



DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Robotics, Cognition, Intelligence

**Tactile Grasp Refinement using Deep
Reinforcement Learning and Analytic Grasp
Stability Metrics**

Alexander Koenig





DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Robotics, Cognition, Intelligence

**Tactile Grasp Refinement using Deep
Reinforcement Learning and Analytic Grasp
Stability Metrics**

**Taktile Griffverfeinerung mit Deep
Reinforcement Learning und analytischen
Griffstabilitätsmetriken**

Author: Alexander Koenig
Supervisor: PD Dr. Tobias Lasser
Advisor: Prof. Dr. Björn Menze
Submission Date: 15.10.2021



I confirm that this master's thesis is my own work and I have documented all sources and material used.

Munich, 15.10.2021

Alexander Koenig

Acknowledgments

Firstly, I would like to thank Robert Howe, Professor of Engineering, for hosting my research stay in the Harvard Biorobotics Lab. I specifically want to thank him for my fellowship at the Harvard John A. Paulson School of Engineering and Applied Sciences (SEAS). Furthermore, I would like to thank Zixi Liu, a Ph.D. student in the lab, for always supporting me when there were any urgent questions regarding the project or life in Boston. I am also grateful that Lucas Janson, Assistant Professor of Statistics, joined the collaboration and provided valuable insights regarding the machine learning aspects of the project. I enjoyed working closely with all of you, and thank you for the freedom you gave me to steer the project in the directions that interested me the most while also providing me with strong guidance along the way. I especially appreciated the quick feedback iterations during the final stages of the submission of our paper [38].

I would also like to thank the administrative staff at Harvard for working hard to get me a visa and a National Interest Exception which allowed me to move to the United States even in the face of the travel ban caused by the ongoing COVID-19 pandemic. I am incredibly grateful that I could conduct my research on-site in Boston from January through July 2021. My time in the US has broadened my horizon, furthered my knowledge of research life at top academic institutions, and allowed me to meet other like-minded robotics researchers through informal chats in the lab. In this regard, I would like to thank the other Ph.D. students of the Biorobotics lab Buse Aktas and Sebastian Roubert Martinez, for welcoming me to the group with open arms and for organizing memorable lab activities such as kayaking on the Charles River.

Furthermore, I would like to thank Professor Dr. Björn Menze and PD Dr. Tobias Lasser for agreeing to supervise this project at my home university, the Technical University of Munich. I appreciate that you supported my plans to conduct research for my master thesis abroad and that the organizational concerns were so unbureaucratic. My gratitude also goes towards my parents and family, who supported me tirelessly throughout my studies. Finally, I would like to mention two fellow robotics students from Munich. I am grateful to Julia Pfeiffer, who stroke the match and inspired me to apply for master theses in the US. Federico Gabriel Wyrwal supported me regarding questions on reinforcement learning and provided me with valuable insights from his extensive experience with this technique.

Lastly, I would like to thank the German Academic Exchange Service (DAAD) for supporting my research stay with a full scholarship through the IFI (German: Internationale Forschungsaufenthalte für Informatikerinnen und Informatiker) program. Furthermore, I am grateful for the PROMOS scholarship which the Technical University of Munich granted me. My research was also partly funded by the United States National Science Foundation (NSF) under Grant No. IIS-1924984, for which I am very thankful. My scholarships (IFI, PROMOS, and NSF) did not overlap.

Abstract

Robotic grasping systems that rely on open-loop controllers often fail when a grasp is subject to unforeseen calibration errors. Controllers should compensate for these errors by continuously integrating sensory feedback and locally refining the grasp to obtain a more stable grip. While it is difficult to refine grasps through vision alone due to occlusion, there is excellent potential for closing the feedback loop in robotic grasping with tactile sensing.

Reinforcement Learning (RL) is becoming an increasingly popular technique for robotic grasping, and reward functions are at the core of every RL algorithm. Rewards of most state-of-the-art grasping algorithms are complex and hand-crafted functions that do not rely on well-justified physical models from grasp analysis. In our first contribution, we demonstrate the potential of using analytic grasp stability metrics as rewards for RL-based tactile grasp refinement controllers. Our algorithms receive only tactile and joint position data from a three-fingered hand and refine the grasp with iterative updates to the robot's wrist pose and finger positions. In large-scale simulated experiments, we find that the best performing reward functions combine metrics concerning finger placement with measures based on current contact forces. This reward framework achieves average success rates of 95.4% for cuboids, 93.1% for cylinders, and 62.3% for spheres across wrist position errors between 0 and 7 cm and rotational errors between 0 and 14 deg. We outperform a binary reward baseline which related works often employ by 42.9%.

In a second contribution, we study the relation between tactile sensing resolution and grasp refinement success. We find that algorithms trained with accurate tactile feedback on contact positions, normals, and forces perform up to 6.6% better than a baseline which only processes proprioceptive information about joint positions. However, even these algorithms, which assume no tactile feedback in their input, reach adequate success rates of 90.7% for cuboids and 87.6% for cylinders across significant calibration errors of up to 7 cm and 14 deg when trained with expressive contact-based reward functions. This result is valuable for robotic hand design since accurate tactile sensors are expensive and delicate hardware items. We make the source code used in this thesis¹ publicly available² and provide supplementary video material³.

¹Partial results of the presented work were submitted to a robotics conference as a publication [38]. Any references to the paper are clearly indicated.

²https://github.com/axkoenig/grasp_refinement

³<https://www.youtube.com/watch?v=9Bg8ZEAE0GI>

Nomenclature

(ξ, η, ζ) Rotation in Euclidean space

(x, y, z) Position in Euclidean space

α Temperature parameter in the soft actor-critic (SAC) framework

$\bar{\mathbf{f}}_{i,cur} \in \mathbb{R}^3$ Tangential force margin of $\mathbf{f}_{i,cur}$ to the friction cone

$\bar{\mathbf{f}}_{i,task} \in \mathbb{R}^3$ Tangential force margin of $\mathbf{f}_{i,task}$ to the friction cone

$\boldsymbol{\tau} \in \mathbb{R}^3$ Torque vector

$\boldsymbol{\tau}_i = \mathbf{r}_i \times \mathbf{f}_i \in \mathbb{R}^3$ Torque resulting from \mathbf{f}_i

$\boldsymbol{\tau}_{i,j} \in \mathbb{R}^3$ The torque resulting from friction cone edge $\mathbf{f}_{i,j}$

$\mathbf{f} \in \mathbb{R}^3$ Force vector

$\mathbf{f}_g \in \mathbb{R}^3$ Object weight

$\mathbf{f}_i \in \mathbb{R}^3$ Contact force at contact i

$\mathbf{f}_{i,add} \in \mathbb{R}^3$ Expected additional contact force at contact i to compensate a task wrench \mathbf{w}_t

$\mathbf{f}_{i,cur} \in \mathbb{R}^3$ Current contact force at contact i

$\mathbf{f}_{i,j} \in \mathbb{R}^3$ The j -th friction cone edge at contact i

$\mathbf{f}_{i,n} \in \mathbb{R}^3$ Normal component of contact force \mathbf{f}_i

$\mathbf{f}_{i,task} \in \mathbb{R}^3$ Expected total contact force at contact i to compensate a task wrench \mathbf{w}_t

$\mathbf{f}_{i,t} \in \mathbb{R}^3$ Tangential component of contact force \mathbf{f}_i

$\mathbf{n}_i \in \mathbb{R}^3$ Contact normal at \mathbf{p}_i

$\mathbf{p}_c \in \mathbb{R}^3$ Object center of mass

$\mathbf{p}_i \in \mathbb{R}^3$ Contact point on object

$\mathbf{r}_i \in \mathbb{R}^3$ Vector pointing from \mathbf{p}_c to \mathbf{p}_i

$\mathbf{w} = (\mathbf{f} \quad \boldsymbol{\tau})^T \in \mathbb{R}^6$ Wrench vector

$\mathbf{w}_t \in D$	Anticipated task wrench
δ_{cur}	Grasp quality based on current contact forces
δ_{task}	Expected grasp quality during task execution
ϵ_τ	Largest-minimum resisted torque
ϵ_f	Largest-minimum resisted force
ϵ_w	Largest-minimum resisted wrench
$\gamma \in [0, 1]$	Discount factor in a Markov Decision Process (MDP)
$\mathbf{F} \in \mathbb{R}^{3 \times n_c}$	Matrix containing n_c contact forces \mathbf{f}_i of a grasp
$\mathbf{G} \in \mathbb{R}^{6 \times n_c \cdot n_{\lambda_i}}$	Grasp matrix where n_{λ_i} depends on the choice of the contact model
$\mathbf{N} \in \mathbb{R}^{3 \times n_c}$	Matrix containing n_c contact normals \mathbf{n}_i of a grasp
\mathcal{A}	Set of all actions $a_t \in \mathcal{A}$ in an MDP
$\mathcal{H}(\cdot)$	Shannon entropy of a random variable
\mathcal{S}	Set of all states $s_t \in \mathcal{S}$ in an MDP
\mathcal{W}	Convex set of wrenches a grasp can apply to an object (i.e., the Grasp Wrench Space (GWS))
\mathcal{W}_τ	Convex set of torques a grasp can apply to an object
\mathcal{W}_f	Convex set of forces a grasp can apply to an object
μ	Coefficient of friction
π_*	Optimal policy
a_t	Action at step t in an MDP. If action is a vector we write \mathbf{a}_t .
$D = \{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_q\}$	Task definition including q wrenches that the grasp must resist
G_t	Return from step t onwards in an MDP
m	Number of edges spanning the approximated friction cone
n_c	Number of contacts of a grasp
$Q(s_t, \mathbf{a}_t)$	Soft action-value function in the SAC framework
$q_*(s, a)$	Optimal action-value function
$q_\pi(s, a)$	Action-value function of a policy

-
- $r(s_t)$ Reward function
- r_t Reward at step t in an MDP
- s_t State at step t in an MDP. If state is a vector we write s_t .
- $t \in \mathbb{N}_0$ Time step in an MDP
- T Final time step t in an MDP
- $V(s_t)$ Soft state-value function in the SAC framework
- $v_*(s)$ Optimal state-value function
- $v_\pi(s)$ State-value function of a policy
- ${}^A_B T \in \mathbb{R}^{4 \times 4}$ Homogeneous transformation matrix from coordinate frame $\{A\}$ to frame $\{B\}$

Acronyms

3D three-dimensional. 24, 35, 39, 53, 63

CPU central processing unit. 61

DART Dynamic Animation and Robotics Toolkit. 20–23, 61

DOF degrees of freedom. 4, 18, 21, 28, 29, 52, 53, 59, 70, 76

GUI Graphical User Interface. xiii, 19, 20

GWS Grasp Wrench Space. viii, 24, 37–39, 41, 44

HF Hard Finger. 46, 47

IMU Inertial Measurement Units. 20

KL Kullback–Leibler. 16, 17

LCP Linear Complementarity Problem. 21, 31, 68

MDP Markov Decision Process. viii, ix, 8, 9, 13

ML Machine Learning. xiii, 6, 7, 18, 27, 28

ODE Open Dynamics Engine. 20–22

PCA Principal Component Analysis. 28

PD proportional–derivative. 23, 51

PPO Proximal Policy Optimization. 14, 30, 34

PwoF Point Contact without Friction. 46

REPS Relative Entropy Policy Search. 28, 34

RL Reinforcement Learning. v, xiv, 4–7, 11, 13–15, 22, 23, 28–30, 33, 35, 48–50, 59, 61, 67, 69, 70, 75, 76

ROS Robot Operating System. 20, 23–25

SAC soft actor-critic. vii–ix, xv, 7, 14, 15, 51, 53, 54, 61, 68

SF Soft Finger. 46

SL Supervised Learning. 7, 10, 29, 34

SVM Support Vector Machine. 28

TCP tool center point. 50, 53

TRPO Trust Region Policy Optimization. 14, 30, 34

TWS Task Wrench Space. 44

UL Unsupervised Learning. 7

List of Figures

1.1	A person uses tactile feedback to refine a grasp (adapted from [26]).	1
1.2	A typical industrial grasping setup with coordinate frames.	2
1.3	A keyword search on Google Scholar for "robot grasp reinforcement learning" indicates that the subject is becoming increasingly popular (results as of October 4, 2021).	4
2.1	Taxonomy of Machine Learning (ML) techniques (adapted from [79]).	7
2.2	Action-perception feedback loop for RL systems (adapted from [90]).	8
2.3	Images of the ReFlex TakkTile robotic hand.	18
2.4	Barometric pressure sensors integrated in the robotic fingers.	19
2.5	Gazebo Graphical User Interface (GUI) with simulated ReFlex TakkTile and a cylinder.	20
2.6	Robotic finger with two revolute joints and contact point p	22
2.7	System design of ReFlex Simulation Stack.	22
3.1	Control paradigms for closed-loop grasping controllers.	27
3.2	Input data for closed-loop grasping controllers.	28
4.1	Friction cone at contact p_i with $m = 5$ and the contact frame $\{n_i, t_i, o_i\}$. The vectors $f_{i,j}$ span the edges of the approximated friction cone. The force f_i with its normal $f_{i,n}$ and tangential component $f_{i,t}$ lies inside the friction cone (graphic adapted from [34]).	36
4.2	Planar example of a grasp and stability analysis with ϵ_f (image source Koenig et al. [38]).	39
4.3	Analyzing a grasp and calculating $f_{i,j}$ and $\tau_{i,j}$ on which \mathcal{W}_f and \mathcal{W}_τ are based.	40
4.4	Grasp configurations with calculated grasp quality metrics.	41
4.5	Friction cones, current contact forces $f_{i,cur}$, contact normals n_i and tangential force margins $\bar{f}_{i,cur}$ used to compute δ_{cur} . Note that we consider the true friction cone now, and not the approximated one as in [19] (image source Koenig et al. [38]).	42
4.6	Friction cones and anticipated task contact forces $f_{i,task}$ used to compute δ_{task} (image adapted from Koenig et al. [38]).	45
4.7	Information flow in the grasping pipeline. The grey arrows indicate data that only flows while training the algorithm. Blue arrows indicate information that flows during training <i>and</i> testing. We also show update frequencies and programming languages (ReFlex image source [77]).	49

4.8	Overview of the RL algorithm. In (A), we generate a new object wrist error combination (O, E) . Afterward, we start the (B) grasp refinement episode using different reward functions (image adapted from Koenig et al. [38]).	50
4.9	Minimum and maximum object sizes. Spheres are placed on a concave mount to prevent rolling (image source Koenig et al. [38]).	52
4.10	Left: wrist error case A (i.e., the ground-truth grasping pose). Right: wrist error case H (i.e., maximum wrist error) (image source Koenig et al. [38]).	53
4.11	Train results for reward frameworks defined in Figure 4.8 (includes data from [38]).	55
4.12	Test results for reward frameworks defined in Figure 4.8 (includes data from [38]).	57
5.1	Train results for contact frameworks defined in Table 5.1 (includes data from [38]).	65
5.2	Test results for contact frameworks defined in Table 5.1 (includes data from [38]).	66
5.3	A grasp with two contact points p_1 and p_2 and friction cones at each contact. The vector $-\mathbf{f}_g$ must balance the object's weight vector \mathbf{f}_g . This is the configuration in which the sum of the contact force magnitudes $\sum_i^{n_c} \ \mathbf{f}_i\ $ is minimized.	72

List of Tables

3.1	Overview of related works using <i>data-driven</i> approaches and <i>tactile</i> information. <i>Reality</i> means that experiments were conducted on physical robot hardware, <i>Simulation</i> refers to algorithms trained and tested only in simulation, and <i>Both</i> means that algorithms were trained in simulation and deployed to real hardware (includes data from Koenig et al. [38, p. 2]).	34
4.1	Contact models and respective selection matrices (partly adapted from [73]).	46
4.2	Grasp quality metrics and the information required to compute them.	48
4.3	Object properties and sampling ranges.	52
4.4	Wrist error cases (source Koenig et al. [38]).	52
4.5	Hyper-parameters for SAC [23] algorithm.	54
4.6	Results of t-test for reward comparison. The mean of framework x is μ_x and ' ≈ 0.0 ' means value was below machine precision (includes data from [38]).	58
5.1	Inputs for different contact sensing frameworks.	63
5.2	Results of t-test for contact sensing comparison (includes data from [38]).	67

Contents

Acknowledgments	iii
Abstract	v
Nomenclature	vii
Acronyms	xi
List of Figures	xiii
List of Tables	xv
1 Introduction	1
1.1 The Human Perspective	1
1.2 The Robot Perspective	2
1.3 Research Questions	4
1.4 Outline	6
2 Background	7
2.1 Reinforcement Learning	7
2.1.1 Taxonomy	7
2.1.2 The Action-Perception Loop	7
2.1.3 Markov Decision Processes	8
2.1.4 Rewards and Returns	9
2.1.5 Policies and Value Functions	10
2.1.6 Optimal Policies and Optimal Value Functions	12
2.1.7 Soft Actor-Critics	14
2.2 Simulating Robotic Grasping	18
2.2.1 Motivation	18
2.2.2 Robotic Hand	18
2.2.3 Simulation Software	19
2.2.4 The ReFlex Simulation Stack	22
3 Related Works	27
3.1 Taxonomy	27
3.2 Data-Driven Tactile Grasp Refinement	28

3.3 Comparison	31
4 Reward Design and Grasp Refinement	35
4.1 Analytic Grasp Stability Metrics	35
4.1.1 Largest-Minimum Resisted Wrench	35
4.1.2 Measuring Resistance to Pure Forces and Torques	38
4.1.3 Force-Agnostic Grasp Stability Metrics	41
4.1.4 Summary	48
4.2 Experimental Setup	48
4.2.1 Algorithm Overview	48
4.2.2 Training Dataset	51
4.2.3 Test Dataset	52
4.2.4 State and Action Space	52
4.2.5 Hyper-parameters	53
4.3 Results	54
4.3.1 Training	54
4.3.2 Testing	56
4.4 Discussion	59
5 Contact Sensing and Grasp Refinement	63
5.1 Experimental Setup	63
5.2 Results	64
5.2.1 Training	64
5.2.2 Testing	66
5.3 Discussion	67
6 Conclusion	75
6.1 Summary	75
6.2 Future Work	75
Bibliography	77

1 Introduction

This chapter introduces the topic of robotic grasp refinement and highlights why it is a pressing issue in robotics. Firstly, we approach the subject by exploring the human perspective on grasp refinement and start with a high-level example to illustrate the topic of this work. Consequently, we draw parallels between the human and robot perspectives by investigating when and how tactile grasp refinement is relevant in modern robotic grasping systems. Further, we define concrete research goals and summarize our contributions. Finally, we outline the structure of this thesis.

1.1 The Human Perspective

Grasping is an innate human capability [84]. The control and sensory processing of our hands occupy a disproportionately large amount of the brain's motor and sensory cortex compared to other body parts [69], which highlights the importance of manipulation and grasping for the survival of the human species. Grasping and manipulation enable us to eat and drink, use tools, and understand our surroundings and are therefore crucial for the advancement of humankind.

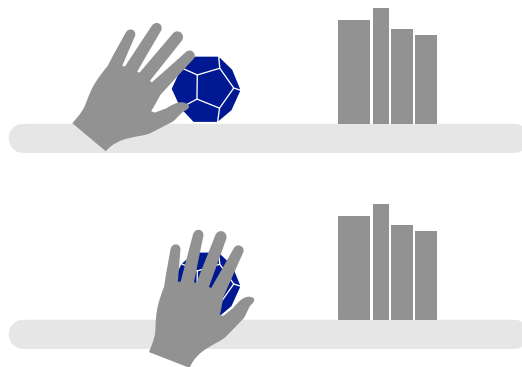


Figure 1.1: A person uses tactile feedback to refine a grasp (adapted from [26]).

Humans heavily rely on tactile sensing when grasping and manipulating objects [31]. We define *tactile grasp refinement* as the process of using sensory information about contact events to improve the stability of a grasp. Figure 1.1 depicts an ordinary situation [26] in which we humans perform tactile grasp refinement. We roughly remember the location of an object that we would like to grasp, but another object (e.g., a shelf mounted high up on the wall) occludes the object of interest. Hence, we cannot use our vision to accurately determine the object's pose and, consequently, use our best guess for an initial grasp attempt. Afterward, we process our

hand's tactile feedback to intuitively decide which direction to move our palm and fingers to get an improved grip on the object.

From the example in Figure 1.1, we identified object occlusion as one primary source of error that humans learned to compensate using tactile feedback. Another common situation where our tactile sense is vital is grasping in the dark. The sources of errors that need to be compensated by iterative refinements through robots are often similar. As we try to move robots from well-controlled factory environments into unstructured and human environments, we must aim to develop similar adaptive grasping strategies that humans evolved throughout evolution. The dexterity and versatility of the human hand are by no means matched by modern autonomous robotic grasping and manipulation systems. Therefore, it is essential to address this fundamental challenge and solve this open research problem in the quest for truly autonomous robots.

1.2 The Robot Perspective

Let us first discuss the hardware setup of a typical robotic grasping system to define the problem more accurately and identify potential sources of error in this system. Figure 1.2 shows a grasping setup that is standard in industrial contexts such as warehouses or factory floors. A typical task such systems perform is bin-picking: moving an object from one bin into another. There are usually four components with four respective coordinate frames in such a gripping robot. Firstly, there is an object of interest with frame $\{O\}$. Most grasping systems feature one or more cameras $\{C\}$ to track the object in space. Finally, there is a robotic arm with a base frame $\{B\}$ and a robotic hand or gripping tool $\{T\}$ mounted on the wrist of the robotic arm.

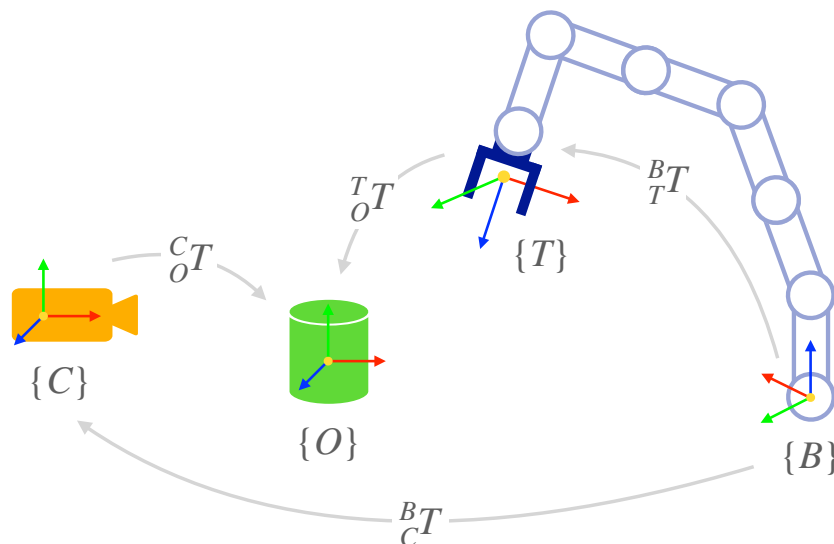


Figure 1.2: A typical industrial grasping setup with coordinate frames.

A central question in modern robotics research is how to precisely and robustly control grasping systems as depicted in Figure 1.2. Traditionally, robotic grasping is separated into a planning

and an execution step [56]. In the *planning* step, computer vision algorithms infer the object's geometry and pose ${}^C_O T$ relative to the camera frame $\{C\}$. A grasp synthesis algorithm then determines an appropriate grasping pose ${}^T_O T$ for the tool with respect to the object that satisfies the constraints at hand (e.g., object geometry, expected grasp stability, reachability, execution time and singularity avoidance). Equation (1.1) shows how the coordinate transformations from Figure 1.2 are connected. In equation (1.2), we obtain the desired tool pose in the manipulator's coordinate frame ${}^B_T T$ by expressing the grasping pose ${}^T_O T$ in the manipulator's coordinate frame via the transforms ${}^B_C T$ and ${}^C_O T$. In the *execution* step, an open-loop controller executes the computed trajectory and thereby moves the robotic hand to the desired pose ${}^B_T T$ which consequently grasps the object.

$${}^B_T {}^T_O T = {}^B_C {}^C_O T \quad (1.1)$$

$${}^B_T T = {}^B_C {}^C_O T ({}^T_O T)^{-1} \quad (1.2)$$

The aforementioned open-loop controllers solely execute a computed trajectory and do not process sensory information online to update the grasp. Therefore, such controllers can not react to calibration errors, which reduces the robustness of open-loop grasping algorithms. Calibration errors can occur in any of the introduced coordinate transformations, as explained below.

- *Perception errors in ${}^C_O T$* : Perception algorithms often struggle to determine the correct object pose and geometry. Reasons for this may be inaccurate camera calibration, noise induced by insufficient lighting conditions, an occluded object (like in Figure 1.1), or limitations on the object pose prediction algorithm.
- *Grasp prediction errors in ${}^T_O T$* : The planning algorithm aims to find a suitable tool pose $\{T\}$ relative to the object $\{O\}$ to grasp it. Errors in this transform can arise due to insufficient object data (e.g., an incomplete geometry model) or poorly performing grasp synthesis algorithms. Furthermore, open-loop grasping algorithms fail if the object moves while executing the grasp, e.g., due to unwanted interactions with the environment or the robotic hand.
- *Registration errors in ${}^B_C T$* : The camera coordinate frame $\{C\}$ and the base frame $\{B\}$ of the manipulator must be co-registered through a known and fixed arrangement of reference points such as visual fiducials in the robot's base frame $\{B\}$. An accurate estimate of this transform ${}^B_C T$ is required to represent the camera's measurements in the robot's frame of reference. The calibration accuracy depends on various parameters such as the convergence of iterative calibration methods [85], the calibration precision of the intrinsic camera parameters, or the discrepancy between the physical calibration phantom and its model.
- *Execution errors in ${}^B_T T$* : Equation (1.2) shows that ${}^B_T T$ is a function of the above transforms ${}^B_C T$, ${}^C_O T$ and ${}^T_O T$. Hence, any errors in these matrices will multiply and result in an overall larger misalignment in the desired grasp pose ${}^B_T T$. Additionally, this desired tool pose ${}^B_T T$

may not always be reached due to inaccuracies caused by mechanical tolerances, which add up along the kinematic chain. Hence, this problem is especially prominent for manipulators with many degrees of freedom (DOF) and large links. A proper manipulator calibration mitigates these errors. However, in practice, robots are not always accurately calibrated, and their calibration parameters may change over time due to wear.

Some approaches use *computer vision* to compensate for such errors at execution time. The work by Morrison et al. [61] is a prime example of vision-assisted closed-loop robotic grasping. They update the grasp based on depth images from a wrist-mounted camera while executing a previously synthesized grasp candidate. They report that the depth sensor struggles with reflective and black objects, worsening the algorithm's performance. More importantly, the depth camera reports no results once the object-to-sensor distance falls below a specific limit. Hence, the algorithm can not further refine the grasp once the fingertips are closer than 7cm to the object. Other challenges for vision-based closed-loop controllers are occlusion through items in the camera's line of sight or self-occlusion (i.e., the side of the object not facing the camera system is not perceivable).

Since vision-assisted reactive grasp control is prone to the above shortcomings, there is excellent potential for *tactile sensing* in closed-loop robotic grasp refinement. Tactile sensors are either integrated into the robotic hand's fingers and palm (e.g., [65]) or are retro-fitted to a non-touch-sensitive hand (e.g., [76]). Unlike vision systems, touch sensors can perceive local contact events and are therefore not susceptible to occlusion.

1.3 Research Questions

Recent innovations in the field of RL attract considerable interest from the robotics community. Robotic grasping in particular benefits from the advancements in RL and Figure 1.3 highlights that the subject is a trending matter.

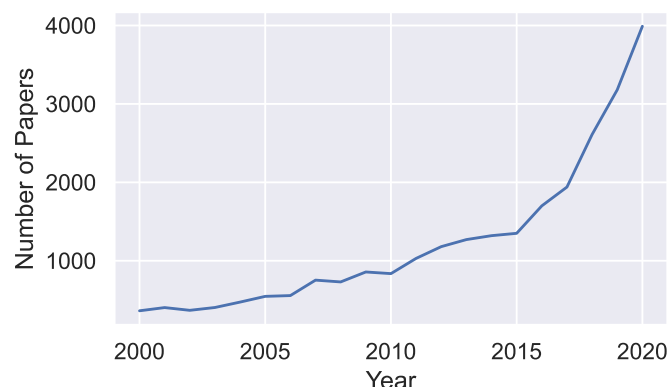


Figure 1.3: A keyword search on Google Scholar for "robot grasp reinforcement learning" indicates that the subject is becoming increasingly popular (results as of October 4, 2021).

Several related works [8, 29, 56, 99] use tactile information to train RL algorithms to grasp. Some of these works make unrealistic assumptions for the algorithm’s input signal. For example, they assume perfect knowledge about the object pose [56] or object geometry [29]. However, such data is commonly only available in simulated environments and not in the real world. In the real world complex and expensive vision systems would be needed to obtain this data at high accuracy. It should therefore be investigated whether grasp refinement algorithms are trainable solely based on information that is intrinsically available from a robotic hand, such as joint positions and contact data.

Besides the algorithm’s input, the reward function is a core component of every RL algorithm [89]. The reward functions of state-of-the-art approaches [8, 29, 56, 99] often consist of hand-crafted cues for grasp success, such as the number of contacts [29, 56]. While such cues may make sense at first glance, they have no well-justified relation with grasp stability: a grasp with many contacts can easily fail if the contact forces are insufficient to perform the desired task. The robotics community does not yet leverage the rich body of research on grasp analysis and grasp stability metrics [78] in the context of closed-loop tactile grasping with RL. Hence, in our first research question **RQ 1**, we empirically investigate the potential of analytic grasp stability metrics as sound optimization objectives for RL algorithms while also relaxing the assumptions on the algorithm’s inputs signals compared to related works.

RQ 1: Which analytic grasp stability metrics are the most expressive reward functions for RL algorithms that refine grasps on three-fingered robotic hands receiving only tactile and joint position data as input?

Recent works revealed that contact sensing improves the success rates of RL grasping [56] and in-hand manipulation algorithms [53, 54]. However, these studies [53, 54] also report that the contact force resolution is not directly correlated with the algorithm’s performance. Melnik et al. [53, 54] found that RL algorithms trained with accurate normal force data perform approximately equally well as ones that only receive binary contact signals. This result is counterintuitive since accurate normal force readings are without a doubt relevant for the execution of the studies tasks in [53, 54] (e.g., in-hand block or pen rotation). To understand this discrepancy more deeply, we study the effect of contact sensing resolution in our second contribution and test if we reach similar results as in [53, 54] in our grasp refinement experiment. There is a second reason to study the effect of contact sensor resolution on grasp refinement success. Tactile sensors are delicate and expensive hardware items. Therefore, it is interesting to investigate whether highly accurate contact sensing is at all required to perform tactile grasp refinement. Hence, our results can be a valuable guide towards robotic hand design.

RQ 2: What is the relation between contact sensing resolution and tactile grasp refinement success?

1.4 Outline

The remainder of this thesis is structured as follows. In chapter 2 we introduce the core concepts of deep RL and review the actor-critic algorithm used in this work. Furthermore, we give an overview of the developed simulation environment through which we generate data to train to RL algorithms. In chapter 3 we summarize and compare related works that also process tactile data with ML. Chapter 4 includes an in-depth explanation of the analytic grasp quality metrics used in this work and discusses the experimental setup as well as the empirical results to answer **RQ 1**. Chapter 5 analyzes our second research objective **RQ 2**. Last but not least, the conclusion in chapter 6 summarizes this research effort and provides ideas for future work.

2 Background

This chapter gives a brief introduction to RL. It summarizes the fundamentals of RL and introduces some vital notation as presented in the works by Sutton and Barto [90] and Russel and Norvig [79]. We will also discuss the SAC [23] algorithm used in this project. Finally, we introduce the simulation framework that we specifically designed for the experiments described in later chapters.

2.1 Reinforcement Learning

2.1.1 Taxonomy

Figure 2.1 shows that RL, along with Supervised Learning (SL) and Unsupervised Learning (UL), is a sub-field of ML.

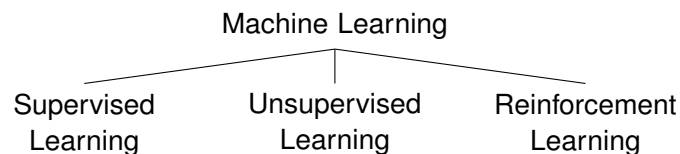


Figure 2.1: Taxonomy of ML techniques (adapted from [79]).

There is a clear distinction between these three fields: *SL* is the process of using labeled training samples to infer a function that maps an input to the desired output [79]. A typical SL problem is classifying if an image shows a car or a truck. *UL* is the problem of identifying a hidden structure within a dataset without using explicit labels [79]. An example of UL is distinguishing three types of tissue in medical images by finding cluster centers in their intensity histogram. *RL* is the process of goal-directed learning by interaction with an environment [90]. An example of RL could be the controller of a robot lawnmower that must avoid running out of battery while mowing as much grass as possible. While there are apparent differences between these methods, a common goal of all ML algorithms is to perform well on unseen data.

2.1.2 The Action-Perception Loop

Sutton and Barto [90] define the two fundamental entities in a RL system: the *agent* and the *environment*. The agent is the learning decision-maker that interacts with the environment at time t through actions a_t where $t \in \mathbb{N}_0$ [90]. The environment responds to these actions and produces the successor state of the system s_{t+1} along with a scalar reward signal $r_{t+1} \in \mathbb{R}$ [90].

The agent's goal is to select actions based on the observation s_{t+1} to maximize the cumulative reward it receives from the environment [90]. Figure 2.2 visualizes the interaction between the agent and the environment in the form of an action-perception feedback loop.

In the example of the robot lawnmower, the actions could be the orientation of the mobile robot and its linear velocity. The state of the environment may be the robot's position and the height of the grass in the immediate surrounding. Finally, the reward could be the ratio of area mowed over battery power used in the last time step.

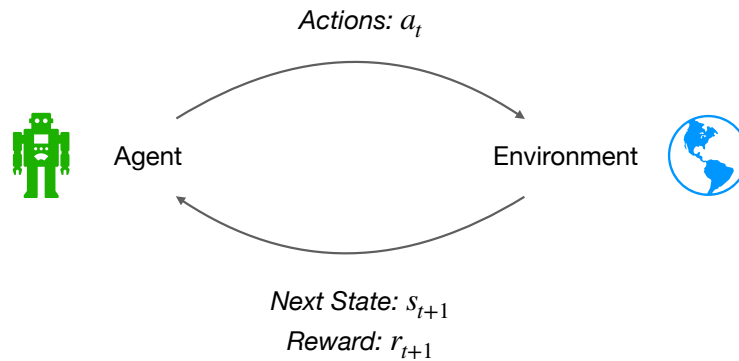


Figure 2.2: Action-perception feedback loop for RL systems (adapted from [90]).

It is worth noting that the agent's state observation s_t may not always represent the complete state of the environment [79]. In the robot lawnmower example, the environment is called *partially observable*, since the agent only receives state information about the grass in its immediate vicinity, while it cannot accurately monitor the state of distant parts of the lawn. On the other hand, an environment is said to be *fully observable* if the agent's state observation contains the complete state of the environment [79] (such as in a chess game).

Furthermore, it is important to distinguish between *continuous* and *discrete* action spaces [90]. Discrete actions are suitable if the task is describable using a finite set of actions (e.g., an agent that solves a maze can either go straight, left, or right). Continuous actions are prevalent in robotics tasks, where the agent controls a real-valued robot property (e.g., end-effector position of a robotic arm). The type of action space is a critical factor in RL algorithm selection.

2.1.3 Markov Decision Processes

The sequential decision making process in RL is modeled as a Markov Decision Process (MDP) [90]. The MDP formalism reduces any goal-directed learning procedure to a five-tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$ [88], where

- \mathcal{S} is the set of all states of the environment (with $s_t \in \mathcal{S}$),
- $\mathcal{A}(s_t)$ is the set of all actions that the agent can take from state s_t (with $a_t \in \mathcal{A}(s_t)$),
- $p(s_{t+1}|s_t, a_t)$ is the transition model which describes the probability of seeing state s_{t+1} after taking an action a_t in state s_t ,

- $r(s_t)$ (sometimes $r(s_t, a_t)$) is the reward function that assigns each state s_t (and action a_t) a utility r_t , and
- $\gamma \in [0, 1]$ is the discount factor that represents the current value of future rewards [90].

MDPs make several assumptions. The state transitions of an MDP satisfy the *Markov property* in equation (2.1): the probability of reaching the state s_{t+1} solely depends on the immediately preceding state s_t and action a_t but not on previous states or actions [90]. That is, the state s_t must capture all information that is relevant for the agent's future decisions [90]. Intuitively, a Markovian transition model assumes that the future solely depends on the present and not on the past [88]. Another assumption of MDPs is that the state is *fully observable*, i.e. the agent has perfect knowledge of the environment's state [79].

$$p(s_{t+1} | s_0, a_0, \dots, s_t, a_t) = p(s_{t+1} | s_t, a_t) \quad (2.1)$$

2.1.4 Rewards and Returns

The trajectory $(s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, \dots, r_T, s_T)$ defines the sequence of states, actions and rewards that an MDP produces, where T is the trajectory's final time step. RL agents aim to maximize the *return*, which is the cumulative reward that the trajectory generates [90].

Sutton and Barto [90] distinguish between two types of returns depending on the nature of the MDP. In *episodic* tasks a terminal state s_t is reached after a finite number of time steps T [90]. In this case, the learning process is separable into multiple independent subsequences, called *episodes* [90]. Playing a timed car racing video game is an example of an episodic task. We can break it down into multiple races, where each new race begins independently of the outcome of the previous race. The optimization objective for episodic tasks is the *finite-horizon undiscounted return* in equation (2.2), which is the sum of all received rewards in the time interval $[t + 1, T]$ [90].

$$G_t \doteq r_{t+1} + r_{t+2} + \dots + r_T = \sum_{k=1}^T r_{t+k} \quad (2.2)$$

On the other hand, Sutton and Barto [90] call tasks that go on without limit *continuing* tasks. For example, a controller for a bipedal walker has no clear terminal state and does not naturally break down into separate episodes. Since the final time step would be $T = \infty$ for continuing tasks, the return could be infinite when computed with equation (2.2), but is difficult to compare two sequences with infinite return [79]. Therefore, Sutton and Barto [90] compute the *infinite-horizon discounted return* with a *discount factor* $\gamma \in [0, 1]$ in equation (2.3).

$$G_t \doteq r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \quad (2.3)$$

The discount factor γ has several advantages. Mathematically, it is convenient because it will yield finite returns for infinite agent-environment interaction sequences if $\gamma < 1$ [79]. Intuitively, it

gives future rewards that are associated with more uncertainty less weight than immediate rewards [79]. This framework imitates animal and human preference of immediate rewards over future rewards [79] (i.e., a banana now is better than a banana later). The discount factor controls the farsightedness of the agent: a $\gamma = 0$ will optimize only for immediate rewards (which will usually lead to lower returns), and a γ close to 1 will assign future rewards more importance [90]. Both return formulations in equations (2.2) and (2.3) are equal if $\gamma = 1$.

We can easily convert any episodic task to a continuing task by infinite-looping in the episodic task's terminal state and assigning a zero reward at each time step. Hence Sutton and Barto [90] (along with most RL literature) simplify notation and use equation (2.3) as the primary return formulation.

A central advantage of RL compared to SL is that the algorithm does not rely on explicitly labeled input-output pairs. While SL would require a dataset of each correct action to be taken in each possible state, RL algorithms learn by optimizing the cumulative reward they receive. Using rewards as learning incentives is a powerful notion and makes RL an attractive and general-purpose tool. In fact, in their *reward-is-enough* hypothesis, Silver et al. [89] state that "[i]ntelligence, and its associated abilities, can be understood as subserving the maximisation of reward by an agent acting in its environment" [89, p. 4]. The reward-is-enough hypothesis means that the sheer maximization of reward gives rise to most if not all phenomena that we observe in artificial and natural intelligence, such as perception, planning, memory, language, cooperation, and creativity [89]. Whether RL is a tool for *artificial general intelligence*, that is, "the ability to flexibly achieve a variety of goals in different contexts" [89, p. 8], remains an active area of research.

When designing an RL system, a key task is finding the right reward function $R(s_t)$ that encodes the desired behavior. Sutton and Barto [90] distinguish between *sparse* and *non-sparse* rewards. Rewards are sparse when the agent does not receive a reward in every time step t , which can be due to the inaccessibility of rewarding states or the nature of the reward function [90]. For example, an agent that controls a football team in a simulated video game may receive sparse rewards about whether or not the team shot a goal in a time step [79]. On the other hand, non-sparse rewards provide intermediate feedback at each time step about the agent's progress towards achieving the desired behavior [90]. The process of providing intermediate rewards to guide the agent's learning is called *reward shaping* [64, 79]. In the football video game, one could provide additional rewards about how close the ball is to the opponent's goal or how often it was kicked [79]. Generally, non-sparse rewards lead to faster learning [64]. Still, there is a risk that agents end up collecting intermediate rewards (i.e., simply passing the football in front of the opponent's goal) rather than achieving the desired outcome (i.e., scoring as many goals as possible) [79].

2.1.5 Policies and Value Functions

Sutton and Barto [90] proceed by defining policies and value functions. The agent acts on the environment according to a *policy* $\pi(a|s)$. The policy is a function that maps the state s to the probability of taking action a [90]. Most RL literature denotes the policy as $\pi_\phi(a|s)$ whenever the policy depends on parameters ϕ , which could be the weights of a neural network. Note that often

the words agent, policy, or actor are used interchangeably in the RL literature.

Many RL algorithms are based on so-called *value functions*. There are two types of value functions. The *state-value function* $v_\pi(s)$ in equation (2.4) specifies the return that the agent can expect from being in a particular state s and following the policy π thereafter [90].

$$v_\pi(s) \doteq \mathbb{E}_\pi [G_t \mid s_t = s] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s \right] \quad (2.4)$$

On the other hand, the *action-value function* $q_\pi(s, a)$ in equation (2.5) defines the expected return by taking the action a in state s and following the policy π thereafter [90].

$$q_\pi(s, a) \doteq \mathbb{E}_\pi [G_t \mid s_t = s, a_t = a] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a \right] \quad (2.5)$$

The expression from equation (2.4) is reformulated in equation (2.9) and we notice that the return at time step t is recursively related to the return in the next step $t+1$ [90]. The state-value $v_\pi(s)$ and action-value $q_\pi(s, a)$ function both satisfy the *Bellman expectation equation* in equations (2.10) and (2.12), respectively [90]. The Bellman expectation equations specify that the value of a particular state s or state-action pair (s, a) equals the expected immediate reward plus the discounted value of the following state s_{t+1} or state-action pair (s_{t+1}, a_{t+1}) [90]. Note how the equations (2.10) and (2.12) also implement a recursive relationship. In equation (2.11), the expectation \mathbb{E}_π is calculated by summing over all actions $a \in \mathcal{A}$ which the policy $\pi(a \mid s)$ generates and over all successor states $s' \in \mathcal{S}$ that the transition model $p(s_{t+1} \mid s_t, a_t)$ produces [88]. Similarly, in equation (2.13) we sum over all successor states s' and actions a' to calculate the expected value [88].

$$v_\pi(s) = \mathbb{E}_\pi [G_t \mid s_t = s] \quad (2.6)$$

$$= \mathbb{E}_\pi [r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots \mid s_t = s] \quad (2.7)$$

$$= \mathbb{E}_\pi [r_{t+1} + \gamma(r_{t+2} + \gamma r_{t+3} + \dots) \mid s_t = s] \quad (2.8)$$

$$= \mathbb{E}_\pi [r_{t+1} + \gamma G_{t+1} \mid s_t = s] \quad (2.9)$$

$$= \mathbb{E}_\pi [r_{t+1} + \gamma v_\pi(s_{t+1}) \mid s_t = s] \quad (2.10)$$

$$= \sum_{a \in \mathcal{A}} \pi(a \mid s) \left(r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s' \mid s, a) v_\pi(s') \right) \quad (2.11)$$

$$q_\pi(s, a) = \mathbb{E}_\pi [r_{t+1} + \gamma q_\pi(s_{t+1}, a_{t+1}) \mid s_t = s, a_t = a] \quad (2.12)$$

$$= r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s' \mid s, a) \sum_{a' \in \mathcal{A}} \pi(a' \mid s') q_\pi(s', a') \quad (2.13)$$

The Bellman expectation equations (2.10) and (2.12) give rise to a system of linear equations, which can be solved analytically for small state and action spaces if the environment dynamics

$p(s_{t+1}|s_t, a_t)$ are known [88]. However, most interesting real-world problems require iterative methods to solve for the value function of a given policy [88].

Through the Bellman expectation equations (2.10) and (2.12), we know how the value functions are recursively related with themselves. Let us explore how the state-value and the action-value function are connected with each other. If we know the action-value function $q_\pi(s, a)$, we can easily calculate the value of a state $v_\pi(s)$ in equation (2.14) by summing over all actions $a \in \mathcal{A}$ that a policy would take [88]. We can now use this definition to derive an expression of $q_\pi(s, a)$ in terms of $v_\pi(s)$ in equation (2.15) by plugging equation (2.14) into the Bellman equation in (2.13). Equation (2.15) states that if we know the value function $v_\pi(s)$, the action-value of a state and action $q_\pi(s, a)$ is calculated as the immediate reward $r(s, a)$ we get from taking that action a in state s plus the discounted returns $v_\pi(s')$ of the successor states $s' \in \mathcal{S}$ that we reach according to the transition model by taking action a in state s [88].

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a | s) q_\pi(s, a) \quad (2.14)$$

$$q_\pi(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s' | s, a) v_\pi(s') \quad (2.15)$$

2.1.6 Optimal Policies and Optimal Value Functions

The value function introduced in equation (2.4) allows us to order policies by their performance [88]. A policy π performs better than or equal than another policy π' if $v_\pi(s) \geq v_{\pi'}(s) \forall s \in \mathcal{S}$ [90]. According to this ordering, there is at least one *optimal policy* π_* which performs better than or equal to all other policies [88]. The optimal policy π_* achieves the *optimal state-value function* $v_*(s)$ in equation (2.16) and the *optimal action-value function* $q_*(s, a)$ in equation (2.17) [90]. The optimal state-value function $v_*(s)$ and the optimal action-value function $q_*(s, a)$ are the maximum value functions over all policies [88].

$$v_*(s) \doteq \max_{\pi} v_\pi(s) \quad (2.16)$$

$$q_*(s, a) \doteq \max_{\pi} q_\pi(s, a) \quad (2.17)$$

For the optimal state-value function $v_*(s)$ we can rewrite equation (2.14) in a special form. The *Bellman optimality equation* for the state-value function in equation (2.18) states that the optimal value of a state corresponds to the value of the best action taken in state s by the optimal policy π_* [90]. Sutton and Barto [90] reformulate to obtain the recursive relationship of the optimal value function $v_*(s)$ in equation (2.21). We can insert equation (2.18) into (2.15) to obtain the Bellman optimality equation in (2.22).

$$v_*(s) = \max_a q_{\pi_*}(s, a) \quad (2.18)$$

$$= \max_a \mathbb{E}_{\pi_*} [G_t \mid s_t = s, a_t = a] \quad (2.19)$$

$$= \max_a \mathbb{E}_{\pi_*} [r_{t+1} + \gamma G_{t+1} \mid s_t = s, a_t = a] \quad (2.20)$$

$$= \max_a \mathbb{E} [r_{t+1} + \gamma v_*(s_{t+1}) \mid s_t = s, a_t = a] \quad (2.21)$$

$$q_*(s, a) = \mathbb{E} \left[r_{t+1} + \gamma \max_{a'} q_*(s_{t+1}, a') \mid s_t = s, a_t = a \right] \quad (2.22)$$

In RL it is desirable to know $v_*(s)$ or $q_*(s, a)$. If the transition model $p(s_{t+1} \mid s_t, a_t)$ is known and the Markov property holds, we can do a one-step look-ahead search on $v_*(s)$ over all actions [90]. The optimal policy π_* is to select the action with the highest return as predicted by $v_*(s)$ and, in RL, this is often called acting greedily with respect to the value function [90]. Note that computational resources may be a limiting factor for exhaustive look-ahead searches and the above assumptions rarely hold for real-world problems [90].

Knowing the optimal action-value function $q_*(s, a)$ is even more desirable, since not even a look-ahead search is necessary [90]. The MDP is considered solved once the optimal action-value function $q_*(s, a)$ is known and we can directly deduce the optimal policy π_* in equation (2.23) [88].

$$\pi_*(a \mid s) = \begin{cases} 1 & \text{if } a = \arg \max_{a \in \mathcal{A}} q_*(s, a) \\ 0 & \text{otherwise} \end{cases} \quad (2.23)$$

The optimal policy π_* in equation (2.23) directly select the action that maximizes the return or if there are multiple optimal actions it assigns a non-zero probability to only those actions [90]. If $q_*(s, a)$ is known, the policy can achieve this without knowing the environments dynamics $p(s_{t+1} \mid s_t, a_t)$ or the returns of successor states [90]. Similar to the Bellman expectation equations (2.10) and (2.12), the Bellman optimality equations (2.21) and (2.22) can be analytically solved to obtain $q_*(s, a)$ if the environment dynamics are known and the MDP is finite. However, as noted before, these assumptions rarely hold for real-world problems. Many RL algorithms attempt to learn approximations of the optimal action-value function $q_*(s, a)$ instead (e.g., Q-learning [93, 97]).

2.1.7 Soft Actor-Critics

The soft actor-critic (SAC) [23] algorithm is a state-of-the-art method for searching the optimal policy π_* in RL problems. The algorithm has several characteristics, and explaining them also allows us to explore some further RL terminology.

- SAC [23] is an actor-critic method. The *actor* represents the policy, while the *critic* approximates the state-value function $v(s)$ [90] or the action-value function $q(s, a)$ [23]. The critic's estimations are used as a proxy to update the actor towards producing actions that yield higher expected returns.
- The stochastic policies that SAC [23] generates are *soft*, meaning that they generate a non-zero probability for selecting all actions in all states (more formally: $\pi(a|s) > 0 \forall s \in \mathcal{S}, \forall a \in \mathcal{A}(s)$) [90].
- SAC [23] is *off-policy* which means that it can reuse past experience. *On-policy* algorithms such as Trust Region Policy Optimization (TRPO) [80] or Proximal Policy Optimization (PPO) [82] rely on new samples to perform iterative improvements to the policy. Off-policy algorithms are usually more sample efficient than on-policy methods [23]. However, off-policy methods tend to be more prone to instability especially in combination with non-linear function approximations [50].
- The SAC [23] framework is a *model-free* RL approach. A model in RL allows making predictions on how the environment reacts to certain stimuli [90]. For example, from a given state and action, a model could approximate the expected next state and reward [90]. Model-free approaches make no assumptions on the behavior of the environment and are purely based on trial-and-error learning [90].

Maximum Entropy Reinforcement Learning

The common optimization objective in RL is the expected discounted return G_t in equation (2.3). Haarnoja et al. [23] extend their objective $J(\pi)$ in equation (2.24) with an entropy regularization term $\alpha \mathcal{H}(\pi(\cdot | s_t))$. This framework is referred to as maximum entropy RL in the literature [22, 101]. The state-action marginal ρ_π in equation (2.24) is the probability distribution of the variable collection (s_t, a_t) produced by the policy π . The optimal policy π_* in equation (2.25) yields the highest return $J(\pi)$ of all policies [23]. Note that equation (2.24) states the return for a finite horizon [23]. The formulation for infinite horizons and discounted rewards is more involved and is stated in the paper [23].

$$J(\pi) = \sum_{t=0}^T \mathbb{E}_{(s_t, a_t) \sim \rho_\pi} [r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot | s_t))] \quad (2.24)$$

$$\pi_* = \arg \max_{\pi} J(\pi) \quad (2.25)$$

Equation (2.26) uses the Shannon entropy [86] from information theory to measure the randomness of the actions a generated by the stochastic policy π [66]. A policy with a larger \mathcal{H} produces more unpredictable actions. Hence optimizing the objective $J(\pi)$ in equation (2.25) yields a policy that collects much reward while acting as randomly as possible [23]. The *temperature* parameter α weighs the entropy term and thereby determines the stochasticity of the learned policy $\mathcal{H}(\pi(\cdot | s_t))$ [23]. The α is a hyper-parameter of the SAC [23] algorithm. It needs to be carefully adjusted for the task at hand which can be done either manually [22] or via an automatic tuning method [23].

$$\mathcal{H}(\pi(\cdot | s_t)) = \mathbb{E}_{a \sim \pi(\cdot | s)}[-\log(\pi(a | s))] \quad (2.26)$$

A common issue in RL is the *exploration versus exploitation* trade-off. An agent must exploit to generate high returns based on its current experience and beliefs of the world, while it must also explore to find behaviors that yield even higher returns [90]. However, exploring the world comes with the potential risk to collect less reward, and hence RL algorithms need to carefully balance to what extent they exploit and explore [90].

Haarnoja et al. [23] note that using the maximum entropy framework has several advantages. First, the entropy term in the objective function directly incentivizes the policy to explore the state and action space more broadly [23] and thereby finds an elegant solution to the trade-off between exploitation and exploration in RL. Other approaches, for example, address the exploration issue by providing external noise to the policy's actions [20]. Secondly, in situations where multiple actions are equally promising, the stochastic policy can capture all of them by yielding high probabilities for all of these actions [23]. Thirdly, the authors mention that due to the improved exploration behavior, the policies train faster than with the regular return formulations [23].

Policy Evaluation

Haarnoja et al. [23] continue by introducing the soft policy iteration framework. *Policy iteration* is a dynamic programming algorithm to find an optimal policy π_* in RL problems by alternating between a *policy evaluation* and a *policy improvement* step [90]. In the policy evaluation step, the current value function of the policy is estimated, while in the policy improvement step, the policy is updated to yield actions that yield higher returns as expected by the value function [90]. These two steps are repeated until convergence to find the optimal policy and the optimal value function.

For the policy evaluation step in equation (2.27), Haarnoja et al. [23] compute the soft action-value Q of a fixed policy π iteratively using the Bellman operator \mathcal{T}^π . Equation (2.27) consists of the reward $r(s_t, a_t)$ collected by performing action a_t in state s_t plus the soft discounted expected value $V(s_{t+1})$ of the following state s_{t+1} . Equation (2.27) therefore is a reformulation of the Bellman equation (2.15) with the Bellman operator, to account for the fact the the current value function estimate is not yet a true value function. Iteratively applying the Bellman operator \mathcal{T}^π will yield the true value function [90]. The soft state-value function V in equation (2.28) is the expected value from state s_{t+1} by following the policy's actions thereafter plus how randomly the policy acts by doing so [23]. The value function V formulation closely resembles the formulation in equation

(2.14) but includes the additional entropy term.

$$\mathcal{T}^\pi Q(\mathbf{s}_t, \mathbf{a}_t) \doteq r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim p} [V(\mathbf{s}_{t+1})] \quad \text{with} \quad (2.27)$$

$$V(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t \sim \pi} [Q(\mathbf{s}_t, \mathbf{a}_t) - \alpha \log \pi(\mathbf{a}_t | \mathbf{s}_t)] \quad (2.28)$$

Policy Improvement

In the policy improvement step, the authors [23] want to update the policy to produce higher returns based on the estimates of the current value function. In equation (2.29), the updated policy π_{new} is a policy π' that minimizes the Kullback–Leibler (KL) divergence between its probability distribution and a function of the Q -value estimate [23]. This function involves an $\exp(\cdot)$ term that gives larger Q -values substantially more weight than smaller ones, guiding the learning process to produce high returns quickly. The term $Z^{\pi_{\text{old}}}$ normalizes the distribution [23], making sure it sums to one. The new policy π' is constrained to a family of distributions Π , for example, Gaussians [23]. A small KL divergence in equation (2.29) means that high probability will be given to those actions by the new policy π_{new} which also receive a high estimated return by the action-value function.

$$\pi_{\text{new}} = \arg \min_{\pi' \in \Pi} D_{\text{KL}} \left(\pi'(\cdot | \mathbf{s}_t) \parallel \frac{\exp\left(\frac{1}{\alpha} Q^{\pi_{\text{old}}}(\mathbf{s}_t, \cdot)\right)}{Z^{\pi_{\text{old}}}(\mathbf{s}_t)} \right) \quad (2.29)$$

Optimizing Critic

The soft policy iteration algorithm by Haarnoja et al. [23] will provably converge to the optimal policy and the optimal value function [23]. However, computing the algorithm in its exact form is only tractable in the tabular case while for continuous problems we have to rely on function approximations [23]. Therefore, the authors [23] parametrize the policy $\pi_\phi(\mathbf{a}_t | \mathbf{s}_t)$ and the action-value function $Q_\theta(\mathbf{s}_t, \mathbf{a}_t)$ with weights of a neural network ϕ and θ , respectively.

Firstly, Haarnoja et al. [23] explain that the parameters for the approximated action-value function $Q_\theta(\mathbf{s}_t, \mathbf{a}_t)$ are updated by minimizing the squared Bellman residual of the soft action-value function in equation (2.30) where \mathcal{D} is the replay buffer storing states, actions, and rewards. Haarnoja et al. [23] note that the value function V is implicitly parametrized by the parameters θ via equation (2.28). For $V_{\bar{\theta}}$, the authors [23] use the parameters $\bar{\theta}$, an exponentially moving average of the parameters θ that stabilizes training [58].

$$J_Q(\theta) = \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \mathcal{D}} \left[\frac{1}{2} \left(Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - \left(r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim p} [V_{\bar{\theta}}(\mathbf{s}_{t+1})] \right) \right)^2 \right] \quad (2.30)$$

Let us analyze equation (2.30) more deeply. Recall that we would like to obtain the Bellman optimality equation for the action-value function in equation (2.22). However, equation (2.22) does not yet hold for the approximated action-value function $Q_\theta(\mathbf{s}_t, \mathbf{a}_t)$ since it is not optimal. Hence, the squared Bellman error effectively calculates the squared difference of the two sides of equation (2.22) while also integrating equation (2.18). In other words, equation (2.30) expresses

the difference between the estimated return that our current action-value function $Q_\theta(s_t, \mathbf{a}_t)$ predicts and the actual reward $r(s_t, \mathbf{a}_t)$ that was received plus the discounted expected return of the following state $\gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim p} [V_{\bar{\theta}}(\mathbf{s}_{t+1})]$.

Interestingly, neuroscience research suggests that dopamine neurons elicit signals that encode a prediction error similar to the Bellman error [83, 90]. Dopamine neurons are activated if the reward is greater than predicted, they remain uninfluenced if the reward is as expected, and they are depressed if the reward is less than predicted [83]. Hence, if the reward is different than expected, the dopamine neuron sends an alerting message which may contribute to a modification of synaptic transmission to improve future reward prediction [83].

The Bellman error is zero for the optimal action-value function. To iteratively update the action-value function $Q_\theta(s_t, \mathbf{a}_t)$ towards parameters θ that produce lower Bellman errors, Haarnoja et al. [23] compute the gradient of the Bellman error using the chain rule in equation (2.31). The algorithm trains two action-value functions with parameters θ_1 and θ_2 and consequently uses the minimum of both to deal with the overestimation bias of value-based methods [23]. The parameters θ_1 and θ_2 are found by iteratively applying stochastic gradient descent $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$ [23].

$$\begin{aligned} \hat{\nabla}_{\theta} J_Q(\theta) = & \nabla_{\theta} Q_{\theta}(s_t, \mathbf{a}_t) (Q_{\theta}(s_t, \mathbf{a}_t) \\ & - (r(s_t, \mathbf{a}_t) + \gamma(Q_{\bar{\theta}}(s_{t+1}, \mathbf{a}_{t+1}) - \alpha \log(\pi_{\phi}(\mathbf{a}_{t+1} | s_{t+1})))) \end{aligned} \quad (2.31)$$

Optimizing Actor

Haarnoja et al. [23] continue by developing an approach to update the policy's parameters ϕ . They suggest to directly minimize the expected KL divergence in equation (2.32) [23]. After dropping the normalizing distribution because it is independent of ϕ and multiplying by α they obtain equation (2.33)[23]. To make the function differentiable with respect to the parameters ϕ in equation (2.34), they apply the reparametrization trick and obtain a sample from the policy $\mathbf{a}_t = f_{\phi}(\epsilon_t; s_t)$ with a noise vector ϵ_t sampled from, for example, a Gaussian [23].

$$J_{\pi}(\phi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}} \left[\text{D}_{\text{KL}} \left(\pi_{\phi}(\cdot | s_t) \parallel \frac{\exp(\frac{1}{\alpha} Q_{\theta}(s_t, \cdot))}{Z_{\theta}(s_t)} \right) \right] \quad (2.32)$$

$$= \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}} [\mathbb{E}_{\mathbf{a}_t \sim \pi_{\phi}} [\alpha \log(\pi_{\phi}(\mathbf{a}_t | s_t)) - Q_{\theta}(s_t, \mathbf{a}_t)]] \quad (2.33)$$

$$= \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}, \epsilon_t \sim \mathcal{N}} [\alpha \log(\pi_{\phi}(f_{\phi}(\epsilon_t; s_t) | s_t)) - Q_{\theta}(s_t, f_{\phi}(\epsilon_t; s_t))] \quad (2.34)$$

Finally, Haarnoja et al. [23] can take the gradient of the objective $J_{\pi}(\phi)$ in equation (2.35) and use it to obtain the desired updated ϕ parameters using stochastic gradient descent $\phi \leftarrow \phi - \lambda_{\pi} \hat{\nabla}_{\phi} J_{\pi}(\phi)$.

$$\begin{aligned} \hat{\nabla}_{\phi} J_{\pi}(\phi) = & \nabla_{\phi} \alpha \log(\pi_{\phi}(\mathbf{a}_t | s_t)) + \\ & (\nabla_{\mathbf{a}_t} \alpha \log(\pi_{\phi}(\mathbf{a}_t | s_t)) - \nabla_{\mathbf{a}_t} Q(s_t, \mathbf{a}_t)) \nabla_{\phi} f_{\phi}(\epsilon_t; s_t) \end{aligned} \quad (2.35)$$

2.2 Simulating Robotic Grasping

2.2.1 Motivation

The motivation behind simulating a robotic setup is straightforward. Firstly, simulations allow researchers to test robot controllers before deploying them on expensive and potentially dangerous real-world equipment. Secondly, simulations can run faster than in real-time and in parallel. Fast simulations are beneficial when looking for corner cases that rarely happen or when working with ML algorithms that often require large amounts of data. Thirdly, researchers can collaborate more easily when developing control software for simulated robots.

2.2.2 Robotic Hand

In this research effort, we work with a three-fingered robotic hand, the ReFlex TakkTile by Right-Hand Robotics [77], which is a commercial product that originated in the i-HY hand [65] research project. Figure 2.3 shows images of the anthropomorphic gripper. The robotic hand has one bending DOF in each of the three fingers and one finger separation DOF between the two adjacent fingers. Each finger consists of a proximal and a distal finger segment connected via an under-actuated rubber flexure. Each of the bending DOF is tendon-driven: a tendon fixed to the distal finger runs inside the proximal finger segment and is pulled by a motor in the hand's base. The finger separation DOF is implemented via a direct gear transmission to a motor. This fourth DOF allows the hand to perform cylindrical, spherical, and pinch grasps. Rubber coats the fingers to increase friction, and the fingers have a round profile.



(a) Robotic hand grasping a coffee container.



(b) Hand mounted on robot arm (source Koenig et al. [39]).

Figure 2.3: Images of the ReFlex TakkTile robotic hand.

Similar to the i-HY hand [65], the ReFlex TakkTile features barometric pressure sensors integrated into the rubber coating of the robotic fingers. Figure 2.4 shows an image of a finger with the tactile sensors revealed on the proximal segment. There are five pressure sensors on the proximal

link and four on the distal one. Hence, there are nine tactile sensors per finger and 27 sensors in total. Note that the tactile sensors output a one-dimensional value. Hence it is only possible to measure normal forces and not tangential forces. If the contact is on an area of the finger that a tactile sensor perceives well, its readings are highly accurate. However, once the contact is located elsewhere (e.g., on the side of the finger), the tactile sensor will output lower or no pressure values. While the rubber-coated tactile arrays produce low noise (< 0.01 N) and have excellent hysteresis behavior [92], the tactile signals from finger-integrated sensors in real grasping scenarios are highly variable despite good experiment repeatability [96]. Hence, the tactile arrays in their current state are not reliably applicable for assessing grasp stability on manipulation systems.



Figure 2.4: Barometric pressure sensors integrated in the robotic fingers.

On top of the tactile sensors, the hand features magnetic angle encoders that measure each proximal finger joint position. There is no dedicated sensor to measure finger separation. However, the motors report their current position, which is a proxy of the measured finger separation. There is no joint encoder at the distal flexure, and hence its accurate position is unknown. However, it can be inferred from the difference between the motor positions and the measured proximal joint positions. This calculation remains an estimate, which makes it hard to apply classical grasp control algorithms [27]. A rough estimate of the contact locations and contact normals can be inferred through forward kinematics using the joint position data and the hand's kinematic information.

2.2.3 Simulation Software

Gazebo Simulator

We use the Gazebo [40] simulation platform in this project. Gazebo is a 3D robot simulator that is widely used in the robotics research community. A 2014 study found that Gazebo was the most popular robot simulation tool among the survey participants [30]. Gazebo is an open-source effort led by Open Robotics and has an active community of developers. Figure 2.5 shows the GUI of

Gazebo with the simulated robotic hand and a cylinder. Gazebo offers various simulated sensor modalities such as laser scanners, RGB cameras, depth cameras, Inertial Measurement Units (IMU) and contact sensors. The DARPA Virtual Robotics Challenge marked an important milestone in Gazebo's history [28] and contributed to its wide adoption among the robotics community. Gazebo is well integrated with the Robot Operating System (ROS) [74], an efficient communication framework for various robot components.

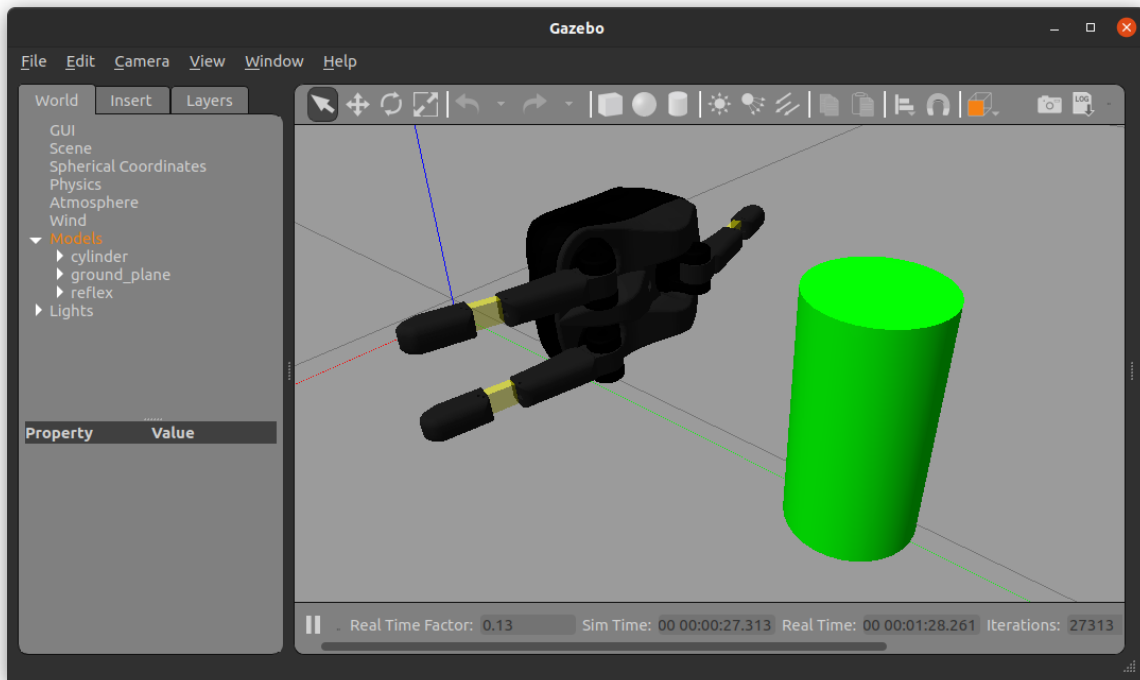


Figure 2.5: Gazebo GUI with simulated ReFlex TakkTile and a cylinder.

Simulating Grasping

Simulating complex contact scenarios can lead to unwanted vibrations and unstable simulations [28], [91]. The grasping scenario is complex because contact points from various robot links are generated on opposing sides of a single, freely moving object. In early experiments, we frequently observed resonance when grasping an object since any errors that occur while resolving the contact forces on one side of the grasp immediately transfer to the opposing side via the object, which may, in turn, lead to even greater errors that escalate over time. This situation is prone to numerical instability and often leads to unrealistic simulation outcomes. Often, considerable parameter tuning is necessary to stabilize grasping simulations. The following factors may help alleviate stability problems: low stiffness in the robot model, high damping of joints, accurate inertia values, low mass differences between the individual robot links, and good solver parameters.

Gazebo integrates four different physics engines into their simulation environment: a custom version of Open Dynamics Engine (ODE) [16], Simbody [87], Bullet [11] and Dynamic Animation

and Robotics Toolkit (DART) [43]. Some physics engines are more robust than others for simulating grasping. While Open Robotics claims that ODE is the most popular rigid body dynamics engine [16], it performs poorly on grasping tasks compared to other engines. Taylor et al. experimentally showed that DART outperforms ODE by a large margin for simulating grasping [91]. It is difficult to pinpoint why some physics engines work better than others since numerous variables can affect a simulation's outcome (e.g., iterative method non-convergence, rounding errors, regularization errors, imprecise contact information [91]).

Physics Engine Differences

The following section highlights some of the differences between Gazebo's physics engines. There are two main paradigms for modeling rigid body dynamics. *Penalty methods* resolve contact forces by enforcing a spring-damper system upon contact that applies restorative forces that are proportional to the amount of inter-penetration of the two colliding objects. On the other hand, *constraint-based methods* resolve contact forces based on a Linear Complementarity Problem (LCP) that enforces constraints, for example for non-penetration. Penalty methods require very small time-steps for stable simulations, whereas the constraint-based formulation is more numerically stable due to less stiffness in the simulated system [28]. Simbody is based on penalty methods, while ODE, Bullet, and DART are constraint-based methods. The Simbody engine is not a good choice for simulating grasping since it is more than 100 times slower than the above-mentioned constraint-based methods [71]. Taylor et al. explicitly exclude Simbody from their experiments because simulating a grasping scenario would last in "the order of days" [91].

A further difference is how the ordinary differential equations that arise from spring-damper systems are numerically solved. Spring-damper systems are common at the joints of articulated robots. Bullet uses an explicit solver method, while all other engines use implicit numerical solvers. Explicit methods are easier to compute; however, they are prone to numerical instability [71].

Another important difference between Gazebo's physics engines is the coordinate representation. Consider the example in Figure 2.6 which shows a simple robotic finger with two revolute joints. The finger is in contact with another object at point p . ODE and Bullet describe this situation in *maximal coordinates*. This representation relies on the fact that a rigid body has six DOF (three for the position and three for the orientation) in a three-dimensional cartesian space. Hence, in the maximal coordinate representation, the pose of both links is described by 12 DOF, which gives rise to a sparse mass matrix $M \in \mathbb{R}^{12 \times 12}$. The kinematic relation between both links and the world reference frame must be explicitly modeled. Each revolute joint has one DOF and therefore constrains the remaining five DOF between the connected links. Overall, ten constraints for the two revolute joints have to be added to the LCP to constrain the links' relative motion. This formulation is simple to implement. However, undesirable simulation outcomes can occur due to numeric errors: a common example is the separation of body parts due to unmet joint constraints. ODE and Bullet use this coordinate representation. DART and Simbody use *generalized coordinates*, which provide a much more compact representation of articulated robots. The robotic finger in Figure 2.6 can be fully described by the joint angles (ξ_1, ξ_2) . No additional constraints need to

be enforced since the kinematic relation is implicit in the formulation. This representation gives rise to a dense mass matrix $M \in \mathbb{R}^{2 \times 2}$ [71]. A problem description in generalized coordinates is beneficial when working with articulated robots that consist of many links and joints.

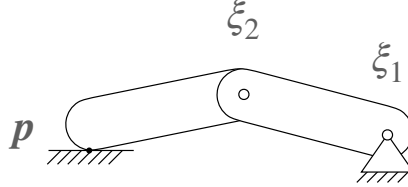


Figure 2.6: Robotic finger with two revolute joints and contact point p .

In initial experiments we validated the results from [91] and concluded that DART is the most suitable physics engine for simulating grasping. We therefore use it in all of our simulations. We find that several related works in robotic grasping also chose the DART physics backend over the default ODE engine in Gazebo [1] [48].

2.2.4 The ReFlex Simulation Stack

The long-term goal of this research project is to train RL algorithms in the simulation where data is abundant and easy to generate and to test these policies on the real robot. Therefore, a major task of this research project was building a simulation framework, named *ReFlex Stack*, that enables a seamless interplay between the simulated world and the real robot hardware. Figure 2.7 shows the system design of the *ReFlex Stack*.

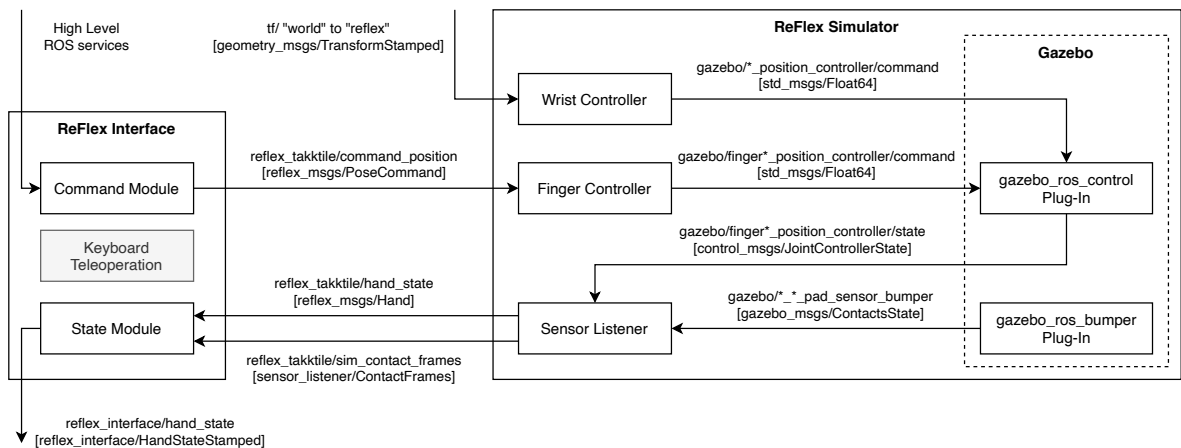


Figure 2.7: System design of ReFlex Simulation Stack.

The simulation stack in Figure 2.7 consists of the *ReFlex Simulator* and the *ReFlex Interface* modules. The below points discuss how these two core modules and the software architecture satisfy the requirements of the software stack.

Requirements of *ReFlex Simulator*

1. *Requirement:* Provide accurate simulation of robotic grasping with the ReFlex TakkTile.

Implementation: We obtain physically realistic simulation of robotic grasping by using the DART [43] physics engine in the *ReFlex Simulator* module. We can obtain other information such as joint positions or contact positions and normals with machine precision from the Gazebo [41] simulation platform.

2. *Requirement:* Achieve a seamless interplay between the real ReFlex hand and the simulation to simplify transferring policies from the simulation to the real world.

Implementation: The ReFlex hand offers a ROS [74] communication interface. Users send position commands to the hand via the ROS topic `reflex_takktile/command_position` and receive its state (e.g., joint positions, tactile sensor pressures, motor positions) by listening to the topic `reflex_takktile/hand_state`. As shown in Figure 2.7, the *ReFlex Simulator* module uses the exact same topics and message definitions as the real hand does. The *Finger Controller* ROS node listens to the command topic that also the real hand would offer and controls joints with a proportional–derivative (PD) controller implemented by a Gazebo plug-in. The *Sensor Listener* node accesses the low-level contact information from Gazebo plug-ins and transforms it into the same message structure that the real hand also uses and publishes it under the respective topic.

3. *Requirement:* Offer an interface to obtain the exact, simulated contact information.

Implementation: The real robotic hand only offers coarse contact sensing through pressure sensors at sparse locations on the finger. Hence, the message definition `reflex_msgs/Hand` only includes a subset of the contact information that would be obtainable from the simulation. Gazebo can output contact data at a higher resolution than the real robot hand can, and we publish this exact contact data (e.g., contact positions, normals, 3-dimensional force vectors, and contact frames) under the topic `reflex_takktile/sim_contact_frames`.

4. *Requirement:* Make the simulation framework applicable to any robotic arm.

Implementation: As discussed in section 1.2, a robot arm is an intricate part of a robotic grasping setup. However, as shown in Figure 2.5 we do not explicitly simulate a robotic arm. We directly control the pose of the simulated ReFlex TakkTile through the *Wrist Controller* node in the *ReFlex Simulator*. Thereby, we reduce the computational load and keep our simulation framework as generic as possible. Users have to publish the desired wrist pose to the `tf` tree, a ROS feature that stores any relative coordinate transformations. The *Wrist Controller* node will consequently send the simulated hand to this pose using a PD controller.

5. *Requirement:* Provide a simple way to spawn geometric primitives (cuboids, cylinders, and spheres) and easily control their size to simulate different starting conditions for RL algorithms.

Implementation: As part of the *ReFlex Simulator*, we provide `roslaunch` scripts that automate the spawning process and allow to parametrize the object size and mass. For example to spawn a sphere, users type `roslaunch description object.launch object_type:=sphere sphere_radius:=0.06 object_mass:=0.4` in their terminal.

6. *Requirement:* Model the distal flexure, the tactile sensors and the hand geometries.

Implementation: By default, Gazebo does not support simulating deformable objects. However, the distal flexure on the real hand is under-actuated and deforms non-linearly. Our simulator provides basic modeling of the under-actuation. We introduce two additional revolute joints at each finger: one between the proximal finger segment and the flexure and one between the flexure and the distal finger segment. We power the joints with virtual motors that execute 20% of the position command of the proximal joint each. Thereby, the hand can perform caging grasps. We model the tactile sensors by separating the finger into nine virtual segments that correspond to the fields of view of each sensor and output the normal force as the sensor's pressure reading. Furthermore, we model the hand's geometry. Depending on the user's preference, the *ReFlex Simulator* can either use accurate three-dimensional (3D) models of the hand's geometry or simplifying cuboids that serve as an approximation of the hand's geometry to reduce computational load.

Requirements of *ReFlex Interface*

1. *Requirement:* Calculate grasp stability using techniques from grasp analysis.

Implementation: We use classical methods from grasp analysis (like the metrics in [78]) to quantify a grasp's stability. Section 4.1 will review the implemented analytic grasp stability metrics in detail. The *State Module* in the *ReFlex Interface* processes the exact contact data from the `reflex_takktile/sim_contact_frames` topic to calculate several quantities from grasp analysis (e.g. Grasp Wrench Space (GWS) [19] or the grasp matrix G [73]). Finally, the calculated grasp quality metrics are published under the `reflex_interface/hand_state` ROS topic which can then be processed by grasping controllers.

2. *Requirement:* Provide a high-level interface to control the simulated and the real robot hand.

Implementation: It is useful to automate some functions that are commonly used by grasping controllers. These functions are pre-configured grasp positions (open, close, pinch, spherical open, and spherical close) and other high-level commands ("close fingers until they made contact", "make an increment to the current finger position", and "tighten the grip"). The *Command Module* offers these functions as ROS services and Listing 2.1 shows their respective names. Users can trigger these services conveniently via the command line as shown in Listing 2.2 or programmatically in their controllers.

```
/reflex_interface/open  
/reflex_interface/close  
/reflex_interface/pinch  
/reflex_interface/spherical_close  
/reflex_interface/spherical_open  
/reflex_interface/close_until_contact  
/reflex_interface/position_increment  
/reflex_interface/tighten_grip
```

Listing 2.1: A list of ROS services that the *ReFlex Interface* offers.

```
rosservice call /reflex_interface/close_until_contact "{}"  
rosservice call /reflex_interface/position_increment "{f1: 0.2, f2: 0.4,  
f3: 0.3, preshape: 1.5, from_measured_pos: true, blocking: false,  
tolerance: 0.01, time_out: 0.0}"
```

Listing 2.2: Calling *ReFlex Interface* ROS services from the command line.

3. *Requirement*: Offer a simple way to teleoperate the robot hand for debugging.

Implementation: The *Keyboard Teleoperation* node reads the user's keystrokes and publishes wrist pose increments to the `tf` tree. Furthermore, users can trigger the above ROS services conveniently via pressing corresponding keys. The keyboard teleoperation helps users debug their custom simulated environments. Further, they can easily check if a grasping task is feasible through human teleoperation before writing autonomous controllers.

4. *Requirement*: Make the software available open-source and facilitate an easy setup.

Implementation: The source code of the *ReFlex Stack* is publicly available¹. Furthermore, we provide pre-built Docker [55] containers to make the simulator setup as easy as possible for other researchers. For more instructions on usage and setup, view the `README.md` file in the repository's parent directory.

¹https://github.com/axkoenig/reflex_stack

3 Related Works

Robotic grasping is an intensively studied subject. This chapter reviews relevant related works in the field of tactile grasp refinement. Grasp synthesis algorithms and open-loop controllers, which make up much of the literature, will not be studied as part of this review. Moreover, we will mainly focus on multi-fingered robotic hands.

3.1 Taxonomy

Most closed-loop grasping controllers are classifiable using two factors.

1. *Control Paradigm*: Figure 3.1 shows that closed-loop grasp controllers can be grouped by to their underlying control paradigm. *Analytical* approaches attempt to solve an optimization problem formulated around geometric considerations, friction models, and equations of motion [46] (e.g., [21, 95]). These models typically require accurate knowledge of the underlying parameters such as friction coefficients or object and gripper pose, which may be hard to obtain in real-world scenarios [56]. On the other hand, *data-driven* (also known as empirical or learning-based) methods avoid an explicit problem formulation and address robotic grasping by large-scale data collection and processing with ML algorithms [46] (e.g., [5, 7, 8, 14, 26, 29, 33, 42, 56, 62, 99]). Efficient simulation environments, the rising availability of computing power, and recent advancements in data-driven methods make learning-based approaches especially appealing in modern robotics research. Furthermore, *hybrid* approaches that combine learned models with analytic controllers exist (e.g, [13, 44, 67, 68, 94]).

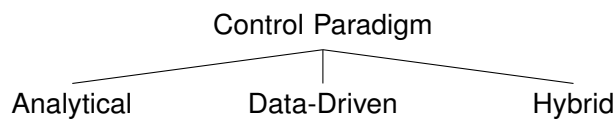


Figure 3.1: Control paradigms for closed-loop grasping controllers.

2. *Input Data*: Closed-loop grasp controllers can also be classified according to their input data, as shown in Figure 3.2. Firstly, some methods primarily rely on *tactile* information from contact sensors (e.g, [8, 14, 26, 29, 56, 62, 67, 99]). Moreover, *visuotactile* approaches combine tactile and visual data from camera systems (e.g., [7, 17, 100]). Thirdly, some approaches purely rely on *vision* systems to close the loop for robotic grasp adjustment (e.g., [5, 33, 42, 60, 61, 68]). While the primary research focus lies on vision-based approaches,

the research body on the sub-problem of tactile grasping is smaller. A search on Google Scholar for the keywords "robot grasping vision" yielded 115000 total results, whereas the keywords "robot grasping tactile" only returned 36200 publications (results as of October 11, 2021).

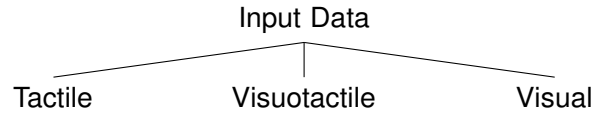


Figure 3.2: Input data for closed-loop grasping controllers.

In the following, we discuss the most prominent works that use *data-driven* controllers which process *tactile* data for grasp refinement. We will not include works using visual information from camera systems, as this project focuses on tactile grasp refinement. For a comprehensive overview of other works which are less related to this project, the reader is referred to review papers on tactile information [45] and ML [37, 46, 59] in robotic grasping.

3.2 Data-Driven Tactile Grasp Refinement

Chebatar et al. (2016) [8] propose a re-grasping approach based on spatio-temporal features and RL. Firstly, they learn a grasp stability predictor from data collected on a real robot equipped with one SynTouch BioTac [63] tactile sensor on each of the three fingers of the Barrett robot hand. Each BioTac sensor consists of 19 electrodes that sense the compression of a surrounding liquid via changes in electrical impedance. The authors create two-dimensional tactile images from the electrode measurements. They incorporate both the spatial and the temporal dimension of the tactile data sequence in a spatio-temporal extension of a Hierarchical Matching Pursuit, which is an unsupervised feature-extraction method [2]. Then they predict the grasp outcome (i.e., success or failure) from the extracted features of a lifting attempt with a Support Vector Machine (SVM) [25]. Furthermore, they learn a re-grasping policy which is used if the SVM algorithm predicts a grasp failure. They control the six DOF gripper pose change using a linear combination of the learned spatio-temporal features. To reduce dimensionality of the problem they apply Principal Component Analysis (PCA) [32] to the extracted features. The weights of the linear combination are learned with RL. They define a policy parametrized by the weights of the linear combination and optimize it with the Relative Entropy Policy Search (REPS) algorithm [70]. The predicted grasp success is their reward. After performing 1000 real-world grasps, they predict grasp success with an accuracy of 93%. They train the policy on a cylindrical object and increase the grasp success rate from 40.2% without re-grasps to 97.1% after three re-grasps. Their approach performs 5% worse on an unseen cylindrical object after three re-grasps. The authors present a generalization of their approach in [9].

Hogan et al. (2018) [26] use a parallel plate gripper equipped with two GelSlim sensors [15], which produce images of tactile imprints on the device's contact-sensitive pads. Their contribution

is two-fold. First, they learn a grasp quality metric based on tactile images and 2800 real-world trials. They train a ResNet50 [24], a convolutional neural network, in a supervised fashion to predict a continuous value $\in [0, 1]$ where large values indicate good grasp stability. Their quality metric predicts the grasp's quality with an accuracy of 85% on known objects and 75% on unknown objects. Secondly, they propose a re-grasping strategy by employing the previously learned metric. They simulate the tactile imprints that different gripper movements would generate via image transformations. Consequently, they analyze these images with the learn grasp quality predictor and select the gripper movement which yields the highest grasp stability. They assume that the object stays in place during the refinement motion and that the two-dimensional transformation of the original image describes the re-grasping action. Their final approach obtains success rates of approximately 60% to 90% on unseen household objects.

Murali et al. (2018) [62] present an approach for blind grasping and re-grasping of unknown objects. They use an adaptive three-fingered Robotiq hand with one high-quality force sensor mounted on each fingertip to obtain haptic data. They collect a publicly available dataset with RGB frames, haptic sensor data, material labels, and grasping actions and outcomes from 7800 grasp interactions. Initially, they place an object inside a box, and the robot roughly localizes the object by monitoring collisions that occur during sweeping motions of the gripper. The robot then performs an initial grasp which is likely to be unstable. A tactile grasp refinement algorithm performs iterative re-grasps in a second stage as long as the estimated grasp quality is below a certain threshold. Murali et al. use a recurrent auto-encoder to learn a low dimensional embedding of the time series of gripper position control signals and measured forces at each finger. They use this embedding as inputs for two neural networks. The first network predicts grasp stability as a scalar $p \in [0, 1]$ and trains with a cross-entropy loss. In contrast, the second network predicts a four-dimensional vector of gripper position and orientation changes parametrizing the re-grasping action. They discretize the control interval of each DOF into five bins which allows them to train the neural network in an SL setting with a cross-entropy loss function. They use the embedded tactile features to classify seven materials with an accuracy of 42.86%. Furthermore, they show that using the tactile re-grasping approach improves the baseline by 14% to achieve an overall accuracy of 40% for the blind grasping scenario. Finally, they demonstrate that tactile re-grasping enhances the performance of a vision-based approach by 10.6% to an overall visuotactile grasping success rate of 61.9%.

Merzić et al. (2019) [56] demonstrate that leveraging contact feedback improves the performance of grasping policies under object pose uncertainty. They propose a closed-loop controller based on model-free deep RL. Their state space consists of the four joint positions of the three-fingered Barrett Hand, three measurements for the object position, four for the objection orientation, and six measurements for the linear and angular velocities of the object. For the experiments with contact feedback, Merzić et al. add a 27-dimensional contact force vector to the state space (three force components at each of the nine links of the hand). The RL agent's actions are four torque commands for each of the hand's joints. The base of the hand remains fixed throughout the episode. The episode lasts for 10 seconds in simulation time, and gravity is disabled. At the end of the episode, the authors enable gravity and perform a drop test. Their reward signal consists of

the change of the total number of links in contact, the change in distance between the gripper's base and the object's center of mass, regularization terms for the joint torques and object's linear velocity, the change in mean distance between fingertips and the object, and a binary reward for the drop test. They use the TRPO [80] algorithm for all their experiments. Their dataset consists of 5 objects with corresponding pre-grasp poses from the database presented in [35]. Their first experiment trains one policy per pre-grasp and evaluates the controller on the previously seen training set. Their results show that the controller learned with contact feedback outperforms the controllers that do not get force feedback or follow more naive approaches such as open-loop grasping with constant torque. In a second experiment, they train one policy for each object and evaluate on a test set of unseen pre-grasps. In this scenario, they report a substantial drop in average performance compared to the first experiment. A third experiment explains that adding noise to the object pose during training can improve controller performance and generalizability.

Wu et al. (2019) [99] present a deep RL framework for closed-loop tactile grasp refinement. They use the Barrett Hand, mounted on a Staubli-TX60 robotic arm, and train their algorithm in a PyBullet [12] simulation environment. The hand has 24 tactile sensors on its palm and each of its fingers. Their state vector contains the recent histories (last 20 time steps) of the absolute values and the incremental changes of the binary tactile contacts, finger joint angles, and contact positions. Their action space consists of a wrist rotation $r \in [-\pi, \pi]$ and five random variables $\in [0, 1]$. The algorithm samples from a Bernoulli distribution given the random variables to trigger five high-level motion primitives: (1-3) closing each of the three finger joints by a pre-defined angle, (4) reopening the hand, and (5) lifting the hand. If the hand reopens, a new wrist position is defined above the center of all contact locations, and the wrist is rotated by r to obtain a more stable grip. When the algorithm closes a finger, the finger increment is relatively large at the beginning (0.4 rad) but is continuously refined throughout the training procedure via curriculum learning. Wu et al. discretize their action space in this way because they argue that continuous robot control leads to a greater sim-to-real gap due to more complex interaction dynamics. However, applying discretized motions leads to an almost identical behavior in simulation and the real world. At the last specified time step, a lifting attempt is automatically triggered. A lifting attempt ends the episode, and the algorithm generates a binary reward that indicates if the lift was successful. A penalty is given if no fingers close more than 0.2 radians after a reopening maneuver. They demonstrate an impressive transfer of their policy to the real robot without retraining. They obtain grasp success rates of up to 98.7% on the real robot and outperform an open-loop baseline which relies purely on vision by 2.5% to 5.2% depending on the experiment. Finally, they show that their tactile controller robustly handles calibration noise of up to 7.5cm and still achieves success rates of approximately 90%. On the other hand, the baseline struggles to handle such cases as success rates are 10% to 20%.

Hu et al. (2020) [29] present a deep RL framework for tackling the problem of reaching, grasping, and re-grasping in one unified policy. They use the PPO algorithm [82] to train a policy that controls the end-effector velocity and finger torques of the Barrett Hand based on various information obtained from the simulation environment. The algorithm's input is the object position, the distances from the hand's fingertips to object key-points, the hand z-rotation, finger angles, and contact force magnitudes as measured on the inside of the fingers. They put particular emphasis on their

task-specific reward function, which consists of six terms. They output a high reward if (1) the distance from finger to object key-points is low, (2) the normals of the key-points point towards the object center, (3) many object key-points are contained within the convex hull of the hand key-points, (4) the number of contacts is large, (5) the number of contacts on the outside of the fingers is low, and (6) the object velocity is low. They show impressive results in simulation. They train the policy by randomly spawning an object in the simulated world, and they apply a disturbance force at a random point in time which pushes the object away if not grasped reliably. The policy can grasp a static object at a 97% success rate while it can also dynamically grasp a moving target and recover from failed grasping attempts at a success rate of 83%.

3.3 Comparison

Table 3.1 compares the selected related works.

Inputs

Firstly, several differences regarding the input space become apparent. While some works only assume tactile feedback [8, 26] for their grasp refinement algorithms, others require perfect object information [29, 56]. The approaches that require object data (e.g., the object's pose or twist) are those that were purely trained and tested in simulation [29, 56]. This observation supports our argument that real-world grasping algorithms should avoid assuming object information in their input, as this data is hard to obtain accurately in the real world. For instance, Hu et al. [29] assume distance measurements between each fingertip and the closest object key-point. Obtaining this information from camera systems is extremely difficult because occlusion (and self-occlusion) are significant challenges. Multi-camera approaches can help with this problem but are complex to integrate.

All presented works in Table 3.1 except Wu et al. [99] use the contact forces as their input in one form or another. Wu et al. [99] only use binary contact signals and demonstrate that algorithms trained without feedback on contact forces can reach impressive success rates of +95%. The work by Wu et al. [99] is also the only algorithm transferred from simulation to the real world. It is known that simulators are prone to producing unphysical behavior if the LCP gets numerically unstable [28, 91]. Therefore the contact forces in simulated environments may not always be physically realistic. Perhaps policies trained in simulations produce better sim-to-real performance when provided with binary contact signals, which should be very close to the values that would also be obtainable in real-world experiments.

Conclusion: We do not want to make any assumptions on object information in our algorithm's input. Even though our algorithms train in a simulation where object data is readily available, using it would substantially hinder their applicability on real robotic hardware. Our algorithms should purely rely on the joint position and contact data, which would be available from intrinsic sensing on a robotic hand. Specifically, our experiments studying **RQ 2** examine which type of contact data is most helpful in grasping. For example, the investigation juxtaposes a binary contact framework

(similar to Wu et al. [99]) with a framework that receives information on normal forces (comparable to [29]).

Outputs

The algorithms' outputs differ by multiple factors. First, [8, 26, 62, 99] control the robot using position commands, while [56] use the torque domain and [29] control the robot with velocity commands. Merzić et al. [56] claim that controlling the hand with joint torques rather than position commands makes the grasp more compliant. Other works argue that position control transfers better to the real world because achieving a low sim-to-real gap with torque or velocity control would require precise physical models [5].

Many works [8, 26, 62] purely adjust the wrist pose with learned controllers and actuate the fingers through open-loop algorithms. On the other hand, Merzić et al. [56] only control the fingers and fix the wrist in place. Presumably, this makes their grasping algorithm less adaptive to calibration errors. Other works [29, 99] control both the wrist and the fingers simultaneously. Of those algorithms that solely learn a gripper pose change [8, 26, 62] only Chebotar et al. [8] predict the full six-dimensional change in gripper pose $\Delta(x, y, z, \xi, \eta, \zeta)$. The other works limit their analysis on top-down [26, 62] or side-ways [29] grasps and hence can leave some dimensions un-actuated. Wu et al. [99] achieve gripper position changes via a hard-coded strategy that analyzes the last contact points and defines a new gripper pose at some distance to these points.

Conclusion: We want to follow most related works in their decision to control the robot in position space. We presume that mapping from contact points (in position space) to the torque or velocity domain may be more complex to learn than keeping the inputs and the outputs in the same domain. Furthermore, we want our algorithm to simultaneously update the fingers and wrist pose, as we believe that only the interplay between both controllers can make robotic grasp refinement truly efficient. Lastly, we do not want to restrict our algorithm to side-ways or top-down grasps and hence wish to predict changes in the full gripper pose $\Delta(x, y, z, \xi, \eta, \zeta)$. Humans who perfect grasp refinement also update their wrist pose in all six dimensions and simultaneously update their fingers to obtain a more stable grip.

Objective

Some works learn a custom grasp stability predictor and aim to maximize its score [8, 26, 62]. From Table 3.1 it is evident that these works also train in the real world. We hypothesize that these authors needed a grasp quality measure tailored to their experimental setup. A learned metric can more easily adapt to experiment-specific factors, while analytic grasp stability metrics may be harder to implement on custom robotic hardware. Especially, Hogan et al. [26] would have difficulty working with classical grasp stability analysis as they are working with tactile images where contact positions and forces are not directly measurable. However, the frameworks trained in simulation [29, 56, 99] rely on hand-crafted optimization objectives, which can get quite complex as shown by the six-term reward function by Hu et al. [29]. More importantly, it is difficult to justify

why, for example, a small "angle between hand key-point normals and vectors pointing from hand key-points to object center" as in Hu et al.'s [29] work relates with grasp stability.

While all presented works in Table 3.1 except [99] use force measurements of some form in their inputs, non of the papers explicitly uses them in their optimization objectives. However, from grasp analysis, it is clear what contact forces are desirable for good grasp stability. It is therefore not obvious why previous research did not integrate them into their optimization objectives.

Conclusion: All presented works in tactile and data-driven grasping use either learned grasp quality predictors or manually engineering grasp stability cues. None of these optimization objectives include analytic descriptions of physical models. However, it is essential to consider the rich body of research on grasp analysis and analytical grasp quality metrics [78] in the context of tactile grasping. While analytical grasp stability metrics are used in the context of grasp synthesis and planning on parallel-jaw grippers [51, 52], and multi-fingered hands [98], their potential for closed-loop grasping with RL remains to be explored. Hence, in our research, we aim to integrate these well-justified metrics in reward functions for RL algorithms. To our knowledge, and by looking at Table 3.1, no prior work attempted this before. Using such analytic grasp quality metrics could avoid constructing complex reward functions as in [29] or sparse rewards as in [99], which are known to be less sample-efficient [64]. Lastly, it is crucial to integrate feedback about the stability of grasping forces in the algorithm's reward functions. As discussed above, none of the analyzed works includes an explicit assessment of the grasping forces in their optimization objective, and hence our algorithms will explore this direction.

Paper	Inputs	Outputs	Objective	Algorithm
Chebotar 2016 [8] <i>Reality</i>	Spatio-temporal tactile features	Weights of a linear combination that maps inputs to gripper pose changes $\Delta(x, y, z, \xi, \eta, \zeta)$	<u>Maximize</u> grasp success predicted by a learned stability estimator	RL, REPS [70]
Hogan 2018 [26] <i>Reality</i>	Images of simulated tactile imprints corresponding to gripper movements	Gripper position change $\Delta(x, y)$ along grasping plane of parallel plate gripper	<u>Maximize</u> grasp success predicted by a learned stability estimator	SL
Murali 2018 [62] <i>Reality</i>	Time series of gripper position control signals and measured forces at each finger	Gripper position and rotation change $\Delta(x, y, z, \xi)$ along with current grasp stability	<u>Maximize</u> grasp success predicted by a learned stability estimator	SL
Merzić 2019 [56] <i>Simulation</i>	Proximal joint angles, object pose and twist, contact force vector on each link of robotic hand	Joint torques of robotic hand	<u>Maximize</u> (1) number of links in contact and (2) binary drop test outcome <u>Minimize</u> (1) distance object to gripper, (2) distance fingertips to object, (3) joint torques and (4) object velocity	RL, TRPO [81]
Wu 2019 [99] <i>Both</i>	Histories of binary contact signals, finger joint angles and contact positions	Absolute wrist rotation angle (ξ) and probabilities for closing fingers by fixed position increment, for finger reopening and lifting	<u>Maximize</u> binary pickup reward at episode end <u>Minimize</u> finger reopening	RL, Soft PPO [82]
Hu 2020 [29] <i>Simulation</i>	Object position, hand z-rotation, finger joint angles, distances from fingertips to closest object key-point, contact force magnitudes on inside of fingers	Hand translational velocities in world (x, y)-plane and rotational velocity around world (z)-axis (v_x, v_y, ω_z), finger torques	<u>Maximize</u> (1) number of contacts and (2) number of object key-points contained in the convex hull of hand and finger key-points <u>Minimize</u> (1) distance from hand key-points to object key-points, (2) angle between hand key-point normals and vectors pointing from hand key-points to object center, (3) number of contacts on outside of fingers, and (4) object's linear velocity	RL, PPO [82]

Table 3.1: Overview of related works using *data-driven* approaches and *tactile* information. *Reality* means that experiments were conducted on physical robot hardware, *Simulation* refers to algorithms trained and tested only in simulation, and *Both* means that algorithms were trained in simulation and deployed to real hardware (includes data from Koenig et al. [38, p. 2]).

4 Reward Design and Grasp Refinement

This chapter investigates the potential of analytic grasp stability metrics as reward functions for RL grasping controllers. Firstly, we will discuss grasp stability metrics from classical grasping theory. Consequently, we describe the experimental setup in detail and present empirical results. The discussion concludes this chapter with an answer to **RQ 1** from section 1.3. The following chapter is a revision of a publication by Koenig et al. [38] which is currently under review and partially contains results from this paper.

4.1 Analytic Grasp Stability Metrics

4.1.1 Largest-Minimum Resisted Wrench

Ferrari and Canny's [19] metric ϵ_w is among the most popular grasp stability metrics in the literature. The metric measures the largest-minimum perturbing wrench that the grasp can resist given the grasp's friction constraints at the contacts. We will introduce the terminology of Ferrari and Canny's [19] paper, which will also be relevant for the following sections, and consequently define the quality metric.

Terminology

Equation (4.1) defines the contact wrench vector $\mathbf{w}_i \in \mathbb{R}^p$ for each point-contact \mathbf{p}_i . The index i refers to the i -th contact and there are n_c contacts in total. A contact wrench vector \mathbf{w}_i consists of a force component \mathbf{f}_i and a torque component $\boldsymbol{\tau}_i$ at that contact. Ferrari and Canny [19] note that in the planar case the dimension is $p = 3$ as the wrench vector consists of two orthogonal force directions and one torque component that lies perpendicular to the force plane. In the 3D case $p = 6$, since both forces and torques lie in a 3D space. In this project, we are working in a 3D cartesian space and a multi-fingered hand, and therefore $\mathbf{w}_i \in \mathbb{R}^6$.

$$\mathbf{w}_i = \begin{pmatrix} \mathbf{f}_i \\ \boldsymbol{\tau}_i \end{pmatrix} \quad (4.1)$$

Coulomb's law of friction [10] in equation (4.2) relates the tangential forces $\mathbf{f}_{i,t}$ at contact \mathbf{p}_i with the contact's normal forces $\mathbf{f}_{i,n}$ via a coefficient of friction μ . Oftentimes, a coefficient of static friction μ_s and a coefficient of kinetic friction μ_k are introduced, where $\mu_s > \mu_k$ means that the tangential forces that the contact can resist decrease as relative motion begins [34]. However,

in Ferrari and Canny's work [19] only a single coefficient of friction μ is used for simplicity. This assumption is reasonable for hard and dry materials [34].

$$\mathbf{f}_{i,t} \begin{cases} = \mu \mathbf{f}_{i,n} & \text{for sliding contact} \\ < \mu \mathbf{f}_{i,n} & \text{for static contact} \end{cases} \quad (4.2)$$

The *friction cone* at contact \mathbf{p}_i shown in Figure 4.1 is a representation of Coulomb's law of friction [34]. All contact forces within the friction cone will result in static contact, whereas sliding begins once the force vector coincides with the cone surface. In [19] and in many other works, the friction cone is approximated as a *friction pyramid* spanned by m unit forces $\mathbf{f}_{i,j}$ that define the edges of the pyramid where i is the index of the contact point and $j \in \{1, 2, \dots, m\}$ is the index of the friction cone edge. The number of friction cone edges m dramatically influences the accuracy of the friction estimation: $m = 4$ leads to an approximated error of roughly 30%, while $m = 8$ gives rise to a lower 8% error [3]. The accuracy of the friction estimation must be traded off with higher computation times for larger values of m [3]. We use $m = 4$ to allow for a fast computation.

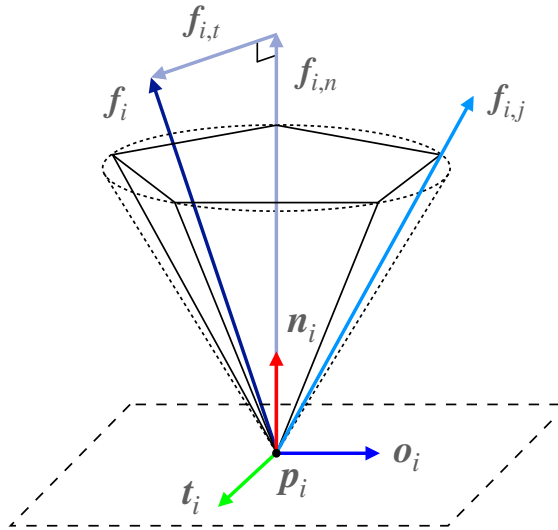


Figure 4.1: Friction cone at contact \mathbf{p}_i with $m = 5$ and the contact frame $\{\mathbf{n}_i, \mathbf{t}_i, \mathbf{o}_i\}$. The vectors $\mathbf{f}_{i,j}$ span the edges of the approximated friction cone. The force \mathbf{f}_i with its normal $\mathbf{f}_{i,n}$ and tangential component $\mathbf{f}_{i,t}$ lies inside the friction cone (graphic adapted from [34]).

Ferrari and Canny [19] show in equation (4.3) that contact force \mathbf{f}_i may be represented as a convex combination of the vectors $\mathbf{f}_{i,j}$ that define the friction pyramid. Similarly, the reaction torque $\boldsymbol{\tau}_i = \mathbf{r}_i \times \mathbf{f}_i$, where \mathbf{r}_i is the vector pointing from the object's center of mass \mathbf{p}_c to the point of contact \mathbf{p}_i , can be expressed as a combination of the $\mathbf{f}_{i,j}$ forces in equation (4.4).

$$\mathbf{f}_i = \sum_{j=1}^m \kappa_{i,j} \mathbf{f}_{i,j} \quad \text{where} \quad \kappa_{i,j} \geq 0 \quad \text{and} \quad \sum_{j=1}^m \kappa_{i,j} \leq 1 \quad (4.3)$$

$$\boldsymbol{\tau}_i = \sum_{j=1}^m \kappa_{i,j} (\mathbf{r}_i \times \mathbf{f}_{i,j}) \quad (4.4)$$

Ferrari and Canny [19] extend the concept of the friction cone to obtain a *wrench cone*, where the contact wrench \mathbf{w}_i at contact point \mathbf{p}_i is represented as a convex combination of the wrenches $\mathbf{w}_{i,j}$ that result from the forces $\mathbf{f}_{i,j}$ which span the friction cone. The wrenches $\mathbf{w}_{i,j}$ define the edges that span the approximated wrench cone. Finally, they compute the total wrench on the object \mathbf{w} as the sum of the wrenches on all n_c contact points. These considerations yield equation (4.5).

$$\mathbf{w} = \sum_{i=1}^{n_c} \mathbf{w}_i = \sum_{i=1}^{n_c} \sum_{j=1}^m \kappa_{i,j} \mathbf{w}_{i,j} \quad \text{with} \quad \mathbf{w}_{i,j} = \begin{pmatrix} \mathbf{f}_{i,j} \\ \mathbf{r}_i \times \mathbf{f}_{i,j} \end{pmatrix} \in \mathbb{R}^6 \quad (4.5)$$

Consequently, the authors [19] define the set of all possible wrenches on the object through convex sets, known as the Grasp Wrench Space (GWS). \mathcal{W}_{L_∞} in equation (4.6) is the convex hull of the *Minkowski sum* of all wrenches $\mathbf{w}_{i,j}$, and \mathcal{W}_{L_1} in equation (4.7) is the convex hull of the *union* of all wrenches $\mathbf{w}_{i,j}$. Ferrari and Canny [19] note that $\mathcal{W}_{L_1} \subseteq \mathcal{W}_{L_\infty}$.

$$\mathcal{W}_{L_\infty} = \text{ConvexHull} \left(\bigoplus_{i=1}^{n_c} \{\mathbf{w}_{i,1}, \dots, \mathbf{w}_{i,m}\} \right) \quad (4.6)$$

$$\mathcal{W}_{L_1} = \text{ConvexHull} \left(\bigcup_{i=1}^{n_c} \{\mathbf{w}_{i,1}, \dots, \mathbf{w}_{i,m}\} \right) \quad (4.7)$$

The force-closure properties of a grasp can easily be analyzed using the GWS. Force-closure means that there exists a combination of contact forces \mathbf{f}_i that satisfy the friction constraints and which can compensate any object wrench [73]. A grasp is force-closure if the origin of the wrench space $\mathbf{0} \in \mathbb{R}^6$ is contained in the GWS \mathcal{W} [19] where \mathcal{W} is either \mathcal{W}_{L_∞} or \mathcal{W}_{L_1} .

Quality Metric

Finally, Ferrari and Canny [19] define the quality of a grasp ϵ_w as the largest-minimum perturbing wrench that the grasp can resist. Remember from equation (4.5) that \mathbf{w} is the sum of all contact wrenches \mathbf{w}_i . If a wrench \mathbf{w} lies on the boundary of the GWS (i.e. $\mathbf{w} \in \partial\mathcal{W}$) a perturbing wrench can directly lead to slippage and a failing grasp. The quality metric ϵ_w in equation (4.8) identifies the magnitude of the smallest of those wrenches $\|\mathbf{w}\|$ that lie on the boundary $\partial\mathcal{W}$ of the GWS. The metric is the distance from the origin of the wrench space to the closest hyper-plane of \mathcal{W} . Geometrically, the quality metric can be interpreted as the radius of the largest sphere centered at the origin and contained in \mathcal{W} . The metric ϵ_w is bounded $\in [0, 1]$ and greater values of ϵ_w indicate

higher grasp quality.

$$\epsilon_w = \min_{\mathbf{w} \in \partial\mathcal{W}} \|\mathbf{w}\| \quad (4.8)$$

There exist directions in the GWS where the resisted wrench is larger, but the metric aims at giving a lower bound of the maximum resisted wrench in all wrench directions. In other words, the metric identifies the magnitude of the wrench in the "worst-case" scenario in which the largest resisted wrench is minimized. The quality metric ϵ_w depends on the choice of the definition and will yield different results for \mathcal{W}_{L_∞} and \mathcal{W}_{L_1} , respectively. We will use the \mathcal{W}_{L_1} definition throughout this work.

4.1.2 Measuring Resistance to Pure Forces and Torques

A significant drawback of the wrench-based metric ϵ_w is the non-comparability of the units of forces (in N) and torques (in Nm) [78]. Therefore, Mirtich and Canny [57] decouple the GWS and propose two alternative metrics ϵ_f and ϵ_τ which evaluate how well a grasp can resist pure forces and torques, respectively.

Planar Example

Let us define Mirtich and Canny's [57] metrics using a planar example. Figure 4.2a shows a grasp on an object with two contact points \mathbf{p}_1 and \mathbf{p}_2 and the approximated friction cones at the contacts. The set of forces the contacts can apply to the object are now defined by \mathcal{W}_f in equation (4.9). Note that, unlike \mathcal{W} from section 4.1.1, \mathcal{W}_f only considers forces. \mathcal{W}_τ in equation (4.10) defines the torques that the grasp can theoretically apply to the object.

$$\mathcal{W}_f = \text{ConvexHull} \left(\bigcup_{i=1}^{n_c} \{\mathbf{f}_{i,1}, \dots, \mathbf{f}_{i,m}\} \right) \quad (4.9)$$

$$\mathcal{W}_\tau = \text{ConvexHull} \left(\bigcup_{i=1}^{n_c} \{\boldsymbol{\tau}_{i,1}, \dots, \boldsymbol{\tau}_{i,m}\} \right) \quad (4.10)$$

Mirtich and Canny's [57] identify the largest-minimum resisted force ϵ_f and torque ϵ_τ in equations (4.11) and (4.12), respectively. The motivation behind this is the same as in section 4.1.1. Figure 4.2b demonstrates that the quality metric is the radius of the largest ball centered at the origin that is fully contained inside \mathcal{W}_f .

$$\epsilon_f = \min_{\mathbf{f} \in \partial\mathcal{W}_f} \|\mathbf{f}\| \quad (4.11)$$

$$\epsilon_\tau = \min_{\boldsymbol{\tau} \in \partial\mathcal{W}_\tau} \|\boldsymbol{\tau}\| \quad (4.12)$$

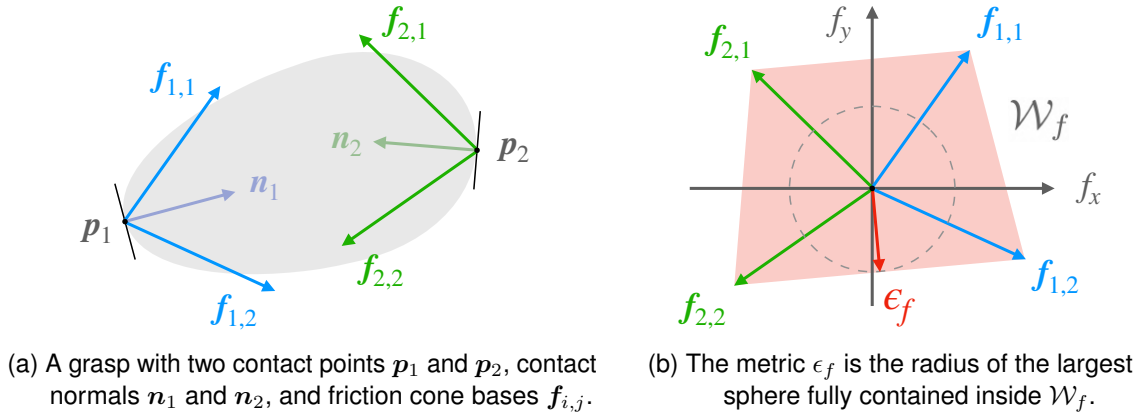
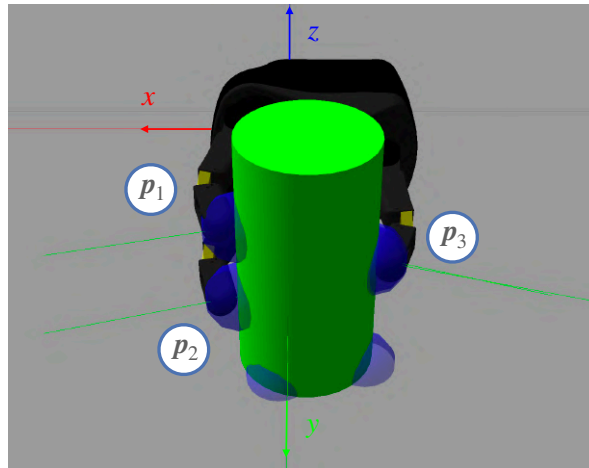


Figure 4.2: Planar example of a grasp and stability analysis with ϵ_f (image source Koenig et al. [38]).

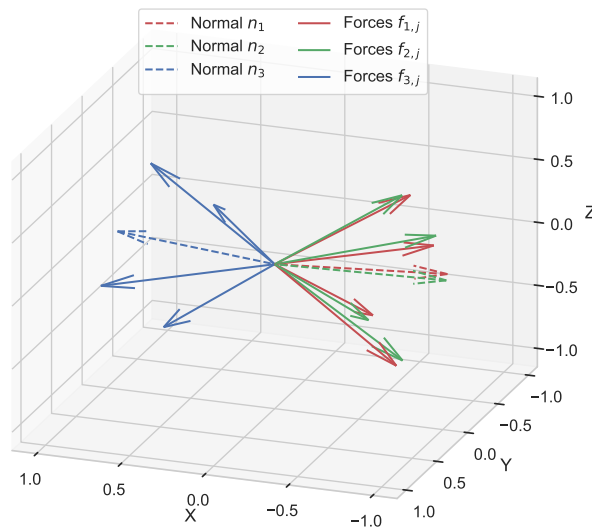
3D Examples

Let us review examples for the calculation of ϵ_f and ϵ_τ on a three-fingered hand in three dimensions. Figure 4.3a shows the simulated grasp with three contact points. We visualize the edges of the approximated friction cone $f_{i,j}$ in Figure 4.3b where $m = 4$. The force component of the GWS \mathcal{W}_f can be obtained by computing the convex hull over the vectors $f_{i,j}$ in bold and ϵ_f is calculated as described earlier. Similarly, we plot the torques $\tau_{i,j}$ that result from the forces $f_{i,j}$ in Figure 4.3c. After constructing the convex set \mathcal{W}_τ using the $\tau_{i,j}$ vectors one can determine ϵ_τ . Notice the different magnitudes of the forces (in N) and torques (in Nm) in Figures 4.3b and 4.3c. The coefficient of friction for both plots is $\mu = 0.9$.

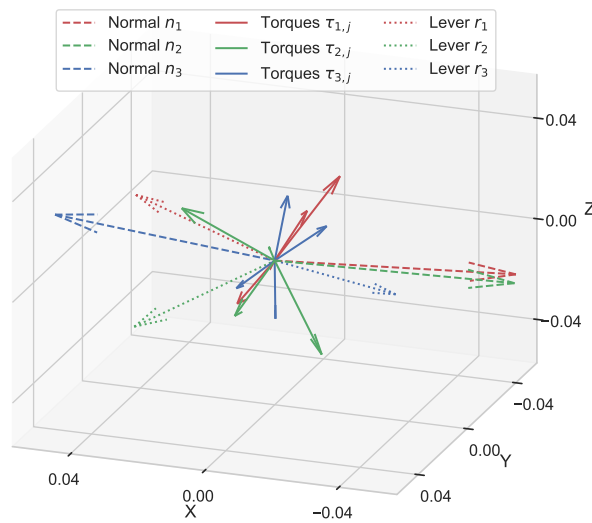
Figure 4.4 briefly reviews some grasps and their corresponding qualities. A *power grasp* is a grasp configuration in which the fingers wrap around the object, leaving no room for the object to move [73]. Firstly, Figures 4.4a and 4.4b show how a power grasp is favored by the ϵ_f metric, since more contact normals with new orientations are generated which consequently span a larger \mathcal{W}_f . If one of the adjacent fingers of the power grasp would lift off (as depicted in Figure 4.4c), the grasp quality as calculated by ϵ_f will be almost unchanged because ϵ_f is solely a function of the contact normals n_i and the friction coefficient. The contact normals of the two-finger power grasp from Figure 4.4c will generate a similar \mathcal{W}_f as the contact normals of the three-finger grasp from Figure 4.4b. Hence, adding the third finger will offer little to no expansion of \mathcal{W}_f . However, a three-finger grasp is more stable in practice and can compensate more object torque. The ϵ_τ metric solves this problem. It also considers the contact positions p_i , and the object's center of mass p_c . Therefore, adding the third finger improves the grasp quality, as shown in Figure 4.4d.



(a) A grasp with three contact points p_1 , p_2 and p_3 (in blue) and contact normals (in green).



(b) Plot of forces $f_{i,j}$ that span the approximated friction cones and contact normals n_i .



(c) Resulting torques $\tau_{i,j}$, contact normals n_i , and lever arms r_i .

Figure 4.3: Analyzing a grasp and calculating $f_{i,j}$ and $\tau_{i,j}$ on which \mathcal{W}_f and \mathcal{W}_τ are based.

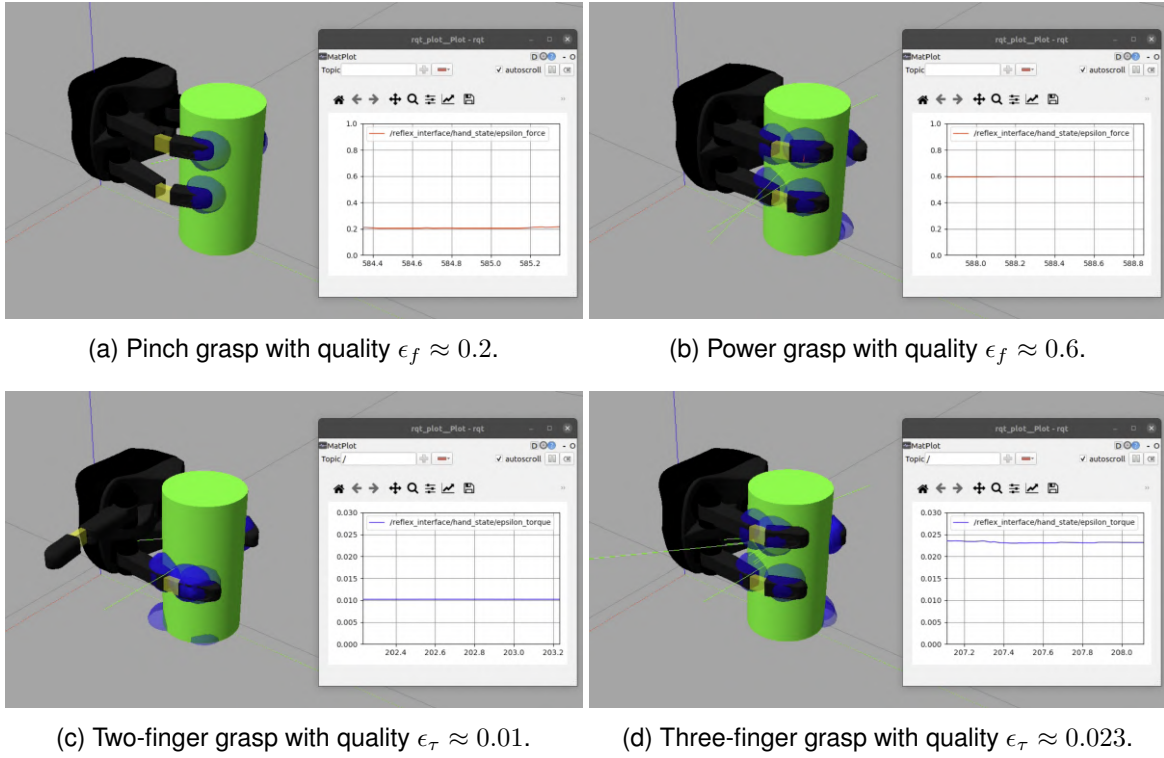


Figure 4.4: Grasp configurations with calculated grasp quality metrics.

4.1.3 Force-Agnostic Grasp Stability Metrics

Motivation for Force-Agnostic Metrics

The grasp quality metrics ϵ_w , ϵ_f , and ϵ_τ provide essential insights into the ability of the grasp to balance external object wrenches. They focus their analysis on the *theoretically applicable contact forces* that are satisfying the friction constraints and are concerned with the correct finger placement to maximize the theoretically applicable object wrench in any direction. Therefore the metrics effectively analyze the grasp's geometric properties because contact positions and normals are the main quantities influencing the calculated grasp stability. Moreover, since *unitary force vectors* $f_{i,j}$ span the GWS (and the decoupled GWS \mathcal{W}_f and \mathcal{W}_τ), the proposed quality metrics allow no assessment of the currently applied grasp forces. While this limitation may not be as apparent in algorithms solely concerned with grasp synthesis, this is a fundamental problem in real-time grasp control. Analyzing solely the grasp's geometrical properties can be problematic since a grasp with a high geometrical grasp quality can easily fail due to slippage (i.e., the grasp fails since the applied contact forces were not adequate).

Buss et al. [6] define grasp stability in terms of the distance of the contact forces to the friction cone. Inspired by this work, we present a metric that evaluates grasp quality based on contact forces. We present two variants of this metric. We call our first metric based on the *current* contact forces, contact normals, and the friction coefficient δ_{cur} . Moreover, our second metric δ_{task} additionally takes *task* wrenches and contact positions into account. In many grasping tasks, the

expected task wrenches can be estimated apriori or at least their upper bounds. For example, for a gripping robot that needs to lift an item from one box into another, it is clear that the grasp must resist the object's weight and any inertial forces that occur while relocating the object, plus a safety margin. While δ_{cur} is a general-purpose grasp quality metric, δ_{task} is applicable when a task definition exists. First, we discuss δ_{cur} before deriving δ_{task} .

Grasp Stability Based on Current Contact Forces

Consider Figure 4.5 which shows an object grasped at two contact points p_1 and p_2 . The current contact forces $f_{i,cur}$ are in equilibrium and there are no external forces acting on the object. The contact normals are n_i . Remember that we refer to each contact with the subscript i and that there are n_c contact points (in this case $n_c = 2$).

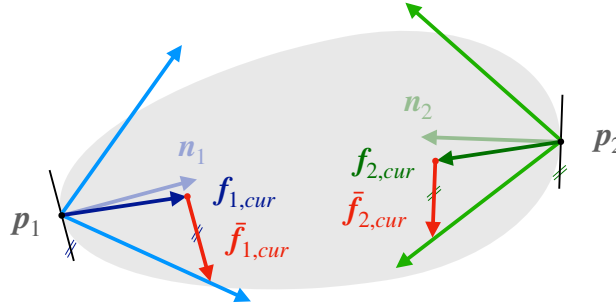


Figure 4.5: Friction cones, current contact forces $f_{i,cur}$, contact normals n_i and tangential force margins $\bar{f}_{i,cur}$ used to compute δ_{cur} . Note that we consider the true friction cone now, and not the approximated one as in [19] (image source Koenig et al. [38]).

The vectors $\bar{f}_{i,cur}$ are the *tangential force margins*, which represent the smallest additional tangential forces that would lead to slippage at contact i . We can also think of the tangential force margins as safety margins to the friction cone. We care most about the direction in which the contact can take the least tangential force before slipping and therefore want to identify the worst-case tangential force. Hence, the tangential force margin $\bar{f}_{i,cur}$ is a lower bound over all tangential forces that contact i can resist. Since the direction of the tangential force margins is determined by the closest distance to the friction cone, we can limit our grasp stability analysis solely on the magnitudes of these safety margins. For simplicity, we denote the magnitudes of the tangential force margins $\|\bar{f}_{i,cur}\|$ as $\bar{f}_{i,cur}$. A tangential force margin $\bar{f}_{i,cur} > 0$ means static contact while $\bar{f}_{i,cur} = 0$ means undesired sliding behavior.

Let us discuss how to calculate the magnitudes of the minimum additional resisted tangential forces $\bar{f}_{i,cur}$. Equation (4.13) shows that using Pythagoras' theorem we can decompose the magnitude $\|f_i\|$ of a generic contact force f_i into the magnitude of the tangential force $f_{i,t}$ and the magnitude of the normal force $f_{i,n}$. Equation (4.14) computes the normal force $f_{i,n}$ using a scalar projection of the contact force f_i along the contact normal n_i , which is a unit vector (i.e. $\|n_i\| = 1$). Finally, we insert equation (4.14) into (4.13) and solve for the magnitude of the tangential force $f_{i,t}$ in equation (4.15).

$$\|\mathbf{f}_i\|^2 = f_{i,t}^2 + f_{i,n}^2 \quad (4.13)$$

$$f_{i,n} = \mathbf{f}_i \cdot \mathbf{n}_i \quad (4.14)$$

$$f_{i,t} = \sqrt{\|\mathbf{f}_i\|^2 - (\mathbf{f}_i \cdot \mathbf{n}_i)^2} \quad (4.15)$$

With equation (4.15) we can calculate the magnitude of the tangential force $f_{i,t}$ as a function of the contact force \mathbf{f}_i and the contact normal \mathbf{n}_i . From the Coulomb friction law in (4.2) we know that the maximum tangential force $f_{i,t}^{max}$ a contact can resist before slipping is the friction coefficient μ times the contact normal force $f_{i,n}$. We write this relation in equation (4.16) and obtain equation (4.17) by substituting equation (4.14) for $f_{i,n}$.

$$f_{i,t}^{max} = \mu f_{i,n} \quad (4.16)$$

$$= \mu \mathbf{f}_i \cdot \mathbf{n}_i \quad (4.17)$$

Finally, the tangential force margin \bar{f}_i in equation (4.19) is simply the *maximum allowed* tangential force $f_{i,t}^{max}$ from equation (4.17) minus the *actual* tangential force $f_{i,t}$ from equation (4.15), thereby representing the minimum tangential force to the friction cone. Equation (4.20) calculates the desired *current* tangential force margin $\bar{f}_{i,cur}$.

$$\bar{f}_i(\mathbf{f}_i, \mathbf{n}_i, \mu) = f_{i,t}^{max} - f_{i,t} \quad (4.18)$$

$$= \mu \mathbf{f}_i \cdot \mathbf{n}_i - \sqrt{\|\mathbf{f}_i\|^2 - (\mathbf{f}_i \cdot \mathbf{n}_i)^2} \quad (4.19)$$

$$\bar{f}_{i,cur} = \bar{f}_i(\mathbf{f}_{i,cur}, \mathbf{n}_i, \mu) \quad (4.20)$$

Our quality metric δ_{cur} builds on the same idea as the work by Buss et al. [6]: a grasp with large tangential force margins $\bar{f}_{i,cur}$ provides a more secure grasp than one with small margins since the contacts are less prone to sliding when an object wrench is applied. Equation (4.21) shows a general formulation of our quality metric $\tilde{\delta}$. It measures the grasp's stability by computing an average of the magnitudes of the tangential force margins $\bar{f}_{i,cur}$ over all n_c contact points weighted by their respective contact force magnitudes $\|\mathbf{f}_i\|$. The matrices \mathbf{F} and \mathbf{N} in equations (4.22) and (4.23) contain the contact forces \mathbf{f}_i and contact normals \mathbf{n}_i as column vectors. A grasp with a higher $\tilde{\delta}$ is more desirable than one with a low $\tilde{\delta}$.

We weigh the tangential force margins \bar{f}_i in equation (4.21) by their respective contact force magnitudes $\|\mathbf{f}_i\|$ because we care more about large contact forces that are not slipping than small ones. Larger contact forces are more important for grasp stability than smaller ones because they can give rise to larger maximum tangential forces (see equation (4.17)) and can thereby compensate more disturbing object wrenches.

$$\tilde{\delta}(\mathbf{F}, \mathbf{N}, \mu) = \frac{\sum_{i=1}^{n_c} \|\mathbf{f}_i\| \bar{f}_i(\mathbf{f}_i, \mathbf{n}_i, \mu)}{\sum_{i=1}^{n_c} \|\mathbf{f}_i\|} \quad \text{with} \quad (4.21)$$

$$\mathbf{F} = \begin{pmatrix} \mathbf{f}_1 & \mathbf{f}_2 & \dots & \mathbf{f}_{n_c} \end{pmatrix} \in \mathbb{R}^{3 \times n_c}, \quad \text{and} \quad (4.22)$$

$$\mathbf{N} = \begin{pmatrix} \mathbf{n}_1 & \mathbf{n}_2 & \dots & \mathbf{n}_{n_c} \end{pmatrix} \in \mathbb{R}^{3 \times n_c} \quad (4.23)$$

Equation (4.21) introduced a generic formulation of the quality metric $\tilde{\delta}$. Finally, we can define the quality metric δ_{cur} in equation (4.24), which quantifies how far the contact forces are *currently* from the boundary of the friction cones. Therefore, we use the current contact forces $\mathbf{f}_{i,cur}$ to span the column space of the matrix \mathbf{F}_{cur} in equation (4.25). Grasp stability optimization aims to increase δ_{cur} .

$$\delta_{cur} = \tilde{\delta}(\mathbf{F}_{cur}, \mathbf{N}, \mu) \quad \text{with} \quad (4.24)$$

$$\mathbf{F}_{cur} = \begin{pmatrix} \mathbf{f}_{1,cur} & \mathbf{f}_{2,cur} & \dots & \mathbf{f}_{n_c,cur} \end{pmatrix} \in \mathbb{R}^{3 \times n_c} \quad (4.25)$$

Grasp Stability Based on Anticipated Contact Forces

We now established a measure of grasp quality based on current contact forces, called δ_{cur} . However, in many grasping tasks, a clear task definition exists (e.g., "move an object from location A to location B"). Let $D = \{\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_q\}$ be the set of task wrenches that the grasp must resist during task execution where q is the total number of wrenches. For example, a task wrench could be the object's weight $\mathbf{w}_1 = (0 \ 0 \ -f_g \ 0 \ 0 \ 0)^T$ if the task definition involves lifting the object. Note that \mathbf{w}_1 is not an object wrench (which would, in this case, be $-\mathbf{w}_1$) but rather the effective object wrench that the grasp forces amount to in order to balance the external object wrenches. Other task wrenches can include wrenches from inertial effects while relocating the object or wrenches that arise from potential collisions with the robot's environment. It is useful to multiply the task wrenches by a scalar safety factor (e.g., for adding a 20% safety margin to \mathbf{w}_1 we write $1.2 \cdot \mathbf{w}_1$).

The Task Wrench Space (TWS) is a convex set of D (e.g., a ball [19] or an ellipsoid [47]) and the GWS [19] is a convex set of all wrenches that the grasp can resist. Then, several related works define a task-oriented quality metric by measuring the maximum scaling factor between the TWS and the GWS such that the TWS is fully contained within the GWS [4, 72]. Unfortunately, since these approaches are based on the GWS they face the same issue as described earlier. Because they reason about the theoretically applicable contact forces, which are commonly bounded to unity [19, 78], it is not possible to evaluate whether the *current* contact forces of a grasp are suitable to balance an anticipated task wrench $\mathbf{w}_t \in D$. In this work, we define an alternative task-oriented metric δ_{task} through which we can calculate the upper bounds of the anticipated task forces during task execution with techniques from grasp analysis.

Figure 4.6 gives a high-level introduction to our metric δ_{task} . Similar to Figure 4.5, we show a

grasp with two contact points p_1 and p_2 . However, now we augment the current contact forces $f_{1,cur}$ and $f_{2,cur}$ by considering the task wrench w_1 from the above example (i.e., in this example the task polytope is $D = \{w_1\}$). We predict the additional contact forces $f_{1,add}$ and $f_{2,add}$ that are required to compensate the task force $-f_g$. By adding these additional forces to the current contact forces $f_{1,cur}$ and $f_{2,cur}$, we obtain the final task contact forces $f_{1,task}$ and $f_{2,task}$ that each contact must resist. Analogously to δ_{cur} , we base our stability analysis on the tangential slip margins. The magnitude of the tangential slip margins $\bar{f}_{i,task} = \bar{f}_i(f_{i,task}, n_i, \mu)$ that we expect during task execution are calculated based on the task contact forces $f_{i,task}$ as depicted in Figure 4.6.

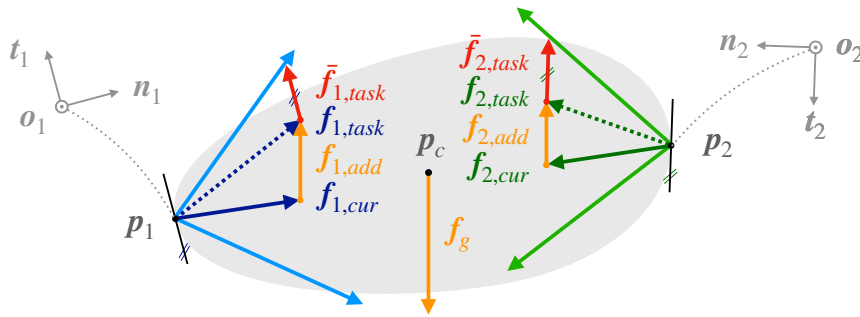


Figure 4.6: Friction cones and anticipated task contact forces $f_{i,task}$ used to compute δ_{task} (image adapted from Koenig et al. [38]).

Let us discuss how we obtain the additional contact forces $f_{i,add}$ that compensate the task wrench. Let $w_t \in D$ be an arbitrary task wrench that the grasp must resist (the object wrench to be resisted would be $-w_t$). We use the *grasp matrix* G (sometimes called the grasp map) to predict how much reacting force each contact i should produce to compensate the task wrench w_t . We adhere to the definitions and notation by Prattichizzo et al. [73] to calculate the grasp matrix.

To determine G , Prattichizzo et al. [73] first define P_i in equation (4.26) which specifies the translation from an arbitrary reference point on the object, usually the object's center of mass p_c , to each contact point p_i . The authors [73] use the cross product matrix $S(r)$ in their calculations.

$$P_i = \begin{pmatrix} \mathbf{I}_{3 \times 3} & \mathbf{0} \\ \mathbf{S}(p_i - p_c) & \mathbf{I}_{3 \times 3} \end{pmatrix} \quad \text{with} \quad \mathbf{S}(r) = \begin{pmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{pmatrix} \quad (4.26)$$

Further, they define a rotation matrix R_i that expresses the orientation of each contact frame in equation (4.27). Three unit vectors span the column space of the matrix R_i : the contact normal n_i and two orthogonal vectors t_i and o_i . See Figures 4.1 and 4.6 for visualizations of the contact frame. Prattichizzo et al. [73] then obtain the blockdiagonal matrix \bar{R}_i in equation (4.28).

$$\mathbf{R}_i = \begin{pmatrix} \mathbf{n}_i & \mathbf{t}_i & \mathbf{o}_i \end{pmatrix} \in \mathbb{R}^{3 \times 3} \quad (4.27)$$

$$\bar{\mathbf{R}}_i = \text{Blockdiag}(\mathbf{R}_i, \mathbf{R}_i) = \begin{pmatrix} \mathbf{R}_i & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_i \end{pmatrix} \in \mathbb{R}^{6 \times 6} \quad (4.28)$$

Prattichizzo et al. [73] introduce the *selection matrix* \mathbf{H}_i that determines which components of the contact wrench $(f_{i,n} \ f_{i,t} \ f_{i,o} \ m_{i,n} \ m_{i,t} \ m_{i,o})^T$ contact i can transmit to the object. The wrench intensity vector λ_i in equation (4.29) contains only those components that are actually transmitted under the assumption of the friction model.

$$\lambda_i = \mathbf{H}_i (f_{i,n} \ f_{i,t} \ f_{i,o} \ m_{i,n} \ m_{i,t} \ m_{i,o})^T \quad (4.29)$$

Depending on the friction model in Table 4.1 we obtain different selection matrices. In the Point Contact without Friction (PwoF) model the contact can only transmit the contact normal force $f_{i,n}$. All other forces or moments in the PwoF model are assumed to be negligible which is a valid assumption for surfaces with a low friction coefficient μ and small contact patches [73]. The Hard Finger (HF) model additionally transmits the tangential forces $f_{i,t}$ and $f_{i,o}$ that arise due to Coulomb friction. The HF model also assumes small contact patches [73]. The Soft Finger (SF) model additionally transmits a contact moment $m_{i,n}$ around the contact normal and is used if the contact patch is sufficiently large to transmit friction moments [73].

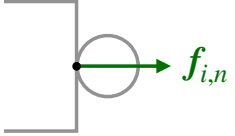
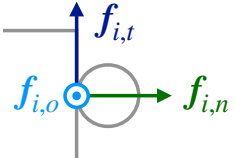
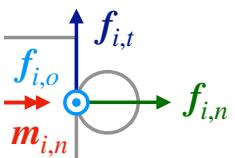
Model	Transmitted Quantities	λ_i	n_{λ_i}	\mathbf{H}_i
PwoF		$\lambda_i = (f_{i,n})$	1	$\mathbf{H}_i = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$
HF		$\lambda_i = \begin{pmatrix} f_{i,n} \\ f_{i,t} \\ f_{i,o} \end{pmatrix}$	3	$\mathbf{H}_i = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$
SF		$\lambda_i = \begin{pmatrix} f_{i,n} \\ f_{i,t} \\ f_{i,o} \\ m_{i,n} \end{pmatrix}$	4	$\mathbf{H}_i = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$

Table 4.1: Contact models and respective selection matrices (partly adapted from [73]).

Prattichizzo et al. [73] go on to define the the partial grasp matrix \mathbf{G}_i of each contact point i in equation (4.30) as the matrix product of \mathbf{P}_i , $\bar{\mathbf{R}}_i$ and \mathbf{H}_i^T . Finally, the grasp matrix \mathbf{G} in equation

(4.31) is the assembly of the partial grasp matrices \mathbf{G}_i of all contacts.

$$\mathbf{G}_i = \mathbf{P}_i \bar{\mathbf{R}}_i \mathbf{H}_i^T \in \mathbb{R}^{6 \times n_{\lambda_i}} \quad (4.30)$$

$$\mathbf{G} = \begin{pmatrix} \mathbf{G}_1 & \mathbf{G}_2 & \dots & \mathbf{G}_{n_c} \end{pmatrix} \in \mathbb{R}^{6 \times n_c \cdot n_{\lambda_i}} \quad (4.31)$$

Prattichizzo et al. [73] stack the contact wrench intensities λ_i at each contact i in the vector $\lambda = (\lambda_1^T \ \lambda_2^T \ \dots \ \lambda_{n_c}^T)^T$. Remember that $w_t \in D$ is the task wrench we would like the grasp to resist. The grasp matrix \mathbf{G} relates the contact wrench intensities λ to the resulting object wrench $\mathbf{G}\lambda$ resulting from the contact interactions. In our case, the object wrench is the task wrench w_t and we obtain equation (4.32).

$$\mathbf{G}\lambda = w_t \quad (4.32)$$

Finally, the only unknown in equation (4.32) are the suitable contact wrench intensities λ that balance the task wrench w_t . Prattichizzo et al. [73] show that using pseudoinverse of the grasp matrix \mathbf{G}^+ we can calculate λ in equation (4.33). Note that there is not necessarily a unique λ that solves the equation $\mathbf{G}\lambda = w_t$. This ambiguity is because there exist internal contact wrenches $\lambda \in \mathbf{N}(\mathbf{G})$, where $\mathbf{N}(\mathbf{G})$ is the null space of the grasp matrix, that only affect the tightness of the grasp but do not translate to an additional object wrench because the individual contact wrench intensities λ_i cancel out [73]. In simple terms: pressing harder will not lead to an external wrench to the object that could accelerate it if the contact wrenches cancel out. Prattichizzo et al. [73] add the term $\mathbf{N}(\mathbf{G})\gamma$ to the right side of the equation and thereby identify all wrench intensities λ that lead to a 0 object wrench. The parameter γ is an arbitrary vector parametrizing the solution [73].

$$\lambda = \mathbf{G}^+ w_t + \mathbf{N}(\mathbf{G})\gamma \quad (4.33)$$

For our quality metric δ_{task} , we use the HF contact model since we are only interested in the additional contact forces (and not the moments). We can therefore replace λ with the desired additional contact forces $f_{i,add}$ in equation (4.34). By assuming that the internal grasp forces stay constant after applying the additional contact forces $f_{i,add}$ we can omit the term $\mathbf{N}(\mathbf{G})\gamma$ in equation (4.34). Furthermore, we assume that the configuration of the grasp (e.g., number of contacts and contact normals) do not change after applying $f_{i,add}$.

$$\begin{pmatrix} f_{1,add}^T & f_{2,add}^T & \dots & f_{n_c,add}^T \end{pmatrix}^T = \mathbf{G}^+ w_t \quad (4.34)$$

Now that we know the additional contact forces $f_{i,add}$ the grasp must react with to balance w_t , we can calculate the anticipated task contact forces $f_{i,task}$ from Figure 4.6. It is important to note that λ expresses the wrench intensities in the contact frames $\{n_i, t_i, o_i\}$. Therefore the current contact forces $f_{i,cur}$ also have to be represented in this reference frame before adding them in

equation (4.35).

$$\mathbf{f}_{i,task} = \mathbf{f}_{i,cur} + \mathbf{f}_{i,add} \quad (4.35)$$

Last but not least, we can calculate our task-oriented grasp quality metric δ_{task} in equation (4.36). We simply insert the contact forces $\mathbf{f}_{i,task}$ that we expect during task execution into the generic formulation of $\tilde{\delta}$ from equation (4.21) to measure the size of the tangential force margins $\tilde{\mathbf{f}}_{i,task}$. Therefore, the metric δ_{task} measures the average distance of the expected task contact forces $\mathbf{f}_{i,task}$ from the friction cones. In equation (4.36), we calculate a lower bound over all task wrenches $\mathbf{w} \in D$, because we want to identify the worst-case task wrench for which the grasp will most likely fail. Larger values of δ_{task} are desirable.

$$\delta_{task} = \min_{\mathbf{w} \in D} \tilde{\delta}(\mathbf{F}_{task}, \mathbf{N}, \mu) \quad \text{with} \quad (4.36)$$

$$\mathbf{F}_{task} = \begin{pmatrix} \mathbf{f}_{1,task} & \mathbf{f}_{2,task} & \dots & \mathbf{f}_{n_c,task} \end{pmatrix} \in \mathbb{R}^{3 \times n_c} \quad (4.37)$$

4.1.4 Summary

Table 4.2 summarizes the required information to calculate each of the presented quality metrics.

Information	ϵ_w	ϵ_f	ϵ_τ	δ_{cur}	δ_{task}
Friction coefficient μ	yes	yes	yes	yes	yes
Contact normals \mathbf{n}_i	yes	yes	yes	yes	yes
Contact positions \mathbf{p}_i	yes	no	yes	no	yes
Object center of mass \mathbf{p}_c	yes	no	yes	no	no
Current contact forces $\mathbf{f}_{i,cur}$	no	no	no	yes	yes
Task definition D	no	no	no	no	yes

Table 4.2: Grasp quality metrics and the information required to compute them.

4.2 Experimental Setup

4.2.1 Algorithm Overview

The Big Picture

Before we discuss the details of the RL algorithm, let us refocus our attention on the vision of this research effort. The grasp quality metrics presented in section 4.1 require various data about the grasp to be known accurately. Table 4.2 summarizes the required data for each metric. Real grasping systems can not always measure this data precisely, but simulators can output it at high resolutions. Therefore, the quality metrics can easily be calculated in the simulated environment, where all the required data is readily available. The metrics can be used as *reward* signals r_t for RL algorithms during training. At test time, RL algorithms do not require access to the

reward and use solely the *state* information s_t which does not necessarily include accurate contact information. Therefore, the high-level vision of this project is to train RL algorithms in *simulation* and consequently deploy them in the *real world*.

Figure 4.7 shows how the information flow of the system implements this idea. In section 2.2.4 we presented the *ReFlex Stack*, a simulation framework for the ReFlex TakkTile robotic hand. We mentioned that one of its primary functional requirements was the seamless interaction between the simulator and the real robotic hand. We achieved this by providing the same message interface in simulation that also the real robot hand uses to report its state s_t information. Similarly, we also want our trained policies to easily communicate with both the simulated and the real robotic hand. As shown in Figure 4.7, our policy π_ϕ receives the state vector s_t as its input. We pipe the policy's *actions* a_t through the *Command Module* of the *ReFlex Interface* and thereby convert the actions into position control commands that both the simulator and the real ReFlex hand can directly execute. Figure 4.7 also demonstrates that the *ReFlex Interface* calculates the quality metrics using the highly accurate contact information from the simulation and provides it to the RL algorithm as the reward signal r_t .

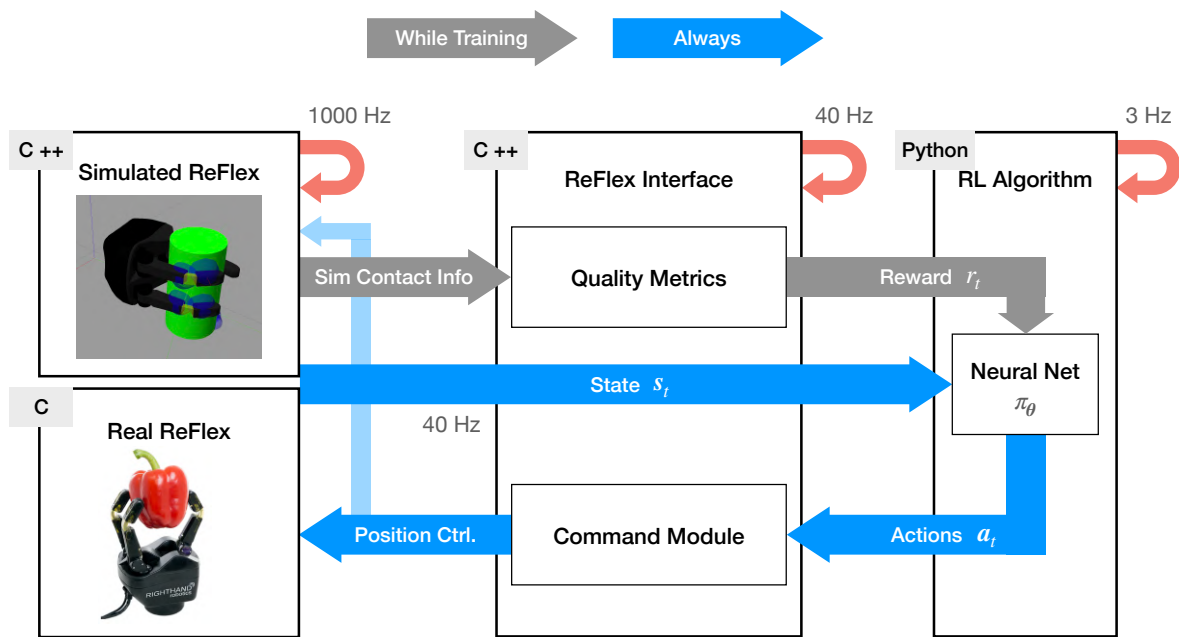


Figure 4.7: Information flow in the grasping pipeline. The grey arrows indicate data that only flows while training the algorithm. Blue arrows indicate information that flows during training *and* testing. We also show update frequencies and programming languages (ReFlex image source [77]).

Grasp Refinement Experiment

In this section we design an experiment to answer **RQ 1** from section 1.3. Namely, we analyze the potential of analytic grasp stability metrics as reward functions for RL algorithms that perform tactile grasp refinement on three-fingered hands. Figure 4.8 shows an overview of our experimental setup. It is split into an (A) *initialization stage* and a (B) *grasp refinement episode*.

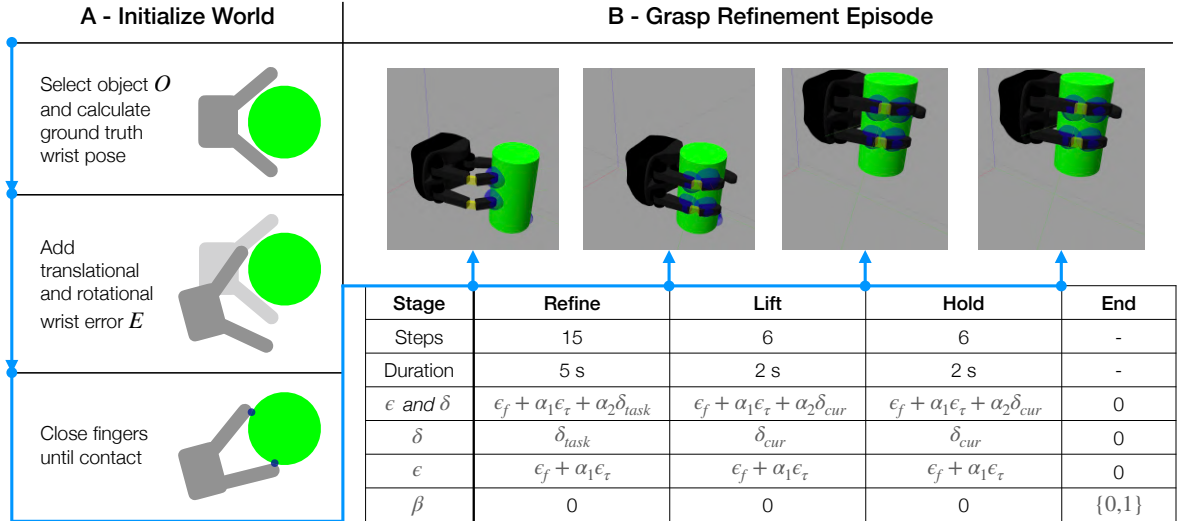


Figure 4.8: Overview of the RL algorithm. In (A), we generate a new object wrist error combination (O, E). Afterward, we start the (B) grasp refinement episode using different reward functions (image adapted from Koenig et al. [38]).

(A) In the *initialization stage*, we create a new grasp configuration that the algorithm should refine. We generate a random object O that the algorithm should grasp and find its corresponding ground-truth wrist pose. Initial grasping configurations are often defined using contact points, but it is sometimes difficult to determine whether they are reachable [18]. Therefore, we represent the ground truth grasping configuration as a gripper pose in the vicinity of the object. Specifically, this wrist pose is defined such that the offset between the tool center point (TCP) and the object's center of mass p_c is 5 cm. The ground-truth pose of the hand is facing sideways towards the object, similar as in Figure 4.8. In a second step of the initialization stage, we add a wrist error E to the pose of the TCP to simulate the calibration errors mentioned in section 1.2. Consequently, the hand closes its fingers in this initial grasp configuration until they make contact.

(B) **Episode Structure:** In the *grasp refinement episode*, the RL algorithm improves the initial grasp through iterative adjustments to the TCP pose and the finger positions. We split the process into four stages. In the (1) *refine* stage, the policy improves the grasp in 15 algorithm steps. Consequently, in the (2) *lift* stage a hard-coded lifting procedure is triggered. In the (3) *hold* stage, we test the stability of the grasp. The algorithm can continue to update the grasp in response to the changing grasping situation through six algorithm steps in the *lift* and

hold stage, respectively. The control frequency of the policy is 3 Hz in all three stages. The update frequency of the low-level PD controllers in the wrist and the fingers is 100 Hz. Finally, in the *end* stage, we record whether the hand successfully lifted the object and whether it stayed in the hand until the end of the holding stage.

Rewards: To answer **RQ 1**, we compare four reward frameworks: (1) ϵ and δ , (2) only δ , (3) only ϵ and (4) the binary reward framework β . The table in Figure 4.8 shows which grasp stability metrics from section 4.1 are provided in which reward framework and stage. For example, we use δ_{task} only in the *refine* stage and δ_{cur} otherwise because we need to predict the stability of the anticipated task forces only before lifting. After lifting, we can directly measure the actual contact forces that occur during task execution, and hence δ_{cur} is the suitable measure. The task only involves lifting the object. Hence our task definition $D = \{(0 \ 0 \ -f_g \ 0 \ 0 \ 0)^T\}$ for δ_{task} only includes resisting the object’s weight $-f_g$. Furthermore, we prefer ϵ_f and ϵ_τ over ϵ_w in our reward functions. We only use ϵ_f and ϵ_τ because, in combination, they provide two stability estimations about a grasp while ϵ_w solely returns the magnitude of a single wrench. We linearly combine the metrics using two factors $\alpha_1 = 5$ and $\alpha_2 = 0.5$ and empirically determine their magnitude. The frameworks based on grasp stability metrics receive dense reward feedback in every algorithm time step. The binary metric $\beta \mapsto \{0, 1\}$ only provides a sparse reward about whether the task was successfully executed $\{1\}$ or not $\{0\}$ at the *end* of the episode. The SAC [23] algorithm is sensitive to reward scaling [23], and therefore we normalize the rewards.

Stopping Criteria: An episode can last a maximum of $15 + 6 + 6 = 27$ algorithm steps. To discourage excessive movement of the object during the refinement stage, we end the episode early if the hand shifts the object by more than 10 cm. Furthermore, we terminate grasp refinement if one of the proximal finger joints exceeds a limit of 3 radians. If the object dropped after the *lifting* stage we do not enter the *holding* stage.

We configure the *ReFlex Simulator* of our open-source *ReFlex Stack* framework. We approximate the hand’s geometry with cuboids to reduce computational load. Furthermore, the simulated gravity in our experiments is activated (unlike in related works [56]). The object can freely interact with the simulated world.

4.2.2 Training Dataset

A training sample consists of the tuple (O, E) , where O is the object, and E is a wrist pose error. O and E are sampled *uniformly* before every episode. Table 4.3 shows the object properties from which we sample. We consider three object types (cuboids, cylinders, spheres) of varying mass and size. Figure 4.9 shows the maximum and minimum object sizes. The wrist error E consists of a translational (e_x, e_y, e_z) and a rotational error (e_ξ, e_η, e_ζ) , and we sample them from $[-5, 5]$ cm and $[-10, 10]$ deg for each variable, respectively.

Property	Sampling Ranges
Geometry	{Cuboid, Cylinder, Sphere}
Mass	$\in [0.1, 0.4]$ kg
Size	Cuboid: height $\in [13, 23]$ cm, length $\in [4, 10]$ cm, width $\in [4, 10]$ cm
	Cylinder: height $\in [13, 23]$ cm, radius $\in [3, 5]$ cm
	Sphere: radius $\in [6.5, 8]$ cm

Table 4.3: Object properties and sampling ranges.

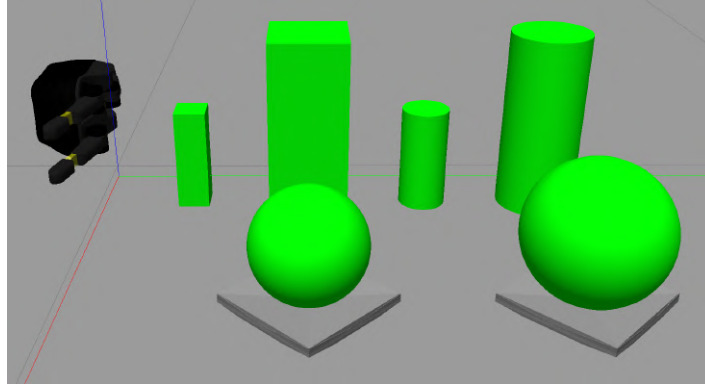


Figure 4.9: Minimum and maximum object sizes. Spheres are placed on a concave mount to prevent rolling (image source Koenig et al. [38]).

4.2.3 Test Dataset

For the test dataset, we define 8 different wrist error cases. The L2 norm of the variables (a, b, c) is $d(a, b, c) = \sqrt{a^2 + b^2 + c^2}$. Table 4.4 shows the wrist error cases in our test dataset. Case A means no error and case H corresponds to the maximum wrist error. Figure 4.10 visualizes two wrist error cases. The test dataset consists of 10 cuboids, 10 cylinders, and 10 spheres (i.e., 30 random objects O in total). For every object O , we generate eight random wrist error cases $\{A, B, \dots, H\}$ from Table 4.4. To test one model, we therefore run $30 \times 8 = 240$ experiments on randomly generated and unseen wrist error, object combinations (O, E) . The test dataset is the same for all models to allow for a fair comparison.

Wrist Error Case	A	B	C	D	E	F	G	H
$d(e_x, e_y, e_z)$ in cm	0	1	2	3	4	5	6	7
$d(e_\xi, e_\eta, e_\zeta)$ in deg	0	2	4	6	8	10	12	14

Table 4.4: Wrist error cases (source Koenig et al. [38]).

4.2.4 State and Action Space

The state vector s_t consists of seven joint positions (1 finger separation DOF, three proximal bending DOF, three distal bending DOF), and contact data on each of the seven links (three proximal links, three distal links, and a palm) that includes contact position, contact normal and

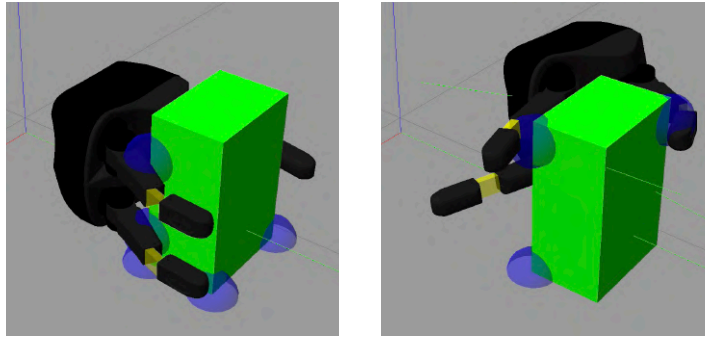


Figure 4.10: Left: wrist error case A (i.e., the ground-truth grasping pose). Right: wrist error case H (i.e., maximum wrist error) (image source Koenig et al. [38]).

contact force, that have three (x, y, z) components each. Therefore, the dimension of the state vector is $\mathbf{s}_t \in \mathbb{R}^{7+7 \times (3 \times 3)} = 70$. Our approach does not assume any information on the object (e.g., object pose, geometry, or mass) in the state vector, unlike related works [29, 56]. The contact normals and positions are provided in the wrist frame, while the contact forces are represented in the contact frame.

In this experiment, we assume perfect contact information (positions, normals, and the full 3D force vectors) that are not obtainable from the real hand. Forward kinematics can make rough approximations about contact positions and normals on the real ReFlex. The tactile sensor readings yield an estimate of the normal forces. However, it is impossible to measure the tangential forces with the barometric pressure sensors on the real robotic hand. In the following chapter 5, which analyzes **RQ 2**, we relax these strong assumptions on perfect contact sensing.

The action vector \mathbf{a}_t consists of 3 finger position increments $(a_{f_1}, a_{f_2}, a_{f_3}) \in [-8, 8]$ deg for the bending DOF, 3 TCP position increments $(a_x, a_y, a_z) \in [-0.5, 0.5]$ cm and 3 TCP rotation increments $(a_\xi, a_\eta, a_\zeta) \in [-0.5, 0.5]$ deg. The action vector’s dimension is $\mathbf{a}_t \in \mathbb{R}^{3+3+3} = 9$. The increments are added to the currently measured finger positions and wrist pose.

4.2.5 Hyper-parameters

We parametrize the policy π_ϕ by a neural network with weights ϕ . The network is a multi-layer perceptron (MLP) with four layers and the following number of neurons in each layer (70, 256, 256, 9). The input layer matches the dimension of the state vector \mathbf{s}_t while the output layer matches the dimension of the action vector \mathbf{a}_t . We use an implementation of the SAC [23] algorithm by the `stable-baselines3` [75] package. The SAC [23] algorithm trains a stochastic policy π_ϕ , which we evaluate deterministically when testing. Table 4.5 shows the hyper-parameters for the SAC [23] algorithm which yielded the best performance in our experiments.

Parameter	Value
ent_coef	0.01
learning_starts	100
batch_size	64
gradient_steps	32
train_freq	32
learning_rate	10^{-4}
tau	10^{-4}

Table 4.5: Hyper-parameters for SAC [23] algorithm.

4.3 Results

4.3.1 Training

Figure 4.11 shows the training results of our experiment. The total number of training steps is 25000, corresponding to roughly 1000 training episodes depending on the episode lengths. For each reward framework from Figure 4.8, we train 40 models, each with a different random seed to avoid overfitting. The error bars in all plots in this work show ± 2 standard errors. Furthermore, we smooth the training results with a moving average filter of kernel size 30. In all plots, *Hold Success* refers to the success rate of the algorithm holding the object in hand at the end of the grasp refinement episode. *Lift Success* refers to whether the object is still in the hand after the lifting stage. In the following, we describe the results from Figure 4.11.

Training Results Figure 4.11

- (4.11-a) **Major Result:** Plot (1) shows that the policies trained with grasp stability metrics reach higher success rates than the binary reward framework β .
- (4.11-b) **Major Result:** Plot (1) also reveals that the frameworks ϵ and δ , and δ , and ϵ are more sample efficient than β because their learning speed is greater (i.e., β has a smaller gain in performance per episode). The δ framework is less sample efficient than the frameworks that include the ϵ_f and ϵ_τ metrics.
- (4.11-c) Plot (1) and Plot (2) are almost identical. Hence, once the algorithm successfully grasps and lifts the object, it rarely drops it in the holding stage.
- (4.11-d) Plot (3) shows that the ϵ framework has the most stable learning procedure since we can observe a constant upwards trend of the episode reward. The δ framework has a reward plot that dips at approximately 350 episodes. Combining δ with ϵ improves the smoothness of the reward plot. Note that the rewards have a different scaling and therefore have different magnitudes.
- (4.11-e) Plot (6) demonstrates that the reward framework ϵ and δ is particularly helpful for

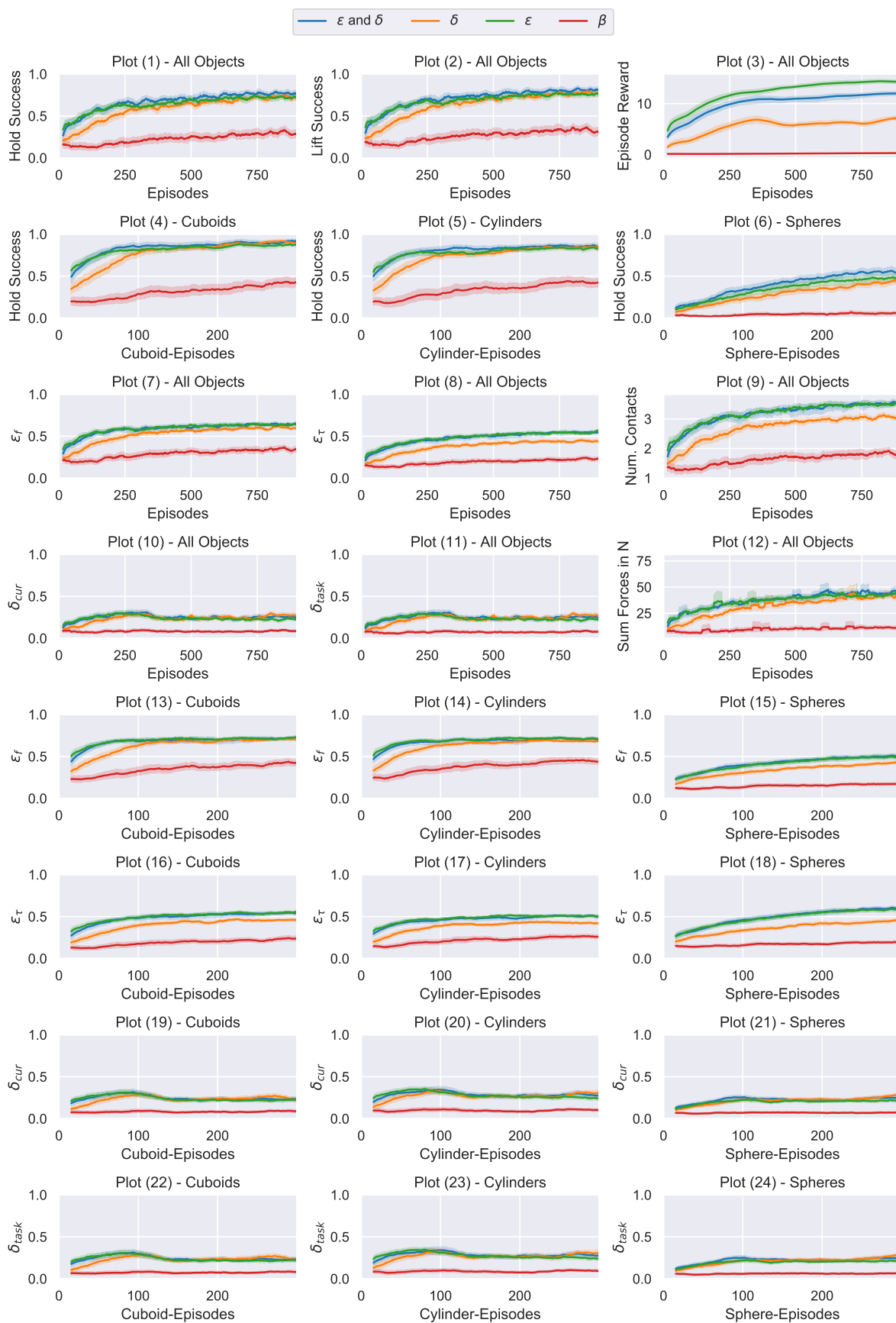


Figure 4.11: Train results for reward frameworks defined in Figure 4.8 (includes data from [38]).

spheres as it outperforms all other frameworks.

- (4.11-f) Plot (6) also shows that the β framework especially struggles with spheres as success rates are close to 0.
- (4.11-g) Plots (7) and (8) show that all reward frameworks improve ϵ_f and ϵ_τ over time. However, the frameworks ϵ and δ and ϵ which explicitly optimize ϵ_f and ϵ_τ reach higher values for these metrics.
- (4.11-h) Plot (9) visualizes that the number of contacts are highest for the ϵ and δ and ϵ frameworks which include the geometric quality metrics ϵ_f and ϵ_τ . The number of contacts for δ is lower, and hence it provides feedback about finger positioning, which differs from rewards that include ϵ_f and ϵ_τ . The δ reward assigns less importance to a large number of contacts. However, interestingly δ reaches similar success rates in Plot (1) as the other grasp quality-based rewards even with a smaller average number of contacts.
- (4.11-i) Plots (10) and (11) shows that all grasp quality based rewards (ϵ and δ , δ , and ϵ) reach similar values for δ_{cur} and δ_{task} , even ϵ which does not explicitly optimize it. Similar to the rewards in Plot (3) we observe a slight but systematic decrease in δ_{cur} and δ_{task} at approximately 350 episodes.
- (4.11-j) Plot (12) graphs the sum of the contact force magnitudes. The framework δ , which mainly produces feedback about the force distribution, leads to smaller contact force magnitudes in the early stages of training. The β framework does not learn to apply sufficient amounts of contact force and hence is the worst performing in Plot (1).
- (4.11-k) In Plots (13) to (18), we observe a similar trend as in Plots (7) and (8). The frameworks ϵ and δ and ϵ reach the highest values for ϵ_f and ϵ_τ . This trend is the same across each object type.
- (4.11-l) Plots (19) to (24) show that the trend from result (4.11-i) also holds for each object.
- (4.11-m) The δ_{cur} Plots (19) to (21) are almost identical to the δ_{task} Plots (22) to (24).

4.3.2 Testing

As described in section 4.2.3, we perform 240 experiments to test one model. We have 40 models with different seeds and four reward frameworks, which gives a total of $240 \times 40 \times 4 = 38400$ testing experiments. Figure 4.12 shows the test results from this large number of grasps. We also evaluate results from the supplementary video material¹ which demonstrate the learned strategies of the best performing models.

¹Video material available under <https://www.youtube.com/watch?v=9Bg8ZEAE0GI>

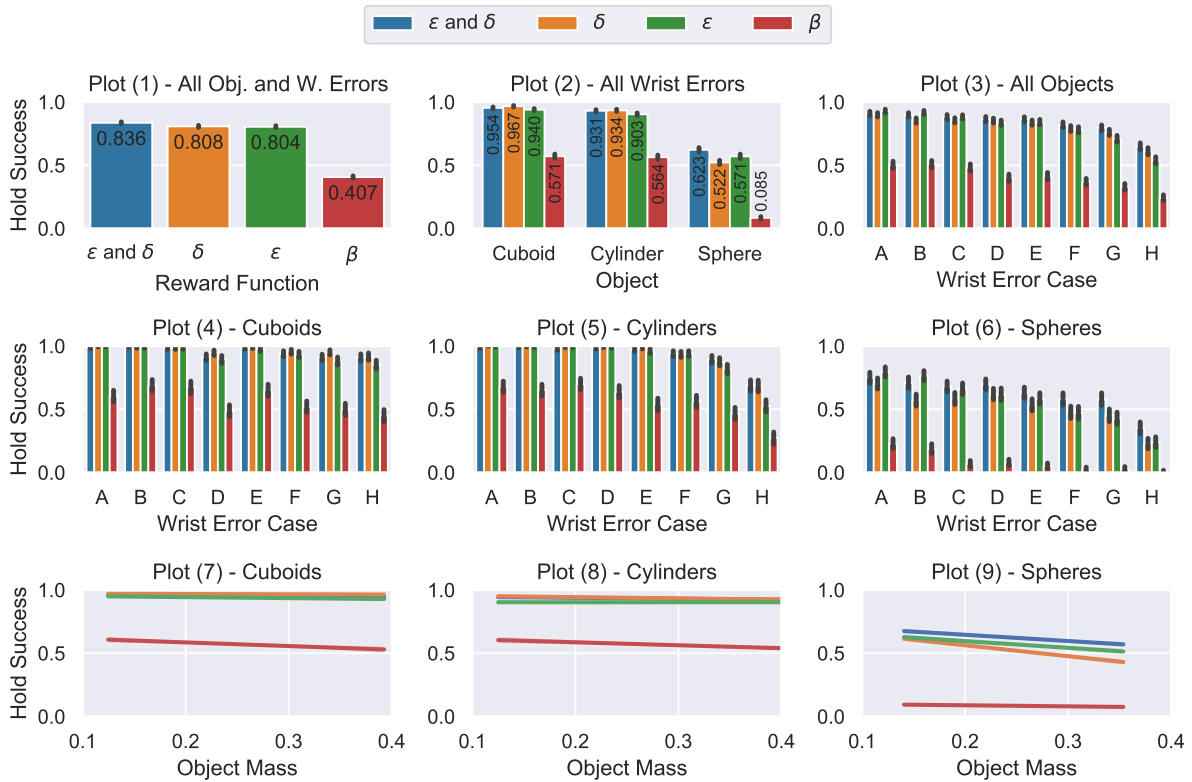


Figure 4.12: Test results for reward frameworks defined in Figure 4.8 (includes data from [38]).

Test Results Figure 4.12

- (4.12-a) **Major Result:** Plot (1) demonstrates that a reward that combines geometric grasp stability metrics ϵ with the force-agnostic metrics δ outperforms all other rewards at an average success rate of 83.6% over L2 wrist errors between 0 and 7 cm and 0 and 14 deg. The best performing reward ϵ and δ outperforms the binary reward baseline β by a large margin of 42.9%. Table 4.6 shows results of a one-sided, paired t-test to support this result.
- (4.12-b) **Major Result:** Plot (2) analyzes algorithm performance per object type. The per-object performance of the best reward framework ϵ and δ in Plot (2) is 95.4% for cuboids, 93.1% for cylinders, and 62.3% for spheres across all wrist errors from Table 4.4.
- (4.12-c) In Plot (2), we also see that, on average, spheres are harder to grasp as success rates are about half of the cuboid and cylinder success rates. Framework β almost always fails to grasp spheres.
- (4.12-d) Plot (2) shows that the metrics ϵ_f and ϵ_τ are especially valuable for spheres as performance of the ϵ and δ reward and the ϵ framework are comparatively large at

62.3% and 57.1%.

- (4.12-e) Plot (3) quantifies how the magnitude of the L2 wrist error relates to successfully lifting and holding the object. The relation appears to be non-linear (i.e., incremental wrist errors close to error case H affect grasp performance proportionally more than around case B).
- (4.12-f) Plots (4) to (6) visualize the success rates per object type across different wrist error cases from Table 4.4. All algorithms are comparatively robust across wrist errors for cuboids and cylinders in Plots (4) and (5), as success rates only slightly decrease for more significant errors. However, for spheres in Plot (6), all algorithms tend to be more sensitive to the size of the wrist error as we observe a more considerable drop in performance for greater errors.
- (4.12-g) Plots (7) to (9) show the relation between grasp success and object mass. The algorithms seem to be largely invariant to changes in object mass for cuboids and cylinders in Plots (7) and (8). However, for the β framework, we observe a slight inverse relation between object mass and grasp success for cuboids and cylinders. For spheres in Plot (9), we detect a more apparent result than for cuboids or cylinders: the grasp quality-based reward frameworks struggle more with heavier spheres.
- (4.12-h) **Major Result:** From the supplementary video material, we see that the ϵ and δ framework generates the most sensible-looking grasps. The δ framework completely fails to use one finger of the robotic hand on the evaluated objects. The strategies learned by the ϵ framework look similar to the behavior of ϵ and δ . The β framework faces a similar issue as δ since it only actuates two of the three fingers. By visual evaluation, it is clear that β does not put an adequate amount of pressure on the object, which we previously noted in result (4.11-j).

Result	$\mu_{\epsilon \text{ and } \delta} > \mu_{\delta}$	$\mu_{\epsilon \text{ and } \delta} > \mu_{\epsilon}$	$\mu_{\epsilon \text{ and } \delta} > \mu_{\beta}$
p-value	$3.1681 \cdot 10^{-10}$	$2.0510 \cdot 10^{-12}$	≈ 0.0

Table 4.6: Results of t-test for reward comparison. The mean of framework x is μ_x and ‘ ≈ 0.0 ’ means value was below machine precision (includes data from [38]).

4.4 Discussion

Answering RQ 1

Let us revisit **RQ 1** which this study investigates.

RQ 1: Which analytic grasp stability metrics are the most expressive reward functions for RL algorithms that refine grasps on three-fingered robotic hands receiving only tactile and joint position data as input?

From result (4.12-a) we conclude that a reward function that combines the geometric grasp quality metric ϵ with the force-agnostic measure δ is most expressive as it outperforms all other studied reward functions. The t-test in Table 4.6 shows that this claim is statistically significant since $p < 0.01$ for all comparisons. This means that ϵ and δ encode different information about grasp stability. They complement each other well as algorithms that trained with feedback from both ϵ and δ resulted in stronger overall policies.

Lower Success Rates for Spheres

The low success rates for spheres in results (4.12-b) and (4.12-c) could be due to the rolling motion that spheres are prone to. Cuboids and cylinders move comparatively less when touched by the robotic fingers or palm. As shown in Figure 4.9, we place spheres on concave mounts to reduce rolling. To understand whether the rolling motion reduces grasp success, we could check if a larger incline on the mount can improve performance for spheres.

Another hypothesis for the lower success rates for spheres is that spheres require different grasping strategies than cuboids or cylinders, and it is hard to unify them under one policy π_ϕ . It would be interesting to modify the action vector \mathbf{a}_t and add actuation to the fourth DOF and thereby allow finger separation. With this modification, spherical grasps would be possible, and it would be worth investigating whether the algorithms learn to use the fourth DOF for spheres.

The fact that spheres are harder to grasp also relates to the result (4.12-f), which states that algorithms that grasp spheres are more sensitive to wrist errors. Moreover, we see from Plot (2) in Figure 4.12 that the reward frameworks that are based on grasp quality metrics have similar success rates for cuboids and cylinders. However, for spheres the results differ by larger margins as noted in result (4.12-d). This means that the performance on spheres is the deciding factor that leads to our final conclusion in Plot (1) of Figure 4.12 that ϵ and δ outperforms all other metrics.

Lower Success Rates for Binary Reward β

In result (4.11-a), we showed that the reward in the β framework is less informative than the rewards that are based on quality metrics. This result is not surprising since it is well understood that shaped rewards are more sample efficient than sparse rewards [64]. The β framework may not represent the best alternative reward not based on analytic grasp quality metrics. Rather, it is a baseline that is often included in rewards of related works such as [56, 99]. We hypothesize that the

low contact forces the β framework applies to the object in results (4.11-j) and (4.12-h) contribute to low success rates. From Plot (1) in Figure 4.11, it is not entirely clear if the β framework already converged. Possibly training longer can increase its success rates further.

Video Demonstrations

Result (4.12-h) revealed that policies trained with the δ framework fail to actuate all three fingers of the hand. However, the frameworks that include ϵ do not face this issue. The latter frameworks include the quality metrics ϵ_f and ϵ_τ . We hypothesize that the frameworks trained with these quality metrics (i.e., the frameworks ϵ and δ and ϵ) learn to use all three fingers because ϵ_τ incentivizes contacts on all three fingers. As shown in Figures 4.4c and 4.4d, the value of ϵ_τ is considerably larger when there are contacts on each finger, because more disturbing torque on the object can be resisted. However, since only vertical acceleration and no external torque is applied in our experiments, grasping the object with two fingers, like in framework δ , can yield similar success rates as a grasp that contacts the object on more fingers. These results are supported by the similar success rates of δ (two-finger grasp) and ϵ (three-finger grasp) in Plot (1) of 4.12.

The fact that δ showed only two-finger grasps in our video demonstrations is also a possible explanation for the result (4.11-h), which showed that δ consistently has fewer contact points than the ϵ -based frameworks that typically perform three-finger grasps. Furthermore, it also explains the comparatively low sphere success rate for δ in result (4.12-d), as grasping a sphere with two fingers will only work robustly if grasped below the sphere’s equator. Cuboids and cylinders can be more easily gripped with two fingers, as the success rates in Plot (2) of Figure 4.12 indicate. Lastly, we have to mention that we could not possibly visually inspect all trained δ models and check whether the two-finger grasp strategy is consistent across all models and objects. However, the lower number of contact points in Plot (9) in Figure 4.11 could indicate that this observation is indeed accurate across the majority of algorithms trained with δ .

Object Mass and Grasp Success

In result (4.12-g), we found a negative correlation between object mass and grasp success. This relation was less prominent for cuboids and spheres. We expect this observation, and it is easily explainable with the larger gravitational forces that act on heavier objects. Heavier objects induce larger tangential forces at each contact and are therefore more likely to slip. Surprisingly, the grasp success of algorithms trained with physically inspired reward functions is largely invariant towards changes in the mass of cuboids and cylinders. It would be interesting to increase object masses further and see at which point the success rates drop. Note that the mass range in Plot (9) of Figure 4.12 is smaller than in Plots (7) and (8), which is a purely random effect because we sample the mass uniformly from the interval $[0.1, 0.4]$ kg.

Using Force-Agnostic Rewards

Result (4.11-d) highlighted a slight decrease in reward for the δ framework after approximately 350 episodes. Possibly this could be a limitation of the SAC [23] algorithm and the interplay with the simulated environment. While building the robot simulator with the DART physics engine, we found that the larger the simulated contact forces of a grasp get, the more unstable they become. Hence, a possible reason for result (4.11-d) could be that as contact forces in Plot (12) of Figure 4.11 rise throughout the learning procedure the contact forces get more unstable which is bound to negatively impact the calculation of the quality metrics in δ because they rely on accurate current force measurements $f_{i,cur}$. This issue needs to be addressed in future work. However, we can say that combining ϵ and δ improves the overall learning stability and can compensate for the drop in reward as shown in Plot (3) of Figure 4.11.

Result (4.11-m) revealed that the plots of δ_{cur} and δ_{task} are almost identical. This result suggests that we could have also only used δ_{cur} and not δ_{task} in our reward functions. It makes sense that a grasp with a high δ_{cur} and contact forces that are well away from the friction limits also has a large δ_{task} because it can compensate more task wrench. But it is counterintuitive that the values are so similar. When calculating δ_{task} , we assume that the contact forces are within realistic magnitudes and directions. However, physics engines often struggle to produce physically meaningful results when simulating robotic grasping [91]. In fact, if the contact forces $f_{i,cur}$ are much larger than the additional contact forces $f_{i,add}$ that appear due to the task wrench $w_t \in D$, then the metric $\delta_{task} \rightarrow \delta_{cur}$. Perhaps adapting the magnitude of the task wrench w_t to the simulated contact forces could make δ_{task} a more informative metric.

Computational Load

The scale of the data collection in this experiment alone is immense. Depending on the episode lengths, we conduct approximately 1000 grasping episodes to train each model. Further, we have four frameworks for which we run experiments over 40 random seeds each. Hence, we train $4 \times 40 = 160$ models in this experiment which conduct an approximate total number of $1000 \times 160 = 160000$ grasps during training. Additionally, we perform 38400 experiments to test the models, as explained in section 4.3.2. Overall, we conducted a total of $160000 + 38400 \approx 200000$ grasps in the training and testing stage combined. If we ran such a large number of experiments on a single real robot, it would take roughly 140 days (assuming one grasp and resetting procedure lasts one minute, and the experiments can run day and night). However, training and testing a single policy in simulation on a four central processing unit (CPU) machine takes approximately 24 hours, and simulations can run in parallel. Therefore, using a research cluster, we could obtain our results in a matter of a few days, demonstrating the power of training agents in simulations. Note that the Gazebo simulator is the most computationally intensive part of the system and not the RL algorithm.

5 Contact Sensing and Grasp Refinement

This chapter investigates the effect of contact sensor resolution on grasp refinement success. We will describe how we change the experimental setup from chapter 4 to examine this question. Further, we present empirical results and consequently answer **RQ 2**. This chapter is also a revision of the paper by Koenig et al. [38] which is currently under review and therefore includes some results presented in the publication. Several results of this experiment are similar to the ones shown in chapter 4 and will not be stated or discussed again.

5.1 Experimental Setup

To investigate **RQ 2**, we train our models with a varying state vector s_t . Except for the changing state vector, the experimental setup is the same as described in section 4.2. As shown in 5.1, we compare four contact sensing frameworks.

Framework	<i>full</i>	<i>normal</i>	<i>binary</i>	<i>none</i>
Joint Positions	yes	yes	yes	yes
Contact Normals	yes	yes	yes	no
Contact Positions	yes	yes	yes	no
Contact Forces	full vector	only normal	binary signal	no
Number of Inputs	70	56	56	7

Table 5.1: Inputs for different contact sensing frameworks.

1. The *full* framework receives the 3D contact force vector and therefore has the same state vector as the algorithms described in section 4.2.4.
2. The *normal* framework omits the tangential forces $f_{i,t}$ and only provides the magnitude of the normal force $\|f_{i,n}\|$ on each finger segment. See Figure 4.1 for a visualization of $f_{i,n}$ and $f_{i,t}$.
3. The *binary* framework receives binary feedback on whether a link is in contact $\{1\}$ or not $\{0\}$.
4. The *none* framework is only provided with the seven joint positions (one finger separation angle, three proximal joint angles and three distal joint angles). It receives no contact information and purely relies on proprioceptive joint position data.

We change the size of the input layer of the neural network that parametrizes our policy π_ϕ to match the dimension of each framework's state vector. The rest of the network remains unchanged to allow for a fair comparison. We use our best performing reward ϵ and δ from section 4.2. Therefore, all of the above contact frameworks indirectly receive contact feedback via the reward. The *full* contact sensing framework is the same as the ϵ and δ reward framework from section 4.2.

5.2 Results

5.2.1 Training

Like in section 4.3, all following plots show results over 40 random seeds per framework, we indicate ± 2 standard errors with the error bars, and we smooth the training plots in Figure 5.1 with a moving average filter of kernel size 30.

Train Results Figure 5.1

- (5.1-a) **Major Result:** The performance of the *full*, *normal*, and *binary* frameworks look almost identical across all training plots of Figure 5.1.
- (5.1-b) **Major Result:** In Plots (1) and (2), the *none* framework initially learns faster than the other frameworks. However, after approximately 300 episodes it plateaus at a slightly lower success rate than the contact-based frameworks (i.e., the *full*, *normal*, and *binary* frameworks).
- (5.1-c) For spheres in Plot (6), the *none* framework learns especially fast in the beginning but is also eventually superseded by the other frameworks.
- (5.1-d) The *none* framework initially learns to optimize the quality metrics in Plots (7) to (8), (10) to (11) and (13) to (24) better and has a larger average number of contacts in Plot (9) than the other frameworks before the 300th episode. Towards the end of the training process the *none* framework converges to similar final values as the other frameworks for the mentioned plots.
- (5.1-e) **Major Result:** The *none* framework consistently applies larger contact forces than the other frameworks in Plot (12).

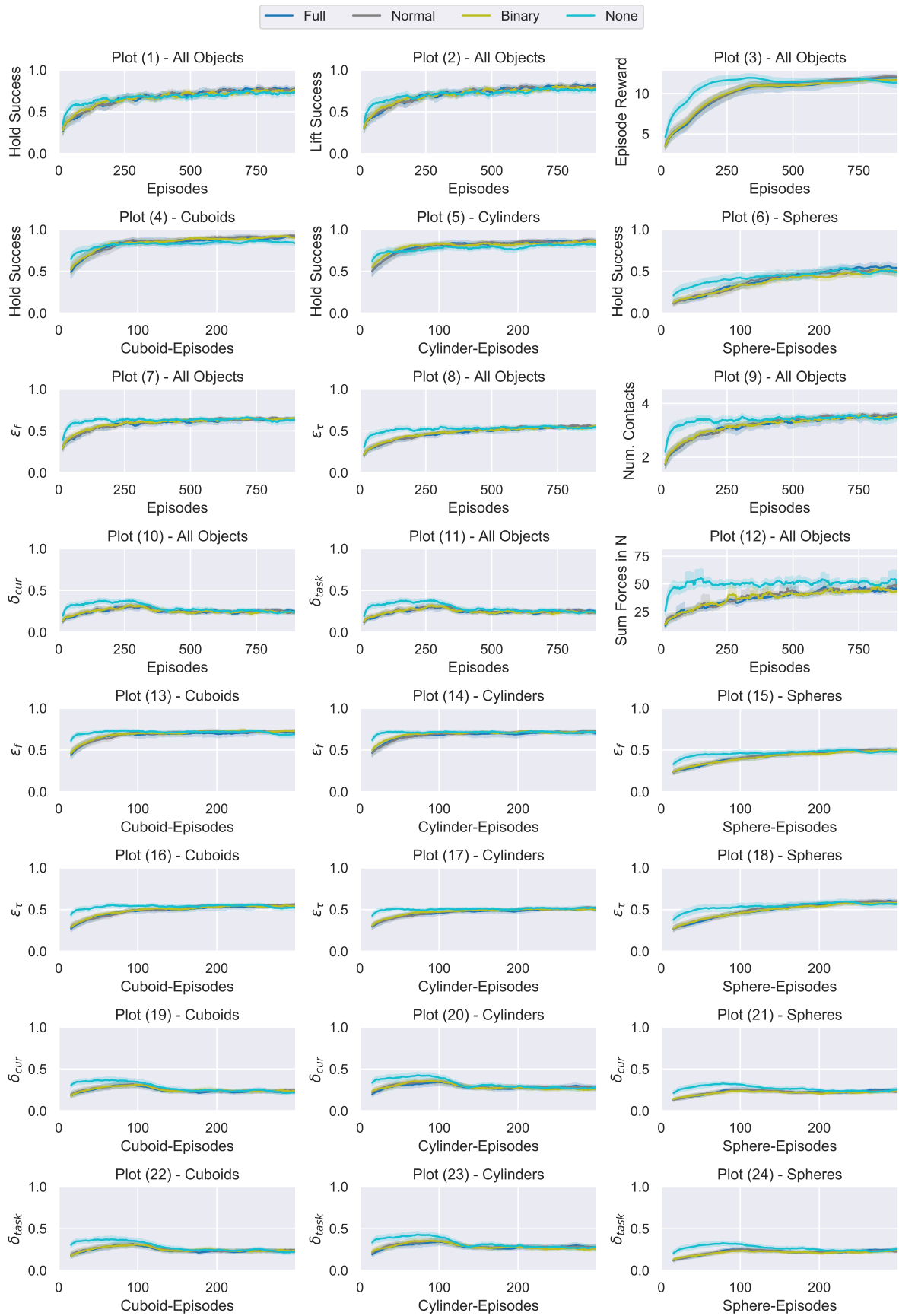


Figure 5.1: Train results for contact frameworks defined in Table 5.1 (includes data from [38]).

5.2.2 Testing

Like in section 4.3.2, we perform 38400 experiments to evaluate the four contact sensing frameworks. Figure 5.2 compares the test results.

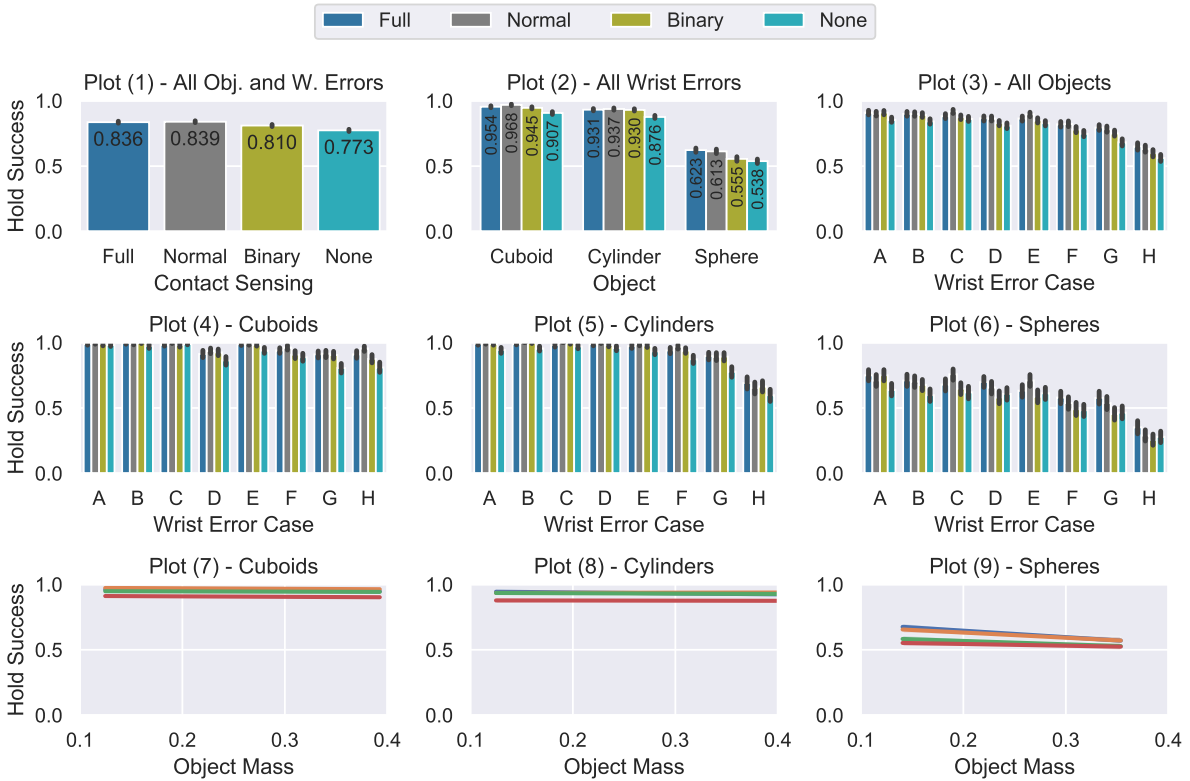


Figure 5.2: Test results for contact frameworks defined in Table 5.1 (includes data from [38]).

Test Results Figure 5.2

- (5.2-a) **Major Result:** Plot (1) shows that the *full*, *normal*, and *binary* frameworks which receive contact feedback outperform the *none* framework by 6.3%, 6.6% and 3.7%, respectively. Table 5.2 shows results of a one-sided, paired t-test investigating the statistical significance of the finding that the normal framework's mean performance μ_{normal} is the best of all frameworks.
- (5.2-b) **Major Result:** We also see in Plot (1) that giving *normal* force feedback marginally improves the *binary* framework by 2.9%. However, providing the *full* contact force vector only provides a benefit of 2.6% compared to the *binary* framework.
- (5.2-c) The *normal* framework, which performs the best after the defined number of training steps, has success rates of 96.8% for cuboids, 93.7% for cylinders, and 61.3% for spheres across all wrist errors.

(5.2-d) In Plot (2), it becomes clear that higher resolution contact feedback is particularly effective for spheres as we observe a more significant improvement from the *binary* to the *normal* and *full* frameworks for spheres than for cuboids and cylinders.

(5.2-e) **Major Result:** From the supplementary video material, we observe no fundamental difference in behavior of the learned grasping strategies. The only exception is that the *none* framework tends towards two-finger grasps on cuboids and cylinders.

Result	$\mu_{normal} > \mu_{full}$	$\mu_{normal} > \mu_{binary}$	$\mu_{normal} > \mu_{none}$
p-value	0.2232	$7.0177 \cdot 10^{-11}$	$1.3087 \cdot 10^{-46}$

Table 5.2: Results of t-test for contact sensing comparison (includes data from [38]).

5.3 Discussion

Answering RQ 2

This study answers our second research objective **RQ 2**.

RQ 2: What is the relation between contact sensing resolution and tactile grasp refinement success?

Result (5.2-a) shows the relation of interest. Our experiments found that the *none* baseline which does not receive contact information is improved by 3.7% through *binary* contact data, by 6.6% with accurate *normal* force information, and by 6.3% via the *full* force feedback. Overall, our main conclusion is that contact sensing improves success rates of tactile grasp refinement algorithms. These results align well with other research in the field which found that contact feedback increases the performance of RL grasping [56] and manipulation algorithms [53, 54]. The improvement for the *normal* force framework when compared to the *binary* and *none* frameworks are statistically significant since the p-values in Table 5.2 for the results $\mu_{normal} > \mu_{binary}$ and $\mu_{normal} > \mu_{none}$ are smaller than 0.01.

Comparing *Normal* and *Binary* Contact Feedback

Previous research [53, 54] concluded that accurate *normal* force feedback leads to similar success rates as *binary* contact information for in-hand manipulation. Therefore, in this experiment we specifically investigate whether these findings are reproducible in the tactile grasp refinement task. We find in result (5.2-b) that the 2.9% improvement from *binary* contact signals to accurate *normal* force readings is surprisingly small. But it is statistically significant, as mentioned above, since the p-value < 0.01 for $\mu_{normal} > \mu_{binary}$ in Table 5.2. However, from this marginal difference of 2.9% we can only conclude that our results are similar to the ones by Melnik et al. [53, 54], who find that their *normal* and *binary* frameworks perform approximately on par. Melnik et al. [53, 54] run on five random seeds, while we train over 40. Perhaps, if Melnik et al. [53, 54] ran more experiments over

more seeds they would also find a marginal, but statistically significant improvement by providing continuous *normal* force readings.

The Full Contact Sensing Framework

It is counter-intuitive that in (5.2-b) the performance increase of 2.9% yielded by the *normal* framework compared to the *binary* one is larger than the 2.6% of the *full* force framework. We expected that the *full* force framework provides valuable information about simulated tangential forces which we hypothesize to be prominent in the grasp refinement and lifting task. It is important to understand that the small performance difference of 0.3% is not statistically significant because p-value in Table 5.2 for $\mu_{normal} > \mu_{full}$ is > 0.01 . We therefore conclude that the *normal* and *full* framework perform approximately equally well. Nevertheless, let us discuss three potential reasons for why the *full* framework does not yield the highest overall performance.

1. As shown in Table 5.1, the *full* framework has the largest state vector $s_t \in \mathbb{R}^{70}$. Therefore, the *full* framework also has the most network parameters since the number of trainable parameters in the first layer of the neural network linearly scales with the number of input features. A larger network requires more training data, and therefore more environment interactions. Future experiments should run for longer and identify if this improves success rates.
2. As explained in section 4.2.4, we represent the tangential forces in the contact frame. To make sense of this representation, the policy π_ϕ will have to internally represent the concept of the friction cone to understand which tangential forces lead to slippage at a contact. This notion may be hard to learn through 25000 steps, a relatively small number of environment interactions. Agents trained with the SAC [23] algorithm often train millions of steps [23]. A factor that could complicate the learning of the friction cone concept is that the data from which the network learns contains mostly sparse contact data. The contact data is only non-zero if there is a contact on a link. Future investigations should explore an alternative representation of these forces to enhance the usability of tangential force for the algorithm. For example, the tangential force margins $\bar{f}_{i,cur}$ to the friction cone from Figure 4.5 could be given instead, which would also directly integrate the concept of the friction cone into the provided tangential force vector. In any case, it would be interesting to scrutinize to what extent the policy π_ϕ uses the tangential force information in its current state through a feature relevance analysis.
3. Simulated physical interactions are prone to instability due to contact chatter, especially in robotic grasping simulations [28, 91]. Therefore, the contact forces may not always point in physically meaningful directions as they attempt to satisfy the constraints of an LCP that becomes numerically unstable. Consequently, these simulated contact forces may not always constitute a good proxy of grasp success in simulation.

Similar Training Plots for *Full*, *Normal*, and *Binary*

Result (5.1-a) showed that the training performance of the *full*, *normal*, and *binary* frameworks looks almost identical across the training plots. In the training plots by Melnik et al. [53, 54], it is also hard to distinguish which framework is the best performing by visual inspection alone because the success rates lie close together. In our training plots, each data point only comes from one wrist error object combination (O, E) . Therefore, we only punctually evaluate the models in the training plots, which reflects poorly on their overall performance. Therefore, to draw solid conclusions, we should focus our attention on the test results, which provide a comprehensive model evaluation in 240 experiments over three object types and all wrist error cases. Using the test results, we made out differences in performance between the contact sensing frameworks as discussed above.

The *None* Framework Stands Out

Result (5.1-b) demonstrated that the *none* framework initially learns fastest, which is most probably due to its comparatively small state vector $s_t \in \mathbb{R}^7$. The *none* framework’s policy has less trainable parameters and therefore requires less training data.

Furthermore, we noted in result (5.1-e) that the *none* framework applied more contact force throughout the learning procedure. We hypothesize that the strategies of the *none* framework learned less adaptive grasping behaviors and output a positive finger increment for most if not all inputs. Policies that close the fingers regardless of the input would result in considerably larger force magnitudes. It would be interesting to study the responses of the network to varying inputs.

Finally, the *none* framework not only receives no feedback about contact forces but also receives no information on contact positions and normals, as shown in Table 5.1. Possibly the information on contact positions and normals also affects grasp refinement behavior.

Performance of *None* Framework

In result (5.2-a), we found statistically significant improvements by providing grasp refinement algorithms with contact information. However, the relative improvements to the *none* framework are surprisingly small at 6.3%, 6.6%, and 3.7%. Previous research [56] draws a similar conclusion and also finds that contact force measurements only improve algorithm performance by a small margin of 2% to 5% depending on the object.

The *none* framework reaches adequate grasp success rates of 90.7% for cuboids, 87.6% for cylinders, and improvable rates of 53.8% for spheres over all wrist errors in Plot (2) of Figure 5.2. These results suggest that when we provide an RL grasping algorithm with an expressive, contact-based reward, such as ϵ and δ in this case, it can abstract much of the information relevant for grasping most objects solely from joint positions. The conclusion that accurate contact sensing may not always be required is valuable for designers of robotic hands since precise tactile sensors are delicate and expensive hardware items. Furthermore, this conclusion once again stresses the importance of the reward function, as we discovered that reducing the information content in the β

reward framework led to a more drastic drop in performance in result (4.12-a) than reducing the expressiveness of the state vector.

Comparison to State-of-the-Art

Let us compare the results to state-of-the-art papers on tactile grasping with RL. Let us begin with the work by Merzić et al. [56]. The authors conducted three experiments.

1. Firstly, they train one policy per pre-grasp, object combination. A pre-grasp is a wrist pose generated in the vicinity of the object and is available in a database [36]. They train and test on the same five objects and pre-grasps. They obtain success rates of 91% for a donut, 62% for a bottle, 48% for a hammer, 46% for a drill, 37% for a tape.
2. Furthermore, they train one policy per object that copes with different initial wrist poses. Success rates drop to 56.3% for a donut, 69.1% for a bottle, 38.7% for a hammer, 25.6% for a drill, 41.5% for a tape.
3. Lastly, they find that adding noise to the object's pose slightly improves results to 56.3% for a donut, 70.6% for a bottle, 39.6% for a hammer, 28.0% for a drill, 44.4% for a tape.

Before comparing the results by Merzić et al. [56] with ours, let us highlight the differences between ours and their experimental setup.

- Most importantly, Merzić et al. [56] assume perfect knowledge on the object pose and twist, joint positions, and contact forces in their state vector. At the same time, we only provide the hand's joint positions in the *none* framework.
- They only actuate the fingers, and the wrist pose remains fixed. Further, we control the fingers via incremental position changes while they actuate the robotic hand through torque commands.
- They actuate all four DOF of the hand while we only actuate three DOF.
- In the first experiment, they test on previously *seen* initial pre-grasps and *known* objects, while we test on 240 *unseen* objects and wrist pose combinations.
- They use a perfect pre-calculated wrist pose from a database [36] and do not add errors to the pre-grasp. However, since their wrist pose is fixed, there would also be no way of compensating errors in the wrist pose.
- When simulating calibration errors, they add noise to the object position information in the state vector, but the actual object position stays the same. In our experiments, we move the initial wrist pose and the object's relative pose to the hand changes.

- The algorithms by Merzić et al. [56] were trained on *one* object, and hence users have to select the right policy to grasp this specific object. The algorithms will likely struggle on unseen objects. We train a single policy that unifies strategies to grasp different object types (cuboids, cylinders, and spheres) of any unknown size and mass within the defined ranges under a wrist error of up to 7 cm and 14 deg.

The *none* framework, which is the worst of all contact sensing frameworks, still reaches success rates of +85% for unseen cuboids and cylinders and +50% for spheres under *significant* calibration errors, and our best-performing frameworks reach picking rates of +93% and +55%, respectively. We can therefore conclude that even the results of the *none* framework, which makes considerably fewer assumptions on the state vector, are at least comparable to the results by Merzić et al. [56] if not exceeding them.

The work by Wu et al. [99] is somewhat more comparable to our project since they also actuate the wrist of the robot. While we learn the incremental wrist pose change, they apply a more classical method to update the wrist pose. They achieve impressive picking rates of approximately 95% for no calibration error and 90% when adding 7.5 cm wrist position errors. They report these results based on experiments in a simulator and mention similar success rates of $\approx 90\%$ for 5 cm error on the real robot. By looking at plots (4) and (5) of Figure 5.2, we see that our results for cuboids and cylinders are approximately equal to Wu et al. [99] when adding little to no wrist pose error. However, as discussed earlier, more significant errors of, for example, 7 cm and 14 deg in wrist error case H decrease performance disproportionately to roughly 60% to 85%. Hence, for significant wrist errors, Wu et al. [99] outperform our approach. Note that Wu et al. add no rotational error to the wrist, which complicates comparing the success rates and presumably makes their task slightly more manageable. Wu et al. [99] include only a few spherical objects in their dataset, and they only report object-specific success rates when adding no calibration error. Hence, we can only compare the success rates on spheres for the wrist error case A. Wu et al.'s [99] picking rates for spheres are approximately 30% higher than ours. While for spheres, Wu et al. [99] certainly outperform our approach, the results for grasp refinement on cuboids and cylinders are comparable.

Large Contact Forces

The contact forces in Plot (12) of Figures 4.11 and 5.1 appear to be quite large. Let us investigate what the minimum contact force on a two-finger grasp is to hold an object in place that weights f_g . Figure 5.3 shows a grasp with two contacts. The friction coefficient in the example is $\mu = 1$. Therefore the opening angle of the friction cone is 45 deg. To obtain the minimum contact forces to balance f_g we must apply the contact forces f_1 and f_2 on the boundary of the friction cone where tangential force is maximized and where the contacts are still in static contact and are not yet sliding. The contact forces must balance the object's weight $f_1 + f_2 = -f_g$. Therefore the tangential contact force at each contact must be $-\frac{f_g}{2}$ (and since $\mu = 1$ also the normal component must have the same magnitude). To achieve this tangential force in the shown configuration, the contact force magnitudes must be $\|f_i\| = \frac{\sqrt{2}f_g}{2}$ at each contact. Hence, the overall smallest

contact force magnitude f_c to balance the object weight f_g using two contacts is $\sqrt{2}f_g$ as shown in equation (5.1).

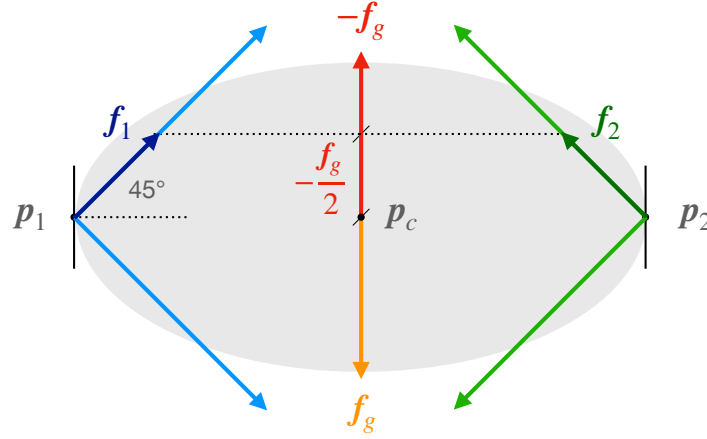


Figure 5.3: A grasp with two contact points p_1 and p_2 and friction cones at each contact. The vector $-f_g$ must balance the object's weight vector f_g . This is the configuration in which the sum of the contact force magnitudes $\sum_i^{n_c} \|f_i\|$ is minimized.

$$\min f_c = \min \sum_i^{n_c} \|f_i\| = 2 \frac{\sqrt{2}f_g}{2} = \sqrt{2}f_g \quad (5.1)$$

The mass range of our objects is 0.1 to 0.4 kg as indicated in Table 4.3. When sampling uniformly from this range the expected mean mass is $\frac{0.4\text{kg}+0.1\text{kg}}{2} = 0.25\text{kg}$. The minimum contact force to hold an object in place averaged over all objects should therefore be $\sqrt{2}f_g = \sqrt{2} \times 0.25\text{kg} \times 9.81 \frac{\text{N}}{\text{kg}} \approx 3.5\text{N}$. Let us make some assumptions to obtain a reasonable minimum force estimate for our experiment in Figure 4.8. Additionally to holding the object, the grasp must resist the vertical acceleration generated by lifting it. Let us be generous with this estimate and say it is half of the object's weight $0.5f_g$. Furthermore, we should add a safety margin to prevent slipping in uncertain situations. Humans tend to apply 10% to 40% more grasp force than would theoretically be necessary to prevent slipping [31]. Let us again be generous with this estimate and say the grasp on our robotic hand should apply 50% more grasp forces than would be required. Finally, we should expect roughly $3.5\text{N} \times 1.5 \times 1.5 = 7.9\text{N}$ of total grasp forces while executing the grasp defined in Figure 4.8. Note that this value may be different for a grasp with three contact points, but let us work with this value as a rough estimate.

The contact force magnitudes reported in Plot (12) of Figure 5.1 are in the order of 30N to 40N for the successful frameworks by the end of the training procedure. Using our previously calculated estimate, we can now say that the contact forces are about four to five times larger than expected. This conclusion is problematic, especially for fragile objects. It is known that physics engines sometimes have problems producing physically meaningful contact forces with realistic magnitudes as discussed in section 2.2.3 and [91]. Therefore, it would be essential to evaluate how large the contact forces get in the real world on robotic hardware to make a genuine claim if

the presented force measurements pose a problem. The *none* framework that receives only the joint position data would be especially suitable for such tests because this state information s_t is available on the real robotic hand.

6 Conclusion

6.1 Summary

In this work, we investigate two research questions. First, we examine the potential of using analytic grasp stability metrics as reward functions for RL algorithms that refine grasps on a three-fingered robotic hand. We design an experiment that includes refining a grasp on an object to compensate for calibration errors in the robot’s wrist pose. We compare the performance of several reward functions by lifting and holding the object in place after refinement and measuring the experiment’s outcome. We find classical methods of grasp analysis to be a valuable toolbox for designing rewards in robotic grasping as all of the rewards based on these techniques outperform a binary reward baseline. Furthermore, we show that stability metrics concerned with finger placement and rewards based on current force measurements complement each other well. A combination of both metrics yields the highest average success rates.

In a second experiment, we study the effect of contact sensor resolution on tactile grasp refinement success. We find that while tactile feedback marginally improves grasping outcomes, the policies trained with crude feedback from joint position data also reach adequate success rates on most objects when trained with rich, contact-based rewards. This conclusion highlights the importance of the thorough design of well-justified reward functions in robotic grasping. It also raises interesting questions about the significance of accurate tactile sensing on robotic hands in the context of RL. Last but not least, we compare our results to previous research in the field and conclude that our success rates are at least similar to the state-of-the-art if not exceeding some works.

6.2 Future Work

There are various exciting avenues for future work.

- As described in Figure 4.7, the agent can communicate with the real robotic hand easily. Future experiments should use this interface and test the performance of the agents trained in simulation on the real robotic hardware. It would be interesting to discover differences in how well each reward framework transfers to the real world. The *none* framework would be an ideal policy to test on the hardware first since it requires no contact data and hence works with little to no setup. The *normal* and *binary* policies are translatable to the real hand by approximating the contact positions and normals in the state vector through forward kinematics. Agents of the *full* contact framework assume tangential forces and can not be

transferred to the real hand as the hardware in its current form can only measure normal forces.

- To improve the success rates of spheres, the hand should actuate its fourth DOF in future experiments. Currently, we approximate the hand geometry with cuboids to allow for fast collision checking. Future training procedures should use the real finger and palm geometries of the hand. The rounded finger crosssections will hopefully improve success rates on spheres.
- It would be interesting to evaluate which input features of the agents are most relevant and influence the agent's action the most. Thereby, we could better understand the contribution of contact position and normal sensing to grasp success, which we did not explicitly evaluate in our experiments. The SHAP (SHapley Additive exPlanations) [49] feature importance algorithm could be an exciting method to conduct these experiments.
- Future work could also study the effect of contact position and normal sensing through ablation studies (i.e., train model without these inputs and check its performance).
- Furthermore, we should compare our results against a non-RL baseline. For example, we could use a simple open-loop controller that closes the hand's fingers using position increments. This comparison would give us an estimate of how helpful the iterative adjustments to the wrist and finger positions through RL are to more basic actuation schemes.
- Future reward functions should also include a force minimization term. Objects can be fragile, and it is also desirable to minimize the power consumption of the hand on mobile robotic systems.
- In the discussion sections 4.4 and 5.3 we already gave instructions for future experiments that investigate the hypotheses we made to explain certain results.

Bibliography

- [1] Y. Bai and C. K. Liu. “Dexterous manipulation using both palm and fingers.” In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 1560–1565. DOI: 10.1109/ICRA.2014.6907059.
- [2] L. Bo, X. Ren, and D. Fox. “Hierarchical Matching Pursuit for Image Classification: Architecture and Fast Algorithms.” In: *Advances in Neural Information Processing Systems*. Ed. by J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger. Vol. 24. Curran Associates, Inc., 2011.
- [3] C. Borst, M. Fischer, and G. Hirzinger. “A fast and robust grasp planner for arbitrary 3D objects.” In: *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*. Vol. 3. 1999, 1890–1896 vol.3. DOI: 10.1109/ROBOT.1999.770384.
- [4] C. Borst, M. Fischer, and G. Hirzinger. “Grasp planning: how to choose a suitable task wrench space.” In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*. Vol. 1. 2004, 319–325 Vol.1. DOI: 10.1109/ROBOT.2004.1307170.
- [5] M. Breyer, F. Furrer, T. Novkovic, R. Siegwart, and J. I. Nieto. “Flexible Robotic Grasping with Sim-to-Real Transfer based Reinforcement Learning.” In: *CoRR* abs/1803.04996 (2018). arXiv: 1803.04996.
- [6] M. Buss, H. Hashimoto, and J. Moore. “Dextrous hand grasping force optimization.” In: *IEEE Transactions on Robotics and Automation* 12.3 (1996), pp. 406–418. DOI: 10.1109/70.499823.
- [7] 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.” In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3300–3307.
- [8] Y. Chebotar, K. Hausman, Z. Su, G. S. Sukhatme, and S. Schaal. “Self-supervised regrasp-ing using spatio-temporal tactile features and reinforcement learning.” In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 1960–1966. DOI: 10.1109/IR05.2016.7759309.
- [9] Y. Chebotar, K. Hausman, O. Kroemer, G. S. Sukhatme, and S. Schaal. “Generalizing regrasping with supervised policy learning.” In: *International Symposium on Experimental Robotics*. Springer. 2016, pp. 622–632.

- [10] C. A. Coulomb. *Théorie des machines simples en ayant égard au frottement de leurs parties et à la roideur des cordages*. Bachelier, 1821.
- [11] E. Coumans et al. “Bullet physics library.” In: *Open source: bulletphysics.org* (2013).
- [12] E. Coumans and Y. Bai. *PyBullet, a Python module for physics simulation for games, robotics and machine learning*. <http://pybullet.org>. 2016–2019.
- [13] H. Dang and P. K. Allen. “Stable grasping under pose uncertainty using tactile feedback.” In: *Autonomous Robots* 36.4 (2014), pp. 309–330.
- [14] Z. Deng, Y. Jonetzko, L. Zhang, and J. Zhang. “Grasping force control of multi-fingered robotic hands through tactile sensing for object stabilization.” In: *Sensors* 20.4 (2020), p. 1050.
- [15] E. Donlon, S. Dong, M. Liu, J. Li, E. Adelson, and A. Rodriguez. “Gelslim: A high-resolution, compact, robust, and calibrated tactile-sensing finger.” In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 1927–1934.
- [16] E. Drumwright, J. Hsu, N. Koenig, and D. Shell. “Extending Open Dynamics Engine for Robotics Simulation.” In: *Simulation, Modeling, and Programming for Autonomous Robots*. Ed. by N. Ando, S. Balakirsky, T. Hemker, M. Reggiani, and O. von Stryk. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 38–50. ISBN: 978-3-642-17319-6.
- [17] C. de Farias, N. Marturi, R. Stolkin, and Y. Bekiroglu. “Simultaneous Tactile Exploration and Grasp Refinement for Unknown Objects.” In: *IEEE Robotics and Automation Letters* (Feb. 10, 2021).
- [18] J. Felip and A. Morales. “Robust sensor-based grasp primitive for a three-finger robot hand.” In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2009, pp. 1811–1816. DOI: 10.1109/IR0S.2009.5354760.
- [19] C. Ferrari and J. Canny. “Planning optimal grasps.” In: *Proceedings 1992 IEEE International Conference on Robotics and Automation*. 1992, 2290–2295 vol.3. DOI: 10.1109/ROBOT.1992.219918.
- [20] S. Fujimoto, H. Hoof, and D. Meger. “Addressing function approximation error in actor-critic methods.” In: *International Conference on Machine Learning*. PMLR. 2018, pp. 1587–1596.
- [21] K. Ganguly, B. Sadrfaridpour, K. B. Kidambi, C. Fermüller, and Y. Aloimonos. “Grasping in the Dark: Compliant Grasping using Shadow Dexterous Hand and BioTac Tactile Sensor.” In: *arXiv preprint arXiv:2011.00712* (2020).
- [22] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor*. 2018. arXiv: 1801.01290 [cs.LG].
- [23] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al. “Soft actor-critic algorithms and applications.” In: *arXiv preprint arXiv:1812.05905* (2018).

- [24] K. He, X. Zhang, S. Ren, and J. Sun. “Deep residual learning for image recognition.” In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.
- [25] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf. “Support vector machines.” In: *IEEE Intelligent Systems and their Applications* 13.4 (1998), pp. 18–28. DOI: 10.1109/5254.708428.
- [26] F. R. Hogan, M. Bauza, O. Canal, E. Donlon, and A. Rodriguez. “Tactile regrasp: Grasp adjustments via simulated tactile transformations.” In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 2963–2970.
- [27] H. van Hoof, T. Hermans, G. Neumann, and J. Peters. “Learning robot in-hand manipulation with tactile features.” In: *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. 2015, pp. 121–127. DOI: 10.1109/HUMANOIDS.2015.7363524.
- [28] J. M. Hsu and S. C. Peters. “Extending Open Dynamics Engine for the DARPA Virtual Robotics Challenge.” In: *Proceedings of the 4th International Conference on Simulation, Modeling, and Programming for Autonomous Robots - Volume 8810*. SIMPAR 2014. Bergamo, Italy: Springer-Verlag, 2014, pp. 37–48. ISBN: 9783319118994. DOI: 10.1007/978-3-319-11900-7_4.
- [29] W. Hu, C. Yang, K. Yuan, and Z. Li. *Reaching, Grasping and Re-grasping: Learning Multimode Grasping Skills*. 2020. arXiv: 2002.04498 [cs.R0].
- [30] S. Ivaldi, V. Padois, and F. Nori. *Tools for dynamics simulation of robots: a survey based on user feedback*. 2014. arXiv: 1402.7050 [cs.R0].
- [31] R. S. Johansson and J. R. Flanagan. “Coding and use of tactile signals from the fingertips in object manipulation tasks.” In: *Nature Reviews Neuroscience* 10.5 (2009), pp. 345–359.
- [32] I. T. Jolliffe. “Principal Component Analysis and Factor Analysis.” In: *Principal Component Analysis*. New York, NY: Springer New York, 1986, pp. 115–128. ISBN: 978-1-4757-1904-8. DOI: 10.1007/978-1-4757-1904-8_7.
- [33] S. Joshi, S. Kumra, and F. Sahin. “Robotic grasping using deep reinforcement learning.” In: *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*. IEEE. 2020, pp. 1461–1466.
- [34] I. Kao, K. Lynch, and J. W. Burdick. “Contact Modeling and Manipulation.” In: *Springer Handbook of Robotics*. Ed. by B. Siciliano and O. Khatib. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 647–669. ISBN: 978-3-540-30301-5. DOI: 10.1007/978-3-540-30301-5_28.
- [35] D. Kappler, J. Bohg, and S. Schaal. “Leveraging big data for grasp planning.” In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 4304–4311. DOI: 10.1109/ICRA.2015.7139793.

- [36] D. Kappler, J. Bohg, and S. Schaal. “Leveraging big data for grasp planning.” In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 4304–4311. DOI: 10.1109/ICRA.2015.7139793.
- [37] K. Kleeberger, R. Bormann, W. Kraus, and M. F. Huber. “A survey on learning-based robotic grasping.” In: *Current Robotics Reports* (2020), pp. 1–11.
- [38] A. Koenig, Z. Liu, L. Janson, and R. Howe. *Tactile Grasp Refinement using Deep Reinforcement Learning and Analytic Grasp Stability Metrics*. 2021. arXiv: 2109.11234 [cs.RD].
- [39] A. Koenig, F. Rodriguez y Baena, and R. Secoli. “Gesture-Based Teleoperated Grasping for Educational Robotics.” In: *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), 2021*. 2021.
- [40] N. Koenig and A. Howard. “Design and use paradigms for Gazebo, an open-source multi-robot simulator.” In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 3. 2004, 2149–2154 vol.3. DOI: 10.1109/IROS.2004.1389727.
- [41] N. Koenig and A. Howard. “Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator.” In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Sendai, Japan, Sept. 2004, pp. 2149–2154.
- [42] O. Kroemer, R. Detry, J. Piater, and J. Peters. “Combining active learning and reactive control for robot grasping.” In: *Robotics and Autonomous Systems* 58.9 (2010). Hybrid Control for Autonomous Systems, pp. 1105–1116. ISSN: 0921-8890. DOI: <https://doi.org/10.1016/j.robot.2010.06.001>.
- [43] J. Lee, M. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S. Srinivasa, M. Stilman, and K. Liu. “DART: Dynamic Animation and Robotics Toolkit.” In: *The Journal of Open Source Software* 3 (Feb. 2018), p. 500. DOI: 10.21105/joss.00500.
- [44] M. Li, Y. Bekiroglu, D. Kragic, and A. Billard. “Learning of grasp adaptation through experience and tactile sensing.” In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2014, pp. 3339–3346. DOI: 10.1109/IROS.2014.6943027.
- [45] Q. Li, O. Kroemer, Z. Su, F. F. Veiga, M. Kaboli, and H. J. Ritter. “A review of tactile information: Perception and action through touch.” In: *IEEE Transactions on Robotics* 36.6 (2020), pp. 1619–1634.
- [46] Y. Li, Q. Lei, C. Cheng, G. Zhang, W. Wang, and Z. Xu. “A review: machine learning on robotic grasping.” In: *Eleventh International Conference on Machine Vision (ICMV 2018)*. Ed. by A. Verikas, D. P. Nikolaev, P. Radeva, and J. Zhou. Vol. 11041. International Society for Optics and Photonics. SPIE, 2019, pp. 775–783. DOI: 10.1117/12.2522945.
- [47] Z. Li and S. Sastry. “Task-oriented optimal grasping by multifingered robot hands.” In: *IEEE Journal on Robotics and Automation* 4.1 (1988), pp. 32–44. DOI: 10.1109/56.769.
- [48] Q. Lu, M. V. der Merwe, B. Sundaralingam, and T. Hermans. *Multi-Fingered Grasp Planning via Inference in Deep Neural Networks*. 2020. arXiv: 2001.09242 [cs.RD].

- [49] S. M. Lundberg and S.-I. Lee. “A Unified Approach to Interpreting Model Predictions.” In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., 2017, pp. 4765–4774.
- [50] H. R. Maei, C. Szepesvari, S. Bhatnagar, D. Precup, D. Silver, and R. S. Sutton. “Convergent temporal-difference learning with arbitrary smooth function approximation.” In: *NIPS*. 2009, pp. 1204–1212.
- [51] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg. “Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics.” In: *CoRR* abs/1703.09312 (2017). arXiv: 1703.09312.
- [52] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kröger, J. Kuffner, and K. Goldberg. “Dex-Net 1.0: A cloud-based network of 3D objects for robust grasp planning using a Multi-Armed Bandit model with correlated rewards.” In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 1957–1964. DOI: 10.1109/ICRA.2016.7487342.
- [53] A. Melnik, L. Lach, M. Plappert, T. Korthals, R. Haschke, and H. Ritter. “Tactile sensing and deep reinforcement learning for in-hand manipulation tasks.” In: *IROS Workshop on Autonomous Object Manipulation*. 2019.
- [54] A. Melnik, L. Lach, M. Plappert, T. Korthals, R. Haschke, and H. Ritter. “Using Tactile Sensing to Improve the Sample Efficiency and Performance of Deep Deterministic Policy Gradients for Simulated In-Hand Manipulation Tasks.” In: *Frontiers in Robotics and AI* 8 (2021), p. 57. ISSN: 2296-9144. DOI: 10.3389/frobt.2021.538773.
- [55] D. Merkel. “Docker: lightweight linux containers for consistent development and deployment.” In: *Linux journal* 2014.239 (2014), p. 2.
- [56] H. Merzić, M. Bogdanović, D. Kappler, L. Righetti, and J. Bohg. “Leveraging Contact Forces for Learning to Grasp.” In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 3615–3621. DOI: 10.1109/ICRA.2019.8793733.
- [57] B. Mirtich and J. Canny. “Easily computable optimum grasps in 2-D and 3-D.” In: *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*. IEEE. 1994, pp. 739–747.
- [58] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. “Human-level control through deep reinforcement learning.” In: *nature* 518.7540 (2015), pp. 529–533.
- [59] M. Q. Mohammed, K. L. Chung, and C. S. Chyi. “Review of Deep Reinforcement Learning-based Object Grasping: Techniques, Open Challenges and Recommendations.” In: *IEEE Access* (2020).

- [60] D. Morrison, P. Corke, and J. Leitner. "Closing the Loop for Robotic Grasping: A Real-time, Generative Grasp Synthesis Approach." In: *Proc. of Robotics: Science and Systems (RSS)*. 2018.
- [61] D. Morrison, P. Corke, and J. Leitner. "Learning robust, real-time, reactive robotic grasping." In: *The International Journal of Robotics Research* 39.2-3 (2020), pp. 183–201. DOI: 10.1177/0278364919859066. eprint: <https://doi.org/10.1177/0278364919859066>.
- [62] A. Murali, Y. Li, D. Gandhi, and A. Gupta. "Learning to Grasp Without Seeing." In: *Proceedings of the 2018 International Symposium on Experimental Robotics*. Ed. by J. Xiao, T. Kröger, and O. Khatib. Cham: Springer International Publishing, 2020, pp. 375–386. ISBN: 978-3-030-33950-0.
- [63] Y. S. Narang, B. Sundaralingam, K. Van Wyk, A. Mousavian, and D. Fox. "Interpreting and predicting tactile signals for the SynTouch Biotac." In: *arXiv preprint arXiv:2101.05452* (2021).
- [64] A. Y. Ng, D. Harada, and S. Russell. "Policy invariance under reward transformations: Theory and application to reward shaping." In: *In Proceedings of the Sixteenth International Conference on Machine Learning*. Morgan Kaufmann, 1999, pp. 278–287.
- [65] L. U. Odhner, L. P. Jentoft, M. R. Claffee, N. Corson, Y. Tenzer, R. R. Ma, M. Buehler, R. Kohout, R. D. Howe, and A. M. Dollar. "A compliant, underactuated hand for robust manipulation." In: *The International Journal of Robotics Research* 33.5 (2014), pp. 736–752.
- [66] S. Olivier. *Lecture notes on the Soft Actor-Critic algorithm*. URL: <http://pages.isir.upmc.fr/~sigaud/teach/sac.pdf>.
- [67] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal. "Online movement adaptation based on previous sensor experiences." In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2011, pp. 365–371. DOI: 10.1109/IR0S.2011.6095059.
- [68] O.-M. Pedersen, E. Misimi, and F. Chaumette. "Grasping unknown objects by coupling deep reinforcement learning, generative adversarial networks, and visual servoing." In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 5655–5662.
- [69] W. PENFIELD and E. BOLDREY. "SOMATIC MOTOR AND SENSORY REPRESENTATION IN THE CEREBRAL CORTEX OF MAN AS STUDIED BY ELECTRICAL STIMULATION¹." In: *Brain* 60.4 (Dec. 1937), pp. 389–443. ISSN: 0006-8950. DOI: 10.1093/brain/60.4.389. eprint: <https://academic.oup.com/brain/article-pdf/60/4/389/948982/60-4-389.pdf>.
- [70] J. Peters, K. Mülling, and Y. Altün. "Relative Entropy Policy Search." In: *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*. AAAI'10. Atlanta, Georgia: AAAI Press, 2010, pp. 1607–1612.

- [71] S. Peters and J. Hsu. *Comparison of Rigid Body Dynamic Simulators for Robotic Simulation in Gazebo*. https://www.osrfoundation.org/wordpress2/wp-content/uploads/2015/04/roscon2014_scpeters.pdf. ROSCon 2014.
- [72] N. Pollard. "Synthesizing grasps from generalized prototypes." In: *Proceedings of IEEE International Conference on Robotics and Automation*. Vol. 3. 1996, 2124–2130 vol.3. DOI: 10.1109/ROBOT.1996.506184.
- [73] D. Prattichizzo and J. C. Trinkle. "Grasping." In: *Springer Handbook of Robotics*. Ed. by B. Siciliano and O. Khatib. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 671–700. ISBN: 978-3-540-30301-5. DOI: 10.1007/978-3-540-30301-5_29.
- [74] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. "ROS: an open-source Robot Operating System." In: *ICRA Workshop on Open Source Software*. 2009.
- [75] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann. *Stable Baselines3*. <https://github.com/DLR-RM/stable-baselines3>. 2019.
- [76] J. Reinecke, A. Dietrich, F. Schmidt, and M. Chalon. "Experimental comparison of slip detection strategies by tactile sensing with the BioTac® on the DLR hand arm system." In: *2014 IEEE international Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 2742–2748.
- [77] RightHand Robotics. *ReFlex Product Page*. URL: <https://www.labs.righthandrobotics.com/reflexhand> (visited on 11/07/2021).
- [78] M. A. Roa and R. Suárez. "Grasp quality measures: review and performance." In: *Autonomous robots* 38.1 (2015), pp. 65–88.
- [79] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. 3rd. USA: Prentice Hall Press, 2009. ISBN: 0136042597.
- [80] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. "Trust Region Policy Optimization." In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by F. Bach and D. Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 1889–1897.
- [81] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. "Trust Region Policy Optimization." In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by F. Bach and D. Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 1889–1897.
- [82] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347 [cs.LG].
- [83] W. Schultz. "Predictive reward signal of dopamine neurons." In: *Journal of neurophysiology* 80.1 (1998), pp. 1–27.
- [84] H. Seyffarth and D. Denny-Brown. "The grasp reflex and the instinctive grasp reaction." In: *Brain: a journal of neurology* (1948).

- [85] M. Shah, R. D. Eastman, and T. Hong. “An Overview of Robot-Sensor Calibration Methods for Evaluation of Perception Systems.” In: *Proceedings of the Workshop on Performance Metrics for Intelligent Systems*. PerMIS '12. College Park, Maryland: Association for Computing Machinery, 2012, pp. 15–20. ISBN: 9781450311267. DOI: 10.1145/2393091.2393095.
- [86] C. E. Shannon. “A mathematical theory of communication.” In: *The Bell system technical journal* 27.3 (1948), pp. 379–423.
- [87] M. A. Sherman, A. Seth, and S. L. Delp. “Simbody: multibody dynamics for biomedical research.” In: *Procedia IUTAM* 2 (2011). IUTAM Symposium on Human Body Dynamics, pp. 241–261. ISSN: 2210-9838. DOI: <https://doi.org/10.1016/j.piutam.2011.04.023>.
- [88] D. Silver. *Lectures on Reinforcement Learning*. URL: <https://www.davidsilver.uk/teaching/>. 2015.
- [89] D. Silver, S. Singh, D. Precup, and R. S. Sutton. “Reward is enough.” In: *Artificial Intelligence* 299 (2021), p. 103535. ISSN: 0004-3702. DOI: <https://doi.org/10.1016/j.artint.2021.103535>.
- [90] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018. ISBN: 0262039249.
- [91] J. R. Taylor, E. M. Drumwright, and J. Hsu. “Analysis of grasping failures in multi-rigid body simulations.” In: *2016 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAN)*. 2016, pp. 295–301. DOI: 10.1109/SIMPAN.2016.7862410.
- [92] Y. Tenzer, L. P. Jentoft, and R. D. Howe. “The Feel of MEMS Barometers: Inexpensive and Easily Customized Tactile Array Sensors.” In: *IEEE Robotics Automation Magazine* 21.3 (2014), pp. 89–95. DOI: 10.1109/MRA.2014.2310152.
- [93] H. Van Hasselt, A. Guez, and D. Silver. “Deep reinforcement learning with double q-learning.” In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 30. 1. 2016.
- [94] F. Veiga, R. Akrouf, and J. Peters. “Hierarchical Tactile-Based Control Decomposition of Dexterous In-Hand Manipulation Tasks.” In: *Frontiers in Robotics and AI* 7 (2020).
- [95] F. Veiga, B. Edin, and J. Peters. “Grip stabilization through independent finger tactile feedback control.” In: *Sensors* 20.6 (2020), p. 1748.
- [96] Q. Wan, R. P. Adams, and R. D. Howe. “Variability and predictability in tactile sensing during grasping.” In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 158–164. DOI: 10.1109/ICRA.2016.7487129.
- [97] C. J. Watkins and P. Dayan. “Q-learning.” In: *Machine learning* 8.3-4 (1992), pp. 279–292.
- [98] J. Weisz and P. K. Allen. “Pose error robust grasping from contact wrench space metrics.” In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 557–562. DOI: 10.1109/ICRA.2012.6224697.

-
- [99] B. Wu, I. Akinola, J. Varley, and P. Allen. “MAT: Multi-fingered adaptive tactile grasping via deep reinforcement learning.” In: *arXiv preprint arXiv:1909.04787* (2019).
 - [100] M. M. Zhang. “Tactile perception and Visuotactile integration for robotic exploration.” PhD thesis. University of Pennsylvania, 2019.
 - [101] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey, et al. “Maximum entropy inverse reinforcement learning.” In: *Aaai*. Vol. 8. Chicago, IL, USA. 2008, pp. 1433–1438.