis torque f_4, and the normal force f_3 [30] [31].

Since friction forces are only resistive, K defines a set of possible contact force vectors enclosed in a cone, whose bounds are derived from the coulomb model (Fig. 2.1 blue cone). This friction cone's aspect angle is governed by the tangential and torsional friction coefficients, μ and γ, respectively.

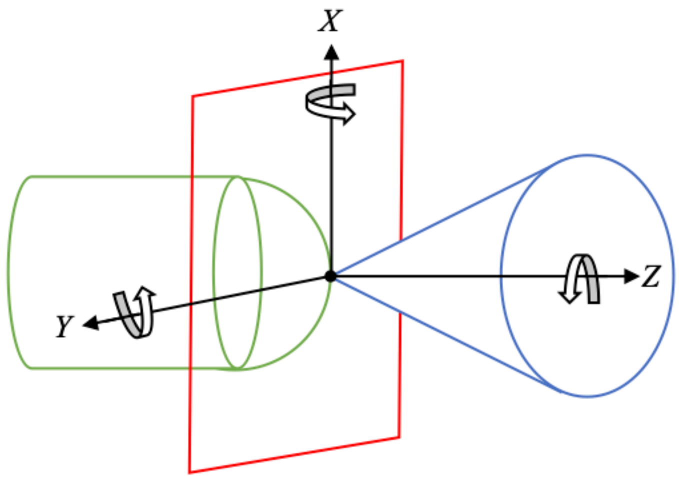


Figure 2.1. Illustration of the point contact model showing contact between a hemispherical endpoint (green) and a flat plane (red). The local contact axes are drawn extending from the contact point along with the friction cone (blue).

A more general form of the friction cone constraints for n arbitrary friction constraints and constraint forces $\{f_1, f_2, \ldots, f_n\}$, can be defined as follows (note that f_1 is now defined as normal force for convenience) [30]:

$$K = \left\{ f \in \mathbb{R}^n : f_1 \geq 0, f_1^2 \geq \sum_{i=2}^{n} \frac{f_i^2}{\mu_{i-1}^2} \right\} \tag{2.4}$$

Given a set of contacts, the sum of all contact wrenches F_{c_i} transformed by some rotational and translational matrices T_i to a common coordinate system, provides the net forces and torques for some object F_o. For simplicity the transformation matrices can be combined with the wrench-basis matrices into a single matrix G. G can be used with the original force vectors f_{c_i} to represent the sum of contact wrenches.

$$f_c = [f_{c_1} \quad f_{c_2} \quad \cdots \quad f_{c_i}]^T \tag{2.5}$$

$$F_o = \sum T_i F_{c_i} = G f_c \tag{2.6}$$

A special case that can be considered for this sum of forces and torques is force-closure, where contact forces exist to allow the net object wrench F_o to have any value. In practical terms, this means that the contact configuration allows any arbitrary resistive or manipulative force and torque to be generated for the object [30] [31]. From the simplified form: $F_o = G f_c$, it follows that if the columns of G are linearly independent (invertable), then the contact configuration is force-closure. If some column of G is not linearly independent, the grasp will not be able to produce or resist any arbitrary forces along that axis.

Given the above definition of contacts and the properties of force-closure, the matrix G can be used in methods of quantifying grasp quality [31]. Some simple metrics of grasp quality include the minimum singular value of G, which indicates how close G is to becoming singular, and the ratio between the minimum and maximum singular values. A more common metric of grasp quality is the radius of the origin-centred hypersphere inscribed within the convex hull formed by the union of all the contact friction cones. In practical terms, the sphere radius would represent the minimum wrench (force and torque) that would be able to exceed the resistive forces of the grasp. However, although these grasp metrics appear reasonable, they are limited in the factors they consider. Evaluations focusing on pose error robustness [32] have shown that grasp metrics can be vulnerable to pose error, and human ratings of grasps [33] [34] do not necessarily correspond to grasp metrics. While any of the above grasp metrics may be useful for conceptualizing grasps in terms of a single quality value, they are less useful in mimicking human capability. On the other hand, the simple contact model is still relevant since it is a common starting point for contact models in simulators and serves as an analytical starting point for inferring grasp forces.

2.2 Mujoco Physics Simulator

Physics simulators are powerful tools for building, training, and testing control methods, and have already proven useful for learning grasp stabilization or manipulation policies [13] [14]. The ability to iterate and predict physical outcomes without material restrictions is a great advantage that allows algorithm development methods such as reinforcement learning to be viable. Mujoco [35] and Pybullet [36] are both premier open-source physics simulators, with multiple modules incorporated in popular reinforcement learning toolkits such as OpenAI Gym [37]. Mujoco specifically is a prime candidate for robot control tasks. According to a 2015 comparison of simulation tools, Mujoco had the best speed-accuracy performance for robot simulation tasks and was able to complete a simulated grasp task with the largest timestep, indicating high stability for grasp tasks [38]. A 2021 comparison had a more mixed conclusion, but noted that Mujoco had high customizability for simulation parameters, and had a lower barrier of entry for reinforcement learning problems [39]. In addition, Mujoco is able to demonstrate several emergent physical behaviors such as the Newton's cradle inertia transfer, the Dzhanibekov effect, and "tippe top" flipping [40]. Finally, learned policies for grasp manipulation trained on synthetic data from Mujoco have been shown to be able to transfer to real physical systems [13] [15], albeit with domain randomization to train a more generalized policy [13]. Physics simulations depend on contact models to accurately approximate reality; thus, contact models limit the conditions where the simulation is accurate. Understanding the simulation contact model is necessary for imitating real-world conditions, especially for importing real-world grasps. Although an explanation of Mujoco's algorithms is outside the scope of this work, an overview of its general process and key characteristics are provided in the next subsections. Further details can be found in the Mujoco documentation [41] and in [42] by the original developer.

2.2.1 Mujoco Simulation Framework

A physics simulation in Mujoco begins with the compilation of a predefined model file into an initial state that can be quantified in terms of relevant system matrices. The simulation then steps forward in time from the initial state using the general equation of motion.

$$M\dot{v} + c = \tau + J^T f \, , \tag{2.7}$$

where M is the inertial matrix in joint space, v are the velocities along all degrees of freedom (DOF), $\dot{v}$ are the DOF accelerations, c represents the system bias force (gravity, Coriolis, centrifugal), τ are the applied forces (externally applied or through actuators), J is the Jacobian for any constraints (which maps the constraints to the DOF) and f are the constraint forces.

Note that instead of using cartesian coordinates, Mujoco operates using generalized joint coordinates, where the axes are defined by the joint degrees of freedom. This ensures that joint constraints are implicit in the coordinate system and thus not violated. This strategy works well for large kinematic hierarchies such as robotic arms. However, this also means that closed-loop kinematics, for example, for a four-bar linkage, cannot be defined without at least one constraint that is not implicit. It can also be observed from the general equation of motion (Eq. 2.7), that the Mujoco state vector only requires time, position, and velocity as internal time-dependent parameters. To compute all unknowns for time integration, the inertial matrix is derived using the Composite-Rigid-Body algorithm described in [43], and the bias forces are calculated using the Recursive Newton-Euler algorithm.

Mujoco solves both the forward and inverse dynamics of the motion equation (rearrangements of Eq. 2.7). The forward dynamics (Eq. 2.8) solve for accelerations, while the inverse dynamics (Eq. 2.9) recover forces given position, velocity, and acceleration:

$$\dot{v} = M^{-1}(\tau + J^T f - c) \tag{2.8}$$

$$\tau = M\dot{v} + c - J^T f \tag{2.9}$$

The inverse dynamics equation (Eq. 2.9) serves as an accuracy check since forward dynamics includes a numerically-solved constraint-force optimization problem. Acceleration solved for in forward dynamics is used to step through simulation time and perform forward kinematic updates through a numerical integrator. In Mujoco, several options available are: Semi-implicit Euler, Implicit-in-Velocity Euler, and 4th-order Runge-Kutta. Specific details on how each of these integrators perform time updates and some stability tradeoffs can be found in the Mujoco documentation [41].

2.2.2 Mujoco Contact Model

A critical part of how Mujoco operates is its constraint-contact model. Without the constraint force, the forward dynamics equation is incomplete. Mujoco handles contacts and general constraints in a similar manner; however, this subsection will focus on how contacts are handled, given that most non-implicit constraints encountered in this research are contacts.

A key preface of the Mujoco contact-collision model is that contacts are considered Coulomb-friction point contacts (Sec. 2.1). This means that for any two contacting geometries, the contact between them will be represented by a single point. This does allow contacts and their constraints to be described with friction cones; however, unrealistic contacts with the wrong geometry can be produced. Managing the expected contact geometries to mitigate the problem of unrealistic contacts must be considered in the design of a Mujoco environment.

A core theme of Mujoco's constraint model is the relationship between the constrained and unconstrained behavior of a physical system. Gauss's principle on constrained dynamics [44] states that a constrained system's accelerations will be as close as possible to the system's unconstrained (free) accelerations with the minimal allowable constraints.

$$a_{constrained} = \underset{a \in constraint}{\operatorname{argmin}} \left\| a - a_{free} \right\|_M^2 \quad , \tag{2.10}$$

where a is the system accelerations (constrained or free), M is the mass matrix, and double bar operator is defined as: $\|a\|_M^2 = a^T M a$.

In terms of the previous equation of motion notation (Eq. 2.7) and x as the acceleration variable:

$$\dot{v} = \underset{Jx=a^*}{\operatorname{argmin}} \|x - M^{-1}\tau\|_M^2 \, , \tag{2.11}$$

where the unconstrained acceleration is rearranged from the unconstrained dynamics $M\dot{v} = \tau$, and is subject to the acceleration constraint: $a^* = J\dot{v}$.

The problem above can be considered as a solution for a system with "hard" constraints that can be solved via optimization, though it is not invertible. Mujoco "softens" the constraints by adding a regularization term R such that $R \to 0$ approaches the original "hard" problem. In addition, the reference acceleration a^* is moved from the constraint into the optimization formulation along with a slack variable y.

$$(\dot{v}, \dot{w}) = \underset{(x,y)}{\operatorname{argmin}} \|x - M^{-1}(\tau - c)\|_M^2 + \|y - a^*\|_{R^{-1}}^{Huber(\eta)} \, , \tag{2.12}$$

where $\dot{w}$ is the velocity of constraint deformations, minimization is subject to the constraint $Jx - y \in K^*$ for contacts, K^* is the acceleration dual of the friction-cone constraints from Eq. 2.4, and the Huber function has an output that transitions between a quadratic at zero to a linear function at a predetermined threshold η. The regularization term R and reference acceleration term a^* are both parameter-dependent values encapsulated by several contact parameters tuned by the user.

Although these values do appear in the primary optimization problem equation, they are best described in the context of the Lagrange dual of the optimization formulation.

For the forward dynamics, the Lagrange dual is:

$$f = \underset{\lambda \in \Omega}{\operatorname{argmin}} \frac{1}{2} \lambda^T (A + R) \lambda + \lambda^T (a^0 - a^*) \ . \tag{2.13}$$

and for the inverse dynamics:

$$f = \underset{\lambda \in \Omega}{\operatorname{argmin}} \frac{1}{2} \lambda^T R \lambda + \lambda^T (a^1 - a^*) , \tag{2.14}$$

where Ω is the force constraint space for contacts in the friction cone $\lambda \in K$, R is the regularization term, and A is the inverse inertial matrix in joint space $A = JM^{-1}J^T$ calculated with the mass matrix M and Jacobian J. The unconstrained and constrained accelerations a^0 and a^1 are rearranged from the equations of motion:

$$a^0 = JM^{-1}(\tau - c) + \dot{J}v \tag{2.15}$$

$$a^1 = J\dot{v} + \dot{J}v , \tag{2.16}$$

where the Jacobian J and mass matrix M define the contribution of the bias forces c, actuated forces τ, and velocities v.

Given the unconstrained minimizer for the dual optimization problem:

$$f^+ = (A + R)^{-1}(a^* - a^0) \ , \tag{2.17}$$

a rearrangement of the equations of motion with these definitions results in:

$$a^1 = A(A + R)^{-1}a^* + R(A + R)^{-1}a^0 \ . \tag{2.18}$$

Once again, just like in the primal optimization form, in Eq. 2.18, $R \to 0$ results in a completely hard contact. On the other hand, it is clear that R interpolates the constrained acceleration a^1 between the reference acceleration a^* when $R \to 0$ and the unconstrained acceleration a^0 when $R \to \infty$. In order words, R allows the interpolation between strong and weak constraints. Both R and A can be redefined as dependent on a single parameter d, which has a scalar value for each constraint. These definitions allow constraint and acceleration interpolation to be controlled by a user-selected parameter that is bounded from $0 \to 1$.

$$R_{ii} = \frac{1 - d_i}{d_i} \hat{A}_{ii} \tag{2.19}$$

$$a_i^1 = d_i a_i^* + (1 - d_i) a_i^0 \ , \tag{2.20}$$

where R_{ii} and $\hat{A}_{ii}$ are the diagonals of the interpolation and inverse inertial matrices, and the diagonal of the A matrix is approximated from the model configuration.

As the approximation of A approaches the actual value, the constrained acceleration turns into an interpolation based on the value of d. For modeling soft constraints or contacts, d itself is defined as dependent on a constraint position value r. In the context of contact, this would correspond to the geometric interference between the two geometries in contact. Mujoco user parameters allow the definition of a position-dependent $d(r)$ based on a shifted or reflected sigmoid function, and these govern the impedance of a simulated soft material.

The reference acceleration parameter a^* is defined as a spring and damper model in the constraint space:

$$a_i^* = -b_i(Jv)_i - k_i r_i \, , \tag{2.21}$$

where k is the spring coefficient, and b is the damping coefficient for the i'th constraint.

For contact, dimensions such as the contact normal include both spring-damper terms, but for frictional dimensions, the position value r is simply zero, leaving just a damping component. Combining the impedance interpolation function with the reference acceleration definition introduces a scaling factor on the spring-damper model. In order to prevent this from impacting the characteristics of the model, b and k can be redefined in terms of a time constant T and damping ratio R, respectively:

$$b = \frac{1}{2}(d_{max}T) \ , \tag{2.22}$$

$$k = \frac{d(r)}{d_{max}^2 T^2 R^2} \ . \tag{2.23}$$

With these definitions, the damping ratio remains constant, while the time constant increases as impedance decreases.

2.2.3 Mujoco Key Characteristics

In summary, the Mujoco physics simulator has several characteristics that are key to understanding the behavior of a simulated kinematic chain with contact constraints. Joints in a kinematic chain are implicit and thus never violated. Kinematic loops will always have at least one constraint that is "soft". The simulation state is only dependent on position, velocity, and time (and actuator activations if included). Contacts are represented by coulomb-friction point contacts and forces are constrained within the friction cones generated by the Coulomb-friction ratios. Constraint forces and accelerations are solved via optimization with a tunable parameter d that governs the strength of constraints and can vary with a position value to model soft contact. Finally, the strong bounds of constraint accelerations are defined via a spring-damper model with tunable damping ratio and time constant values.

2.3 Machine Learning

Machine learning is a set of automated methods of deriving inferences from data, usually using a function approximation model [45] [46]. The function approximator is shaped by exposure to (training) data and evaluated on separate (test) data. A specific type of function approximator that has become highly popular since breakthroughs around 2015 is neural networks [45]. Neural networks have demonstrated the ability to generalize for high-dimensional problems even without expert knowledge [46]. However, the ability of a neural network to approximate complex functions also allows neural networks to overfit training data. Finding the right balance of approximator complexity matched to the training data richness is a major challenge not only for neural networks but machine learning in general. Most of the time, more data and with variance in features relevant to the target function helps neural networks generalize.

Machine learning and neural networks have been successfully applied many times to deriving robotic grasp control policies [13]-[21],[24]-[29],[47]-[49]. The distinctions between each application of machine learning for grasp control are the chosen training environment and the input data provided to the machine learning algorithm.

2.3.1 Training Environment

Both real-world and simulated data are used to train grasp control policies. Real-world training data has the advantage of being sourced from the same environment a trained grasp policy is expected to operate in. However, real-world data is difficult to gather, as it involves creating grasps in a physical setup. The difficulty of creating enough physical grasps with variance in grasper, object, and grasp type often leads to trained policies that are platform specific and overfit on undesirable characteristics such as object or grasp orientation [20] [21] [24].

Synthetic training data created by simulations offers a solution to generating large amounts of varied training data. Simulations can be automatically created and adjusted, and often run faster than real-time. However, the issue with synthetic data is that it is only an approximation of reality. Grasp policies trained in simulation often mix in real-world data or inputs to try to prevent the final policy from overfitting on the unrealistic simplifications of the simulation. A common process is to first train a policy on synthetic data, then perform final tuning with real-world data [13]. The drawback is that a partially viable grasp policy must be trained in simulation first. Another popular method of bridging synthetic and real-world data specifically for grasp tasks is to use human grasps as demonstrations. The demonstrations can range from processed video of grasp tasks [29] to interactive episodes measured using proprioceptive gloves and VR equipment [26] [27] [28]. In almost all cases, human grasp demonstration increases the robustness of the final trained policy [26].

2.3.2 Policy Input Data

Policy input data is the other factor that distinguishes each instance of grasp control with machine learning in the literature. From the perspective of this research, the most important policy input distinction is between policies that incorporate external object information, or only use tactile information. External object information includes any data not collected from tactile sensors on the grasper, such as visual information or prior object characteristics. Policies that incorporate external object information have demonstrated impressive grasp control capability, but requires external camera views [13] [15] [29], or directly measured object pose [26]. Additional research focuses on fusing large amounts of tactile, visual, and depth information [16] [17] [21] [47] for use in grasp stabilization. Policies trained with only grasper-situated tactile information are generally more limited in scope and focus on predicting or classifying grasp stability through slip detection or prediction [20] [21] [24] [48]. The type of input a policy is trained with often aligns with the type of data the policy is being trained with, real or synthetic. Many of the most capable grasp policies using external object information are at least partially trained in simulation with synthetic data and take advantage of the large breadth of generated data. Most policies trained with only tactile information are trained with real-world data, since contact models in simulations are not as detailed as in real-world contacts. As stated in the third research objective, this research aims to bridge this divide by replicating slip predictions in simulation and then using that input to train a grasp-stabilization policy.

2.3.3 Reinforcement Learning

A particular class of machine learning methods that aligns well with interactive learning in simulation is reinforcement learning. Reinforcement learning generally centres around learning from states and action sequences to find an optimal policy that maximizes a reward function [45] [46] for all actions taken in the environment. Deep learning, where deep neural networks are used for policy approximation, has been a relatively recent innovation that has allowed reinforcement learning to scale to very high-dimensional problems, such as natural language processing, computer games, and more recently, robotics [45] [46]. The primary feature of reinforcement learning is that the agent learns through environmental interaction, which requires significant trial and error in the process. However, the environmental dynamics do not need to be known or characterized if sufficient interaction data is available. Also, rewards can be sparse, for example, only being defined at the end of a specific series of actions and not at every state.

2.3.3.1 Reinforcement Learning Definitions

Most reinforcement learning setups can be split into several definitions. A policy that produces actions given a state, a reward generated by the current state, action, or state transition, a cumulative reward score, and a policy-dependent value that estimates the expected cumulative rewards given the current state. An additional useful augmentation to the value function is the quality function, which performs value evaluation given some initial policy action and serves as an estimate for the quality of some action given the current policy. Finally, the difference between the value and quality functions can be used to infer the advantage of some action over the current policy.

For a policy π that maps states s to actions:

$$\pi(s) \rightarrow a\,, \tag{2.24}$$

and an immediate environmental reward r_t and cumulative reward R_t:

$$r_t = r(s_t, s_{t+1}, a) \tag{2.25}$$

$$R_t = r_{t+1} + r_{t+2} + \cdots. \tag{2.26}$$

The value V^π, quality Q^π, and advantage A^π for the policy π are defined as:

$$V^\pi(s) = \mathbb{E}(R_t|s) \tag{2.27}$$

$$Q^\pi(s,a) = \mathbb{E}(R_t|s,a) \tag{2.28}$$

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)\,. \tag{2.29}$$

The quality function conveniently provides a definition for the optimal policy:

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}\, Q^\pi(s,a)\ . \tag{2.30}$$

Note that there can be many variations. For example, a policy might be stochastic rather than deterministic and thus would be defined as $\pi(s,a)$, or the cumulative reward can be calculated with a time horizon, decay factor, or numerous other discount functions. These common definitions of reinforcement learning systems illustrate some of the core components of reinforcement learning methods: learning the optimal policy, while also determining the value or quality function.

Reinforcement learning is often initially described as a Markov decision process [45] [46], where an agent with an action policy acts on a series of environmental states that encapsulate all information required to move to the next state. With this property, it is possible to summarize the expected environmental reward of all policy actions recursively into a value function dependent on the expected cumulative reward R_t.

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k} = r_{t+1} + \gamma R_{t+1}\,. \tag{2.31}$$

With this definition of cumulative reward, the value and quality functions can be redefined:

$$V^\pi(s) = \mathbb{E}(R_t|s) = \mathbb{E}(r_t|s) + \gamma\mathbb{E}(R_{t+1}|s') = \mathbb{E}(r_t|s) + \gamma V^\pi(s') \tag{2.32}$$

$$Q^\pi(s,a) = \mathbb{E}(R_t|s,a) = \mathbb{E}(r_t|s,a) + \gamma\mathbb{E}(R_{t+1}|s') = \mathbb{E}(r_t|s,a) + \gamma V^\pi(s')\,. \tag{2.33}$$

The Markov property allows the quality and value functions to depend on their own values given the successor state s'. A decay parameter γ has also been added to prevent the cumulative sum from increasing indefinitely. The recursive nature of the value and quality functions means that the problem of optimizing the policy and quality can be broken into simpler step-by-step convergence processes. The caveat is that not all reinforcement learning problems can be considered Markov decision processes, often because environments are not entirely observable.

2.4 Summary

In summary, contact representation, simulation contact model, and machine-learning characteristics are all key background factors to this research. Point contact models represent the most basic but generalizable form of contact representation and open the door for limited analytical grasp solutions as well as more complex contact models. The Mujoco contact model implements soft contact on top of the point contact representation, but relies on tuned parameters such as the spring-damper parameters and the contact-hardness interpolation value. Finally, machine learning, specifically reinforcement learning, is the policy training method of choice for interactive control applications such as grasp control, but literature results are split between real-world and synthetic data as well as policy inputs such as external object information.

Chapter 3 Methods

3.1 Overview

Creation of a policy learning pipeline using human grasps in simulation involves two steps: 1) importing human grasps from a motion-capture dataset into simulation, and 2) a training loop. Dataset conversion includes all the preparatory steps to transfer human grasps in a dataset into some viable representation in simulation, and the training loop consists of the simulation running in parallel with policy training. Each part can be further broken down into nested steps and methods, which are individually addressed in this section.

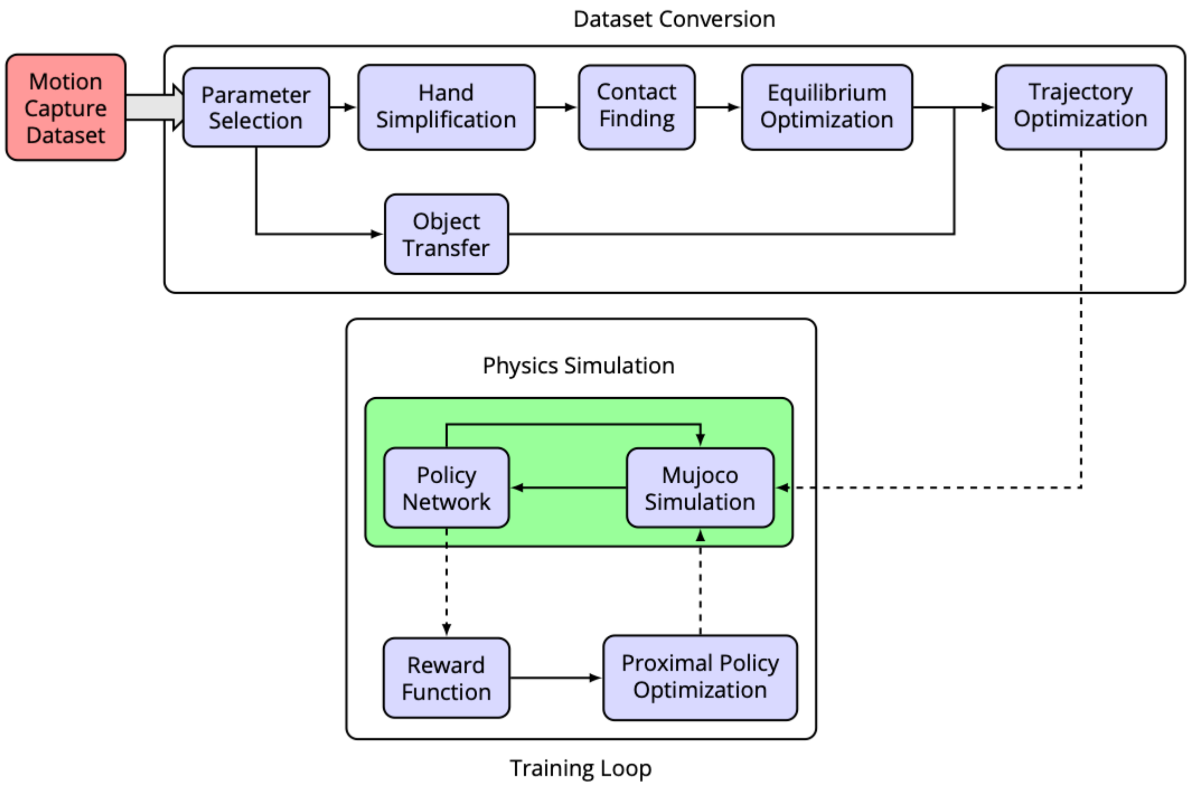


Figure 3.1. Methods organization flowchart showing the two major sections: motion-capture dataset conversion, and the simulation training loop. The green subsection encapsulates steps run within the physics simulation.

The starting point for the entire pipeline is the motion-capture dataset. In this thesis, the GRAB Dataset [49] was used. The dataset presents multiple object-grasp episodes using common hand representation models that provide the basic parameters for later hand-simplification steps. These representation models include hand and object poses, but not force information. Thus, force information must be inferred later based on pose and object dynamics after the poses are imported into simulation. Before any data conversion (required for inputting the dataset to the model), simulation parameter selection provides the first opportunity to tune simulation behavior. Real human hands incorporate contact properties that cannot be replicated completely in simulation, such as a deformable skin and large contact area. Simulation contact parameters allow contact behavior to be adjusted for more realistic grasp behavior across all hand and object poses in the dataset. After parameter selection, a simplified analog hand that retains as many of the original grasp poses as possible, is created. Grasp contacts have a low margin for error. Therefore, contact errors that already existed in the dataset or were introduced during hand-model simplification, must be removed. Next, the hand trajectories for each grasp episode are replicated in simulation to ensure that the simulated grasp is subjected to disturbances that are similar to those experienced by the real hand. Finally, the last step of dataset conversion is to infer the initial grasp forces based on the corrected hand, object, and trajectory poses. Once the grasps are replicated in simulation, the training loop is used to learn a control policy through proximal policy optimization (PPO). PPO is a common reinforcement learning method used in contemporary control simulation literature; however, this research diverges from the literature in the choice of policy network and chosen reward function.

3.2 Body and Hand Representation Models

To represent a hand in motion, a parameterized hand model, which incorporates the kinematic hierarchy, is necessary. The MANO (hand **M**odel with **A**rticulated and **N**on-rigid def**O**rmations) hand model [50] is a parameterized hand mesh model derived from high resolution three dimensional (3D) scans of 31 subject hands. The MANO parameters include shape parameters, blend parameters, and blend weights that are applied to a hand template to shape it to an individual's hand. The template also includes joint regression parameters that allow the extraction of estimated joint positions through a weighted sum of specific mesh vertices. The MANO model was originally associated with the SMPL (**S**kinned **M**ulti-**P**erson **L**inear Model) body model to create the SMPL+H combined body and hand model, until it was superseded by the SMPL-X model [51], which included expressive facial features.

3.3 GRAB Dataset

The GRAB (**Gr**asping **A**ctions with **B**odies) Dataset [52] is a series of whole-body motion-captured episodes of object interaction. What separates the GRAB Dataset from most human-motion-capture datasets is that the interaction objects have known meshes, and the object poses (position and orientations) are also tracked. Human and object 3D poses are recorded using tracking targets with known locations on a body suit and object. The body of each human subject is scanned beforehand and approximated using the SMPL-X [51] body model to account for differing body and hand shapes, generate a full body mesh from tracked points, create pose parameters, and derive joint positions. The interaction objects are 3D printed from the ContactDB [49] dataset, which has known 3D models.

Each interaction episode consists of a subject who is instructed to interact with an object with some intent ("use", "pass", "lift", and "off-hand pass"). The episodes always start with the subject in a T-pose (arms outstretched laterally) a short distance from the object, and end in approximately the same pose. Each episode is repeated across ten subjects (five women, five men). A single episode could contain multiple grasps and object interactions separated by periods of no contact depending on the episode intent. For each timestep of the episode, a SMPL-X model and object pose are constructed to represent the instantaneous pose of both the subject and interaction object.

The GRAB Dataset as a motion-captured dataset does not contain any force or contact information. Contact must therefore be assumed based on surface proximity. The threshold distance was tuned empirically by the dataset authors to 4.5 mm, but this was tuned to produce object-contact heatmaps. No actual poses were altered. The assumption of contact through proximity combined with pose errors and the use of a body-mesh instead of a true surface scan causes most instances of assumed contact to either underpenetrate with a gap between the contact surfaces, or overpenetrate with excessive overlapping geometry between the contact objects. To successfully import these grasp poses into a simulation that does model contact forces, finger and object under/overpenetration must be corrected.

3.4 Mujoco Contact Parameter Selection

Human fingers have much greater contact areas than rigid fingers due to skin deformation. Reducing the behavioral differences between the real and simulated hand contacts is primarily accomplished by adjusting the Mujoco contact parameters. These parameters are not able to correct for the differences between simulated and real contact under all circumstances, and therefore must be tuned specifically for the grasps included in the GRAB Dataset. Mujoco contacts depend on two classes of values: instantaneous contact properties and static contact parameters. Some of the instantaneous contact properties change according to the contact context and include values such as the contact position, surface normal, and contact distance. The behavior of these instantaneous values is partially governed by the static contact parameters that are chosen on simulation initialization, and are tuned to achieve a desired contact behavior. These parameters include friction coefficients, Solimp, Solref, and Condim. The next paragraphs explain the meaning of the contact values, starting with the definitions of the instantaneous contact properties and followed by how each static parameter value alters instantaneous contact property behavior.

3.4.1 Instantaneous Contact Properties

Not all contact values directly depend on the static contact parameters. Contact position and surface normal are governed by where two objects intersect to create contact. The contact surface normal is the axis perpendicular to the plane formed by overlap between surfaces, and the contact position is determined by the mid-point between the two contact surfaces along the normal axis. Although initially, contact position and surface normal are independent of the static contact parameters, the behavior of the contact, such as slip or sliding, will affect future position and normal values.

Contact distance in Mujoco is the negative value of the shortest distance along a contact normal between two intersecting surfaces, as shown in Fig. 3.2. The contact distance generally represents the geometry overlap or penetration distance along the contact normal. Contact distance is always negative, as a positive value or zero would mean that there is no geometric overlap, no contact constraint impedance, and thus no contact force. Contact distance is an artifact of Mujoco's contact model with no true corresponding value in real contacts.

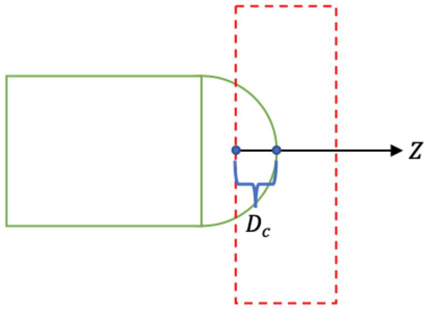


Figure 3.2. Contact distance D_c for contact between a spherical endpoint (green) and box (red).

3.4.2 Condim

Condim is a static parameter with possible integer values of 1, 3, 4, and 6. Condim corresponds to the number of local contact axes that are allowed to exert resistive friction force. As in the point contact model, discussed in Sec. 2.1, the Condim value determines what friction forces map onto the contact wrench F_c for each contact.

$$F_c = [F_x \quad F_y \quad F_z \quad \tau_x \quad \tau_y \quad \tau_z]^T , \tag{3.1}$$

where F_c is the contact wrench vector containing the XYZ forces F_x, F_y, F_z, and axis torques τ_x, τ_y, τ_z. In the local contact frame of reference, the z-axis is colinear with the contact surface normal. Table 3.1 illustrates the mapping of contact forces onto the wrench vector for Condim values.

Table 3.1. Condim values corresponding to friction forces and local contact force matrices.

Condim	**Local contact force matrix**	**Friction forces**
1	$F_c = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} f_z$	Normal Force (no friction)
3	$F_c = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix}^T$	Normal and tangential forces (tangential friction only)
4	$F_c = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_x \\ f_y \\ f_z \\ \tau_z \end{bmatrix}^T$	Normal and tangential forces, torque about contact normal (tangential and normal twist friction)
6	$F_c = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_x \\ f_y \\ f_z \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix}^T$	Normal and tangential forces, torque about all local axes (tangential friction, normal twist friction, rolling friction)

3.4.3 Friction

Contact friction is denoted by a vector of three values that represent the coulomb friction coefficients for separate types of friction: tangential sliding (μ), rolling (ω), and torsional (γ).

$$\sqrt{f_x^2 + f_y^2} \leq \mu f_z \tag{3.2}$$

$$\sqrt{\tau_x^2 + \tau_y^2} \leq \omega f_z \tag{3.3}$$

$$\tau_z \leq \gamma f_z \, . \tag{3.4}$$

Each additional friction coefficient corresponds to the local contact forces and torques added by increasing the Condim value. Altogether, the friction coefficients form the point-contact model friction-cone constraint K covered in Eq. 2.4, Sec. 2.1.

$$K = \left\{ f \in \mathbb{R}^n : f_1 \geq 0, f_1^2 \geq \sum_{i=2}^{n} \frac{f_i^2}{\mu_{i-1}^2} \right\} \tag{3.5}$$

By default, the torsional friction coefficients in Mujoco are small. However, almost all the human grasps in the GRAB Dataset were unstable without high torsional friction coefficients (in the order of 10^{-1}). By observation, this requirement is likely due to the high finger-contact areas and thus high torsional friction found in human grasps. This is most evident in human pinch grasps, which heavily rely on torsional friction for stability.

3.4.4 Solimp

The defining equation for the static Solimp parameter is one of the characteristic Mujoco contact model equations defined in Sec. 2.2.2:

$$a_i^1 = d_i(r) a_i^* + \big(1 - d_i(r)\big) a_i^0, \qquad 0 \leq d(r) \leq 1 \ , \tag{3.6}$$

where the constrained acceleration a_i^1 is defined as an interpolation by the parameter $d(r)$ between the unconstrained acceleration a_i^0 and the reference acceleration a_i^* for some contact index i. $d(r)$ is dependent on the value of the contact distance r.

The Solimp parameter defines the relationship between the interpolation value d and the contact distance r though a set of values: *dmin*, *dmax*, midpoint, and power. In simple terms, the Solimp parameters determine how the contact constraint impedance increases as the penetration and contact distance between the two objects increases. The specific curve is generated as a spline from the Solimp values. *dmin* and *dmax* determine the maximum impedance values, power determines the degree of the spline function, and midpoint determines the inflection point.

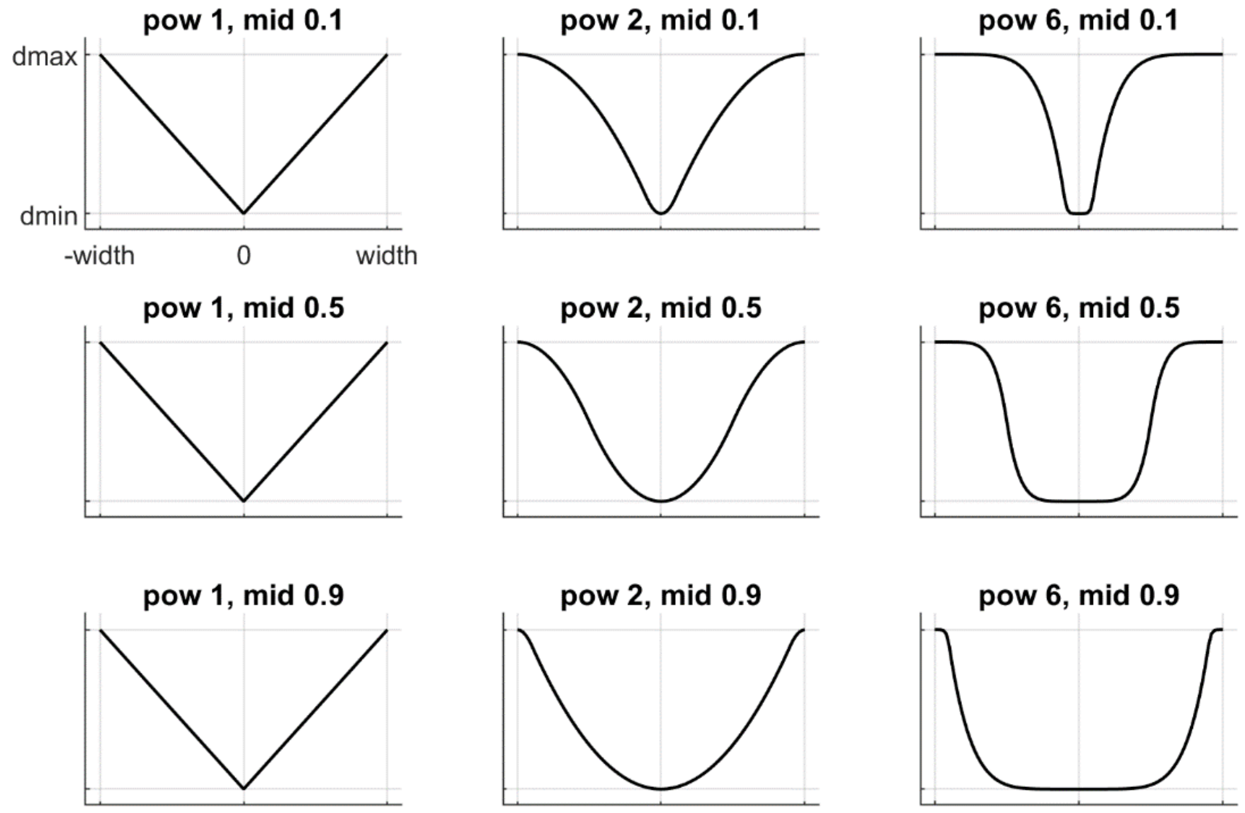


Figure 3.3. Graphs of the value of impedance d vs. contact distance r for different midpoint and power values [41]. The impedance graphs are symmetric and centre about $r = 0$ to accommodate constraints with positive distance values such as weld constraints.

The dynamic nature of contact impedance results in contact distances that move towards some steady state, which can result in a loss of contact. Contacts that are unstable in this way often have contact distances that oscillate until contact is completely lost, or some stable value is found. This oscillation causes slip and sliding in Mujoco to be characterized as contact losses (contact-distance increases above zero) of high frequency because the contact distance temporarily exceeds zero. Fig. 3.4 illustrates the difference in behavior between an unstable contact undergoing slip (3.4a), and a stable contact that has initial contact distance oscillation but settles on a stable distance value (3.4b).

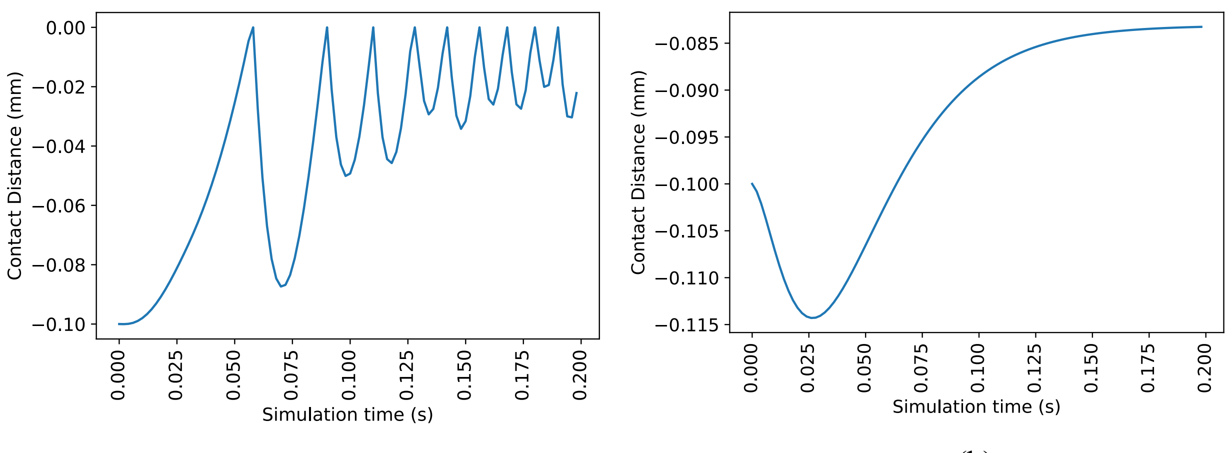


Figure 3.4. Contact distance over simulation time showing contact distance settling for a friction contact with a constant normal force: (a) contact distance over simulation time, where the normal force is sufficient to stabilize the contact, and (b) contact distance with a normal force not large enough to maintain contact through friction, resulting in sliding and contact distance oscillation.

3.4.5 Solref

The Solref parameter is also defined in the characteristic Mujoco contact equations from Sec. 2.2.2:

$$b = \frac{1}{2}(d_{max}T) \ , \tag{3.7}$$

$$k = \frac{d(r)}{d_{max}^2 T^2 R^2} \ . \tag{3.8}$$

Solref is composed of two parameters: time-constant T and damping-ratio R, which are used to calculate the stiffness k and damping b. A characteristic of contact in Mujoco is a small but sustained slip regardless of contact force, since Mujoco constraints only approach but do not actually achieve complete hardness (when all forces within the friction cone are completely resisted). The sustained slip is characterized by a small but constant change of position. One method of reducing the magnitude of unwanted slip is to reduce the *Solref* time-constant parameter. A small time-constant value increases both the stiffness b and damping k, thus scaling the impedance contact constraint and resulting in stiffer contacts, in general.

Figs. 3.5 and 3.6 demonstrate the effect of a smaller time-constant T on contact distance and slip. The example data shown are from a sphere held via friction against a vertical wall with different constant normal force scenarios ranging from 4.1 to 100 N. The minimum normal force to resist downward slip under gravity in this context is 4.1 to 4.2 N.

Fig. 3.5 shows the contact distances between the sphere and wall over time for each constant normal force scenario given a time-constant value of 0.02 (Fig. 3.5a), and 0.005 (Fig. 3.5b). Similar to the behavior shown in Fig. 3.4, the contact distances for normal forces above the minimum force required to resist gravity (stable contacts) reach a stable negative value. The only normal force scenario (4.1 N) below the minimum force (unstable contact) has a contact distance that oscillates towards zero. The decreased time-constant value between Fig. 3.5a and Fig, 3.5b causes the contact distances for stable normal force scenarios to decrease, and the contact distance oscillation frequency for the insufficient force scenario (4.1 N) to increase.

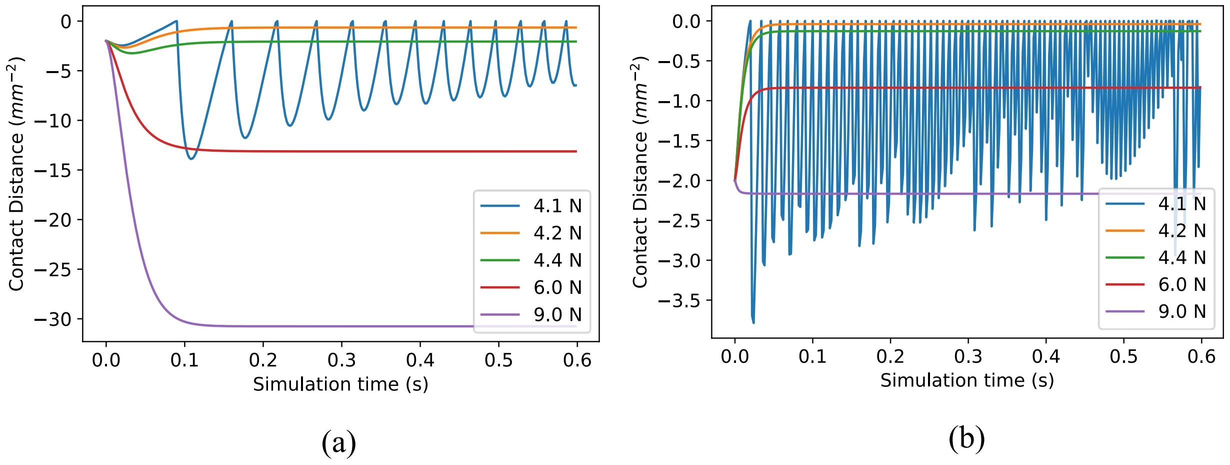


Figure 3.5. Comparison of the effect of the time-constant T on contact distance under different normal forces. A smaller time-constant of 0.005 (b) results in an increased contact distance oscillation frequency for contacts with insufficient normal forces (4.1 N) compared to a higher time-constant value of 0.02 (a). The smaller time-constant value also decreases the overall steady-state contact distances for all stable contacts.

Fig. 3.6 shows the sphere's vertical position over time for the same constant normal forces scenarios in Fig. 3.5 given the same two time-constant values of 0.02 (Fig. 3.6a), and 0.005 (Fig. 3.6b). Regardless of the normal force, a degree of constant slip always occurs because Mujoco's contact model only approaches completely "hard" contacts (Sec. 2.2.2). In this example situation, detuned simulation parameters result in an exaggerated degree of constant slip (in the order of mm/s). Forces above the minimum have close to linear constant slip while the force below the minimum (4.1 N) has downwards vertical acceleration. Decreasing the time-constant decreases the degree of constant slip, while also increasing the divergence between the constant slip experienced for stable contacts, and the slip for unstable contacts. The minimum value for the time-constant is the timestep of the simulation itself; however, to achieve simulation stability, a more realistic minimum time-constant value is several times the simulation timestep t. Ultimately a time-constant $T = 0.005$ was chosen for this research. Smaller time-constant values were observed to introduce instability, since they were too close to the simulation timestep $t = 0.001\ s$.

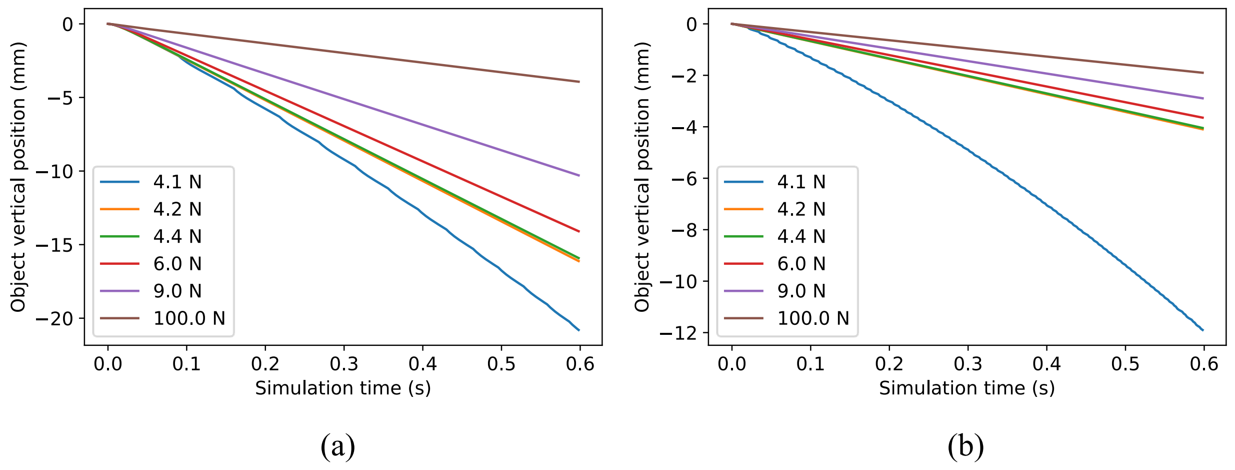
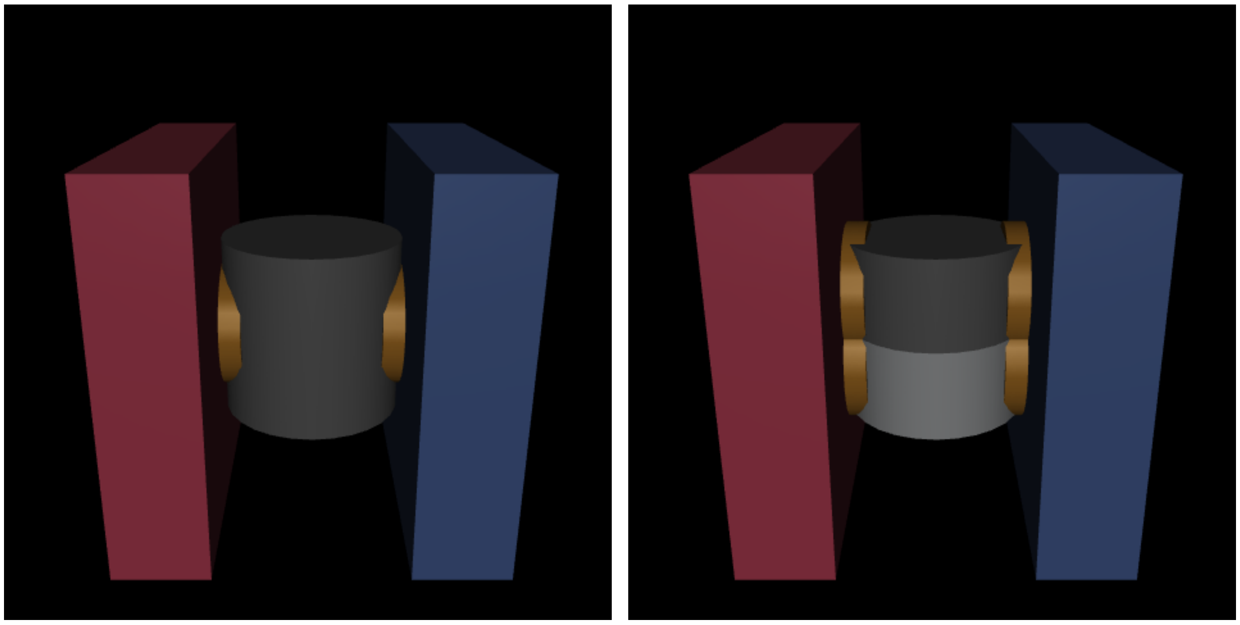


Figure 3.6. Comparison of the time-constant parameter T on the effects of force on contact slippage. In both graphs all forces except for 4.1 N are sufficient to theoretically prevent slip. As such, the position vs. time relationships for all forces, except for 4.1 N, are linear. The smaller time-constant (b) of 0.005 decreases the rate of constant slip compared to a time-constant value of 0.02 (a).

3.5 GRAB Dataset Object Transfer to Mujoco

As mentioned in Sec. 2.2.3, several key limitations restrict contacts simulated in Mujoco. First, all collision geometries are convex, and generally only one contact point can exist between two geometries. This restriction can only be bypassed by dividing up non-convex geometry into individual concave pieces that can form their own contact pairs. However, this must be done considering the object contact. For example, an edge-on-edge contact between two cylinders produces a single contact that oscillates between the top and bottom endpoints of the theoretical contact edge. The solution is to split one of the contact geometries in two, resulting in two contact points that are stable.

Fig. 3.7 depicts how the contact points of a cylinder suspended via friction between two boxes can be manipulated by dividing the cylinder into multiple bodies. Each contact is represented by a gold disc indicating the contact location (centre of the disc), and normal vector (disc longitudinal axis). Theoretically, contact between the cylinder and the boxes would have the shape of a line. Mujoco restrictions cause all contacts to be points. In Fig. 3.7(a), the cylinder is a single body, thus only has a single contact point between the cylinder and each box. Single contact points in this configuration allow the cylinder to unrealistically rotate about the horizontal axis. Fig. 3.7(b) shows the same cylinder divided into two bodies (which are welded together), each with their own contact points. This new configuration, although not matching the theorical contact shape, now has two contacts between the box and the composite cylinder, making horizontal rotation realistically more difficult.



(a) (b)

Figure 3.7. Demonstration of resolving Mujoco single contact restriction: (a) single-body cylinder (dark grey) held between two boxes (red and blue). One contact (gold disc) is present between the cylinder and each box; and (b) two-body cylinder (dark and light grey), where each cylinder section has a contact point with the boxes.

The single contact restriction and its solution drastically effect the choice of geometry for both the objects and simulated hands. Many of the GRAB Dataset objects are complex and concave. Although tools such as VHACD [53] can perform automatic convex decomposition, the results are not controllable, and the number of subdivided bodies (shards) increases significantly to reach a good approximation of the original object shape. Fig. 3.8 provides an example of convex decomposition for a detailed object using VHACD.

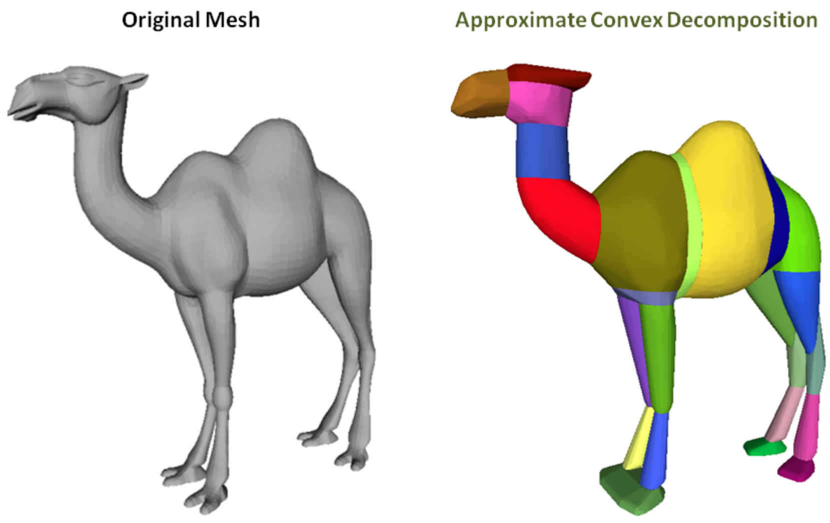


Figure 3.8. VHACD decomposition example [53]. Finer details of the original mesh, such as the camel's toe and knee joints, are lost in the decomposition process.

The original concave grasp objects from the GRAB dataset could be decomposed with high accuracy. However, the high number of convex shards produced changes the number of contacts in the grasp and alters the grasp forces. Some of these contacts are also likely to be on the edges or corners of multiple shards causing erroneous contact normals. To keep the expected types of grasp-contact consistent, the training data from the GRAB Dataset is filtered to only use episodes with convex interaction objects. This restricts the interaction objects to cubes (small, medium, large), cylinders (medium, large), spheres (small, medium, large), and pyramids (small, medium). Although this filtration greatly reduces the number of interaction objects, a reasonable portion of the interaction episodes remain to be used for training.

3.6 Simplified Simulation Hand

The GRAB Dataset is composed of the SMPL-X mesh poses for each subject for each frame of the episode. The MANO hand model is a single parameterized mesh that generally follows hand kinematics, but also deforms significantly with movement since the model does not preserve mesh sizes between hand poses. Given the restrictions of convex contact bodies in Mujoco, directly recreating the hand mesh model in Mujoco is not feasible. Using a preset hand model of an existing robot in Mujoco as a replacement would not be able to capture the differences in hand size and shape between the subjects in the dataset. Instead, a simplified hand is generated.

Mujoco can simulate composite objects attached by deformable constraints; however, directly simulating the hand mesh as a deformable composite object would not take the hand's hard kinematic hierarchy into account. Thus, a generated hand, composed of discrete rigid pieces and jointed based on the hand kinematic hierarchy, is used. Fig. 3.9 compares the original hand mesh (Fig. 3.9a) with the generated hand (Fig. 3.9b). The generated hand is composed of several capsule shapes, sized to fit the individual finger segments of the subject's hand. Each capsule has a length generated from the initial hand pose skeleton, and a diameter centred around an estimated fingertip radius from the initial hand pose. The resulting hand is more skeletal than the original mesh, and in almost all cases, is inscribed in the original hand mesh volume. Even after simplification, the original hand mesh and generated hand still have the same pose errors. For example, in Fig. 3.9 the index finger penetrates the cube (over-penetrating contact), and the thumb has a gap above the object surface without contact (hovering contact).

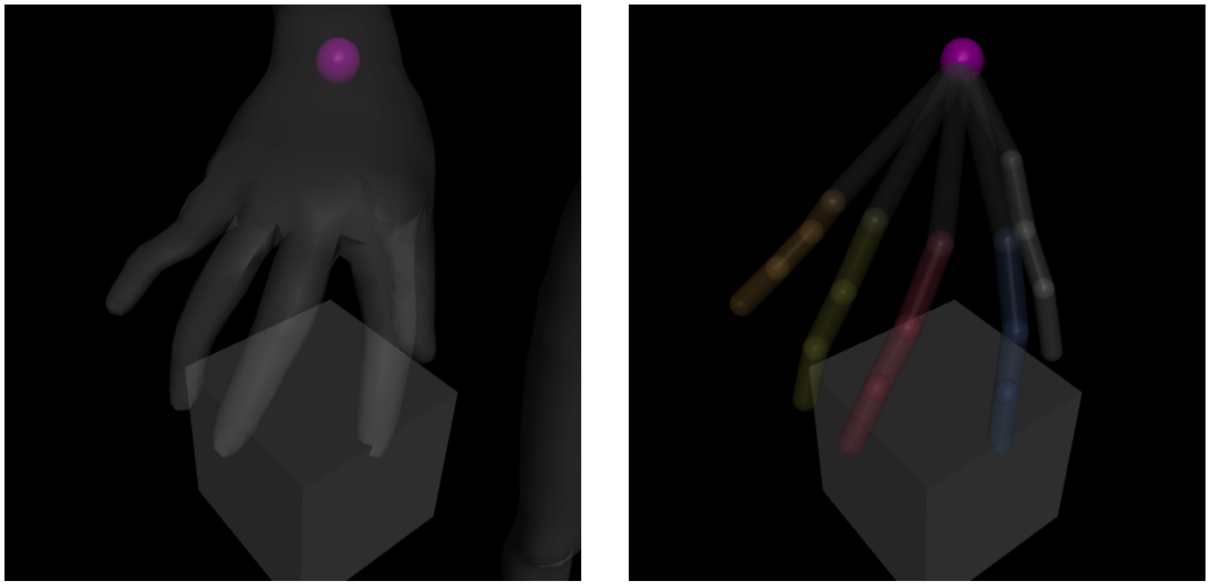

(a) (b)

Figure 3.9. Comparison of (a) original hand mesh with (b) simplified hand model in the same pose. The wrist joint is marked by a purple sphere and each finger has a separate color.

Generated hands also address the Mujoco contact limitation of convex collision geometries. The simple shapes of the simplified hand segments are already convex and reduce the probability of contact situations where the single contact restriction is not realistic.

3.6.1 Initial Frame Choice

Each GRAB Dataset episode starts and ends with the subject in a T-pose with no contact with the interaction object. Although the entire episode could be replicated in Mujoco from the beginning, this would include grasp approach and grasp initiation, which would be outside the scope of this thesis. Furthermore, the sensitivity of contacts to error could easily cause even small differences in pose or shape to result in unintended collisions on grasp approach and initiation. To avoid these potential complications, the initial frames of the grasp are chosen after the GRAB Dataset contact distance threshold has been reached and the number of fingers in contact remains stable.

3.7 Pose Adaptation

As mentioned in Sec. 3.3, the GRAB Dataset does not directly measure contacts. Detecting contacts via visual motion capture is difficult because proximity and contact are often only distinguishable from a few angles. Furthermore, contact is discontinuous, allowing small pose errors to potentially lead to contact misclassification. Finally, the hand and finger surfaces are not directly measured. The surfaces are generated using a tuned body and hand model, which may not exactly represent reality. The GRAB Dataset avoids this issue by using a large proximity margin of 4.5 mm for hand and object mesh vertices to define contact. Although defining contact by proximity would produce an acceptable contact heatmap, the result is raw hand poses that often have fingers that should be in contact with the grasp object, but actually have a small gap between the contact surface. These "hovering" contacts carry over when the hand poses are transferred into simulation and are sometimes exacerbated because the simplified geometry does not completely match the body mesh. Overlap between the fingers and grasp object geometries (overpenetration) is another type of contact error present, but with less frequency than hovering contact.

Even with hovering contacts, over-penetrating contacts, and other pose errors after hand pose importation, the resulting pose will still be close to a viable pose. Thus, grasp synthesis is not necessary. Hovering and over-penetrating contacts must be resolved before allowing the simulation to proceed because they would disrupt the viability of the grasp. Alg. 3.1 uses a combination of brute force search and contact normal position adjustment to resolve the contact errors. The existing GRAB Dataset contact distance threshold is initially used to determine which fingers within a proximity margin of m should have contact and need adjustment (Alg. 3.1 Line 2). To resolve hovering contacts, a brute-force search of joint angles close to the original pose is conducted to find the closest finger poses that contact the interaction object. Permutations of joint changes are stepped by a value j_{step} up to a maximum of j_{max}. The change with the minimum magnitude that produces contact (negative proximity) is selected as the new adjusted finger pose (Alg. Line 3). Overpenetration (contact distance greater than r_{max}) due to the initial pose or the brute-force contact search is resolved by moving the geometries apart along the contact normal until the penetration distance is below a manually selected threshold r_{max} (Alg. 3.1 Line 5:6). The current pose Jacobian $J(\vec{P})$ is used to translate motion along the contact normal to joint pose changes.

Algorithm 3.1. Hovering and over-penetrating contact pose correction.

```
Input: Finger joint poses: $\vec{P_i} = [p_{i_1} \quad p_{i_2} \quad p_{i_3} \quad p_{i_4}] \in [0,2\pi)^4$
        …for five fingers: $F_0 = \{P_1 \quad P_2 \quad P_3 \quad P_4 \quad P_5\}$
        Proximity margin $m$, joint change steps $j_{step}$ up to a maximum of $j_{max}$,
        Maximum contact distance $r_{max}$
Output: Corrected finger joint poses $\vec{P_i}$ for five fingers $F_1$
1:  for $\vec{P}$ in $F_0$ do
2:      if $m \geq Proximity(\vec{P}) > 0$ then
            /* Change $\vec{P}$ by the minimum joint step $j_{step}$ up to $j_{max}$ that has negative proximity */
3:          $\vec{P} \leftarrow \underset{\vec{P}}{argmin} \left\| \{0 \quad j_{step} \quad 2j_{step} \quad \cdots \quad j_{max}\}^4 \in Proximity(\vec{P} + \vec{J}) < 0 \right\|$
4:      if $0 \geq Proximity(\vec{P})$ then
5:          while $contactDist(\vec{P}) > r_{max}$ do
                /* Step contact position along contact normal by $r_{step}$ */
6:              $\vec{P} \leftarrow \vec{P} - J(\vec{P})^{-1} \cdot r_{step} \cdot contactNormal(\vec{P})$
```

3.7.1 Initial State Grasp Balancing

Although the GRAB Dataset includes essential information such as grasp and object pose, force information is unfortunately not present. The finger control policy is responsible for managing finger forces but acts on the initial state of the grasp. If the initial state is not stable, then the policy would likely be given irrecoverable grasp states, potentially hindering the learning process. Since contact forces depend on both the actuator forces and contact impedance (which depends on contact distance), both the actuator forces and finger poses need to be adjusted to bring the initial grasp state into equilibrium.

3.7.1.1 Analytical force-closure solution

Given the simple point-contact and friction-cone model, described in Sec. 2.1, the forces required for grasp equilibrium within the friction-cone constraints can be solved as a convex optimization problem [54]:

$$-F_e = Gf_c \,, f_c \in K \tag{3.9}$$

where F_e is the external force wrench on the object that the grasp is resisting, G is the transformation matrix that maps local contact forces onto the global wrench space, and f_c is a matrix of all the contact forces subject to the friction cone restrictions K (Eq. 2.4).

Most convex solvers can readily solve for f_c; however, the force solution is only for completely rigid contacts and does not incorporate any contact parameters other than position and contact normal. Since Mujoco contact forces depend on contact impedance, and thus contact distance, the analytical point-contact force-closure method is not sufficient to find a complete solution for an equilibrium grasp state by itself. However, analytical point-contact solutions are used as starting solutions for other methods of finding the grasp forces.

3.7.1.2 Equilibrium pose optimization

Since contact forces, and thus grasp equilibrium, in Mujoco, depend on both external actuated force and contact position, some combination of applied force and pose close to the original dataset pose would produce a grasp in equilibrium and minimize undesired acceleration. Since the contact impedance model implies that changes in external force and contact distance always change the contact constraint force, convex optimization might be a possible method for finding equilibrium values. The process is unlikely to be straightforward, as the optimization inputs are the finger contact poses, and the outputs are affected by the surface curvature, and thus the shape of the object. Furthermore, multiple solutions exist for the acceleration minimization problem. It is possible that for equilibrium pose optimization, an initial guess at the contact forces and finger poses might be good enough to permit optimization to converge to a final solution.

Several variations of pose optimization were tested as potential solutions for the initial grasp forces and pose adjustments. These included direct optimization of pose and force values via gradient descent and Nelder-Mead algorithm [55] with and without analytical force-closure initial conditions and force solutions, and gradient descent optimization of only contact positions along the contact normal instead of joint angles. Table 3.2 lists each tested variation of optimization. They key differences between variations was the optimization algorithm, and the parameters being optimized. Variations 1 and 2 optimized over the joint poses and joint forces directly, with the initial force values determined by the analytical force-closure method. Variations 3 and 4 only optimized the joint poses with the forces determined for each pose value via analytical force-closure. Finally, Variation 5 optimized the contact positions along the contact normal instead of directly altering the joint poses, while the contact forces were once again determined by analytical force closure.

Table 3.2. Pose optimization variations (X indicates use).

Variation	**Optimization algorithm**	**Optimization parameters**	**Used force-closure solutions for force**
1	Nelder-Mead	Joint poses and forces	
2	Gradient descent	Joint poses and forces	
3	Nelder-Mead	Joint poses	X
4	Gradient descent	Joint poses	X
5	Gradient descent	Contact normal position	X

Unfortunately, all optimization variants converged onto a local minimum, which did not produce equilibrium. The most common optimization result was the trivial case of no contact force and minimal contact distance, thus minimizing the object acceleration to gravity only. It is possible that the stiff contact impedance parameters chosen to reduce constant contact slip also narrowed the transition region between stable and unstable contact, thus increasing the difficulty of optimization. Clearly, without carefully crafted constraints, convex optimization would not be viable for direct pose optimization in Mujoco, even with a reasonable initial guess.

3.7.1.3 Contact constraint force balancing

Since the general optimization techniques failed to converge on viable pose and force solutions for the initial grasp state, a separate method similar to optimization Variation 5 was explored. The new method expands on the analytical force-closure solution by adding a step to consider the effect of contact impedance on contact force in Mujoco. As discussed in Sec. 3.4.4, contact distance determines contact impedance, which then impacts contact force. Without sufficient impedance, a contact would not be able to properly exert force, since the contact constraint would be interpolated close to the unconstrained acceleration (Eq. 2.12). Thus, for any desired grasp force, a matching impedance must be found to allow the desired contact force.

Alg. 3.2 is the process used to find the joint forces and pose adjustments for the initial grasp. The original grasp pose is used to generate joint forces via the analytical force-closure method. For each contact, the pose is adjusted along the contact normal by r_{step} to increase or decrease the contact distance and find an impedance that can generate the force-closure contact force. Force-closure forces are solved for the new poses, and the cycle is repeated until the total pose adjustment falls beneath a chosen threshold m. Once again, the Jacobian J is used to translate pose changes along the contact normal to joint angle changes.

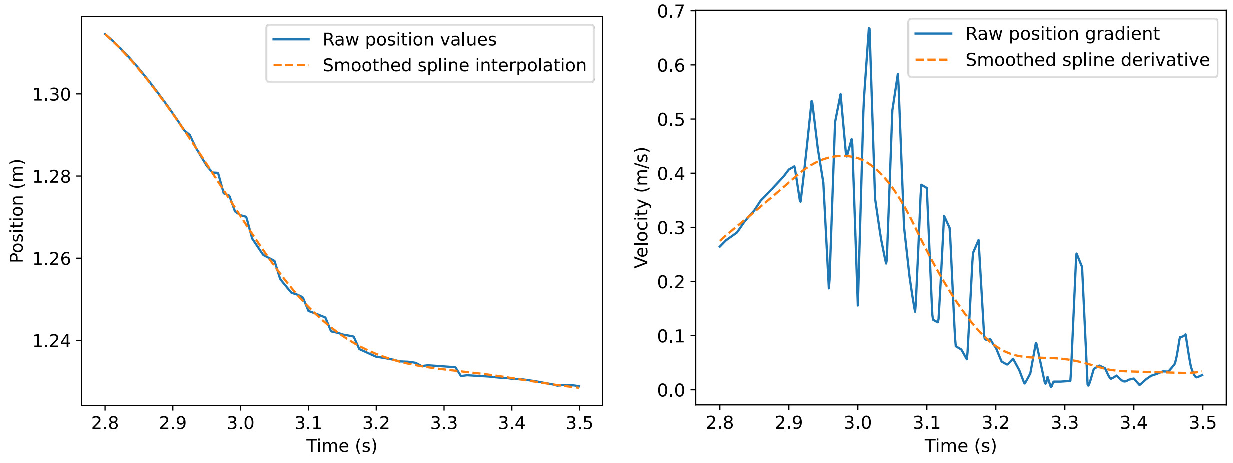
Algorithm 3.2. Contact constraint force balancing.

```
Input: Finger joint poses: $\vec{P_i} = [p_{i_1} \quad p_{i_2} \quad p_{i_3} \quad p_{i_4}] \in [0,2\pi)^4$
        …for five fingers: $F_0 = \{P_1 \quad P_2 \quad P_3 \quad P_4 \quad P_5\}$
        Pose change margin $m$, contact distance step $r_{step}$
Output: Corrected finger joint poses $\vec{P_i}$ for five fingers $F_1$
1:  while $\|\Delta F\| > m$ do
2:      $f_a \leftarrow forceClosure(F)$
3:      for $\vec{P_i}$ in $F$ do
4:          while $contactForce(\vec{P_i}) - f_{a_i} < e$ do
5:              $d \leftarrow contactForce(\vec{P_i}) - f_{a_i}$
6:              $\vec{P_i} \leftarrow \vec{P_i} + sign(d) \cdot J(\vec{P})^{-1} \cdot r_{step} \cdot contactNormal(\vec{P})$
```

The contact constraint force balancing method finds contact impedances to match the grasp forces found via analytical force-closure optimization. However, the contact distance for each contact may not be stable and could oscillate before settling at a stable value such as in Fig. 3.4 in Sec. 3.4.4. The overall effectiveness of the contact constraint force balancing method for initial state balancing is evaluated in the results chapter.

3.8 Hand Trajectory Optimization

In addition to the grasp-internal forces, the trajectory of the hand during the grasp episode is replicated to recreate the external forces applied to the object. Before the trajectory can be imported into simulation, the raw dataset poses must be preprocessed to smooth pose wobbling and match the simulation sampling rate. The GRAB dataset only includes pose data without velocity or acceleration information. Thus, any velocity or acceleration reference value for the trajectory must be estimated from the original poses. Fig. 3.10 illustrates the issue of velocity estimation with pose wobbles in the raw dataset. The small position wobbles in Fig. 3.10a cause discontinuities in the velocity estimate in 3.10b. Cubic splines with a tuned smoothing factor remove the pose wobbles and make the velocity and acceleration estimates smooth.



(a) (b)

Figure 3.10. Comparison of smoothed dataset pose values for: (a) position, and (b) velocity.

The smoothing factor restricts the maximum divergence of the spline from the sample points according to the function:

$$s \geq \sum_i (g(x_i) - y_i)^2 \ , \tag{3.10}$$

where s is the smoothing parameter, x_i and y_i represent the points for i samples to be smoothed, and $g(x)$ is the smoothed spline function.

The smoothing parameter values were chosen manually by inspecting the velocity and acceleration estimates for discontinuities. The final smoothing values were $s = 0.0004$ for the XYZ positions and $s = 0.014$ for the quaternion poses. Since quaternions must have a magnitude of one, the SQUAD spherical cubic interpolation algorithm [56] was used for smoothing all quaternions together. The XYZ positions were smoothed with independent splines. In addition to removing wobbles, spline smoothing has the dual effect of interpolating the poses, allowing a simulation timestep not matching the dataset sampling rate.

The smooth pose trajectories are suitable for replication in simulation. Two methods were attempted, each with a distinct advantage and disadvantage. The first method is to use convex optimization to find the optimal forces that produce the desired pose trajectory, and the other is to use Mujoco's "Mocap" property to arbitrarily set the position of a virtual object to the trajectory, and then use a soft constraint to fix the wrist pose to that object.

The force optimization method involves searching for the forces that would allow the currently tracked values (position, velocity, or acceleration) to match the desired values of the next step in the trajectory. Since the simulation can be stepped and resampled for different force inputs, it is possible to empirically converge on a force that achieves the desired trajectory, regardless of the disturbances from manipulation. In initial explorations, this method of optimization worked best for matching the velocity of the desired trajectory, while position and acceleration matching often drifted from the target trajectory until optimization failed. The advantage of this method is that it can instantly adjust forces to stay on target, regardless of the disturbances encountered during the grasp manipulation. The drawbacks are that the optimization significantly increases computation time because the simulation state is sampled multiple times per optimization step. Furthermore, without restrictions, the forces found by the optimization process can be discontinuous, and although those forces would make the hand follow the desired trajectory, discontinuous forces on contacts in the grasp could cause overpenetration or unrealistic changes in contact distance.

The other method of replicating the hand trajectories in simulation uses a solution already incorporated in Mujoco: the Mocap object. In Mujoco, the Mocap object is a geometry that has its position arbitrarily set for each timestep, for use with motion-capture. This Mocap geometry alone does not solve the issue of trajectory following, because it essentially is a teleporting geometry that would cause issues with the grasp contacts. Instead, the Mocap geometry can be used as an intangible target, onto which the actual hand wrist is fixed, via a soft constraint. As the Mocap geometry is moved to new positions, the constraint produces forces that try to keep the wrist fixed to the Mocap geometry's pose. This does not guarantee that the constraint forces will make the hand follow the trajectory. The procedure will produce forces according to the constraint parameters that try to maintain the fixed constraint in a viable simulation solution. The resulting trajectory can thus exhibit oscillations and divergence from the target trajectory based on disturbances from the grasp. The greatest advantage of the Mocap method is that it is already integrated in Mujoco, and is significantly computationally less intensive than the force optimization method (from testing, this has been found to be one to two orders of magnitude faster). Even though the trajectory is not guaranteed from the soft constraint, tests demonstrated that the trajectory divergence is most often an issue only when the grasp contact is already lost. For these reasons, the Mocap method of trajectory following was chosen for this research.

3.9 Slip Prediction in Mujoco

Slip prediction for human and robotic hands in the real-world use temporal and spatial tactile information from pressure, force, and surface texture sensors. Replicating such a setup in simulation would be highly complex and require introducing slip classification as an additional problem to be solved. The simplified contact model of Mujoco might not exhibit the same slip signals that are detected in real slip; therefore, any slip detection and prediction method must be tailored to Mujoco's contact model to ensure accuracy.

Mujoco's soft contact model allows frictional forces to not always be completely resistive and contact normal force and velocity to both be positive. A consequence of this model is that slip initiation results in motion along the contact normal and vice versa [41]. As introduced in Sec. 3.4.1, contact distance is the shortest normal distance between two penetrating geometries in contact. The coupling of incipient slip and normal-distance changes is thus reflected in contact distance values. This can be seen in the oscillation and settling behavior in Figs. 3.4 and 3.5. Unstable contacts have contact distances that oscillate up to zero, resulting in zero crossings of regular frequency. This oscillation characterizes the rate of slip and separates non-constant slip from the constant unavoidable slip addressed in Sec. 3.4.4.

Fig. 3.11 displays the relationship between the frequency of contact distance zero crossings to the degree of non-constant slip for a contact. The situation illustrated is an object held via friction against a vertical wall simulated across different friction forces. The minimum number of zero crossings, 1, corresponds to a maximum friction force of zero and an immediate and permanent loss of contact, since zero contact distance corresponds to no contact. The discontinuity at approximately 4.1 N of maximum friction force corresponds to the minimum ideal force required to prevent non-constant slip. Forces between 0 and 4.1 N experience slip with a degree that corresponds to the frequency of contact distance zero crossings. Forces greater than this threshold have contact distances that stabilize, and only suffer from the small unavoidable constant slip. This relationship between contact-distance zero-crossing frequency and slip-type allows the clear detection of non-constant slip in Mujoco. Slip prediction is a simple extension of slip detection. Since simulations can be copied, saved, and reverted, prediction of slip can be simply performed by saving the current simulation state, stepping the simulation forward in time until slip is detected, then reverting to the original state.

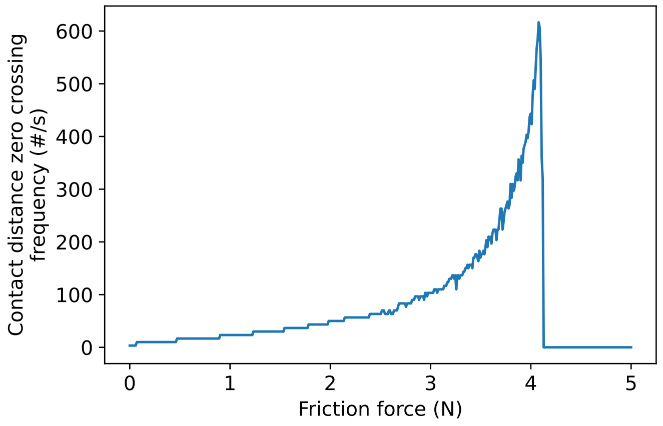


Figure 3.11. Friction force vs. contact distance zero crossing frequency. Contacts with friction forces below 4.1 N are experiencing accelerating (non-constant) slip.

3.10 Reinforcement Learning Pipeline

After the GRAB dataset is converted into simulation, the next major step is running the learning process. As discussed in Sec. 2.3.3, reinforcement learning is well-suited for interactive tasks like grasp stabilization. While many reinforcement learning methods focus on the value or quality functions, policy gradient methods focus on updating a parameterized policy representation itself, usually taking parameter steps that increase the expected value distribution

[57]. For many modern reinforcement learning setups, the policy gradient parameterization is done through a neural network, and thus the stepped policy parameters are the weights of the neural network. Similar to neural network backpropagation, parameter adjustment is performed via a gradient of the expected policy return over possible trajectories. For settings with black-box environments, these gradients need to be estimated. Expressed in the simplest generic form, the update functions for the parameters θ are:

$$\theta_{k+1} = \theta_k + \alpha_k \nabla_\theta J(\theta) \tag{3.11}$$

$$U(\theta) = \mathbb{E}\left(\sum r_t | \pi_\theta\right) , \tag{3.12}$$

where the gradient of the objective function U is used to update the parameters θ from iteration k to iteration $k+1$ with a learning rate α. Note that the objective function does not need to be the expected policy reward $r_t|\pi_\theta$. Other combinations of value, quality, and advantage function objectives can be used. More sophisticated methods also adjust the learning rate in order to achieve better results.

The reinforcement learning method used in this research is Proximal Policy Optimization (PPO). There were two reasons PPO was chosen for this research. First, PPO has widespread and recent use for continuous control problems [46], including grasp-related control problems in simulation [13] [15]. Second, the primary advantage of PPO is that its update method restricts parameter updates based on characteristics of the current policy, thus improving policy stability. The specific variation is the *clipped advantage function* variation of PPO. PPO alters the generic objective function to rely on a clipped advantage function $L_{PPO}(\theta)$ as the objective function [57]:

$$\theta_{k+1} = \underset{\theta}{\operatorname{argmax}}\, L_{\text{PPO}}(\theta) \tag{3.13}$$

$$L_{\text{PPO}}(\theta) = \mathbb{E}\min\left(r_p A_t, clip\left(r_p, 1-\epsilon, 1+\epsilon\right) A_t\right) \tag{3.14}$$

$$r_p = \frac{\pi(s,a)}{\pi_{\text{prev}}(s,a)} , \tag{3.15}$$

where the parameters θ are updated from iteration k to $k+1$ by the maximum argument of the L_{PPO} function. r_p is the ratio of the probability for some given action compared to the probability under the previous policy, A_t is the current advantage (Eq. 2.29) , the $clip(value, min, max)$ function restricts values between a maximum and minimum, and ϵ is a hyperparameter that limits the risk that the policy takes by constraining the probability ratio to $[1-\epsilon, 1+\epsilon]$.

The L_{PPO} objective function constrains the parameter update steps by clipping the update based on the estimated risk using the advantage values [45] [57]. Note that this method does require advantage estimates and is thus contingent on having a method to provide such estimates.

3.10.1 Policy Function Approximator

This research uses simple multi-layer perceptron neural networks for both the actor and critic functions. Both actor and critic neural networks are restricted to a single hidden layer, inputting the simulation-environment observation-vector and outputting force for each of the finger-joint actuators. Table 3.3 lists the policy input elements and output vectors. Since all the input components can be represented by vectors of fixed size, complex convolutional layers or size-invariant structures are not necessary. Existing literature training policies in simulation [15] [29] and on physical systems [47], use convolutional layers for image pre-processing, and also use multi-layer perceptrons for fusion of vector information to calculate the final policy output.

Table 3.3. Actor and critic input and output vector component description.

Input/Environment Observation Size	**Element Size**	**Number**	**Total**
Finger segment lengths	4 (per finger including palm segment)	5 (fingers)	20
Joint angles	4 (per finger)	5 (fingers)	20
Joint velocities	4	5	20
Previous joint forces	4	5	20
Wrist linear velocity	3	1 (hand)	3
Wrist angular velocity	3	1	3
Contact relative position	3	15 (finger segments)	45
Contact normal force	1	15	15
Slip prediction	1	15	15
Total			**161**
Output Size			
Joint forces	4 (per finger)	5 (fingers)	20
Total			**20**

3.10.2 Reward function

During each trajectory episode, a reward is calculated for each timestep based on the simulation state. The reward function should be chosen to shape the desirable qualities of the control policy when maximized during the learning process. For grasp stability, the desired qualities are maintaining the grasp, and minimizing the force required to maintain the grasp. Grasp episodes are run until the trajectory data ends, or contact is lost, thus any positive reward will reward maintaining the grasp. The simplest such reward function is just a constant reward of 1 for each timestep.

$$R(t) = 1 \tag{3.16}$$

Smaller grasp-force magnitudes can be favored by adding a penalty based on the magnitude of the grasp forces; however, this works in opposition to the positive maintained grasp score. Since there are no known minimum grasp force bounds for each object grasp, a negative bias based on force magnitude cannot be introduced without potentially exceeding the positive reward value. Instead of using a bias or constant penalties for both the grasp maintenance reward and the force minimization penalty, the ratio of the corresponding rewards can be used as the final reward.

$$R(t) = \frac{e^t}{F_{grasp} + 1} \tag{3.17}$$

The above reward function (Eq. 3.17) exponentially increases the reward for later timesteps, while inversely penalizing grasp-force magnitude. Since the grasp maintenance reward numerator increases faster than the force minimization denominator, the reward will always favor maintaining a grasp over reducing the grasp force.

3.11 Experimental Setup

Experiments were divided into two stages: grasp conversion stability analysis, and policy training. For general Mujoco contact parameters, a Condim of 6 was chosen to include all types of frictional forces (tangential, torsional, rolling). The Solimp values were tuned to $dmin = 0.9$, $dmax = 0.95$, $width = 0.001$, and $power = 2$. Finally, the Solref values were tuned to $T = 0.005$ and damping ratio $d_r = 1$.

The grasp conversion process started by generating simplified hands in simulation for each grasp episode in the GRAB dataset (Sec. 3.6), followed by contact pose adaptation to eliminate hovering or over-penetrating contacts (Sec. 3.7). Final grasp initial state balancing was performed using the constraint force balancing method (Sec. 3.7.1.3). The grasp trajectory smoothing was performed with tuned smoothing weights of $s = 0.004$ and $s = 0.014$ for XYZ and quaternion poses, respectively. To analyze the converted grasp's stability over the imported trajectories, each grasp episode was simulated with the initial grasp forces held constant.

The policy training experiments were conducted with actor and critic input sizes of 161, output sizes of 20, and a hidden layer with a size of 256. The dataset was divided in two sets using an 80% - 20% split to produce a training set and test set, respectively. The test dataset was set aside for final policy validation and was never used to alter policy parameters or training hyperparameters. The PPO training process was run using a constant sampled data batch length of 16,384 timesteps for 500 batches. A policy output standard deviation value of 0.01 was selected to promote grasp policy exploration during training, and a maximum L_{PPO} clip value of 0.2 was used. Two reward functions were tested: the constant time reward (Eq. 3.16), and the shaped exponential reward function (Eq. 3.17).

Chapter 4 Results and Discussion

4.1 Introduction

This chapter presents the research results of the two primary objectives: the GRAB dataset transfer into Mujoco and the PPO learning with the multi-layer perceptron neural network.

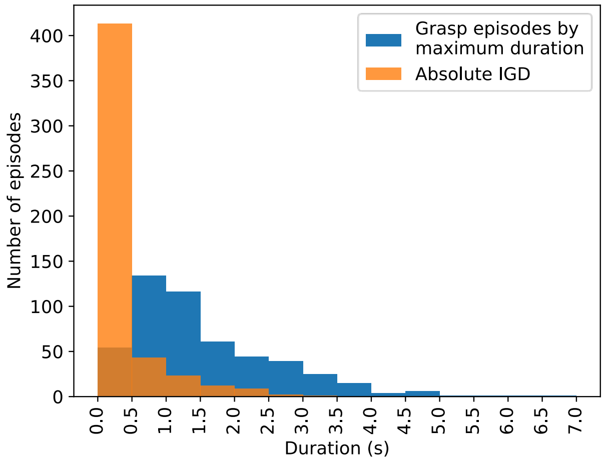
Sec. 4.2 addresses the effectiveness of converting human grasps from the GRAB dataset into Mujoco. The objective was to ensure that the viable human grasps remained viable after being replicated in simulation. The primary metric used to evaluate the grasp viability was the duration of grasp stability given the converted initial grasp state, or “initial grasp duration”. This value also served to establish a baseline for the learned policies evaluated in the second results section (Sec. 4.3). The initial grasp duration was compared to episode characteristics such as subject, grasp object, maximum acceleration, and number of contacts, to determine if the dataset conversion process introduced unwanted biases or had specific weaknesses.

Sec. 4.3 discusses the attempt to learn a grasp policy using a PPO reinforcement learning process. The losses for the neural network structure (Sec. 3.10) during PPO policy training indicate learning effectiveness. Two reward functions were tested: a generic constant non-scaling reward for each timestep (Sec. 3.10.2, Eq. 3.16); and an exponential reward function scaling with time, while penalizing grasp force magnitude (Sec. 3.10.2, Eq. 3.17).

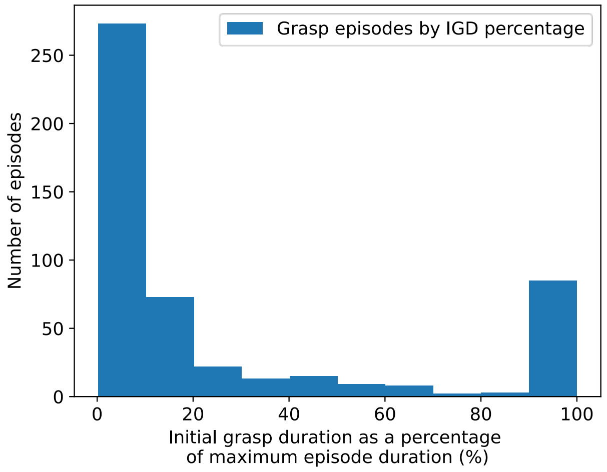
4.2 Grasp Conversion Results

To test the stability of the calculated initial states, each grasp episode was simulated with no additional inputs. The initial grasp forces calculated by the contact constraint force balancing process, described in Sec. 3.7.1.3, were held constant until the grasp was lost or the episode completed. The stable-grasp duration with only the initial forces is referred to as “initial grasp durations” (IGD) for the rest of this section. The initial forces were not expected to maintain the grasp for the entire grasp episode, since the forces do not adapt to the grasp trajectory, external accelerations, or disturbances. However, the initial grasp forces were expected to keep the grasps stable for the initial portion of the episode so that a learning policy had a starting point that is nominally stable. Ideally, the initial grasp forces would maintain initial stability across episodes that vary in number of contacts and disturbances. The results also served as a baseline for the trained policy effectiveness.

The stability results were collected with a force overgrip of 20%, within the typically observed human overgrip range of 10-40% [2] [4]. Fig. 4.1 is a histogram of number of episodes for varying stable grasp duration, showing the distribution of the stable initial grasp durations and maximum episode durations (Fig. 4.1a), and the distribution of the stable initial grasp durations relative to the maximum episode duration, or relative stable duration (Fig. 4.1b). As expected, Fig. 4.1a shows that the stable grasp durations distribution was shifted left from the maximum duration distribution, representing a smaller mean duration compared to the maximum durations. The relative episode stability duration distribution in Fig. 4.1b shows that the initial grasp stability durations were not equally distributed relative to the maximum episode length. Out of the total 503 grasp episodes, 81 (16.1%) were stable for the entire episode duration. However, the mode of the relative stable duration distribution was within the 10% duration bin. This could indicate that some factor of the grasp initial condition separates grasps that are stable for the full episode duration from grasps that are only stable for the initial portion of the episode.



(a)



(b)

Figure 4.1. Histogram of number of episodes for varying stable grasp durations given the constant initial grasp forces: (a) with distribution of maximum episode duration, and (b) with the stable initial grasp duration as a percentage of the maximum episode duration.

Since the initial grasp forces were held constant, the initial grasp stability was expected to be disrupted by high accelerations or simply accumulated slip. Fig. 4.2 is a histogram of number of episodes for varying stable grasp durations, showing the distribution of episode maximum durations in the dataset and the average initial stable grasp duration. As expected, the average initial grasp duration decreased with a greater maximum episode length, since greater maximum episode length decreases the relative duration of the stable grasp region. The relative initial grasp duration also clearly had a negative correlation to the maximum episode XYZ and angular accelerations, as shown in Figs. 4.3a and 4.3b. This is also expected, as the initial grasp forces did not compensate for disturbances; accelerations greater than the acceleration at the initial state are thus expected to upset or degrade the grasp. Table 4.1 contains the calculated correlation values for the maximum durations and accelerations and shows that the negative correlations with the initial grasp duration are all strong with low p-values.

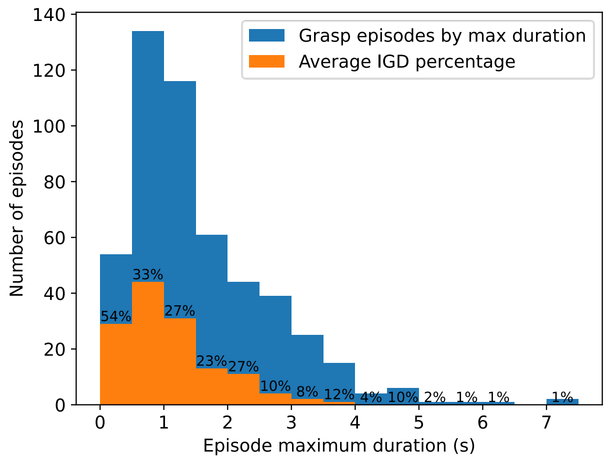
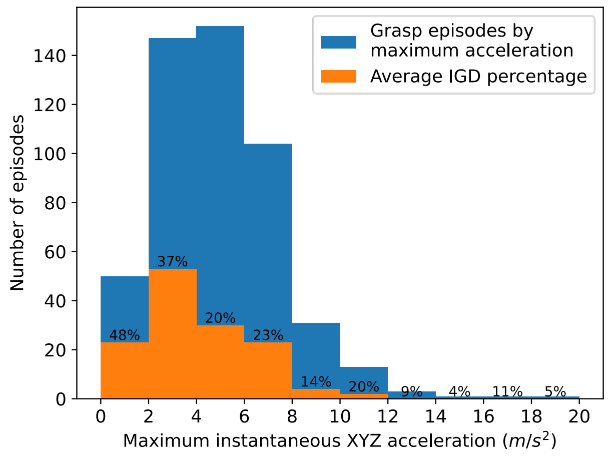
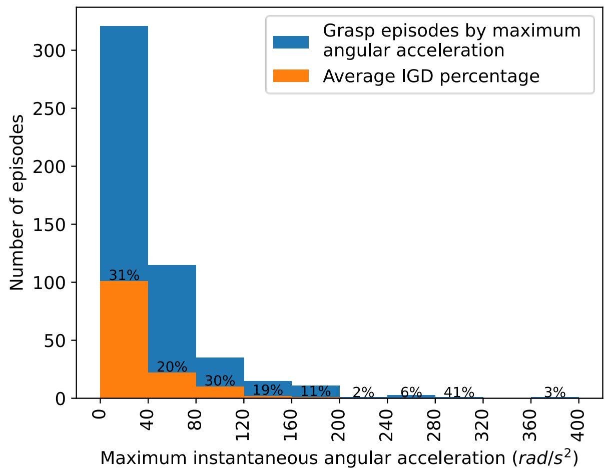


Figure 4.2. Histogram of number of episodes for varying grasp episode maximum duration (blue) and stable grasp duration expressed as a percentage of entire episode duration (orange).



(a)



(b)

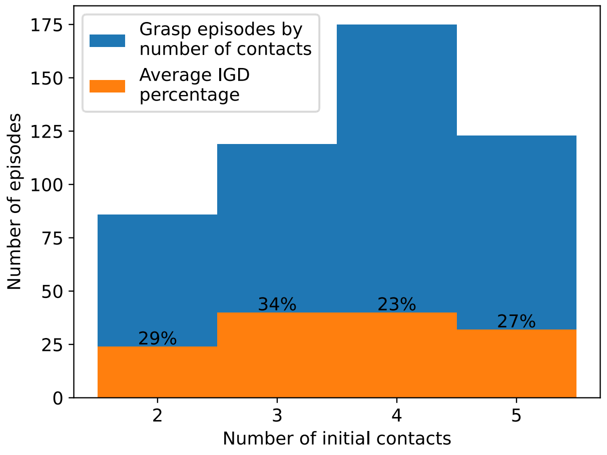
Figure 4.3. Histogram of number of episodes for varying grasp episodes distributed according to: (a) maximum XYZ acceleration, and (b) wrist angular acceleration in the episode.

Table 4.1. Episode metrics correlation to stable grasp duration ratio.

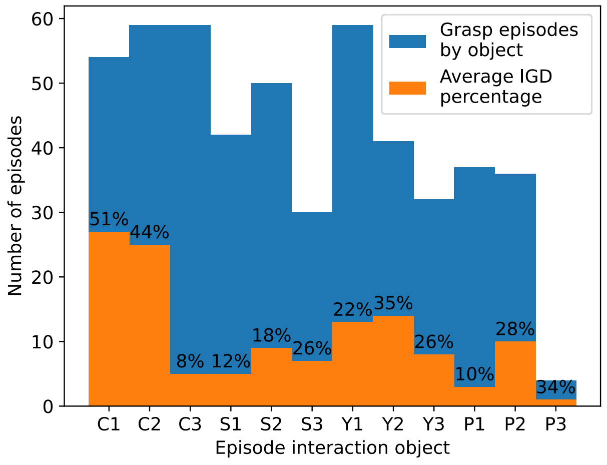
	Pearson Correlation	**Pearson p-value**	**Spearman Correlation**	**Spearman p-value**
Max. episode length	-0.30	5.80e-10	-0.52	2.13e-36
Max. XYZ wrist acceleration	-0.23	1.13e-7	-0.27	7.96e-10
Max. angular wrist acceleration	-0.14	2.19e-3	0.22	7.68e-7

A key requirement of the dataset conversion into simulation was to ensure that the initial grasps were stable for episodes with a variety of starting conditions. Fig. 4.4 shows the distribution of average initial grasp stability vs. episode characteristics, such as the initial number of grasp contacts, grasped object, subject, and episode task. Each histogram contains the distribution of the dataset according to a specific metric, as well as the average initial grasp duration as a percentage of the maximum duration. None of these episode characteristics were expected to correlate to the initial grasp duration, as a correlation might indicate an unwanted bias in the dataset. The distributions in Fig. 4.4 do not demonstrate any correlations between the initial grasp duration and the number of initial contacts, subject, or episode task; however, the distribution in Fig. 4.4b does highlight specific objects that the initial grasp struggles to keep stable. The large pyramid (P1) was only kept stable for on average 10% of the episode durations. Upon review of the raw dataset footage, the large pyramid's low initial grasp duration appeared to be due to the large pyramid being too large to be grasped from opposing edges, and often having to be grasped using the top face, unlike the small (C3) and medium pyramids (C2). In pyramid grasps using the top face, the contact normal forces oppose the friction forces lifting the object, making the grasp very sensitive to error and disturbances. The large sphere (S1) had a low relative grasp duration of 12% for similar reasons, as the sphere was too large for the hand to fully enclose it. This may be the reason that the initial grasp duration progressively increased to 18% for the medium sphere (S2) and 26% for the small sphere (S3), though this is only an observation for three sphere sizes. The last poorly performing object was the small cube with a average relative grasp duration of 8%, which did not have a clearly observable explanation.

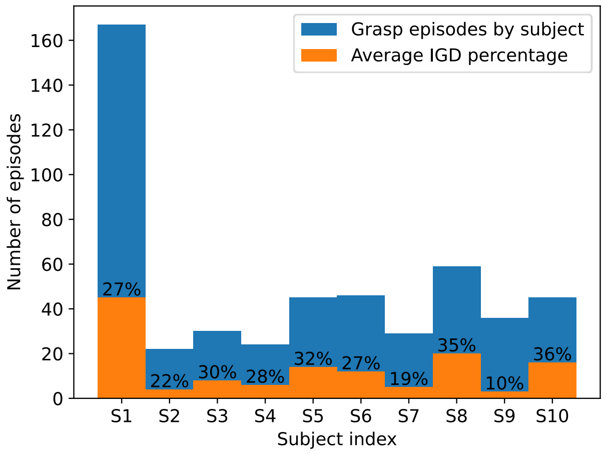
Although size could have made the small cube grasps more vulnerable to pose error, the small cylinder (Y3) was of similar size, yet had a much higher stable duration (26%). Overall, differences in initial grasp stability due to the grasped object are expected, as objects have different characteristic grasps with varying robustness.



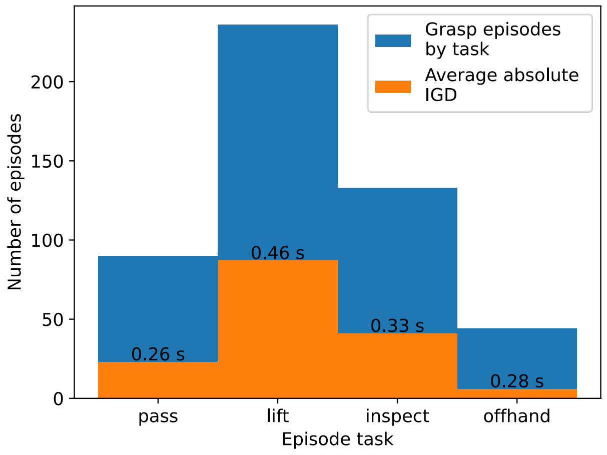
(a)



(b)



(c)



(d)

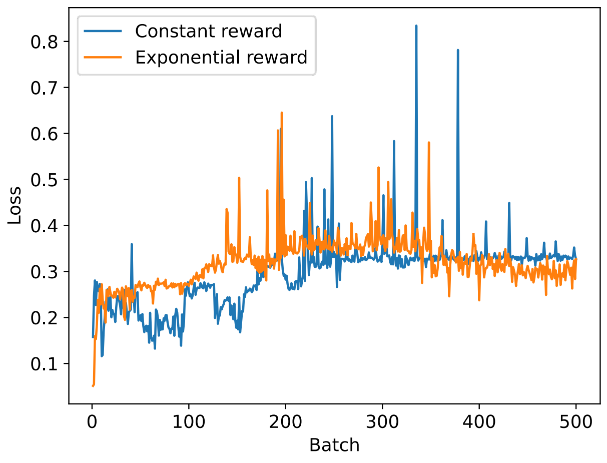
Figure 4.4. Histograms showing number of episodes for average initial stable grasp duration and episode characteristic such as: (a) number of initial contacts, (b) grasped object, (c) subject performing the task, and (d) interaction task. The initial grasp durations for (d) are in absolute terms, since each task category had different average maximum durations. The grasped objects in (b) are labeled according to object (Cube: C, Sphere: S, Cylinder: Y, Pyramid: P) and size (Large: 1, Medium: 2, Small: 3).

4.2.1 Grasp Transfer Discussion

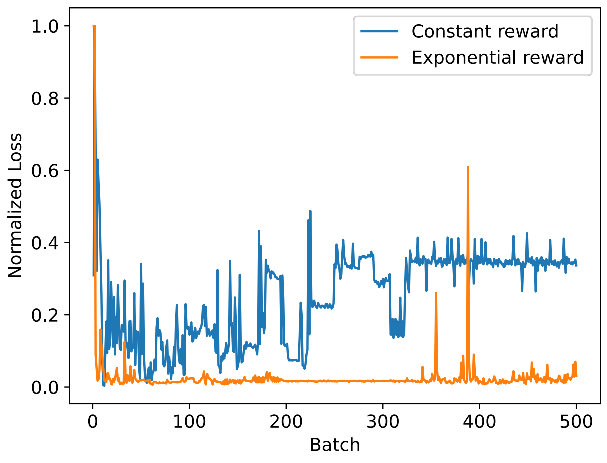
Overall, the stability of the transferred grasps behaved as expected. Episode characteristics such as acceleration and maximum duration, which were expected to decrease the initial grasp stability, were shown to have the expected negative correlations. Characteristics such as number of contacts, subject, and task, broadly had the same initial grasp stability, while stability varied based on object type and size. A manual review of the original grasp tasks suggests that differences in initial grasp duration within grasp object types (sphere, pyramid) were a result of the most common grasp used for the object. Although the initial grasp durations broadly behaved as expected, as shown in Fig. 4.2, 4.3, and 4.4, the divided distribution of grasp duration ratios in Fig. 4.1 with the most common relative stable duration bins of 0-10% and 90-100%, suggest that some other property of each grasp episode divides grasps that are stable for the entire duration (100%) from grasps that only maintain stability for a short time (10% or less). Although the contact constraint force balancing method did not completely stabilize all grasp episodes, the initial grasp stability results show that finger and contact pose information with the addition of contact distance values is sufficient to analytically infer reasonable initial grasp forces from the motion-captured initial poses.

4.3 PPO Training Results

Two training runs were performed using PPO and the multi-layer perceptron structure from Sec. 3.10.1. One run used a constant reward for each timestep (Eq. 3.16), while the other used a reward function that exponentially scaled with timestep and inversely with grasp magnitude (Eq. 3.17). The progression of the actor and critic networks was measured through the network losses (Fig. 4.5), specifically mean squared-error loss. The actor network used the estimated value to calculate loss, while the value network used the accumulated reward. Since each simulated grasp episode ran until either the grasp object was dropped or the maximum episode duration was reached, data from multiple grasp episodes were concatenated to form training data batches of fixed size. The number of batches trained on was used instead of epochs in the Fig. 4.5 loss graphs to keep the loss value relative to a constant size of observation and action sets, as well as a constant number of parameter updates.



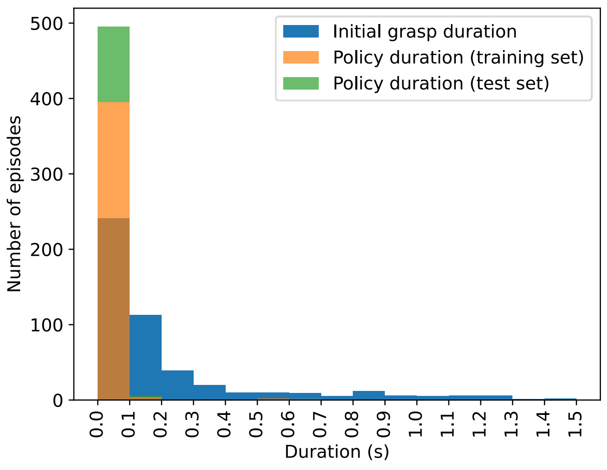
(a)



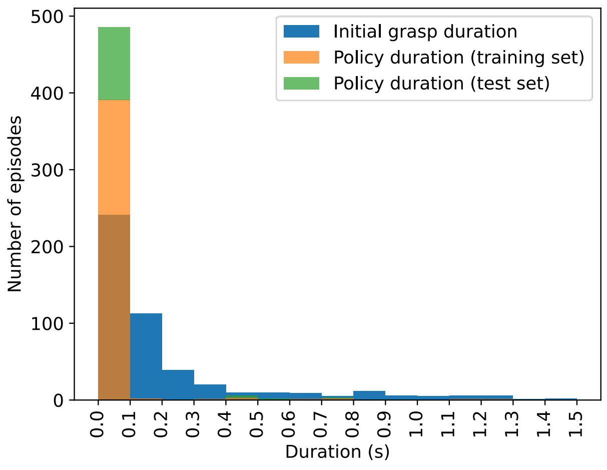
(b)

Figure 4.5. PPO training loss over training batches for exponential (Eq. 3.6), and constant (Eq. 3.5) reward functions: (a) Actor (policy) network loss, and (b) Critic (value estimation) network loss. The value estimation loss in Fig. 4.5(b) is normalized to allow both runs to be visible.

Figs. 4.6 and 4.7 are histograms of the number of episodes for different grasp durations, showing the direct comparison between the stable grasp durations under the trained policy compared to the baseline initial grasp stability after 500 training batch iterations. The leftward shift of almost all grasp episodes into the smallest duration bin, indicating all episodes lost grasp stability in less than 1 second, confirmed what the loss graphs in Fig. 4.5 indicated: the trained policies performed worse than the initial grasp duration baseline. Fig. 4.6a shows that the policy trained with the constant reward function immediately loses grasp stability, with 94% of the test episodes retaining stability for less than 0.1 s. The policy trained with the exponential reward function performed better with only 69% of the episodes retaining grasp stability for less than 0.1 s. However, the exponential reward policy still underperforms compared to the initial grasp baseline. In the test set, the average reduction in grasp stability time was -0.11 s over all episodes. A minority (23%) of the grasp episodes were stable longer under the learned policy, though the average increase in the grasp duration of 0.38 s was higher than the average duration reduction. The change of grasp duration in seconds for episodes not bounded by the minimum or maximum episode lengths was found to negatively correlate with the initial grasp duration with a Pearson correlation of -0.69 and p-value of 9.79e-11. This shows that grasps that were not held stable by the initial grasps benefited most from the learned policy.

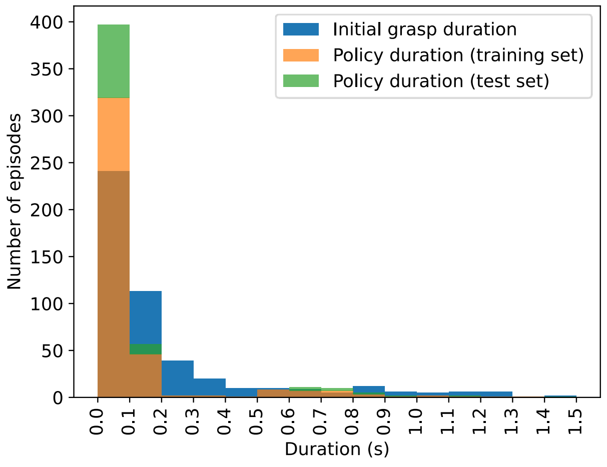


(a)

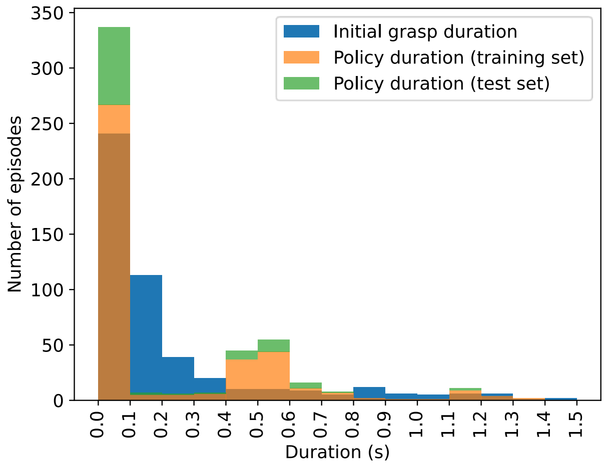


(b)

Figure 4.6. Histograms of the number of episodes for different grasp durations. Grasp duration distribution for policy trained with constant reward function (Eq. 3.16) for (a) before training, and (b) after 500 training batches. The histograms plot the policy episode durations on the training (orange) and test (green) additively stacked together. For comparison the baseline initial grasp duration (blue) is plotted to reveal the overlap (dark orange) between the trained policy durations and the baseline durations.



(a)



(b)

Figure 4.7. Histograms of the number of episodes for different grasp durations. Grasp duration distribution for policy trained with exponential reward function (Eq. 3.17) for: (a) before training, and (b) after 500 training batches.

Fig. 4.8 evaluates a policy trained using the exponential reward function without slip prediction inputs. The actor loss (Fig. 4.8a) generally increases over the number of batches trained, indicating that similar to the constant reward loss, the policy does not converge. The grasp stability duration (Fig. 4.8b) confirms this conclusion, as 99% of the test grasps were stable for less than 0.1 s.

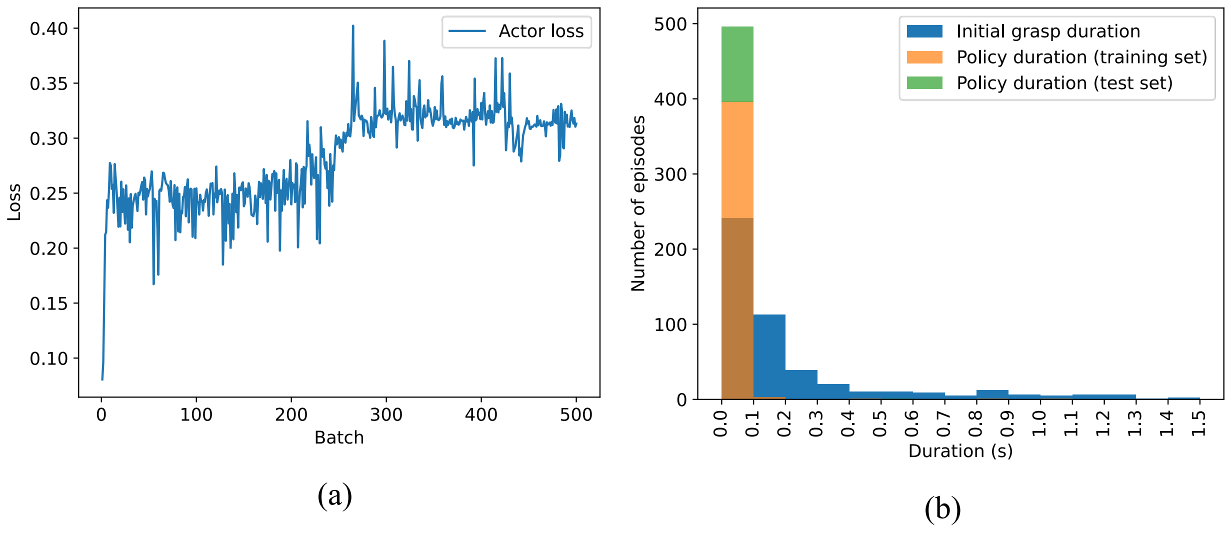


Figure 4.8. Grasp policy trained without slip prediction inputs using the exponential reward function: (a) actor loss, (b) grasp duration distribution after 500 training batches.

4.3.1 PPO Training Discussion

The training losses from Fig. 4.5a and Fig 4.8.a clearly show that in most cases the policy network did not learn a viable grasp stabilization policy. In a successful training session, the actor loss would be expected to decrease as the policy parameters are altered according to the policy loss. Although the critic loss (Fig. 4.5b) quickly fell from the initial value, the actor loss (Fig 4.5a) increased and remained relatively stable at a value higher than the initial loss. The only policy that had an actor loss that decreased slightly through training was the policy trained with the exponential reward and slip prediction inputs. However, the final actor loss was still above the initial starting value. The distributions of grasp stability durations in Figs 4.6, 4.7, and 4.8(b) demonstrate that only the policy with the exponential-reward with slip prediction inputs converged to a grasp stability solution, though it was a suboptimal solution unable to keep the episode grasps stable for as long as the initial grasp forces.

Although no trained policy exceeded the performance of the baseline initial grasp, there was a clear difference in the before-and-after training results for the policy using the exponential reward function (Eq. 3.17). The duration distribution after training in Fig. 4.7b clearly showed a minor improvement over the original duration distribution in Fig. 4.6a. In contrast, the policy trained with a constant time-based reward function (Eq. 3.16) showed no improvement through training, with almost all episodes (94%) losing stability immediately. The difference in training improvement between the reward functions suggests that incorporating a reward penalty for total grasp force magnitude prompted a poor stabilization policy to be learned.

The comparison between the exponential-reward policies trained with and without slip prediction inputs clearly demonstrates that the slip prediction inputs do benefit the learned policy, through to a very small degree. The policy without slip predictions did not converge, and like the failure of the force-closure method, discussed in Sec. 3.7.1.1 for balancing the initial states, a policy only using hand positions and forces could not maintain grasp stability in the simulation.

Chapter 5 Conclusion

5.1 Conclusion

The objective to transfer a set of human grasps into simulation was largely successful. The grasp poses were recreated with a simplified hand model, and the inferred grasp forces kept the initial grasp stable for the initial portion of most grasp episodes. The initial grasp durations negatively correlated to the maximum episode accelerations as expected, while not correlating to factors such as subject, object, task, or number of contacts.

The objective of predicting slip in simulation was also successful. Leveraging the coupled nature of contact distance and contact impedance of Mujoco's soft contact model, slip could be detected through direct measurement of contact-distance zero-crossing frequency, bypassing any need for contact surface information. This model of slip detection was extended to make slip predictions by branching the simulation and stepping forward in time until slip was detected. So far, no other research appears to have used this method to detect or predict contact slip in Mujoco.

The objective to learn a grasp stabilization policy based on slip prediction inputs produced mixed results. A comparison between policies trained with and without slip prediction inputs showed that the slip predictions did contribute to a better learned policy. However, the failure of the policy to converge on a solution superior to the initial grasp baseline indicates that slip prediction was not sufficient to replace visual information for learning grasp stability policies via a multi-layer perceptron in simulation. The difference in training outcome also depended on reward function, suggesting that slip prediction as a policy input may require specific adaptations through reward shaping. Other literature [26] has made similar conclusions on the importance of reward shaping for learning grasp policies in simulation. Ultimately, the performance of the grasp stabilization policies using slip prediction in simulation suggests that while slip predictions in simulation may assist in learning a grasp stabilization policy, the learned policy does not compare to policies trained on physical systems with real-world slip prediction or simulated systems trained with visual information.

5.2 Future Work

Expansion of the scope of the dataset imported into Mujoco to include more objects and grasps could create a better foundation for future grasp policy learning attempts. The most significant improvement would be to expand the number of imported objects. Since Mujoco's contact model restricts collision geometries to convex meshes only, this research only included convex grasp objects. Decomposing non-convex objects into convex subcomponents was attempted with the VHACD library; however, the grasp contacts were found to rapidly change location when close to the edges where the decomposed object pieces were joined. A possible method of addressing this issue would be to decompose the object surface based on proximity to the grasp fingers so that surfaces near contact points are preserved as single geometries.

It is important to consider that the function approximator structures used in this research for policy learning were simple and not specifically altered for processing inputs such as slip prediction. The chosen policy structure was a simple multi-layer perceptron, one of the basic forms of neural network. The same type of structure was used in other grasp simulation studies [15] [29] to combine processed visual pose features and directly sampled grasp-pose information (Sec. 2.3). However, the hypothesis that slip prediction could be processed together with proprioceptive information directly may not be true. Given the limited degree of learning displayed by the policy network, perhaps just as visual information was processed through specialized convolutional-neural-network layers before insertion into the multi-layer perceptron, slip prediction information may also require specialized processing before insertion into a multi-layer perceptron.

One initial expectation for the learned policies was for the policy parameters to converge in passing through the previous joint forces as outputs. This configuration would replicate the initial grasp solution by simply holding the initial calculated forces constant. However, this convergence did not occur, suggesting that the local minimum represented by a constant initial force policy was a form of unstable equilibrium not easily accessible though gradient descent, at least from the randomly initialized policy weights. An avenue of further exploration would be to initialize the policy in a pass-through state so that policy exploration starts immediately from the constant force policy.

References

[1] A. L. Ciancio, F. Cordella, R. Barone, R. A. Romeo, A. D. Bellingegni, R. Sacchetti, A. Davalli, G. Di Pino, F. Ranieri, V. Di Lazzaro, E. Guglielmelli and L. Zollo, "Control of Prosthetic Hands via the Peripheral Nervous System," *Frontiers in Neuroscience,* vol. 10, no. 116, 2016, doi: 10.3389/fnins.2016.00116.

[2] R. S. Johansson and J. R. Flanagan, "Tactile Sensory Control of Object Manipulation in Humans," in *E. Gardner, J. H. Kaas (Eds.), The Senses: A Comprehensive Reference, Vol. 6*, San Diego, Academic Press, 2008, pp. 67-86.

[3] H. P. Saal and S. J. Bensmaia, "Touch is a team effort: interplay of submodalities in cutaneous sensibility," *Trends in Neurosciences,* vol. 37, no. 12, pp. 689-697, 2014.

[4] G. Westling and R. S. Johansson, "Factors Influencing the Force Control During Precision Grip," *Experimental Brain Research,* vol. 53, pp. 277-284, 1984.

[5] R. A. Romeo and L. Zollo, "Methods and Sensors for Slip Detection in Robotics: A Survey," *IEEE Access,* vol. 8, pp. 73027-73050, 2020.

[6] A. C. Eliasson, H. Forssberg, K. Ikuta, I. Apel, G. Westling and R. Johansson, "Development of Human Precision Grip," *Experimental Brain Research,* vol. 106, no. 3, pp. 425-433, 1995.

[7] F. Cordella, A. L. Ciancio, R. Sacchetti, A. Davalli, A. G. Cutti, E. Guglielmelli and L. Zollo, "Literature Review on Needs of Upper Limb Prosthesis Users," *Frontiers in Neuroscience,* vol. 10, no. 209, 2016, doi: 10.3389/fnins.2016.00209.

[8] A. Spiers, J. Cochran, L. Resnik and A. Dollar, "Quantifying Prosthetic and Intact Limb Use in Upper Limb Amputees via Egocentric Video: An Unsupervised, At-Home Study," *IEEE Transactions on Medical Robotics and Bionics,* vol. 3, no. 2, pp. 463-484, 2021.

[9] S. Weiwei, F. Song, Z. Zhao, H. Gao, S. Cong and Z. Li, "Deep Learning Methods for Grasping Novel Objects Using Dextrous Hands," *IEEE Transactions on Cybernetics,* vol. 52, no. 5, pp. 2750-2762, 2022.

[10] C. Shi, D. Yang, J. Zhao and H. Liu, "Computer Vision-Based Grasp Pattern Recognition With Application to Myoelectric Control of Dexterous Hand Prosthesis," *IEEE*

Transactions on Neural Systems and Rehabilitation Engineering, vol. 28, no. 9, pp. 2090-2099, 2020.

[11] G. Ghazaei, A. Alameer, P. Degenaar, G. Morgan and K. Nazarpour, "Deep learning-based artificial vision for grasp classification in myoelectric hands," *Journal of Neural Engineering,* vol. 14, p. 036025, 2017.

[12] A. Billard and D. Kragic, "Trends and Challenges in Robot Manipulation," *Science,* vol. 364, no. 6446, p. eaat8414, 2019.

[13] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng and Zaremb, "Learning Dextrous In-hand Manipulation," *The International Journal of Robotics Research,* vol. 39, no. 1, pp. 3-20, 2020.

[14] F. Veiga, R. Akrour and J. Peters, "Hierarchical Tactile-Based Control Decomposition of Dexterous In-Hand Manipulation Tasks," *Frontiers in Robotics and AI,* vol. 7, p. 521448, 2020.

[15] C. Wang, Q. Zhang, Q. Tian, S. Li, X. Wang, D. Lane, Y. Petillot and S. Wang, "Learning Mobile Manipulation through Deep Reinforcement Learning," *Sensors,* vol. 20, no. 3, p. 939, 2020.

[16] R. Calandra, A. Owens, M. Upadhyaya, W. Yuan, J. Lin, E. H. Adelson and S. Levine, "The Feeling of Success: Does Touch Sensing Help Predict Grasp Outcomes?," in *Conference on Robotic Learning*, Mountain View, 2017, arXiv:1710.05512.

[17] J. Li, S. Dong and E. Adelson, "Slip Detection with Combined Tactile and Visual Information," in *IEEE International Conference on Robotics and Automation*, Brisbane, 2018, pp. 7772-7777, doi: 10.1109/ICRA.2018.8460495.

[18] J. A. Corrales, F. Torres and V. Perdereau, "Finger Readjustment Algorithm for Object Manipulation Based on Tactile Information," *International Journal of Advanced Robotic Systems,* vol. 10, pp. 1-9, 2013.

[19] K. Hang, "Hierarchical Fingertip Space: A Unified Framework for Grasp Planning and In-Hand Grasp Adaptation," *IEEE Transactions on Robotics,* vol. 32, no. 4, pp. 960-972, 2016.

[20] J. W. Jasper and N. F. Lepora, "Slip detection for grasp stabilization with a multi-fingered tactile robot hand," *IEEE Transactions on Robotics,* vol. 37, pp. 506-519, 2020.

[21] J. Kwiatkowski, D. Cockburn and V. Duchaine, "Grasp Stability Assessment Through the Fusion of Proprioception and Tactile Signals Using Convolutional Neural Networks," in *International Conference on Intelligent Robots and Systems*, Vancouver, 2017, pp. 286-292.

[22] R. Fernandez, I. Payo, A. S. Vazquez and J. Becedas, "Slip detection in Robotic Hands with Flexible Parts," in *ROBOT13: First Iberian Robotics conference* , Madrid, 2013, pp. 153-167.

[23] F. Veiga, B. B. Edin and J. Peters, "In-Hand Object Stabilization by Independent Finger Control," *arXiv preprint arXiv:1806.05031,* 2018.

[24] B. S. Zapata-Impata, "Tactile-Driven Grasp Stability and Slip Prediction," *Robotics,* vol. 8, no. 4, p. 85, 2019.

[25] F. R. Sanchez, S. Redmond, K. McGuinness and N. O'Connor, "Towards Advanced Robotic Manipulation," in *2022 Sixth IEEE International Conference on Robotic Computing*, Naples, 2022, pp. 302-305.

[26] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov and S. Levine, "Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations," *arXiv preprint arVix:1709.10087,* 2017.

[27] I. Radosavovic, X. Wang, L. Pinto and J. Malik, "State-Only Imitation Learning for Dexterous Manipulation," in *IEEE International Conference on Intelligent Robots and Systems*, Prague, 2021, pp. 7865-7871.

[28] D. Jain, A. Li, S. Singhal, A. Rajeswaran, V. Kumar and E. Todorov, "Learning Deep Visuomotor Policies for Dexterous Hand Manipulation," in *International Conference on Robotics and Automation*, Montreal, 2019, pp. 3636-3643.

[29] P. Mandikal and K. Grauman, "DexVIP: Learning Dexterous Grasping with Human Hand Pose Priors from Video," in *5th Conference on Robot Learning*, London, 2021, pp. 651-661.

[30] R. M. Murray, Z. Li and S. S. Sastry, "Multifinger Hand Kinematics," in *A mathematical introduction to robotic manipulation*, Boca Raton, CRC Press, 1994, pp. 211-229.

[31] M. A. Roa and R. Suárez, "Grasp quality measures: review and performance," *Autonomous Robots,* vol. 38, pp. 65-88, 2015.

[32] J. Weisz and P. K. Allen, "Pose Error Robust Grasping from Contact Wrench Space Metrics," in *IEEE International Conference on Robotics and Automation*, Saint Paul, 2012, pp. 557-562.

[33] M. A. Roa, R. Kõiva and C. Castellini, "Experimental Evaluation of Human Grasps Using a Sensorized Object," in *4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics*, Roma, 2012, pp. 1662-1668.

[34] C. Rubert, D. Kappler, J. Bohg and A. Morales, "Predicting grasp success in the real world - A study of quality metrics and human assessment," *Robotics and Autonomous Systems,* vol. 121, p. 103274, 2019.

[35] E. Todorov, T. Erez and Y. Tassa, "Mujoco: A Physics Engine for Model-Based Control," in *International Conference on Intelligent Robots and Systems*, Vilamoura, 2012, pp. 5026-5033.

[36] E. Coumans and Y. Bai, "Pybullet, a Python module for physics simulation for games, robotics, and machine learning," [Online]. Available: https://pybullet.org. [Accessed September 2021].

[37] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang and W. Zaremba, "OpenAI Gym," *arXiv preprint arXiv:1606.01540,* 2016.

[38] T. Erez, Y. Tassa and E. Todorov, "Simulation Tools for Model-Based Robotics: Comparison of Bullet, Havok, MujoCo, ODE and PhysX," in *2015 IEEE International Conference on Robotics and Automation*, Seattle, 2015, pp. 4397-4404.

[39] M. Körber, J. Lange, S. Rediske, S. Steinmann and R. Glück, "Comparing Popular Simulation Environments in the Scope of Robotics and Reinforcement Learning," *arXiv preprint arXiv:2013.04616,* 2021.

[40] "Deepmind Mujoco Purchase," Google DeepMind, October 2021. [Online]. Available: https://www.deepmind.com/blog/opening-up-a-physics-simulator-for-robotics. [Accessed September 2021].

[41] "Mujoco Documentation," Google DeepMind, [Online]. Available: https://mujoco.readthedocs.io/en/latest/overview.html. [Accessed September 2021].

[42] E. Todorov, "Convex and analytically-invertible dynamics with contacts and constraints: Theory and implementation in MuJoCo," in *2014 IEEE International Conference on Robotics and Automation*, Hong Kong, 2014, pp. 6054-6061.

[43] R. Featherstone, "Forward Dynamics - Inertia Matrix Methods," in *Rigid Body Dynamics Algorithms*, New York, Springer, 2008, pp. 104-110.

[44] E. Ramm, "Principles of Least Action and of Least Constraint," in *E. Stein (Ed.), The History of Theoretical, Material and Computational Mechanics - Mathematics Meets Mechanics and Engineering*, Heidelberg, Springer, 2014, pp. 23-43.

[45] V. Francois-Lavet, P. Henderson, R. Islam, M. G. Bellemare and J. Pinaeu, "An Introduction to Deep Reinforcement Learning," *Foundations and Trends in Machine Learning,* vol. 11, no. 3-4, pp. 219-354, 2018.

[46] K. Arulkumaran, M. P. Deisenroth, M. Brundage and A. A. Bharath, "Deep Reinforcement Learning: A Brief Survey," *IEEE Signal Processing Magazine,* vol. 34, no. 6, pp. 26-38, 2017.

[47] R. Calandra, A. Owens, D. Jayaraman, J. Lin, W. Yuan, J. Malik, E. H. Adelson and S. Levine, "More Than a Feeling: Learning to Grasp and Regrasp using Vision and Touch," *IEEE Robotics and Automation Letters,* vol. 3, no. 4, pp. 3300-3307, 2018.

[48] F. Veiga, "Stabilizing Novel Objects by Learning to Predict Tactile Slip," in *IEEE International Conference on Intelligent Robots and Systems*, Hamburg, 2015, pp. 5065-5072.

[49] S. Brahmbhatt, H. Cusuh, C. Kemp and J. Hays, "ContactDB: Analyzing and Predicting Grasp Contact via Thermal Imaging," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, 2019, pp. 8701-8711.

[50] J. Romero, D. Tzionas and M. J. Black, "Embodied Hands: Modeling and Capturing Hands and Bodies Together," *ACM Transactions on Graphics,* vol. 36, no. 6, pp. 1-17, 2017.

[51] G. Pavlakos, V. Choutas, N. Ghorbani, T. Bolkart, A. Osman, D. Tzionas and M. J. Black, "Expressive Body Capture: 3D Hands, Face, and Body from a Single Image," in *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, 2019, pp. 10967-10977.

[52] O. Taheri, N. Ghorbani, M. J. Black and D. Tzionas, "GRAB: A Dataset of Whole-Body Human Grasping of Objects," in *European Conference on Computer Vision (ECCV)*, Glasgow, 2020, pp. 581-600.

[53] K. Mamou, "V-HACD," 2014. [Online]. Available: https://github.com/kmammou/v-hacd. [Accessed August 2022].

[54] M. Buss, H. Hashimoto and J. B. Moore, "Dextrous Hand Grasping Force Optimization," *IEEE Transactions on Robotics and Automation,* vol. 12, no. 3, pp. 406-418, 1996.

[55] J. A. Nelder and R. Mead, "A Simplex Method for Function Minimization," *Computer Journal,* vol. 7, pp. 308-313, 1965.

[56] D. Eberly, "Quaternion Algebra and Calculus," 2 March 1999. [Online]. Available: https://www.geometrictools.com/Documentation/Quaternions.pdf. [Accessed July 2022].

[57] J. Schulman, F. Wolski, P. Dhariwal, A. Radford and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv preprint arVix:1707.06347,* 2017.