UNIVERSITY OF
CAMBRIDGE
Department of Computer
Science and Technology

Investigating How Dexterity Affects Imitation Learning in Humanoid Robotic-Hand Simulations

Dylan Moss

Queens' College

June 2024

Submitted in partial fulfillment of the requirements for the
Computer Science Tripos, Part III

Total page count: 86

Main chapters (excluding front-matter, references and appendix): 42 pages (pp 1–42)

Main chapters word count: 11997

Methodology used to generate that word count:

```
$ texcount -inc -total -alphabets=Latin -sum=1,0,1 dissertation.tex
```

Declaration

I, Dylan Moss of Queens' College, being a candidate for Computer Science Tripos, Part III, hereby declare that this project report and the work described in it are my own work, unaided except as may be specified below, and that the project report does not contain material that has already been used to any substantial extent for a comparable purpose. In preparation of this project report I did not use text from AI-assisted platforms generating natural language answers to user queries, including but not limited to ChatGPT. I am content for my project report to be made available to the students and staff of the University.

Signed: Dylan Moss

Date: 27.05.24

Abstract

For nearly forty years, humanoid robotics research has strived to develop biomimetic robotic hands which match the dexterity of human hands. Although this goal has yet to be achieved, recent advancements in the field suggest that this milestone will soon be reached.

Despite approaching human-like dexterity in robotic hands, limited research has explored how these improvements might affect a robot's ability to complete human-orientated tasks. To address this gap, our project proposes a **novel investigation** to answer the question: *does improving a robotic-hand's dexterity (up to and including that of a human hand) improve the robot's ability to be controlled by, and to learn from humans?*

We compare the dexterity of existing robotic hands — which deliberately omit degrees of freedom (DOF) for engineering practicality — to a robotic-hand model with human-like dexterity. We investigate this question through the lens of teleoperation (human operation of a robotic hand), reinforcement learning (RL), and imitation learning (IL).

No existing robotic hands — whether physical or simulated — have achieved 27 DOF. Our project builds the **first robotic-hand simulation with 27 DOF**, achieving **state-of-the-art dexterity** by developing the **first robotic-hand simulation to match the dexterity of a human hand**.

Using this hand, we perform **novel analysis** into how dexterity affects teleoperation, reinforcement learning, and imitation learning. Our work is the **first** to compare existing, less-dexterous robotic hands to a robotic hand with human-like dexterity. We achieve this by systematically removing DOF from our 27 DOF robotic-hand simulation, and comparing their performance on *manipulation tasks* (using a subset of objects from the GRASP taxonomy [1]). To our knowledge, this is the **first time** a robotic hand with *adjustable dexterity* has been used to conduct teleoperation, RL, and IL dexterity trials.

Our results show our **27 DOF hand model to outperform all other (less-dexterous) hand models on the majority of benchmarks**. Notably, this hand model **improves the average teleoperation success rate** by more than **15%** compared to the other models. Our work is the first to provide empirical evidence that improving dexterity — up to and including that of a human hand — improves the performance of teleoperation, reinforcement learning, and imitation learning.

Acknowledgements

I would firstly like to thank my supervisors, Neil Lawrence and Chapa Hewa Pelendage, for their support and guidance throughout this project. Thank you also to my bachelor's project supervisor, Alan Mycroft, for teaching me how to write technically and precisely.

I would like to acknowledge the vital role that the Queens' Computer Science community has played in life at Cambridge over the last four years. Thank you to everyone for building such a wonderful and welcoming community, which I hope will last for decades to come. Thank you especially to the Queens' DoSes (Andy, Alastair, Neil, Ramsey, Jasmin, and Challenger) for your care and support over the years, and your continued efforts to foster the Queens' Computer Science community.

I am very grateful to my friends and family, who have inspired and believed in me throughout my time at Cambridge. Special thanks to Nita, Peter, and Conall, for your love and encouragement over the past 22 years. Your unwavering support has given me opportunities which I could never have dreamed of.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Robots have revolutionised industries from manufacturing [2, 3, 4] to agriculture [5, 6] through their simple, robust, and efficient designs. Despite this, a crucial barrier has prevented their integration into our everyday lives: *our world is fundamentally designed for humans*. For robots to effectively interact with our world, they must achieve *human-like dexterity* [7, 8] — a goal not yet achieved due to the intricate complexity of the human body [9]. For nearly forty years, humanoid robotics research has strived to develop biomimetic robotic hands which match the dexterity of human hands. Although this goal has yet to be achieved (due to engineering constraints [10, 11]), recent advancements in the field suggest that this milestone will soon be reached [12, 13].

Despite approaching human-like dexterity in robotic hands, limited research has explored how these improvements might affect a robot's ability to complete human-orientated tasks. To address this gap, our project proposes a **novel investigation** to answer the question: *does improving a robotic-hand's dexterity (up to and including that of a human hand) improve the robot's ability to be controlled by, and to learn from humans?*

We compare the dexterity of existing robotic hands — which deliberately omit degrees of freedom (DOF) for engineering practicality — to a robotic-hand model with human-like dexterity. We investigate this question through the lens of teleoperation (human operation of a robotic hand), reinforcement learning (RL), and imitation learning (IL). RL trains a robot from scratch, whereas IL uses *expert demonstrations* to guide the training process.

The human hand is widely regarded as the pinnacle of human dexterity [14, 15]: with each hand containing 27 degrees of freedom (DOF) [16], representing the number of controllable parameters which govern the hand's state. Since hands are the primary mechanism humans use to interact with the world [17, 18], significant effort has been dedicated to refining robotic hands to achieve human-like dexterity. Despite this, no existing robotic hands — whether physical or simulated — have achieved 27 DOF. Our project builds the **first robotic-hand simulation with 27 DOF**, achieving **state-of-the-art dexterity** by developing the **first robotic-hand model to match the dexterity of a human hand**.

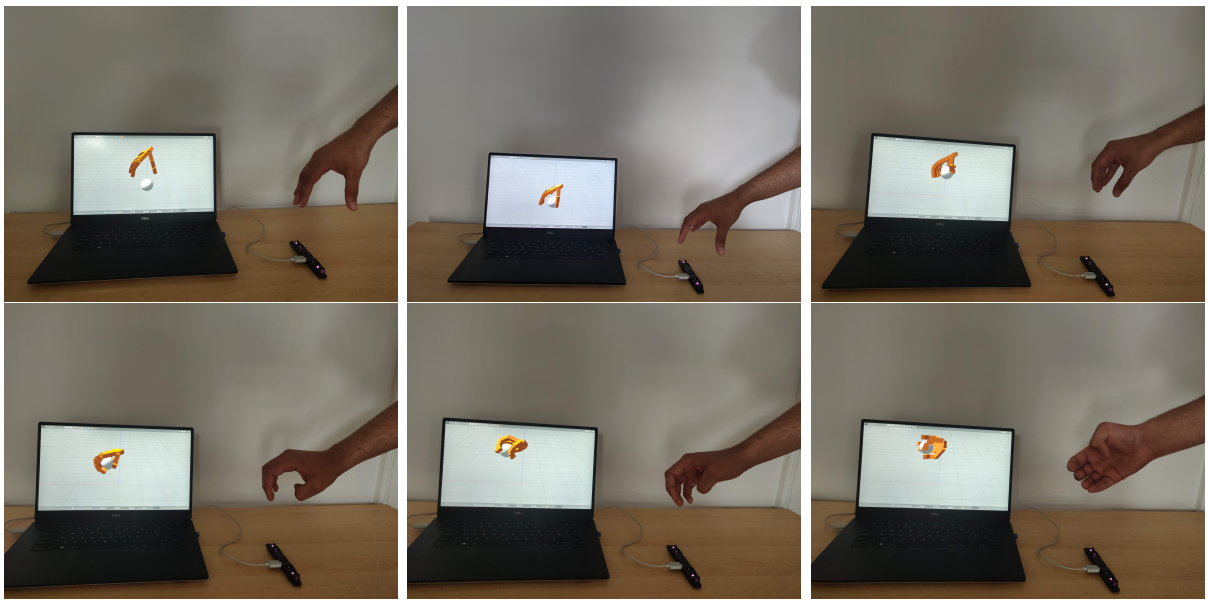

Figure 1.1: Pictures of a participant taking part in our *teleoperation* experiment: the robotic hand is controlled by the participant's hand movements (detected by the hand-tracking camera beneath their hand). This data is used to evaluate the teleoperation performance of each robotic-hand model (which contain varying levels of dexterities), as well as to collect *human expert demonstrations* for our imitation learning algorithms.

Using this hand, we perform **novel analysis** into how dexterity affects teleoperation, reinforcement learning, and imitation learning. Our work is the **first** to compare existing, less-dexterous robotic hands to a robotic hand with human-like dexterity. We achieve this by systematically removing DOF from our 27 DOF robotic-hand simulation, and evaluating its performance on *manipulation tasks* (using a subset of objects from the GRASP taxonomy [1]). To our knowledge, this is the **first time** a robotic hand with *adjustable dexterity* has been used to conduct teleoperation, RL, and IL dexterity trials.

Participant experiments are used to evaluate teleoperation performance, and to collect *human expert demonstrations*. These demonstrations act as training data, to teach our robotic hand to mimic human behaviour using *imitation learning* (IL). Intuitively, robotic hands with more human-like dexterity should be capable of learning from humans more efficiently and effectively. However, previous research is limited, with existing analyses restricted to rudimentary robotic hands with few degrees of freedom [19, 20]. Our thesis seeks to address this gap, by providing **novel insights** into how hand dexterity — ranging from basic to human-level — affects this learning process.

1.1 Project Overview & Contributions

This project explores how the dexterity of a robotic hand affects its ability to complete grasping tasks via teleoperation, reinforcement learning, and imitation learning. Our investigation aims to answer the following questions:

- *How does robotic-hand dexterity affect a human's ability to accurately control the robot (via teleoperation)?*
- *How does robotic-hand dexterity affect a robot's ability to learn **without** human demonstrations (via reinforcement learning)?*
- *How does robotic-hand dexterity affect a robot's ability to learn **with** human demonstrations (via imitation learning)?*

To the best of the author's knowledge, the project's novel contributions are as follows:

- We build the **first robotic-hand simulation simulation with 27 DOF**. Our hand achieves **state-of-the-art dexterity** for rigid-body human-hand simulations, being the **first to match the dexterity of a human hand**.
- We perform a **systematic literature review**, spanning **220 papers** and **254 robotic hands**, to show that no existing physical or simulated robotic hands have achieved 27 DOF.
- We perform **novel analysis into how dexterity affects teleoperation, reinforcement learning, and imitation learning**. To our knowledge, this is the **first time** a robotic hand with *adjustable dexterity* has been used to conduct teleoperation, RL, and IL dexterity trials.
- Our 27 DOF robotic-hand model (matching human dexterity) **outperformed all other (less-dexterous) models on the majority of benchmarks**, notably achieving a **15% higher average teleoperation success rate**. Hence, we are the first to provide empirical evidence that greater dexterity — up to and including that of a human hand — **improves the performance of teleoperation, reinforcement learning, and imitation learning**.
- We publish an open-source ROS2 (Robot Operating System[1]) package containing both our robotic hand and libraries for analysing the effect of hand dexterity. We hope the package will stimulate further work in this field.

[1]https://docs.ros.org/en/iron/index.html.

Chapter 2

Background

This chapter outlines the background material required to contextualise our project's methodology. Our investigation revolves around the construction of a robotic-hand simulation, which can complete grasping tasks via teleoperation, reinforcement learning, and imitation learning. Section 2.1 covers the *kinematics* knowledge required to describe our *human-hand kinematic model*. This model lays the foundation for our robotic-hand simulation implementation, and provides an abstract representation of the hand's dexterity and range of motion. Section 2.2 describes how the robotic-hand simulation can be trained to complete tasks in a reinforcement-learning environment. We discuss the remaining robotics concepts required for this thesis in Section 2.3: rigid vs soft bodied robots, PID controllers, and the ROS (Robotics Operating System) framework.

2.1 Kinematics

Kinematics studies the motion of objects, without consideration of their physical properties or forces which act on them [21]. In particular, we examine how different representations of *angular kinematics* can describe the rotation of an object around a fixed point (Section 2.1.1). This enables us to characterise the rotational joints which form the majority of the hand's *degrees of freedom* (Section 2.1.2). From this, we devise a *human-hand kinematic model* (Section 2.1.3): a kinematic abstraction over the true biomechanical structure of a human hand. The model describes the hand's bone and joint structure, which we use to manipulate the hand's dexterity during experiments.

2.1.1 Angular Kinematics

Angular kinematics — also referred to as *rotational kinematics* — describes the *rotation* and *orientation* of an object. The orientation of an object is defined by its rotations around the X, Y, and Z axes. These axes are defined *locally* to the joint's base; altering the joint's orientation changes the direction of its rotational axes. Quaternions are commonly used

in hand-tracking software to represent bone orientations in the human hand. These are converted into *Tait-Bryan* angles, which describe the rotation of robotic-hand joints along each axis (Section 2.1.1).

Quaternions

Quaternions extend complex numbers by adding two additional complex dimensions $\mathbf{j}$ and $\mathbf{k}$, such that $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1$. Quaternions are generally written in the form:

$$q = w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$$

Where the coefficients w, x, y, z are real numbers. The real component of a quaternion w acts as a scalar in $\mathbb{R}$, and the imaginary coefficients (x, y, z) act as a vector $\vec{v}$ in $\mathbb{R}^3$. Hence, quaternions can also be written as the conjunction of a scalar and vector component: $q = w + \vec{v}$. Multiplication and reciprocals of quaternions are defined by the following equations.

$$q_1 q_2 = (w_1 + \vec{v}_1)(w_2 + \vec{v}_2) = (w_1 w_2 - \vec{v}_1 \cdot \vec{v}_2) + (w_1 \vec{v}_2 + w_2 \vec{v}_1 + \vec{v}_1 \times \vec{v}_2)$$
$$q^{-1} = (w + \vec{v})^{-1} = \frac{w - \vec{v}}{w^2 + \|\vec{v}\|^2}$$

Unit quaternions (those who obey the property $\|q\| = \sqrt{w^2 + x^2 + y^2 + z^2} = 1$) are commonly used to represent *orientations*. Given two objects A and B with corresponding quaternion orientations q_1 and q_2, we can calculate the relative quaternion of B with respect to A using the equation $q_1^{-1} q_2$. In essence, we treat A's orientation as the new X-Y-Z axes basis, and calculate B's orientation with respect to this basis.

Tait-Bryan angles

Tait-Bryan angles represent rotations around the X, Y, and Z axes in any permutation. We use the notation $X_1 Y_2 Z_3$ to represent a rotation around the X axis first, followed by the Y axis, then the Z axis (where the numerical subscripts denote the ordering). We will focus on the $Z_1 Y_2 X_3$ Tait-Bryan angle representation, also referred to as *yaw-pitch-roll* (illustrated in Figure 2.1). All rotations can be described as three separate yaw (Z-axis), pitch (Y-axis), and roll (X-axis) rotation angles.

Given the relative quaternion between two bones, we wish to derive the quaternion's $Z_1 Y_2 X_3$ Tait-Bryan angle representation. This tells us the yaw-pitch-roll rotations of the joint which connects these two bones. The yaw-pitch-roll rotations of a quaternion $q = w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$ can be described by the following equation [22], where `atan2` is the 2-argument arctangent[1].

[1] `https://en.wikipedia.org/wiki/Atan2`.

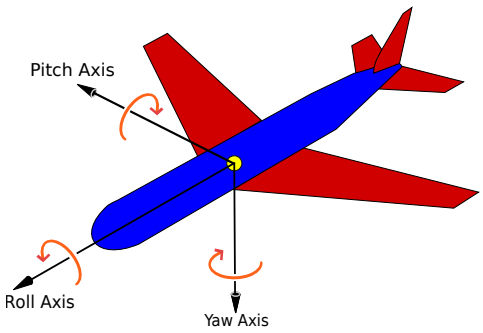

Figure 2.1: An illustration of the $Z_1Y_2X_3$ Tait-Bryan angle representation (also known as yaw-pitch-roll) [23].

$$\begin{bmatrix} \text{yaw} \\ \text{pitch} \\ \text{roll} \end{bmatrix} = \begin{bmatrix} \text{atan2}\left(2(wx+yz), 1-2(x^2+y^2)\right) \\ -\frac{\pi}{2} + 2\text{atan2}\left(\sqrt{1+2(wy-xz)}, \sqrt{1-2(y^2+z^2)}\right) \\ \text{atan2}\left(2(wz+xy), 1-2(y^2+z^2)\right) \end{bmatrix} \tag{2.1}$$

Key Takeaway. ***Unit quaternions*** *(written as $q = w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$, where $\|q\| = 1$) are useful for representing spacial orientations.* $Z_1Y_2X_3$ ***Tait-Bryan angle representations*** *(also known as* ***yaw-pitch-roll****) are useful for representing robotic joint rotations. Given the* ***relative quaternion*** *between two bones, we can calculate the yaw-pitch-roll angle representation of the joint connected these two bones.*

2.1.2 Degrees of Freedom

The number of *Degrees of Freedom* (DOF) in a robot refers to the smallest number of *controllable* parameters required to completely define the robot's state [24]. We refer to both the number of parameters, and the parameters themselves, as the robot's DOFs. In a robotic hand, these may consist of its: global position, global orientation, and joint rotation angles.

A hand's global position and orientation refers to the position and orientation of its *root* (from which all other hand components stem). The global position is defined as a vector from the environment's origin, and the global orientation can be described with a unit quaternion. Parameters which control a robot's global position and orientation are collectively known as its *global DOF*.

Joint rotations control the angle between two bones in a robotic hand. As described in Section 2.1.1, joint rotations can be represented as $X_1Y_2Z_3$ Tait-Bryan angles, with three controllable parameters (one for each axis). For example, a freely rotating joint would have 3 DOF, whereas a joint which can only rotate along the X-axis would have only 1 DOF. Degrees of freedom located *within* the robot, such as joint rotations, are known as *local DOF*.

However, it is possible for a robot to alter its state in ways which are not directly controllable. An example of this is a joint which cannot be controlled manually, but may bend when pressed up against a solid object. While the robot's state might change, this movement is not classified as a DOF as the parameters are not controllable. We instead refer to these parameters as *Degrees of Actuation* (DOA), which include both controllable and non-controllable parameters. Whilst all DOF are DOA, not all DOA are necessarily DOF.

Key Takeaway. ***DOF*** *are the smallest number of controllable parameters required to define a robot's state, whereas* ***DOA*** *encapsulate both controllable and uncontrollable variables.* ***Global DOF*** *refer to the robot's global position and orientation, whereas* ***local DOF*** *are DOF contained within the robot itself (such as joint rotations).*

2.1.3 Human-Hand Kinematic Model

The human hand contains a complex array of bones, muscles, ligaments, and tendons, which all interact when producing movement. Due to this complexity, humanoid robotic-hand research is far from being able to accurately replicating these biomechanisms [10, 11]. Instead, we can produce an *abstraction* over the hand's true structure using a *kinematic model*: a mathematical description of the hand's functional dimensions and DOF. This allows us to develop *rigid-body* robotic hands, which differ structurally from human hands but exhibit near-identical ranges of motion.

A kinematic model contains *links*, *joints*, and a set of *functional dimensions*. Links are one dimensional *rigid* structures connected together with joints; throughout this thesis, we assume joints to be only *rotational* (as opposed to *translational*). We refer to joints with 1 DOF (yaw or pitch) as *revolute joints*, and joints with 2 DOF (yaw and pitch) as *universal joints*. Functional dimensions refer to key measurements and parameters which describe a robot's physical structure and capabilities.

The functional dimensions of a human hand include the length of its links, and limits on the hand's joint rotations (for example, limiting joint angles to between 0 and $\frac{\pi}{2}$ radians). The functional dimensions of an average human hand have been well-studied; our model uses the dimensions measured by Cobos et al. [25].

Our human-hand kinematic model includes both the hand and wrist, but excludes the rest of the arm. This is because the arm only controls the position and orientation of the hand[2] — information already captured by the hand's global DOF.

Since the hand's kinematic model is only an abstraction of the complex underlying hand structure, there are often disagreements regarding where the joints are located, and how

[2]In reality, the arm may restrict the hand's position or orientation (for example, preventing the hand from extending beyond the arm's reach). However, our experiments assume the arm does not restrict the hand's movement.

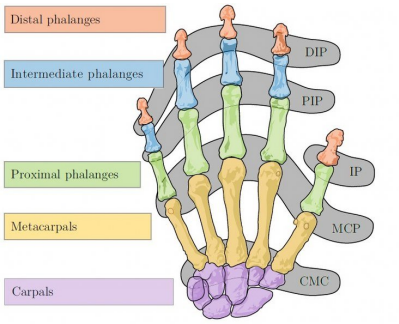

Figure 2.2: A diagram of the human hand's bones and joints [33].

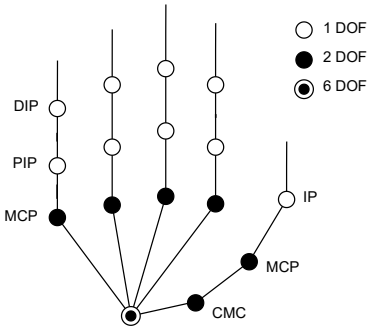

Figure 2.3: Our 27 DOF human-hand kinematic model.

many DOFs the hand has. Some papers argue that the hand's kinematic model has 24 [26, 27], 25 [28], or 26 [29, 30] DOF, however, the majority agree that it has 27 DOF [16, 31, 32]. We use a commonly accepted 27 DOF model, proposed by Chen et al. [16], throughout the remainder of this thesis. The choice of model does not affect our investigation as, despite their structural differences, most 27 DOF models exhibit near-identical ranges of motion.

The rest of this section describes our human-hand kinematic model (Figure 2.3), based off the structure of a real human hand (Figure 2.2). Our kinematic model contains 6 global DOF, and 21 local DOF; for simplicity, we call all kinematic-model links and joints by their corresponding bone and joint names in the human hand. The hand's wrist (which we refer to as the *root*) is controlled via the hand's *global DOF*: 3 DOF for position, and 3 DOF for orientation.

There are four fingers and one thumb connected to the hand's root, which we collectively refer to as *digits*. In each finger, *carpal bones* and *metacarpal bones* connect the root to the *metacarpophalengeal* (MCP) joint (located at the knuckle). We refer to the joint connecting the root to these bones as the *root joint*: a fixed-rotation joint containing no DOF. The MCP joint connects the carpal and metacarpal bones to the *proximal phalanges*. The MCP joint is a *universal joint*, permitting both "forwards-and-backwards" (pitch) and "side-to-side" (yaw) movements. Bending this joint forwards is known as *flexion*, and bending it backwards is known as *extension*. We refer to these movements collectively as *flexion-extension*. Bending your fingers side-to-side at the knuckle is known as *abduction* (away from the middle finger) and *adduction* (towards the middle finger). These movements are collectively known as *abduction-adduction*.

The finger's *proximal interphalangeal* (PIP) joint connects the proximal phalanges to the *intermediate phalanges*. The intermediate phalanges are then connected to the *distal phalanges* at the tip of the finger via the *distal interphalangeal* (DIP) joint. Both the PIP and DIP joints are revolute joints, which only permit flexion-extension (pitch) motions. The finger's MCP joint contains 2 DOF, whereas the PIP and DIP joints contain 1 DOF each. Figure 2.4 illustrates the MCP and PIP joint motions of the finger.

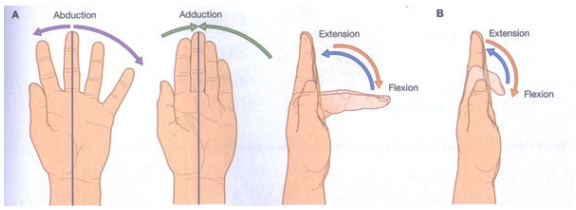

Figure 2.4: Illustration of the finger's movements [34]. Diagrams 1–3 represent the MCP's *abduction-adduction* and *flexion-extension* motions, and diagram 4 represents the PIP joint's *flexion-extension* movements.

Figure 2.5: Illustration of the thumb's movements [35]. Diagrams 1 & 2 represents the CMC's *abduction-adduction* motions, and diagrams 3 & 4 represent the CMC's *flexion-extension* movements.

The thumb's structure differs from the rest of the fingers, allowing it to produce the *opposable thumb* motions only found in human hands [36]. A fixed root joint joins the hand's root to the thumb's carpal bones, which are then connected to the thumb's metacarpal bones via the *carpometacarpal joint* (CMC) joint. This is a *universal joint*, which allows the thumb to perform both *abduction-adduction* and *extension-flexion*. Unlike the fingers, the thumb is *abducted* when it points away from the palm, and *adducted* when it aligns with the palm plane, and is *extended* if it bends away from the middle finger, and *flexed* when bending towards the middle finger (pictured in Figure 2.5). The metacarpal bones connect to the proximal phalanges with the MCP joint, which are connected to the distal phalanges via the interphalangeal (IP) joint (skipping the intermediate phalanges found in the fingers). The MCP joint is also a *universal joint*, whereas the IP joint is a *revolute joint* only permitting *flexion-extension* motions (similar to the finger's DIP and PIP joints). The thumb has 2 DOF each in its CMC and MCP joints, and 1 DOF in the IP joint, totalling 5 DOF.

Key Takeaway. *The kinematic model of the human hand describes the* ***links*** *and* ***joints*** *which mimic the hand's true biomechanical structure. These links and joints correspond to the bones and joints found in the human hand. The most widely accepted hand kinematic model contains* ***27 DOF****: 6 global and 21 local (4 in each finger, and 5 in the thumb).*

2.2 Reinforcement Learning

Reinforcement learning (RL) encompasses a range of techniques which train *agents* to perform desirable behaviour in a given environment. The environment is typically formulated as a *Markov Decision Process* (MDP), which acts as a mathematical framework for decision-making. Section 2.2.1 provides the formal definition of a MDP, and Section 2.2.2 describes how RL can be used to train agents in MDP environments.

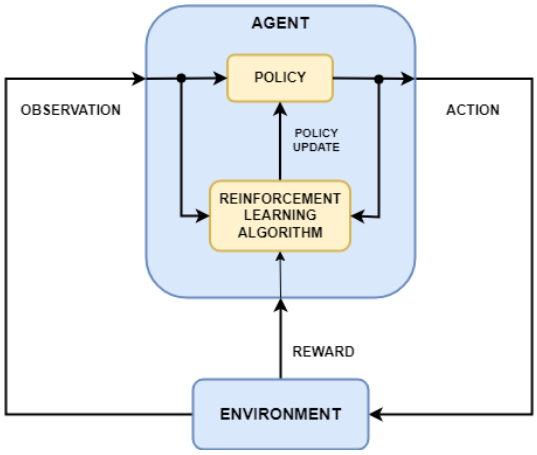

Figure 2.6: The feedback loop for a RL system [37]. The agent takes in observations from its environment, and produces a new action based on its policy. The *current state*, *action*, *next state*, and *reward* from taking this action are used to update the agent's policy.

2.2.1 Markov Decision Processes

A Markov decision process (MDP) is a discrete-time stochastic control process. They are used to model *decision-making* in task-orientated environments under the (full or partial) control of a *decision maker*. Formally, a MDP is defined with the 4-tuple (S, A, T, R), describing the environment's: *State*, *Action*, *Transition*, and *Reward*.

State (S)**.** The set of possible environment (and agent) states — capturing the full environment information, even when viewed in isolation. *Observations* are the state which the agent can see, however we use the terms *state* and *observations* interchangeably as we assume the environment to be full observable.

Action (A)**.** The set of possible actions an agent can take in the environment.

Transition $(T : S \times A \to S)$**.** The *transition function* characterises the dynamics of the environment. The transition function maps a current state s and the agent's chosen action a to the new environment state s' (also known as a *step*). Transitions may either be *probabilistic* $(s' \sim T(s, a))$ or *deterministic* $(s' = T(s, a))$. Throughout this thesis, we assume the simulator, and hence transitions, to be deterministic.

Reward $(R : S \times A \times S \to \mathbb{R})$**.** The *reward function* maps the current state s, chosen action a, and next state s' to a real value r, representing how desirable these states and actions are. In a deterministic system, this can be reduced to $R : S \times A$, as s' is deterministically described by $T(s, a)$.

2.2.2 Reinforcement Learning

A reinforcement learning system consists of an *environment* and an *agent*, which interact with each other in a feedback loop (as illustrated in Figure 2.6). The agent is governed by a *policy* π, which maps the current state s onto a distribution of possible actions

$a_0, \ldots, a_n$. The agent choses an action every *step* by sampling from the distribution $a \sim \pi(s)$. This produces an *experience* (s, a, r, s'): containing the state s, chosen action a, reward $r = R(s, a)$, and next state $s' = T(s, a)$. Experiences are used to update the policy π, to maximise current and future rewards, and are typically stored in an *experience replay buffer* to be replayed back during training.

The reward function is typically hand-crafted to push an agent towards completing its task. For example, if a robot were tasked with picking up a glass, it may be rewarded for touching the glass, and rewarded even further for picking up the glass, but punished (with a negative reward) for dropping the glass. In this example, dropping the glass may be considered "failing the task", and so the environment would be considered *terminated*, and reset for another attempt. Similarly, if the agent takes too long to complete a task, we may *truncate* and reset the environment early. The period between environment initialisation and reset is known as an *episode*. Agents aim to maximise their total reward each episode, also referred to as their *episode reward*.

We further explore how RL can be applied to humanoid robotic systems in Section 3.3.

Key Takeaway. *A RL environment is described by a **MDP**, consisting of: a state space, an action space, a reward function, and a transition function. An agent interacts with this environment in a feedback loop, learning from past experiences to **maximise its episode reward**.*

2.3 Robotics & Control

The two main approaches to building humanoid robots are *soft-bodied* and *rigid-bodied* robotics — Section 2.3.1 explains the differences and trade-offs between these approaches. Rigid-body robot control is typically implemented with a *control loop feedback mechanism*, the most common of which being the *PID (proportional-integral-derivative) controller* (Section 2.3.2). Our implementation relies upon ROS2 (Robotics Operating System): an industry-standard framework for developing robot applications. Section 2.3.3 describes the ROS2 framework, and the core ROS2 components used in our implementation.

2.3.1 Rigid vs Soft Bodied Robots

Rigid-body robots (also known as *hard robots*) are fabricated from rigid structural materials, whereas soft-body robots (also known as *soft robots*) are built from flexible materials. Soft robots aim to mimic soft lifeforms, which exhibit complex behaviour with simple and robust designs. For example, star-fish inspired robots contain a simple, flexible mesh which is able to grasp a variety of different objects [38].

Despite their promises, soft robots are much earlier in their technological development than hard robots [39]. Compromises are often made when developing soft robots, such

as dexterity and control complexity [40]. While soft robots are able to achieve moderate dexterity with simple designs, this approach does not scale to the dexterity required to mimic a human hand [41]. Soft materials can deform unpredictably, which leads to difficulties when attempting to precisely manipulate an object with fine motor control [42].

To achieve the level of control required, the majority of highly-dexterous robotic hands are hard robots (despite being more complex and expensive to build). However, it is common to incorporate some soft-bodied elements, so that the hand flexes more naturally when performing tasks. Our investigation primarily focuses on hard robots, where dexterity is easy to define and control (such as with a rigid-body kinematic model, Section 2.1.3).

Key Takeaway. *Rigid-bodied robots can achieve greater levels of control and dexterity than soft-bodied robots, at the expense of cost and complexity.*

2.3.2 PID Controllers

When controlling a robot, our goal is to align its current state with a target value. For example, consider the scenario where we wish to move a car $5m$ forward. We lack direct control over the car's position: we cannot tell the car's wheels to jump forward $5m$ in one go. Instead, we might instruct the car's wheels to rotate at a velocity of $1m/s$ for $5s$. In a perfect world, this would move the car forward $5m$.

However, the majority of real-world systems are imperfect. If the car is blown backwards by a slight wind, it may only advance $4.8m$ forward. This becomes dangerous when the car's true position deviates from where it thinks it is (due to the accumulation of small errors). The solution to this problem is to implement a *closed-loop control system*: feeding the car's true state back into the control system to adjust for inaccuracies.

Figure 2.7 illustrates a typical closed-loop control system. First, we give the car an *input* (change position state to $5m$) and receive a *feedback signal* about the cars current state (current position = $4.8m$). We then subtract the input from the current state in the *error detector* to obtain the *error* offset ($0.2m$). The car's input and error are passed into the controller, which determines the *actuating signal* to be sent to the *plant* (the object whose output we control, in this case the car). The car adjusts its position based on the actuating signal, and its new state is collected by the *feedback elements* (such as sensors). The cycle repeats until the error is within a reasonable threshold.

The controller can calculate the *actuating signal* in different ways. The most common approach is known as a PID controller: a summation of proportional (P), integral (I), and derivate (I) responses.

Proportional (P) Response. This response ensures the output moves towards the expected input. This is achieved by multiplying the error (between the input and feedback signal) by the tunable parameter K_p.

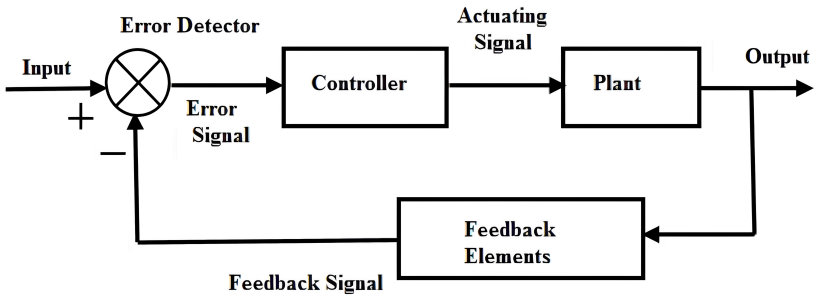

Figure 2.7: A closed-loop control system [43]. Given an input (the desired end-state) and the error (difference between the current state and desired end-state), the controller produces outputs to align the object's current state with the desired end-state.

Integral (I) Response. This response helps eliminate *steady-state error*: the difference between the input and output of a system in the infinite time limit. To resolve this, the integral response accumulates the error term over time, and multiply this value by K_i.

Derivative (D) Response. Using proportional and integral responses alone may cause the object to overshoot its target, causing unwanted oscillations. The derivate response dampens the system's response by multiplying the derivative of the error (with respect to time) by the tunable parameter K_d.

Summing these terms gives us:

$$\mathrm{u}(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de}{dt}$$

Where $u(t)$ is the controller output at time t, and $e(t)$ is the error at time t. The values of K_p, K_i, and K_d depend on the specific robot system, and must be tailored to match the system's dynamics. The previous example uses distance as the input and error, and velocity as the output — however, this does not necessarily need to be the case. For example, we could take *angle rotation* as the input, *angle rotation* and *rotational velocity* as the error, and return *angle effort* (torque) as the output.

Key Takeaway. *The most common* ***closed-loop control system*** *uses a PID controller: a component which adjusts for inaccuracies using* ***Proportional (P), Integral (I), and Derivative (D)*** *responses.*

2.3.3 Robot Operating System

ROS (Robot Operating System) [44], and its successor ROS2, are open-source robotics systems which provide tools and libraries for developing robot applications. This project focuses on ROS2, as ROS is no longer supported. Despite its name, ROS2 is not actually an operating system — it instead acts as a middleware layer for packages to communicate via message passing. ROS2 is commonly used in academia [45], as it provides common software interfaces and hardware abstractions for developing robotic systems.

A ROS2 application consists of *nodes*, which run asynchronously and intercommunicate via *message-passing*. The primary form of message-passing is the *publisher-subscriber (pub-sub)* model. In this model, nodes can publish or subscribe to messages on a *topic*, which acts as a named channel in the ROS2 middleware.

ROS2 is closely integrated with the simulation engine *Gazebo*[3]; messages can be sent and received from Gazebo using *plugins*. The three core plugins used in this project are: `gazebo_ros2_control` [46], `gazebo_ros_bumper` [47], and `gazebo_step_control` [48]. The `gazebo_ros2_control` plugin communicates with ROS2's `ros2_control` package: a framework for handling real-time control of robots. By defining `ros2_control` controllers, we can control and view the state of robots within the Gazebo simulation. `ros2_control`'s state does not include collision detection, so we add the `gazebo_ros_bumper` plugin to recieve collision-detection information. Finally, the `gazebo_step_control` plugin tells Gazebo how many milliseconds to simulate before pausing, to control RL step sizes.

Key Takeaway. *A ROS2 system consists of asynchronous **nodes**, which intercommunicate via pub-sub message-passing. One such node is **Gazebo**, a simulation engine which communicates with other ROS2 nodes via **plugins**.*

2.4 Summary

Before building our robotic-hand simulation, we first define a *rigid-body kinematic model* of the human hand (Section 2.1.3), which serves as the foundation for our simulation model. The kinematic model abstracts the structure of the human hand, outlining the 27 DOF (Section 2.1.2) that describe the hand's motions. These DOFs are represented either as *joints* (described by Tait-Bryan angle rotations, Section 2.1.1), *positions* (described by positional coordinates), or *orientations* (described by unit quaternions, Section 2.1.1).

Our rigid-body (Section 2.3.1) robotic-hand simulation is implemented in *ROS2* (Section 2.3.3) using the *Gazebo* simulation engine, and is controlled with a *PID controller* (Section 2.3.2) to adjust for simulation inaccuracies. This allows the robotic hand to be controlled via both teleoperation and a RL agent. By formulating the simulation environment as a MDP (Section 2.2.1), reinforcement-learning agents can learn to control the hand to complete grasping tasks (Section 2.2.2).

[3] `https://gazebosim.org/home`.

Chapter 3

Previous Approaches

Before developing our own robotic-hand simulation, we first investigate existing robotic hands (Section 3.1). The most common method for evaluating robotic-hand control is via *manipulation tasks* [49], which typically involve grasping or moving objects in an environment (Section 3.2). To teach our robotic hand to complete these tasks, we explore previous approaches to robotic hand training, both *with* (Section 3.3) and *without* (Section 3.4) the aid of expert demonstrations.

3.1 Biomimetic Robotic Hands

Section 3.1.1 outlines the historical progression of dexterous robotic hands. In the Introduction (Section 1.1), we claim to achieve **state-of-the-art** dexterity with our 27 DOF rigid-body robotic-hand simulation. To substantiate this claim, we perform a systematic literature review (Section 3.1.2), spanning **220 papers** and **254 robotic hands**, which reveals that no existing rigid-body robotic hands — neither physical nor simulated — achieve 27 DOF. We discuss potential explanations and implications for these results in Section 3.1.3. Our review finds that existing robotic hands simplify their designs in similar ways, by excluding common sets of DOFs. To analyse how this affects the hand's dexterity, we remove these DOFs from our full kinematic hand model to produce a set of less-dexterous robotic-hand models (Section 3.1.4), which we later use throughout our experiments.

3.1.1 History

The history of dexterous robot hands has been well-reviewed: Sampath et al. [50], Yu and Wang [51], and Rebollo and Molina [52] survey state-of-the-art robotic hands found in previous literature and industry. We briefly summarise their findings:

- The UTAH/MIT hand [53] was one of the first biomimetic robotic hands. It was developed in 1985 by the University of Utah and the Massachusetts Institute of

Technology, and it contains three fingers and a thumb (for a total of 22 DOF, if attached to a robotic arm).

- In 1993, the Harbin Institute of Technology and the German Aerospace Centre developed the four-fingered DRL hand [54]. While this hand only contains 18 DOF, it was used during the Spacelab-D2 Mission to confirm the functionality of robotic hands in space [50].
- In recent years, robot hands tend to prioritise either dexterity (such as the Shadow [55] and Allegro [56] hands), or cost and simplicity (such as the TriFinger [57] and DClaw [58] hands).
- The most dexterous robotic hand to date is arguably the Shadow Hand [55]. Developed in 2013, it contains four fingers and one thumb for a total of 24 DOF and 27 DOA.

3.1.2 Systematic Literature Review

The surveys discussed in Section 3.1.1 do not provide a comprehensive review of the field: they fail to identify all relevant robotic hands, and they omit robotic-hand *simulations*. Therefore, we conduct our own systematic literature review to ensure all relevant biomimetic hand models are considered.

We first generate a list of keywords, which we update throughout our review:

- Dexterous, Manipulation
- Human, Humanoid, Human-like, Biomimetic
- Robot, Robotic
- Hand, Hands, Arm, Arms
- Simulated, Simulation
- Learning, Reinforcement Learning, Imitation Learning

We use these keywords to generate queries, allowing us to identify relevant works in previous literature. Our literature review spans **220 papers**, from which we analyse **254 robotic hands**. We exclude irrelevant robotic hands, such as soft robots, and those which do not intend to mimic the structure of the human hand. This leaves us with 45 dexterous robotic hands, partially listed in Table 3.1. The full details and results of our literature review can be found in Appendix A.

The surveys from Section 3.1.1 do not apply consistent definitions of DOF: often confusing DOF and DOA, and inconsistently including global DOF. We resolve these ambiguities with the following rules: we explicitly disambiguate DOF and DOA (as defined in Section 2.1.2), and we assume that the hand is always attached to an arm and wrist (so that global DOFs are always included). We also rectify errors identified in these surveys, such as miscounted digits and incorrect DOF counts.

Robotic Hand	Year	Real-world Based	Simulation Based	Number of Digits	DOF	DOA
Shadow Dexterous Hand [55]	2013	✓	✓	5	23	27
ADROIT Hand [59]	2013	✓	✓	5	23	27
Shadow EDC Hand [60]	2002	✓	✓	5	23	23
NTU Hand [61]	1998	✓	✗	5	23	23
Unnamed [62]	2016	✓	✗	5	23	23
Unnamed [63]	2024	✓	✗	5	23	23
Unnamed [64]	2005	✓	✗	5	23	23
CEA Dexterous Robot Hand [65]	2013	✓	✗	5	23	23
Asada Hand [66]	2007	✓	✗	5	23	23
Unnamed [67]	2005	✓	✗	5	22	22
Allegro Hand [56]	2015	✓	✓	4	22	22
Gifu hand I [68]	1999	✓	✗	5	22	22
Gifu hand II [69]	2002	✓	✗	5	22	22
Gifu hand III [70]	2004	✓	✗	5	22	22
UTAH/MIT [53]	1985	✓	✓	4	22	22
Blackfingers [71]	2000	✓	✗	5	22	22
Dist Hand [72]	1998	✓	✗	4	22	22
BUAA [73]	2001	✓	✓	4	22	22
KITECH Hand [74]	2016	✓	✗	4	22	22
DART Hand [75]	2011	✓	✗	5	22	22
SKKU Hand II [76]	2006	✓	✓	4	22	22
PUCP hand [77]	2022	✓	✗	5	22	22
SmartHand [78]	2011	✓	✗	5	22	22
SCHUNK S5FH [79]	2017	✓	✗	5	21	26
TH-3R Hand [80]	2009	✓	✗	5	21	21
DLR/HIT Hand II [81]	2008	✓	✗	5	21	21
GCUA [82]	2011	✓	✗	5	21	21
HRI Hand [83]	2020	✓	✗	5	21	21
KYTECH HAND [84]	2012	✓	✗	4	20	20
TUAT/Karlsruhe [85]	2000	✓	✗	5	20	20
Dextrus Hand [86]	2016	✓	✗	5	20	20
DLR Hand II [87]	2001	✓	✗	4	19	23
Unnamed [88]	2017	✓	✗	5	19	19
RHC-1 [89]	2013	✓	✗	5	19	19
Ultralight Anthropomorphic Hand [90]	2001	✓	✗	5	19	19
DLR Hand I [54]	1993	✓	✗	4	18	22
Robotnaut Hand [91]	1999	✓	✓	5	18	18
Robotnaut 2 Hand [92]	2012	✓	✓	5	18	18
Karlsruhe Dextrous Hand II [93]	2013	✓	✗	5	18	18
Sandia Hand [94]	2014	✓	✓	4	18	18
DexHand [95]	2011	✓	✗	4	18	18
ZAR3 Hand [96]	2004	✓	✗	5	18	18
MPL [97]	2010	✓	✗	5	17	26
EthoHand [98]	2016	✓	✗	5	17	24
FRH4 [99]	2008	✓	✗	5	17	18
IH2 Azzurra [100]	2016	✓	✗	5	17	17

Table 3.1: The **truncated** results of our systematic literature review, identifying **45** relevant physical and/or simulated dexterous robotic hands across **220 papers**. The results are listed in descending DOF order, followed by descending DOA order. The full details and results from the literature review can be found in Appendix A.

Our findings (Table 3.1) show that no robotic hands (neither physical nor simulated) achieve 27 DOF. This highlights the benefit of our work: designing a 27 DOF biomimetic robotic hand which can operate in a 3D simulated physics environment is a **novel** achievement.

3.1.3 Implications

Engineering constraints have thus-far prevented the development of physical robotic hands with the full 27 DOFs found in human hands [10, 11]. Simulated hands, however, are not bound by such constraints. Although it is theoretically possible to create a 27 DOF robotic-hand simulation, our review indicates that one has never before been built.

All the robotic-hand simulations found from our literature review have a physical robotic-hand counterpart (such as the Shadow [55], ADROIT [59], and Allegro [56] hands). This suggests that dexterous robotic-hand simulations are built only to support existing physical robotic hands. We believe that the reason no simulated robotic hands with 27 DOF have been built, is because there are no corresponding physical hands with this level of dexterity.

The absence of 27 DOF robotic-hand models means that no prior research has explored how this level of dexterity might affect robotic-hand control and training. With the growing interest in this field [101, 13], it is likely that the engineering constraints preventing the development of physical 27 DOF robotic hands will soon be overcome [12, 13] — giving rise to a new generation of dexterous robotic hands. However, it remains unclear whether such hands will perform better at teleoperation, RL, and IL than their predecessors. This motivates the core question of our thesis: *how does dexterity (up to and including that of a human) affect a robot hand's ability to be operated by, and to learn from humans.*

3.1.4 Robotic-Hand Models

Given that existing robotic hands cannot yet achieve human-like dexterity, they must simplify their designs. We find that these models reduce their complexity in similar ways, by removing common sets of DOFs. To experimentally assess the impact that these changes have on the hand's dexterity, we produce modified hand models (derived from our kinematic hand model, Section 3.1.4), which reflect the loss of these common sets of DOFs. We propose the following models to use throughout our experiments:

- `full` (27 DOF): Our full 27 DOF robotic-hand model (Section 2.1.3).
- `finger-no-abd` (23 DOF): We omit the abduction-adduction motions in all MCP finger joints (finger splay). This change can be found in existing hand models, such as the LMS [102], AR10 [103], MPL [97], and Deka [104] hands, in order to simplify the finger's structure.

- `finger-interdep` (19 DOF): We model the joint angles in each finger as interdependent. Although they can be controlled separately, the movement of a human's MCP, PIP, and DIP finger joints are often correlated[1]. For example, Robson et al. [105] models these joints as linearly dependent: $\theta_{\text{PIP}} = 0.75\,\theta_{\text{MCP}}$ and $\theta_{\text{DIP}} = 0.5\,\theta_{\text{MCP}}$. This allows the finger to use a simpler tendon-driven design, rather than having to control each joint individually. Finger *flexion-extension* movements can now be controlled through a single parameter: reducing the three flexion-extension DOFs in each finger to one DOF.

- `thumb-no-mcp` (25 DOF): We remove the MCP joint from the thumb (the thumb's intermediary joint), leaving most of the thumb's control to the CMC joint (the thumb's base joint). Since the thumb is the most complex component of the human hand [36, 106], it is often modelled with two joints rather than three to reduce complexity. This can be found in the EthoHand [98] and Deka [104] hands from our review.

- `thumb-no-flex` (25 DOF): We remove the flexion-extension movement in the thumb's CMC and MCP joints. Restricting this motion leaves only the abduction-adduction motion in the thumb (towards-and-away from the palm). This means that the thumb is no longer opposable, and so acts more like a fifth finger. The UTAH/MIT [53], Dist [72], and DLR I [54] hands all make this simplification.

- `three-fingers` (23 DOF): We remove the pinky finger from the hand model. Models often choose to remove this finger as it is not required for many types of manipulation tasks. This change can be found in Allegro [56], UTAH/MIT [53], DLR I [54], and DLR II [87] hands.

3.2 The GRASP Taxonomy

Grasping taxonomies synthesise and categorise different types of human-hand grasps. One such taxonomy is Feix et al's GRASP taxonomy [1]: a systematic arrangement of static, stable, single-handed grasps, collated from previous literature. This taxonomy is commonly referenced when performing RL on humanoid hands, to ensure a wide range of grasps can be learnt by the algorithm [107, 108, 109]. Table 3.2 summarises the 33 grasps found in the taxonomy.

Our experiments are constrained by the fact that we cannot tell the RL agent *how* to grasp an object without introducing bias into the learning process. Instead, we choose a set of *objects* which encourage the agent to exhibit different types of grasping behaviour, rather than choosing the types of grasps it should perform. We select a subset of the

[1]For example, imagine curling your fist into a ball. The rate at which the MCP, PIP, and DIP joints bend are linearly correlated. This correlation extends to other common types of hand movements.

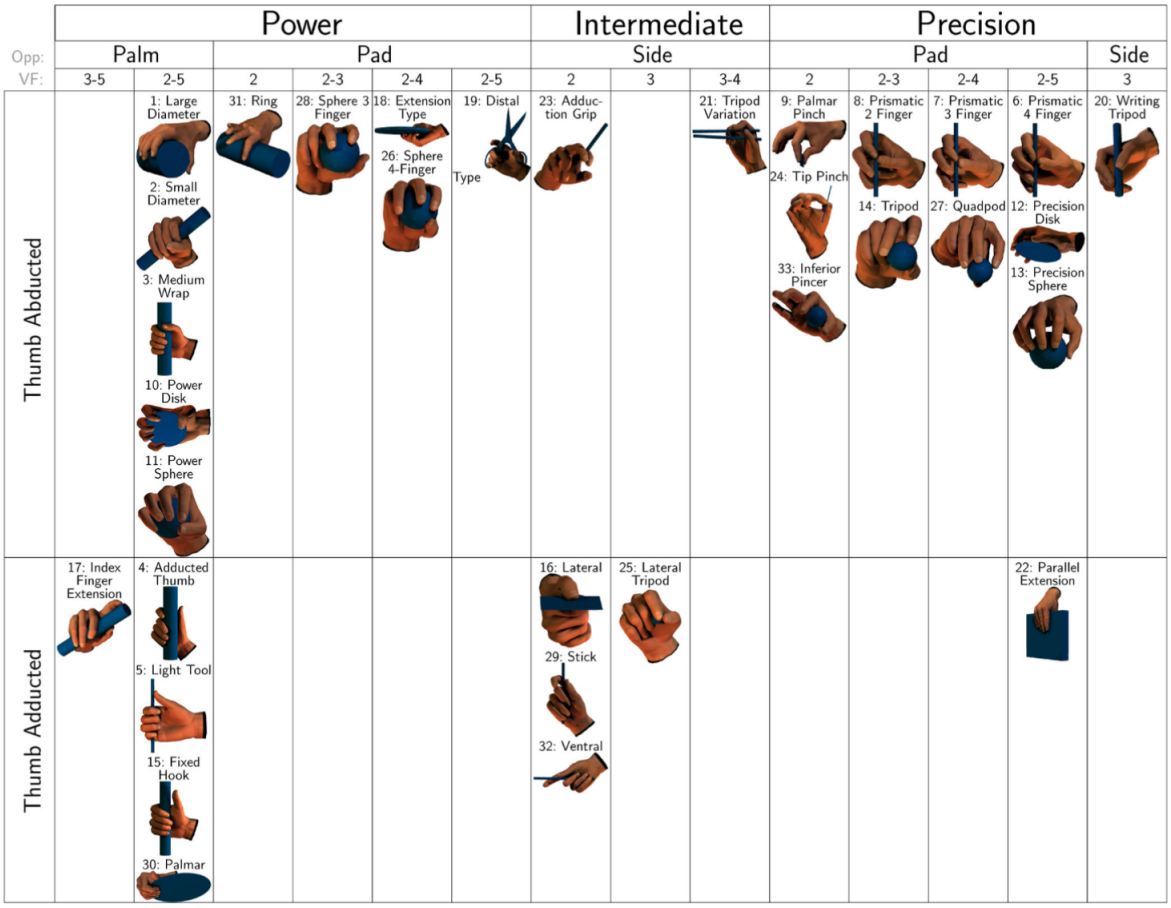

Table 3.2: A summary of the 33 grasps in the GRASP taxonomy [1]. These grasps are categorised by: opposition type, virtual finger assignments, type (in terms of power, precision or intermediate grasp), and position of the thumb.

objects listed in Table 3.2, as it would be impractical to perform user-experiment studies on all the objects in the taxonomy. Our chosen objects are:

- `thin-cyl`: Thin cylinder *(7: Prismatic 3-finger)*
- `med-cyl`: Medium-sized cylinder *(3: Medium Wrap)*
- `thick-cyl`: Thick cylinder *(1: Large diameter)*
- `small-sphere`: Small sphere *(13: Precision sphere)*
- `med-sphere`: Medium-sized sphere *(26: Sphere 4-finger)*
- `thin-sheet`: Thin upright sheet *(22: Parallel extension)*

These objects allow the hand to exhibit a range of grasping types, providing a diverse set of manipulation tasks for our dexterity analysis.

3.3 Reinforcement Learning

This section describes the three high-level approaches to reinforcement learning: *value-based RL* (Section 3.3.1), *policy-based RL* (Section 3.3.2), and *actor-critic models* (Section 3.3.3). We explore the trade-offs of these approaches in the context of humanoid robotics, and present practical implementations of these algorithms in Section 3.3.4.

3.3.1 Value-based RL

Value-based RL algorithms attempt to predict the *quality* of taking different actions given the current state. Quality is measured through *Q-values*: a combination of the action's immediate reward and expected future returns (scaled by the discount factor γ). This is represented by the Bellman equation:

$$Q(s,a) = r(s,a) + \gamma \max_{a' \in A} Q(s',a')$$

Q-networks are trained to approximate the Q-value function $Q(s,a)$, by minimising the loss between the original and updated Q-values. This is typically formulated as the mean-squared error loss:

$$L = (r + \gamma \max_{a'} Q(s',a';\theta') - Q(s,a;\theta))^2$$

Where θ represents the current parameterisation of the Q-network, and θ' represents the updated parameterisation of the Q-network.

The most common implementation of value-based RL is Deep Q-Networks (DQNs), which use deep neural networks (DNNs) as the Q-network [110].

Value-based RL attempts to pick the action a which maximises the Q-value function $Q(s,a)$. This works well with discrete action spaces, where we can enumerate over the action space to find the optimal action. However, value-based RL struggles with *continuous actions* as the action-space is no longer enumerable. This approach is poorly suited to humanoid robotics, where most joints are controlled via continuous-angle actions.

3.3.2 Policy-based RL

Policy-based RL overcomes this constraint by learning policies without calculating a value function, instead representing the policy explicitly as $\pi_\theta(a|s)$. The parameters θ are optimised to maximise the performance objective $J(\pi_\theta) = E_{\tau \sim \pi_a}[R(\tau)]$. This is typically implemented via gradient ascent, with the iterative formula: $\theta_{t+1} = \theta_t + \alpha \nabla J(\pi_{\theta_t})$, where α is the learning rate and $\nabla J(\pi_{\theta_t})$ represents the policy gradient [111].

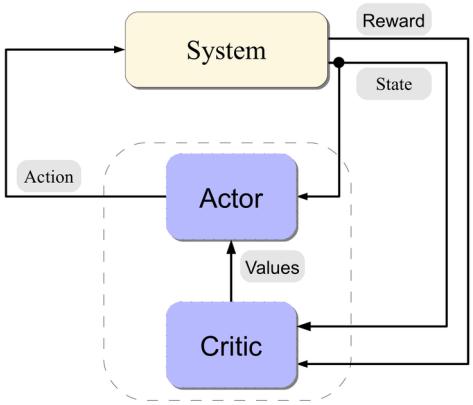

Figure 3.1: The actor-critic RL model [113]. The *actor* chooses actions based on its policy to maximise expected return, and the *critic* assesses the quality of the action taken, adjusting the model accordingly.

However, unlike value-based methods, policy-based RL suffers from slow learning (due to high variance in gradient estimates) [112]. This is problematic when dealing with highly dexterous robots in complex environments: policy-based approaches struggle to learn quickly when dealing with large observation and action spaces. Ideally, we would like to combine the performance benefits of value-based methods with policy-based RL's support for continuous action spaces.

3.3.3 Actor-Critic Models

Actor-critic models combine the best aspects of value-based and policy-based RL. The *actor* component makes decisions using a policy-based approach, and the *critic* component critiques the quality of the model's action using value-based RL (depicted in Figure 3.1). The actor component generates actions, and is policy-based, allowing it to handle *continuous action spaces*. Introducing a value-based critic reduces the variance in an action's estimated return, preserving the performance benefits of value-based RL.

3.3.4 Implementations

It is uncommon to use policy-based RL in humanoid robotics due to its poor performance. However, one successful example is the **Proximal Policy Optimisation (PPO)** algorithm [114], which combats slow learning rates by constraining the optimisation of a policy to a *trusted region*. This region is defined as the area in which local approximations of the function are accurate; the trust region is then iteratively expanded or shrunk based on how well the new approximation performs [111].

However, due to their greater performance, the majority of humanoid-robot RL algorithms are actor-critic models. We describe the most popular actor-critic implementations below:

Deep Deterministic Policy Gradient (DDPG) [115]

The DDPG algorithm combines DQNs [110] with deterministic policy gradients [116]. The model contains two neural networks: a deterministic policy network (for the actor), and a DQN (for the critic). The policy network trains via gradient ascent (Section 3.3.2), whereas the DQN trains by minimising the loss between the original Q-value and the updated Q-value (Section 3.3.1).

Twin Delayed DDPG (TD3) [117]

Although DDPG can perform well, training can be unstable and the algorithm is heavily reliant on well-chosen hyperparameters. This is caused by Q-value overestimation in the critic network. These errors can build up over time, leading the agent into undesirable local optima, or potentially causing the agent to experience catastrophic forgetting [111]. To address these problems, TD3 focuses on reducing the overestimation bias found in DDPG, such as delaying policy and target network updates (*twin delay*), and introducing noise to the target action.

Advantage Actor Critic (A2C) [118]

The advantage actor-critic (A2C) is a variant of the actor-critic algorithm which introduces the concept of an *advantage function*. This function measures the benefit of an action compared to the average action in a given state. The critic is trained to estimate the advantage function instead of the Q-function, so that the critic's evaluation of an action considers not only its success but its success relative to other actions.

Soft Actor Critic (SAC) [119]

The soft actor-critic algorithm employs a maximum entropy approach. The algorithm's goal is to find the optimal policy which maximises both the expected long-term reward and the long-term entropy [120]. This approach helps balance exploration (via long-term entropy optimisation) and exploitation (via long-term reward optimisation), by explicitly separating out these terms into seperate networks.

3.4 Imitation Learning

Imitation learning extends RL with *expert demonstrations*, accelerating training by using these demonstrations as a starting point. This approach also promotes more human-like behaviour, as the agent learns to mimic the expert's actions.

This section covers the four main approaches to imitation learning: behavioural cloning (BC), direct policy learning (DPL), inverse reinforcement learning (IRL), and generative adversarial imitation learning (GAIL). We view these algorithms in the context of

humanoid robotics, where our expert demonstrations are collected *offline* (without live interaction between the IL algorithm and the demonstrator).

Behavioural Cloning (BC)

Behavioural cloning samples state-action pairs from the expert demonstrations, to directly train the RL neural-network policy via supervised learning. However, it assumes state-action pairs to be independent and identically distributed, which is not the case (new states always follow on from the previous states). Hence, errors can accumulate, putting the expert in new and unfamiliar situations.

Direct Policy Learning (DPL)

Direct Policy Learning attempts to address the issues of BC, by interleaving policy-training and demonstration-collection — the algorithm presents the demonstrator with unfamiliar situations to fill in the policy's gaps. However, this approach is unsuitable to our situation as it requires *online* training (live interactions with the demonstrator).

Inverse Reinforcement Learning (IRL)

Instead of using a hand-crafted reward function, as with traditional RL, inverse reinforcement learning uses the expert demonstrations to *learn* the reward function. It then applies standard RL with this learnt reward function to complete the given task. A popular implementation is adversarial IRL (AIRL) [121], which adversarially trains a discriminator to distinguish *expert demonstrations* from the *current learnt policy*, in order to recover a generalisable reward function. Another approach is density-based reward modelling (DBRM) [122], which uses *kernel density estimation* to model the underlying distribution of expert demonstrations.

Generative Adversarial Imitation Learning (GAIL) [123]

Generative adversarial imitation learning combines IRL with generative adversarial networks (GAN). The *generator* component tries to mimic the behaviour of the expert, while the *discriminator* attempts to distinguish the generator's actions from that of the expert (similar to AIRL).

3.5 Summary

After reviewing the history of dexterous robotic hands (Section 3.1.1), and conducting a systematic literature review into this field spanning **220 papers** (Section 3.1.2), we conclude that no existing robot hands (physical or simulation) have yet to achieve 27 DOF. Engineering constraints have prevented the development of physical 27 DOF

robotic hands, and no dexterous simulated hands have been developed independent of physical hands (Section 3.1.3). Hence, the 27 DOF robotic-hand simulation we build in the Implementation chapter achieves **state-of-the-art** dexterity.

Our thesis evaluates the performance of teleoperation, RL, and IL when grasping different *objects* using different *robotic-hand models*. We use the GRASP taxonomy [1] to derive a list of objects (Section 3.2), which will be used to conduct our manipulation tasks. Examining previous robotic hands, we find examples where hands commonly omit similar sets of DOF to reduce complexity. From this, we produce six robotic-hand models (one with the full 27 DOF, and five with missing sets of DOF), which we use to investigate the effect of dexterity on grasping (Section 3.1.4).

Finally, we describe the implementations of RL (PPO, DDPG, TD3, A2C, SAC) and IL (BC, DPL, IRL, GAIL) algorithms commonly found in the context of humanoid robotics, in Sections 3.3 and 3.4 respectively.

Chapter 4

Implementation

This chapter covers our teleoperation, RL, and IL implementations. We document the framework architectures and key design decisions which underpin our approach. Section 4.1 describes our *data collection framework*: a ROS2 system designed to collect and store expert demonstration data via teleoperation. Section 4.2 outlines the experimental procedure we use to collect human participant data using this framework. We then design a second ROS2 system — our *imitation learning framework* — to conduct IL on the collected data (Section 4.3).

4.1 Data Collection Framework

To collect teleoperation data, we first build a robotic-hand simulation which can be controlled by a human participant (Section 4.1.1). Teleoperation is implemented via a hand-tracking camera, which maps the user's hand positions onto our robotic-hand simulation (Section 4.1.2). A log of the simulator's state and user's actions are stored as *expert human demonstrations* for the given task (Section 4.1.3). Figure 4.1 illustrates a rough architectural overview of this framework, highlighting the key implementation components.

4.1.1 Robotic Hand Simulation

To build a robotic-hand simulation, we provide a robot specification to the Gazebo simulation engine.

Gazebo Simulation Engine. The Gazebo simulation engine can simulate robots described in SDF (Simulation Description Format). However, since it is difficult to design robots using this format, we instead use URDF (Unified Robotics Description Format), which can compile to SDF. URDF is an XML format primarily consisting of *links* and *joints*. A joint connects a *child link* to a *parent link*; in the human hand, parent links are positioned closer to the heart, and child links are positioned farther from the heart.

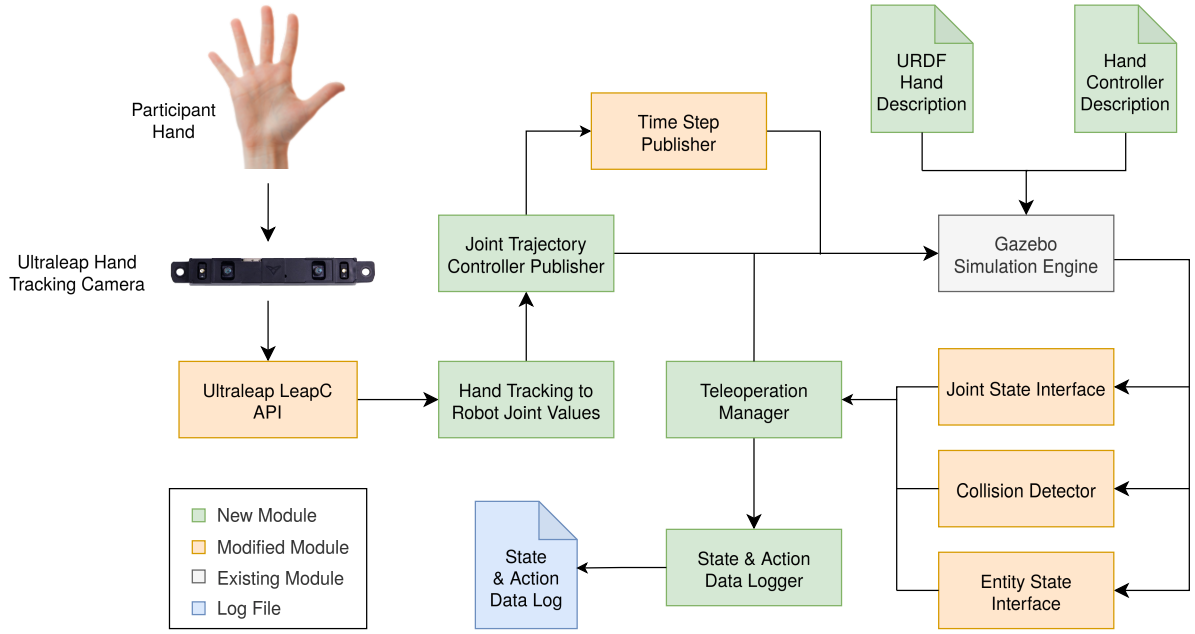

Figure 4.1: An architectural overview of the *data collection framework* (Section 4.1). Arrows represent data-flow, and modules loosely represent nodes in the ROS2 architecture. *New modules* refer to modules we write from scratch, and *modified modules* refer to existing modules which we modify and/or write a new interface for.

URDF Hand Description. The URDF hand description directly corresponds to our human-hand kinematic model (Section 2.1.3): links and joints in the kinematic model are directly converted into links and joints in the URDF format. However, URDF does not support 2 or 3 DOF joints — these are implemented as a series of 1 DOF URDF *revolute* joints along orthogonal axes[1]. For example, we implement a 2 DOF *universal joint* from the kinematic model as two URDF revolute joints which rotate along the yaw and pitch axes respectively. The hand's wrist is implemented with three URDF revolute joints for orientation, and three URDF *prismic* (translational) joints for position. Full details of our URDF hand description can be found in Appendix B. Our work presents the **first** robotic hand description which matches the dexterity of the human hand, and is compatible with physics simulation engines (such as Gazebo).

Hand Controller Description. The hand controller description is an XML and YAML description of the `ros2_control` controllers (Section 2.3.3) attached to the URDF Hand Description file. These controllers allow us to pass joint-state information in and out of the simulation environment. In particular, we use the `JointTrajectoryController` to manipulate the joints in the robotic hand, and the `JointStateBroadcaster` to publish the hand's current state. The `JointTrajectoryController` controls the hand's joint angles with a PID feedback loop: taking *angle rotation* as the input, *angle rotation* and *rotational velocity* as the error, and returning *angle effort* (torque) as the output. We implement a PID feedback loop for each individual joint; all 81 (27×3) PID parameters

[1]Unlike our definition of revolute joint in kinematic models (Section 2.1.3), URDF revolute joints can rotate along any fixed axis.

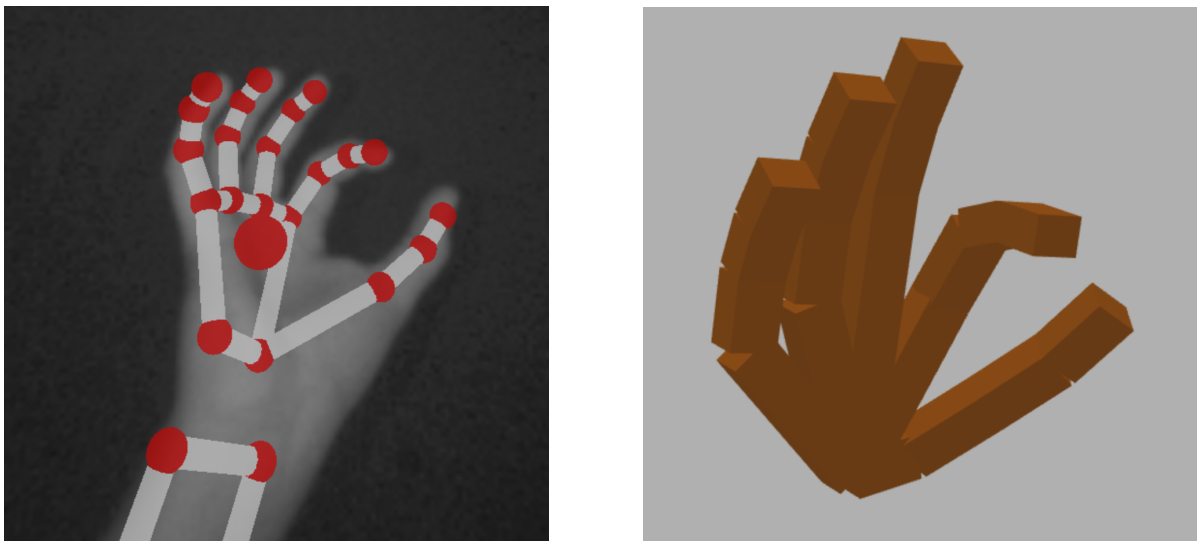

Figure 4.2: **Left:** A visualisation of the hand-tracking data collected by the Ultraleap camera on a participant's hand. **Right:** Our robotic-hand model teleoperated using the hand-tracking data: the participant hand's position and joint information is mapped onto the robotic hand.

were **fine-tuned by hand**, as is standard for most robotics systems [124]. Our final PID values can be found in Appendix C.

4.1.2 Hand Tracking & Joint Controllers

To teleoperate the robotic hand, joint angles are tracked on a participant's hand and mapped onto our robotic-hand simulation.

Ultraleap[2] Hand Tracking Camera. We place an Ultraleap Stereo IR 170 camera underneath the participant's hand to capture hand-tracking data. The device forwards this data onto our computer (running the ROS2 system) via a USB cable.

Ultraleap LeapC API. To parse the incoming data, we use Ultraleap's C interface: LeapC[3]. We wrap the interface as a C++ ROS2 node, which publishes the hand-tracking data as a custom `UltraleapJointInfo` message. The data contains the coordinates and orientations for each hand bone, where orientations are represented as quaternions.

Hand Tracking to Robot Joint Values. The `UltraleapJointInfo` message is captured by this node, and converted into a `JointState` message containing 27 floats (one controlling each DOF in the robotic-hand model). The hand's global position is calculated from the coordinates of the hand's wrist (root) joint. However, obtaining joint rotation angles for the remaining 24 DOF is more difficult: our hand kinematic model does not align with Ultraleap's hand model, and our joint angles must conform to Tait-Bryan's angle representation (whereas Ultraleap represents bone orientations as quaternions).

[2]Previously known as Leap Motion.
[3]`https://docs.ultraleap.com/api-reference/tracking-api/api-reference.html`.

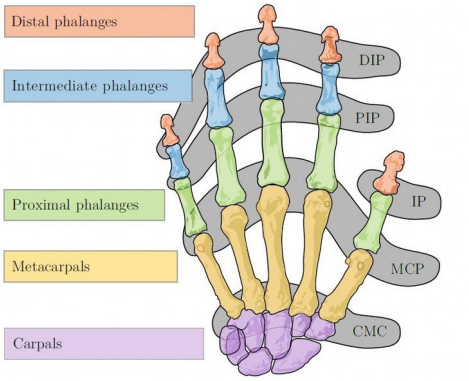

Figure 2.2: A diagram of a human hand's bones and joints (from Section 2.1.3) [33].

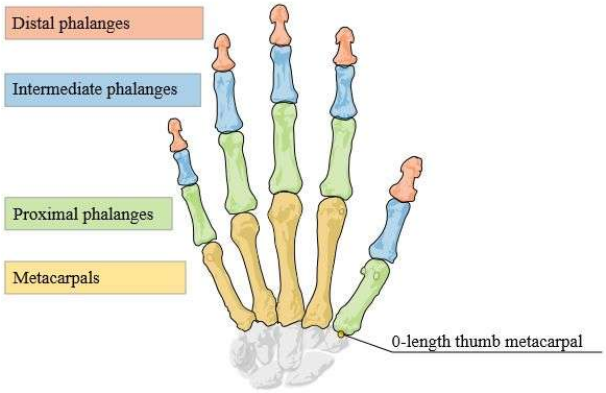

Figure 4.3: Ultraleap's model of the human hand [125]. The thumb's metacarpal bone is removed, and the proximal phalanges are shifted down to replace it. The gap left by the proximal phalanges is filled with intermediate phalanges (which are not found in a real human thumb).

To solve the second problem, we convert bone orientations into their joint angle representation. For example, let us calculate the angles at a finger's MCP joint given the quaternion orientations of its (parent) metacarpal bone q_1 and (child) proximal phalanges q_2. We first calculate the relative quaternion orientation of the child bone with respect to its parent bone: $q_1^{-1}\, q_2$ (Section 2.1.1). From this, we extract the yaw, pitch, and roll rotational values using Equation 2.1. The finger's MCP joint permits *flexion-extension* rotations along the yaw axis, and *abduction-adduction* rotations along the pitch axis. Hence, we map the yaw and pitch angle values onto the MCP's first (yaw) and second (pitch) revolute joints respectively.

However, this method is unable to calculate the thumb's joint angles, as Ultraleap's hand model is misaligned with our kinematic model (Figures 2.2 and 4.3). We are unable to calculate the relative quaternion at the CMC joint, as the child bone is undefined in Ultraleap's data representation. Our solution is to use the wrist as the joint's child-bone reference point. We adjust the wrist's quaternion to align with the thumb, to act as a virtual link between the root and CMC joint. The joint angles at the CMC joint can be approximated by taking the relative quaternion (Section 2.1.1) between this virtual link and the thumb's proximal phalanges.

Joint Trajectory Controller Publisher. This node converts incoming `JointState` messages into `JointTrajectoryController` messages. `JointTrajectoryController` messages control the simulated robot's joints via Gazebo's `gazebo_ros2_control` plugin (Section 2.3.3), allowing us to map the participant's hand movements directly onto the robot. Once this message has been sent, we also send a `TimeStep` message which tells Gazebo to process the next 10ms of the simulation (via the **TimeStepPublisher** node, and Gazebo's `gazebo_step_control` plugin). This produces consistent step sizes when capturing IL demonstration data during teleoperation.

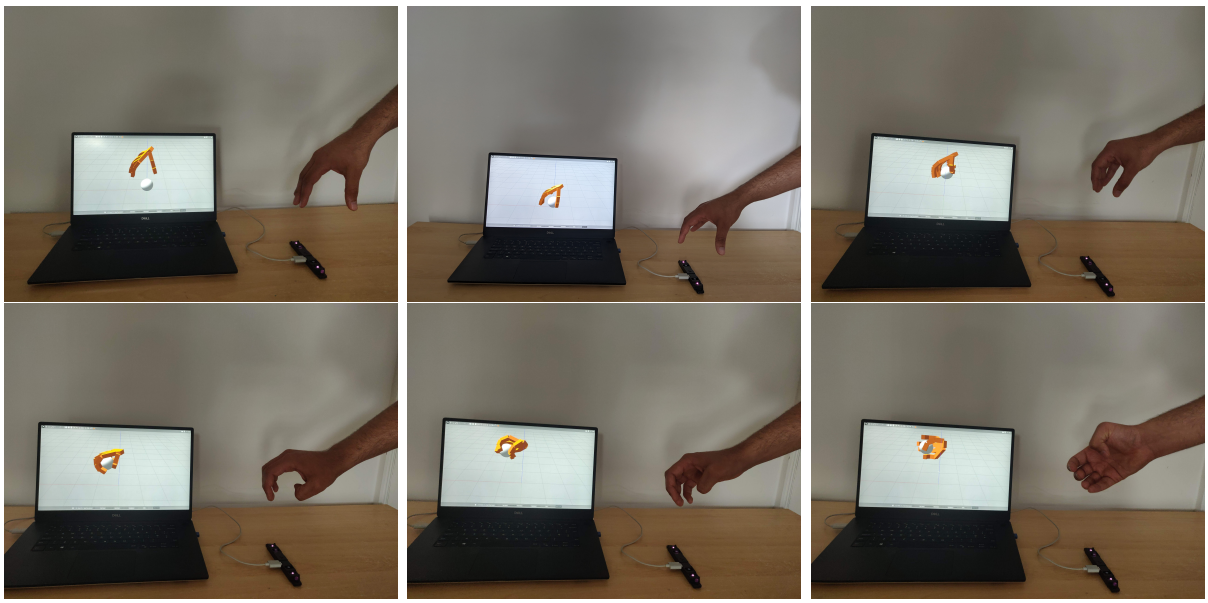

Figure 4.4: Pictures of a participant taking part in our *teleoperation* experiment: the robotic hand is controlled by the participant's hand movements (detected by the Ultraleap hand-tracking camera beneath their hand). The experiment depicted above is the `med-sphere` grasping task using the `full` robotic-hand model.

4.1.3 State Interfaces & Data Logging

The **Teleoperation Manager** queries the Gazebo simulator at each step for the state of the robotic hand and other objects in the scene. This includes: the position and velocity of all hand joints (**Joint State Interface**), the object's position, orientation, velocity, and angular velocity (**Entity State Interface**), and collision-detection information (**Collision Detector**). This state, along with the participant's teleoperation actions, are captured as *expert demonstrations* by the **State & Action Data Logger**. The demonstrations are stored in the **State & Action Data Log**, to be recalled during our *imitation learning framework*.

4.2 Experiment Details

We use the *data collection framework* to gather information on how participants teleoperate robotic hands to complete grasping tasks. Our experiments have two core objectives:

Objective 1. To evaluate how the dexterity of a robotic hand affects a participant's ability to teleoperate the robotic hand.

Objective 2. To collect *expert demonstration* training data for our IL algorithms, on simulated robotic hands of varying dexterities.

Experiment Procedure

We ask participants to teleoperate the robotic hand to grasp and pick up objects within the simulation. Participants do not grasp the objects in real life, as this introduces occlusions and inaccuracies when capturing hand-tracking data. The participant completes a task when they lift the object 20cm above its start point, but fail the task if they take longer than 30 seconds, or knock the object out of the environment area. Figure 4.4 pictures a participant taking part in our experiment: teleoperating the robotic-hand model to pick up a ball within the simulation.

The participant attempts ten grasps on each object from Section 3.2. This process is repeated for each robotic-hand model (Section 3.1.4), which are selected in a random order. This helps avoid the situation where a participant learns to better control the hand throughout the experiment, as the cumulative results from multiple participants average out this effect.

The `full` hand model is described with our 27 DOF URDF hand description. The remaining models are generated by systematically removing DOF from the hand-description URDF files (or in the case of `finger-interdep`, altering the **Hand Tracking to Robot Joint Values** module to constrain the finger's MCP, PIP, and DIP flexion-extension joint movements to a single DOF). The full details of the URDF changes made to each model are provided in Appendix B.

From the experiments, we record the teleoperation success rate of each participant on each object and hand model (Objective 1). The full data collected for this objective is listed in Appendix E.

We also track the state of the simulator, and the actions performed by the participant to pick up each object (Objective 2). This provides us with our *expert demonstrations*: sets of state-action (s, a) pairs that describe the human's decision a made in each environment situation s.

The full experiment details, along with our ethics approval, can be found in Appendix D. We present the results of this experiment in the Results chapter (Section 5.1).

4.3 Imitation Learning Framework

Our *imitation learning framework* differs from the *data collection framework* by sending joint messages via a RL/IL agent, instead of human-participant teleoperation. Despite being called our *imitation* learning framework, we can also implement RL by removing the expert demonstrations, and replacing the IL algorithms with RL algorithms. Section 4.3.1 describes our `Gymnasium` environment node, which interfaces between our RL/IL algorithms and the robotics-simulation environment. We also discuss the RL/IL reward functions, termination conditions, and truncation cut-offs in Section 4.3.2.

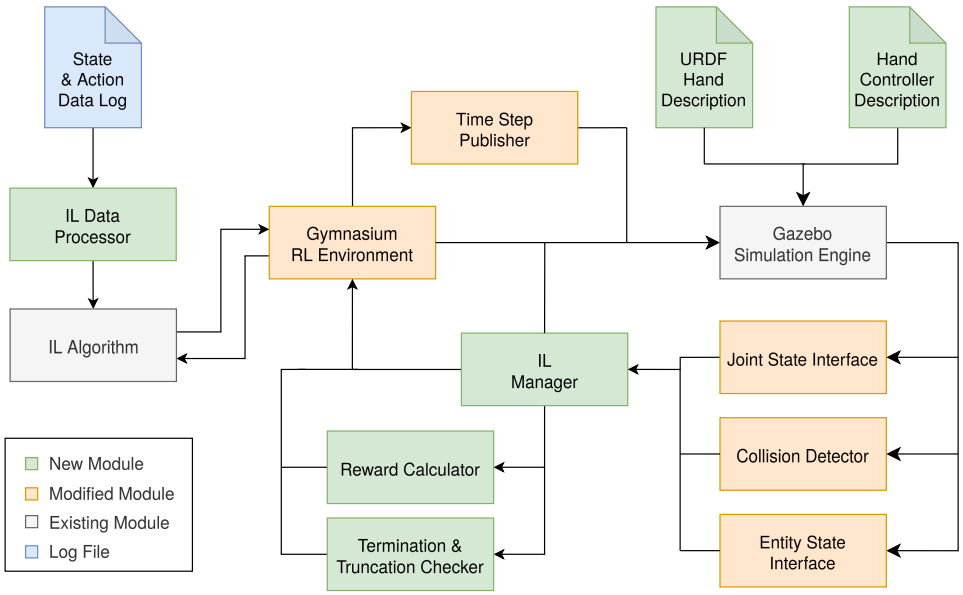

Figure 4.5: An architectural overview of the *imitation learning framework* (Section 4.3). As before, arrows represent data-flow, and modules loosely represent nodes in the ROS2 architecture. *New modules* refer to modules we write from scratch, and *modified modules* refer to existing modules which we modify and/or write a new interface for.

4.3.1 Gymnasium RL Environment

`Gymnasium` [126] is an open source Python library which provides an API between RL/IL algorithms and MDP environments. By formulating our robotic-hand simulation task as an MDP environment, we can use any RL/IL algorithms compatible with the `Gymnasium` interface to train the hand. In our case, we use RL algorithm implementations from `stable-baselines3` [127] and IL algorithms from the `imitation` library [128].

State (S)**.** The **IL Manager** captures the simulator's state: the same state collected by the **Teleoperation Manager** in Section 4.1.3.

Action (A)**.** These are the 27 floating-point values which control each DOF in the simulated robotic hand.

Transition ($T : S \times A \to S$)**.** This is the Gazebo simulator itself: given the simulator's state s and the robot's current joint-angle values a, Gazebo produces the new state s' by running the next simulation step.

Reward ($R : S \times A \times S \to \mathbb{R}$)**.** Our task's reward function is described in Section 4.3.2.

The **IL Data Processor** processes our state-action (s, a) pairs into the format expected by each IL algorithm.

4.3.2 Rewards, Termination & Truncation

The **Reward Calculator** module produces a reward depending on the simulator's current state. Our reward function is inspired by Lu et al. [129], who reward the robot for

interacting, grasping, and picking up the object. We measure *interaction* through collision detection: we add a reward of $r_{\text{collision}} \cdot d$ where $r_{\text{collision}}$ is a tunable parameter, and d is the number of digits touching the object. If the thumb is opposed, and at least the thumb and two other fingers are touching the object, we say that the object is *grasped* and add a r_{grasped} reward. To reward the robot for picking up the object, we add a reward of $r_{\text{picked-up}} \cdot h$, where h is the height (in cm) the object has been lifted above its starting point.

We use the same task completion, truncation, and termination rules as in our teleoperation experiments, calculated in the **Termination & Truncation Checker**. If the robot completes the task it receives a r_{complete} reward, however, if the episode is truncated or terminated it receives a $-r_{\text{trucated}}$ or $-r_{\text{terminated}}$ punishment.

The values for each parameter r_x were tuned manually based on observation. We found the optimal parameters to be:

$$\begin{aligned} r_{\text{collision}} &= 2.0, \\ r_{\text{grasped}} &= 5.0, \\ r_{\text{picked-up}} &= 0.5, \\ r_{\text{trucated}} &= 500.0, \\ r_{\text{terminated}} &= 500.0, \\ r_{\text{complete}} &= 5000.0 \end{aligned}$$

4.4 Summary

Our *data collection framework* collects hand-tracking data from a participant's hand (Section 4.1.2) to control the joints of our robotic-hand simulation (described with a URDF hand description, and a hand controller description, Section 4.1.1). Our hand achieves **state-of-the-art** dexterity, and to the best of our knowledge, we are the **first** to use a robotic-hand simulation with *varying dexterity* to conduct teleoperation, RL, and IL dexterity trials. The data collected from the participant experiments (Section 4.2) is stored as state-action pairs (Section 4.1.3) to be used as training data in our imitation learning algorithms.

Our *imitation learning framework* describes how our robotic-hand simulation can be trained using existing RL and IL algorithms. We formulate the simulation environment as a MDP, which can be implemented as a `Gymnasium` [126] environment (Section 4.3.1). We then describe the termination and truncation rules for the grasping task, and the reward functions used to guide the training algorithms (Section 4.3.2).

Chapter 5

Results

This chapter explores our experimental results. Recalling our core research questions from Section 1.1: we examine how robotic-hand dexterity affects a robot's ability to be *controlled by a human* (Section 5.1), to *learn without human demonstrations* (Section 5.2), and to *learn with human demonstrations* (Section 5.3). We conclude by comparing the performance of the teleoperation, RL, and IL approaches in Section 5.4.

5.1 Teleoperation

This section presents the results of our participant teleoperation experiments from Section 4.2. Recalling Section 3.1.4, the `full` model matches the dexterity of a human hand, whereas the other models omit degrees of freedom (corresponding to sets of DOF often excluded in previous robotic-hand designs). Our experiments analyse the effect of removing these DOF, to determine whether participants find it easier to control a more human-like hand.

Our results (Figure 5.1) show that this is indeed the case: on average, participants teleoperating the `full` model have a **15% higher success rate** than for all the other hand models. Furthermore, the `full` model outperforms (or matches) all other models on 5 out of the 6 grasping tasks. This provides strong empirical evidence that future robotic hands (with human-like dexterity) will be easier to teleoperate than existing robotic-hand designs. The full experimental results can be found in Appendix E.

We hypothesise two reasons for the `full` model performs better:

Hypothesis 1. More dexterous hands allow the participant to exhibit more complex behaviour, which is necessary to complete the grasping tasks effectively.

Hypothesis 2. It is easier for a human to control a robotic hand whose structure and dexterity is closer aligned to their own.

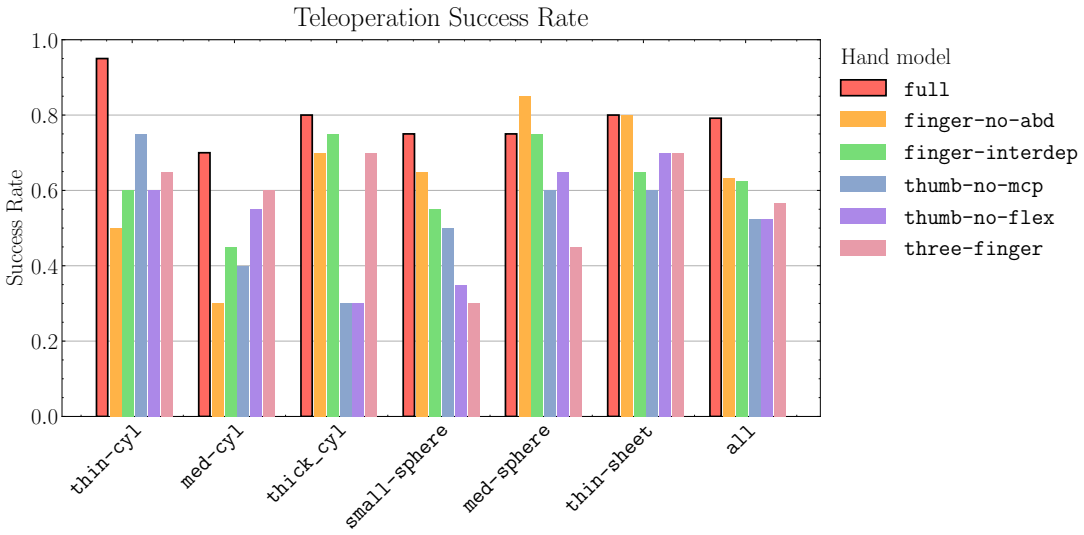

Figure 5.1: The **teleoperation** success rate for completing each grasping task using each robotic hand. The `full` hand model (bolded) matches human dexterity, whereas all remaining hands omit some DOFs. Each bar represent the success rate after 20 grasps, except the `all` bar which aggregates results across all previous experiments. Error bars are omitted as few participants grasped each object.

To investigate these hypothesise, we invited participants to an optional post-experiment interview, where we asked them to describe their teleoperation experience with each hand model (without telling them how each model was different). Three out of the seven participants agreed to this interview. Their responses, alongside our results, support both of our hypothesise, which suggests that they are both true in conjunction.

When asked to describe the `thumb-no-mcp` (removing the thumb's intermediary joint) and `thumb-no-flex` (only allowing thumb movements towards-and-away from the palm) models, all three interviewees mentioned that they struggled to grasp objects without the full dexterity of the thumb. One participant noted that the "unnaturally" long thumb prevented them from reaching around the object, resulting in a weaker grip (Hypothesis 1). Although the thumb was no longer than in the other models, the restrictions on the `mcp` joint made the thumb feel longer than it really was. This suggests that small differences between the participant and model's joint structures can make control feel unnatural and foreign, despite the similarities between their overall structure (Hypothesis 2). These comments, along with the fact these models performed worst on average, highlight the importance of the thumb's dexterity when completing manipulation tasks.

The `three-finger` model (removing the pink finger) performed worst on the sphere objects. This result is unsurprising as, unlike the other grasps, the sphere-based grasps in the GRASP taxonomy [1] use all four fingers (Hypothesis 1). Despite discovering a new approach to compensate for the missing dexterity, two interviewees mentioned that they found it challenging to adapt to the new grasping method. Since this technique was new and unfamiliar, participants struggled to execute it consistently: achieving a success-rate 20% lower than for the other objects using this model (Hypothesis 2).

When interviewed about the `full` model, all three participants unanimously agreed that it was the easiest to control — albeit not by a significant margin compared to the `finger-no-abd` (removing finger splay) and `finger-interdep` (modelling the finger joint's *curl* movements as linearly correlated) models. Overall, the participant's comments indicate that they felt the least restricted and most comfortable with a robotic hand which matched their own dexterity: supporting both Hypotheses 1 & 2.

Hence, we believe that more dexterous models are both able to exhibit more complex behaviour, and are easier for participants to control.

5.2 Reinforcement Learning

This section presents the results of our reinforcement-learning experiments. We begin by training the `full` hand model using various RL algorithms, to identify which is the most effective in our simulation environment. We test each algorithm (from Section 3.3) on the `med-sphere` grasping task, and record their episode reward histories in Figure 5.2.

Both PPO and A2C outperformed the TD3, SAC, and DDPG algorithms by a large margin. The key difference between these algorithms, is that PPO and A2C are *on-policy* algorithms, whereas TD3, SAC, and DDPG are *off-policy* algorithms. Unlike on-policy algorithms, off-policy algorithms explicitly separate out the *behaviour* policy (for selecting actions) and the *target* policy (the optimal policy learnt throughout training). While this improves their ability to explore the environment, performance may suffer due to using an outdated learning policy for action selection at each step.

From manually observing the simulation, we discovered that the off-policy algorithms immediately knock the object out of the environment area, causing them to quickly fail and terminate the environment. This suggests that the on-policy approach of committing to a good (though potentially suboptimal) local minima is more effective than pursuing greater exploration.

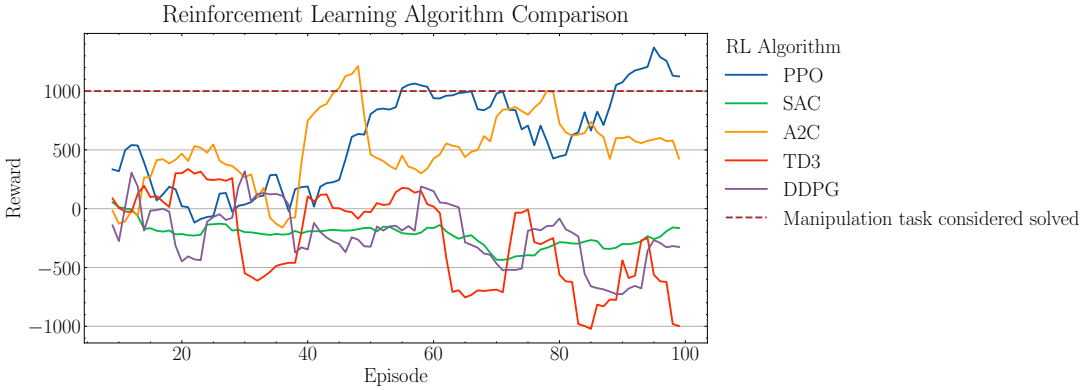

Figure 5.2: The rolling-average reward history of **RL** algorithms on the `med-sphere` grasping task using the `full` hand. Each algorithm is trained for 100 episodes (∼400K steps). We consider the grasping task to be complete with a reward of 1000 ($= r_{\text{complete}}$).

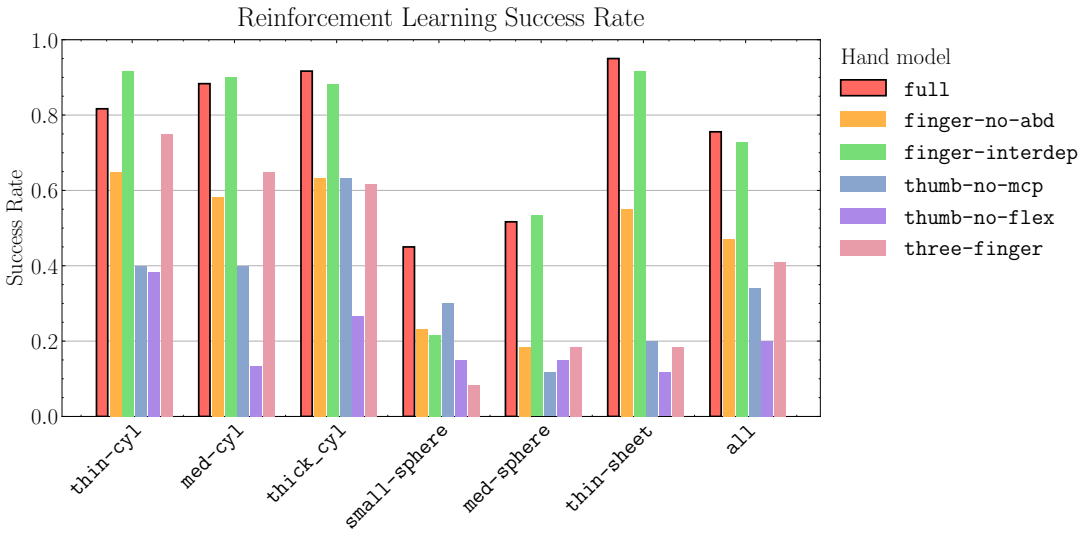

Figure 5.3: The **RL** success rate for completing each grasping task using each robotic hand, after training for 300 episodes (∼1M steps). Error bars are omitted as each experiment was run only once (due to resource constraints).

We used PPO to conduct our RL trials, as it performed better and more consistently than A2C overall. Our results can be found in Figure 5.3, where we find that our **`full` model performs the best on average, outperforming all other models (except `finger-interdep`) by over 25%**.

Similar to our teleoperation experiments, the `thumb-no-mcp`, `thumb-no-flex`, and `three-finger` models performed the worst. The lack of dexterity from these missing sets of DOF consistently limits the performance of these models across our trials.

The `finger-interdep` model performs surprisingly well, achieving only a 2.5% lower success rate than the `full` model. One explanation is that these DOFs may not have a large effect, so by removing them we give the agent less parameters to learn (improving its learning efficiency). Or, the `full` model agent may learn to not make use of these DOFs, hence producing similar results to `finger-inderdep`. Manually observing the simulation shows us that the `full` hand model does in fact use these DOF effectively, suggesting the former explanation to be true.

5.3 Imitation Learning

Similar to the RL experiments, we compare the performance of our chosen IL algorithms (Section 3.4), to identify which is the most effective in our simulation. We find both of the IRL algorithms (AIRL and DBRM) to perform the best (Figure 5.4). This suggests that our manually-defined reward function was suboptimal, and these algorithms were able to learn a more effective function. We use the DBRM algorithm throughout our remaining IL experiments, as it achieved the highest episode reward overall.

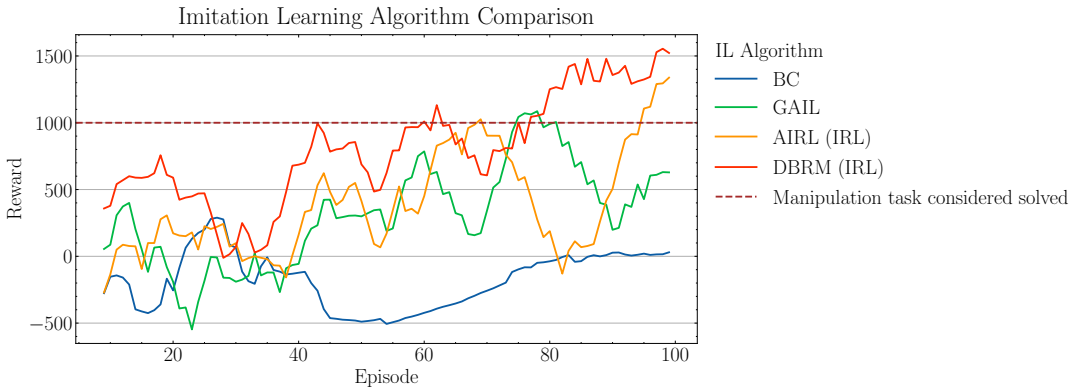

Figure 5.4: The rolling-average reward history of **IL** algorithms on the `med-sphere` grasping task using the `full` hand. Each algorithm is trained for 100 episodes (∼400K steps). We preliminarily measure algorithm performance against our manually-defined reward function. The expert demonstrations contain all teleoperation attempts on only the `med-sphere` object.

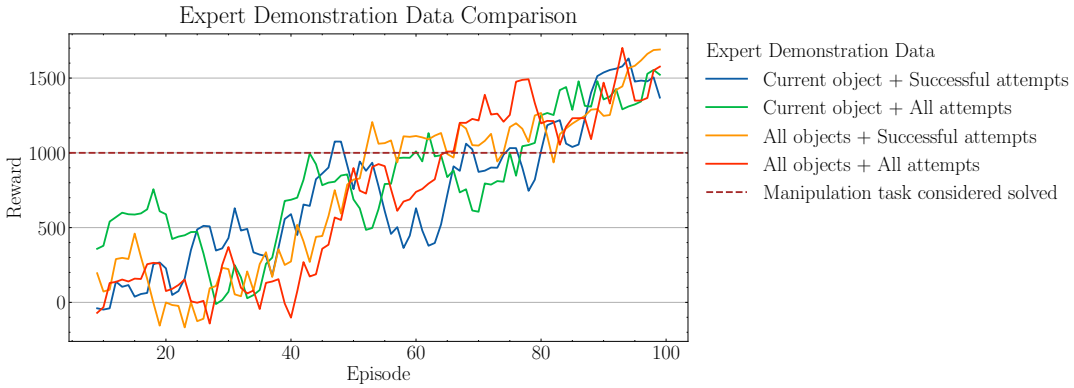

Figure 5.5: The rolling-average reward history of the **DBRM (IRL)** algorithm on the `med-sphere` grasping task using the `full` hand, using different expert demonstration datasets. Each algorithm is trained for 100 episodes (∼400K steps).

Our initial benchmark (Figure 5.4) draws expert demonstrations from all teleoperation attempts (pass or fail) on only the `med-sphere` task. However, we wish to explore how different expert demonstration datasets may affect our results. For example, does performance improve if we only use *successful* teleoperation attempts, and should the demonstration data only include the object we are *currently* training on, or all objects? Our results (Figure 5.5) show that the choice of demonstration data makes little difference, suggesting that the algorithm learns the same general grasping action regardless of the dataset. Throughout our remaining experiments, we use *all* teleoperation attempts on the *current* object (and hand model), as its success rate marginally outperforms those from the other datasets.

Our results closely align with those from our RL experiments, with the *comparative* performance of each hand model being near identical. Notably, the **`full` model outperforms `finger-interdep` by 7%, and all other models by over 20% on average**. This reinforces our belief that more dexterous hands (closer in structure to human hands) are able to learn better from humans.

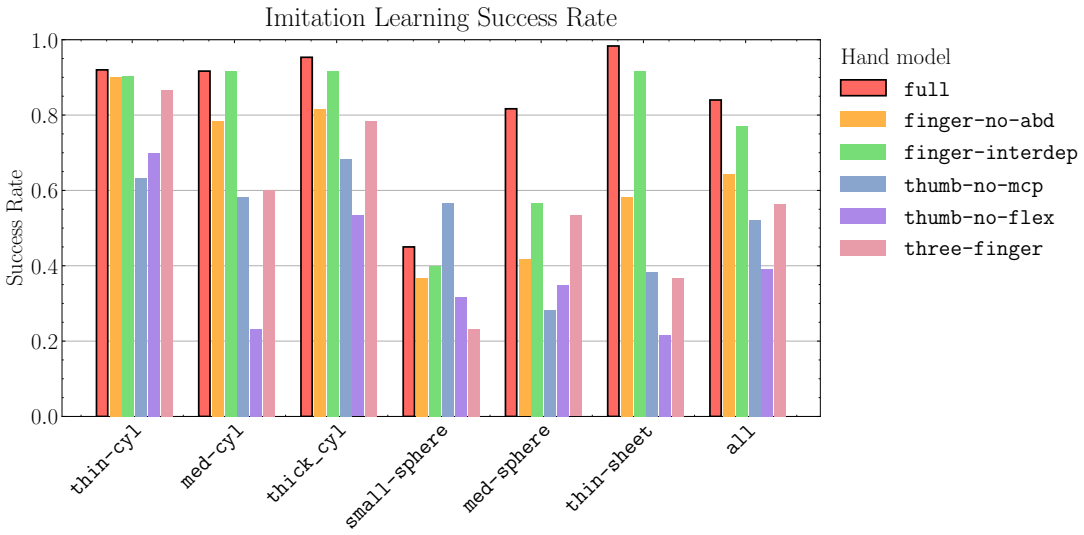

Figure 5.6: The **IL** success rate for completing each grasping task using each robotic hand, after training for 300 episodes (∼1M steps). The expert demonstrations contain all teleoperation attempts on only the current object.

5.4 Approach Comparison

Although our investigation does not focus on comparing the teleoperation, RL, and IL approaches, we include this analysis for the sake of completeness (Figure 5.7).

It is promising to find that IL consistently outperforms RL, given that it has more information to work with (in the form of expert demonstrations). Somewhat surprisingly, **IL also outperforms teleoperation** overall. This suggests that our hand was able to not only to learn from demonstrations, but to improve upon them.

Notably, IL outperforms teleoperation on the majority of cylinder tasks. This suggests that these tasks have a consistent grasping strategy, which can be reliably learnt from watching human demonstrations. However, compared to teleoperation, IL struggles with the *thin sheet* and *medium sphere* tasks. These tasks require fine motor control, which suggests that humans are better than IL at performing precise tasks.

5.5 Summary

The **teleoperation** results show the **`full` hand model to outperform all other (less-dexterous) hand models by an average of 15%**. Our analysis suggests that more dexterous hands, closer in structure the participant's own, are easier to control and are able to exhibit more complex behaviour.

The RL and IL results show that the **`full` model performs the best on average, outperforming all other models (except `finger-interdep`) by over 20%**. The missing sets of DOF from the `thumb-no-mcp`, `thumb-no-flex`, and `three-finger` models

consistently limits their performance across our trials.

These results **provide strong empirical evidence that improving a robotic-hand's dexterity (up to and including that of a human hand) improves the robot's ability to be controlled by, and to learn from humans**.

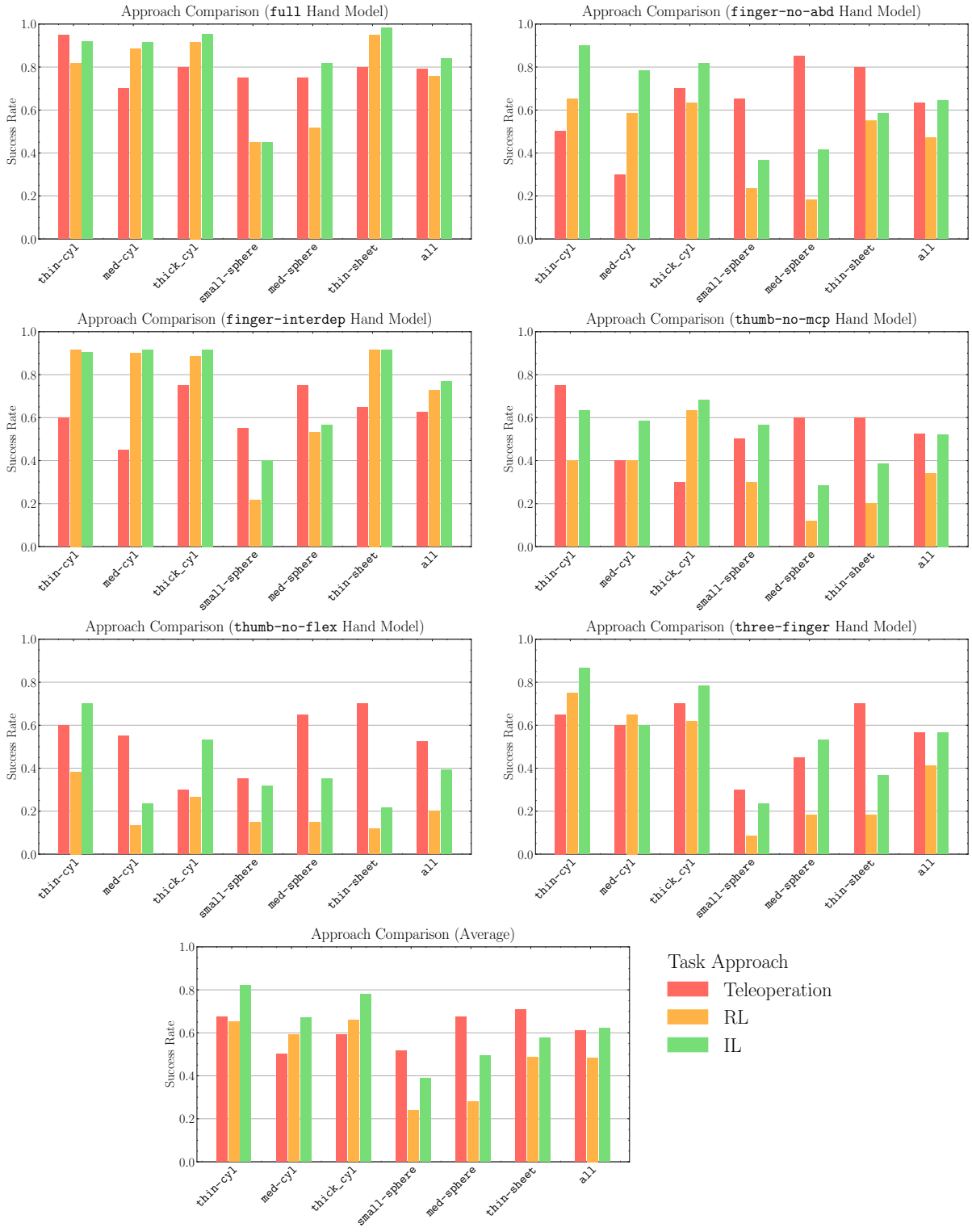

Figure 5.7: Comparison of the teleoperation, RL, and IL approaches (consolidating Figures 5.1, 5.3, and 5.6). The *Average* plot takes the average across all hand models.

Chapter 6

Summary and Conclusions

In this project, we aim to answer the question: *does improving a robotic-hand's dexterity (up to and including that of a human hand) improve the robot's ability to be controlled by, and to learn from humans?*

We achieve this by analysing how the dexterity of humanoid robotic hands affect their ability to complete tasks via teleoperation, RL, and IL. Our 27 DOF hand model (with human-like dexterity) **outperforms all other (less-dexterous) hand models on the majority of benchmarks** (Section 5). Notably, this model achieves a **15% higher average success rate** on our teleoperation experiments (Section 5.1), and outperforms all other models (except `finger-interdep`) by an average of **20%** on our RL and IL benchmarks (Sections 5.2 and 5.3).

Hence, we are the **first** to provide empirical evidence that improving robotic-hand dexterity — up to that of a human hand — improves their ability to be controlled by, and to learn from humans. This highlights the potential of soon-to-be-developed 27 DOF physical robotic hands: our analysis indicates that these hands will be capable of completing tasks beyond the reach of today's models.

As part of this project, we build a rigid-body robotic-hand simulation with **state-of-the-art dexterity** (Section 4.1.1). Conducting a systematic literature review of dexterous robotic hands (Section 3.1.2) reveals that our robot is the **first simulated or physical robotic model to achieve human-level dexterity**.

To our knowledge, this is the **first time** a robotic hand with *adjustable dexterity* has been used to conduct teleoperation, RL, and IL dexterity trials. This allows us to perform novel analysis on robotic-hand dexterity: analysing the effect of systematically removing different DOFs from a robotic-hand model.

We show that sets of DOF often omitted by previous robot models (Section 3.1.4) have a large impact on their dexterity. In particular, excluding the thumb's MCP joint (`thumb-no-mcp`) or flexion-extension motions (`thumb-no-flex`), or removing the pinky

finger (`three-finger`), leads to poor performance across all our benchmarks. However, our analysis finds that more dexterous hands (closer in structure the participant's own) are easier to control, and are able to exhibit more complex grasping behaviour.

With future advancements in robotics and machine learning, problems surrounding human-robot interaction will grow evermore important. Our thesis seeks to contribute to this field, by introducing a new approach to understanding the effect of dexterity on biomimetic robotic hands. We **publish our robotic-hand models and framework implementations** (Section 4) as an open-source ROS2 package[1], with the hopes of stimulating further work in the field.

6.1 Reflections & Future Work

The effect of dexterity on biomimetic robots is a largely unexplored field; we propose the following future directions based upon the results of our investigation.

- Our experiments are restricted to *manipulation tasks*, which despite being a major component, only represent part of robotic-hand control. Future work could build upon our experiments to explore other forms of control, such as *gestures* [130, 131, 132], *tool manipulation* [133, 134], and *human interaction* [135, 136, 137]. Although benchmarks for such experiments have been created [138, 139, 140, 141, 142], they have not been used to systematically analyse the effect of dexterity.
- We explore the impact of dexterity on humanoid robots from a purely practical perspective. However, our investigation would be benefit from connecting our work to theoretical findings drawn from existing biological literature. For example, extensive research has investigated how the anatomy of the *opposable thumb* affects a human's ability to complete tasks (compared to other mammals lacking this dexterity) [14, 143, 144, 36, 145, 106]. This research could act as the foundation for a systematic analysis into the role of each DOF in a humanoid robotic hand.
- Our experiments explore the effect of removing different sets of DOF from the 27 DOF hand model. However, we do not explore the *compounding* effect of removing multiple sets of DOF at once. The majority of robotic hands surveyed in our literature review (Section 3.1.2) omit more than one set of DOF (such as the UTAH/MIT hand [53], which excludes both the pinky finger and finger abduction-adduction movements). Future work could explore interactions between different sets of DOF in a robotic-hand model.

Important questions, such as how dexterity affects a robot's ability to work alongside humans, are yet to be answered. We hope to see subsequent work build upon our analysis, exploring the future impact of robots which are soon to achieve human-like dexterity.

[1] `https://github.com/DylanMoss1/Humanoid-Robotic-Hand-Dexterity-Analysis`

Bibliography

[1] Thomas Feix et al. "The grasp taxonomy of human grasp types". In: *IEEE Transactions on human-machine systems* 46.1 (2015), pp. 66–77.

[2] Mary Edwards. "Robots in industry: An overview". In: *Applied ergonomics* 15.1 (1984), pp. 45–53.

[3] Antoni Grau et al. "Robots in industry: The past, present, and future of a growing collaboration with humans". In: *IEEE Industrial Electronics Magazine* 15.1 (2020), pp. 50–61.

[4] Sara Bragança et al. "A brief overview of the use of collaborative robots in industry 4.0: Human role and safety". In: *Occupational and environmental safety and health* (2019), pp. 641–650.

[5] Robert Sparrow and Mark Howard. "Robots in agriculture: prospects, impacts, ethics, and policy". In: *precision agriculture* 22 (2021), pp. 818–833.

[6] Juan Jesús Roldán et al. "Robots in agriculture: State of art and practical experiences". In: *Service robots* 12.2 (2018), pp. 67–90.

[7] Helge Ritter and Robert Haschke. "Hands, dexterity, and the brain". In: (2015).

[8] Salah Bazzi and Dagmar Sternad. "Human control of complex objects: towards more dexterous robots". In: *Advanced Robotics* 34.17 (2020), pp. 1137–1155.

[9] John Rasmussen. "Challenges in human body mechanics simulation". In: *Procedia Iutam* 2 (2011), pp. 176–185.

[10] A. Bicchi. "Hands for dexterous manipulation and robust grasping: a difficult road toward simplicity". In: *IEEE Transactions on Robotics and Automation* 16.6 (2000), pp. 652–662. DOI: 10.1109/70.897777.

[11] Saeed Saeedvand et al. "A comprehensive survey on humanoid robot development". In: *The Knowledge Engineering Review* 34 (2019), e20.

[12] Deepa Ahirwar et al. "The recent advancements in humanoid robot technology". In: *2022 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*. IEEE. 2022, pp. 1–6.

[13] Yuchuang Tong, Haotian Liu, and Zhengtao Zhang. "Advancements in Humanoid Robots: A Comprehensive Review and Future Prospects". In: *IEEE/CAA Journal of Automatica Sinica* 11.2 (2024), pp. 301–328.

[14] Lynette A Jones and Susan J Lederman. *Human hand function*. Oxford university press, 2006.

[15] Robert J Schwarz and C Taylor. “The anatomy and mechanics of the human hand”. In: *Artificial limbs* 2.2 (1955), pp. 22–35.

[16] Fai Chen Chen et al. “Constraint study for a hand exoskeleton: human hand kinematics and dynamics”. In: *Journal of Robotics* 2013 (2013).

[17] Ethel J Alpenfels. “The anthropology and social significance of the human hand”. In: *Artificial limbs* 2.2 (1955), pp. 4–21.

[18] Mary W Marzke and Robert F Marzke. “Evolution of the human hand: approaches to acquiring, analysing and interpreting the anatomical evidence”. In: *The Journal of Anatomy* 197.1 (2000), pp. 121–140.

[19] Fan Xie et al. “Deep imitation learning for bimanual robotic manipulation”. In: *Advances in neural information processing systems* 33 (2020), pp. 2327–2337.

[20] Bin Fang et al. “Survey of imitation learning for robotic manipulation”. In: *International Journal of Intelligent Robotics and Applications* 3 (2019), pp. 362–369.

[21] Joseph Stiles Beggs. *Kinematics*. CRC Press, 1983.

[22] José Luis Blanco-Claraco. “A tutorial on E3 transformation parameterizations and on-manifold optimization”. In: *arXiv preprint arXiv:2103.15980* (2021).

[23] Jrvz. *Yaw Axis Corrected.* 2010. URL: `https://commons.wikimedia.org/w/index.php?curid=9441238` (visited on 05/05/2024).

[24] William L. Luyben. “Design and Control Degrees of Freedom”. In: *Industrial & Engineering Chemistry Research* 35.7 (1996), pp. 2204–2214. DOI: `10.1021/ie960038d`. eprint: `https://doi.org/10.1021/ie960038d`. URL: `https://doi.org/10.1021/ie960038d`.

[25] Salvador Cobos et al. “Efficient human hand kinematics for manipulation tasks”. In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2008, pp. 2246–2251. DOI: `10.1109/IROS.2008.4651053`.

[26] Salvador Cobos et al. “Efficient human hand kinematics for manipulation tasks”. In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2008, pp. 2246–2251. DOI: `10.1109/IROS.2008.4651053`.

[27] J. Chalfoun et al. “Muscle forces prediction of the human hand and forearm system in highly realistic simulation”. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 2. 2004, 1293–1298 vol.2. DOI: `10.1109/IROS.2004.1389574`.

[28] Esteban Peña-Pitarch, Jingzhou Yang, and Karim Abdel-Malek. “SANTOS™ hand: A 25 degree-of-freedom model”. In: (June 2005). DOI: `10.4271/2005-01-2727`.

[29] Huan Du and Edoardo Charbon. “3D Hand Model Fitting for Virtual Keyboard System”. In: *2007 IEEE Workshop on Applications of Computer Vision (WACV '07)*. 2007, pp. 31–31. DOI: `10.1109/WACV.2007.3`.

[30] L. Van Gool. “Stochastic optimisation for high-dimensional tracking in dense range maps”. English. In: *IEE Proceedings - Vision, Image and Signal Processing* 152 (4 Aug. 2005), 501–512(11). ISSN: 1350-245X. URL: https://digital-library.theiet.org/content/journals/10.1049/ip-vis_20045113.

[31] Esteban Pena-Pitarch, Neus Ticó Falguera, and Jingzhou Yang. “Virtual human hand: model and kinematics”. In: *Computer methods in biomechanics and biomedical engineering* 17.5 (2014), pp. 568–579.

[32] Frank PJ Van Der Hulst et al. “A functional anatomy based kinematic human hand model with simple size adaptation”. In: *2012 IEEE International Conference on Robotics and Automation*. IEEE. 2012, pp. 5123–5129.

[33] Mahmoud Tavakoli, Rafael Batista, and Lucio Sgrigna. “The UC Softhand: Light Weight Adaptive Bionic Hand with a Compact Twisted String Actuation System”. In: *Actuators* 5 (Dec. 2015), p. 1. DOI: 10.3390/act5010001.

[34] Outlander Anatomy. *Anatomy Lesson*. 2024. URL: https://www.outlanderanatomy.com/harming-hands-helping-hands-healing-hands/ (visited on 05/27/2024).

[35] Corinne Viscogliosi. *Physical Therapy*. 2024. URL: https://www.pinterest.co.uk/pin/217509856978240707/ (visited on 05/27/2024).

[36] Visakha K Nanayakkara et al. “The role of morphology of the thumb in anthropomorphic grasping: a review”. In: *Frontiers in mechanical engineering* 3 (2017), p. 5.

[37] Mathworks. *What Is Reinforcement Learning?* 2024. URL: https://www.mathworks.com/help/reinforcement-learning/ug/what-is-reinforcement-learning.html (visited on 05/27/2024).

[38] Shixin Mao et al. “Gait study and pattern generation of a starfish-like soft robot with flexible rays actuated by SMAs”. In: *Journal of Bionic Engineering* 11.3 (2014), pp. 400–411.

[39] George M Whitesides. “Soft robotics”. In: *Angewandte Chemie International Edition* 57.16 (2018), pp. 4258–4273.

[40] Gursel Alici. “Softer is harder: What differentiates soft robotics from hard robotics?” In: *MRS Advances* 3.28 (2018), pp. 1557–1568.

[41] Ebrahim Mattar. “A survey of bio-inspired robotics hands implementation: New directions in dexterous manipulation”. In: *Robotics and Autonomous Systems* 61.5 (2013), pp. 517–544. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2012.12.005. URL: https://www.sciencedirect.com/science/article/pii/S0921889013000031.

[42] Mariangela Manti, Vito Cacucciolo, and Matteo Cianchetti. “Stiffening in soft robotics: A review of the state of the art”. In: *IEEE Robotics & Automation Magazine* 23.3 (2016), pp. 93–106.

[43] elprocus. *What is a Closed Loop Control System and Its Working*. 2024. URL: https://www.elprocus.com/what-is-a-closed-loop-control-system-its-working/ (visited on 05/27/2024).

[44] ros. *ROS - Robot Operating System*. 2016. URL: https://www.ros.org/ (visited on 05/27/2024).

[45] José M Cañas et al. "A ROS-based open tool for intelligent robotics education". In: *Applied Sciences* 10.21 (2020), p. 7419.

[46] ros-controls. *gazebo-ros2-control*. 2020. URL: https://github.com/ros-controls/gazebo_ros2_control (visited on 05/27/2024).

[47] ros-simulation. *gazebo-ros-pkgs*. 2019. URL: https://github.com/ros-simulation/gazebo_ros_pkgs (visited on 05/27/2024).

[48] arshadlab. *gazebo-step-control*. 2021. URL: https://github.com/arshadlab/gazebo_step_control (visited on 05/27/2024).

[49] Tsuneo Yoshikawa. "Multifingered robot hands: Control for grasping and manipulation". In: *Annual Reviews in Control* 34.2 (2010), pp. 199–208.

[50] Suhas Kadalagere Sampath et al. "Review on human-like robot manipulation using dexterous hands". In: *Cognitive Computation and Systems* 5.1 (2023), pp. 14–29.

[51] Chunmiao Yu and Peng Wang. "Dexterous manipulation for multi-fingered robotic hands with reinforcement learning: a review". In: *Frontiers in Neurorobotics* 16 (2022), p. 861825.

[52] Daniel R Ramirez Rebollo, Pedro Ponce, and Arturo Molina. "From 3 fingers to 5 fingers dexterous hands". In: *Advanced Robotics* 31.19-20 (2017), pp. 1051–1070.

[53] Steve Jacobsen et al. "Design of the Utah/MIT dextrous hand". In: *Proceedings. 1986 IEEE International Conference on Robotics and Automation*. Vol. 3. IEEE. 1986, pp. 1520–1532.

[54] Joerg Butterfass et al. "DLR's multisensory articulated hand. I. Hard-and software architecture". In: *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*. Vol. 3. IEEE. 1998, pp. 2081–2086.

[55] Company SR. *Shadow Hand*. 2016. URL: https://www.shadowrobot.com/dexterous-hand-series/ (visited on 05/27/2024).

[56] Allegro Hand. *Allegro Hand*. 2023. URL: https://www.allegrohand.com/ (visited on 05/27/2024).

[57] Manuel Wüthrich et al. "Trifinger: An open-source robot for learning dexterity". In: *arXiv preprint arXiv:2008.03596* (2020).

[58] Henry Zhu et al. "Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 3651–3657.

[59] Vikash Kumar, Zhe Xu, and Emanuel Todorov. “Fast, strong and compliant pneumatic actuation for dexterous tendon-driven hands”. In: *2013 IEEE international conference on robotics and automation*. IEEE. 2013, pp. 1512–1519.

[60] Company SR. *Shadow Hand*. 2016. URL: `https://www.shadowrobot.com/dexterous-hand-series/` (visited on 03/01/2016).

[61] Li-Ren Lin and Han-Pang Huang. “NTU hand: A new design of dexterous hands”. In: (1998).

[62] Ahmet Atasoy et al. “24 DOF EMG controlled hybrid actuated prosthetic hand”. In: *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE. 2016, pp. 5059–5062.

[63] Davide Liconti, Yasunori Toshimitsu, and Robert Katzschmann. “Leveraging Pretrained Latent Representations for Few-Shot Imitation Learning on a Dexterous Robotic Hand”. In: *arXiv preprint arXiv:2404.16483* (2024).

[64] Ikuo Yamano and Takashi Maeno. “Five-fingered robot hand using ultrasonic motors and elastic elements”. In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE. 2005, pp. 2673–2678.

[65] Mathieu Grossard, Javier Martin, and Gabriel Felippe da Cruz Pacheco. “Control-oriented design and robust decentralized control of the CEA dexterous robot hand”. In: *IEEE/ASME Transactions on Mechatronics* 20.4 (2014), pp. 1809–1821.

[66] Christopher Y Brown and H Harry Asada. “Inter-finger coordination and postural synergies in robot hands via mechanical implementation of principal components analysis”. In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2007, pp. 2877–2882.

[67] Kiyoshi Hoshino and Ichiro Kawabuchi. “Pinching with finger tips in humanoid robot hand”. In: *ICAR’05. Proceedings., 12th International Conference on Advanced Robotics, 2005.* IEEE. 2005, pp. 705–712.

[68] Haruhisa Kawasaki and Tsuneo Komatsu. “Mechanism design of anthropomorphic robot hand: Gifu hand I”. In: *Journal of Robotics and Mechatronics* 11.4 (1999), pp. 269–273.

[69] Haruhisa Kawasaki, Tsuneo Komatsu, and Kazunao Uchiyama. “Dexterous anthropomorphic robot hand with distributed tactile sensor: Gifu hand II”. In: *IEEE/ASME transactions on mechatronics* 7.3 (2002), pp. 296–303.

[70] Haruhisa Kawasaki, Tetsuya Mouri, and Satoshi Ito. “Toward next stage of kinetic humanoid hand”. In: *Proceedings World Automation Congress, 2004.* Vol. 15. IEEE. 2004, pp. 129–134.

[71] Michele Folgheraiter and Giuseppina Gini. “Blackfingers an artificial hand that copies human hand in structure, size, and function”. In: *Proc. IEEE Humanoids* (2000), p. 4.

[72] Andrea Caffaz and Giorgio Cannata. “The design and development of the DIST-Hand dextrous gripper”. In: *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*. Vol. 3. IEEE. 1998, pp. 2075–2080.

[73] Yuru Zhang et al. “Design and control of the BUAA four-fingered hand”. In: *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*. Vol. 3. IEEE. 2001, pp. 2517–2522.

[74] Dong-Hyuk Lee et al. “KITECH-hand: A highly dexterous and modularized robotic hand”. In: *IEEE/ASME Transactions on Mechatronics* 22.2 (2016), pp. 876–887.

[75] Nicholas Thayer and Shashank Priya. “Design and implementation of a dexterous anthropomorphic robotic typing (DART) hand”. In: *Smart Materials and Structures* 20.3 (2011), p. 035010.

[76] Byungjune Choi et al. “Development of anthropomorphic robot hand with tactile sensor: SKKU Hand II”. In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2006, pp. 3779–3784.

[77] Enzo Romero and Dante A Elias. “Design of a haptic palmar-finger feedback system for upper limb myoelectric prosthesis model PUCP-Hand”. In: *2022 International Conference on Electrical, Computer and Energy Technologies (ICECET)*. IEEE. 2022, pp. 1–6.

[78] Christian Cipriani, Marco Controzzi, and Maria Chiara Carrozza. “The SmartHand transradial prosthesis”. In: *Journal of neuroengineering and rehabilitation* 8 (2011), pp. 1–14.

[79] Ilaria Cerulo et al. “Teleoperation of the SCHUNK S5FH under-actuated anthropomorphic hand using human hand motion tracking”. In: *Robotics and Autonomous Systems* 89 (2017), pp. 75–84.

[80] Wenzeng Zhang et al. “Super under-actuated multi-fingered mechanical hand with modular self-adaptive gear-rack mechanism”. In: *Industrial Robot: An International Journal* 36.3 (2009), pp. 255–262.

[81] Hong Liu et al. “Multisensory five-finger dexterous hand: The DLR/HIT Hand II”. In: *2008 IEEE/RSJ international conference on intelligent robots and systems*. IEEE. 2008, pp. 3692–3697.

[82] Demeng Che and Wenzeng Zhang. “GCUA humanoid robotic hand with tendon mechanisms and its upper limb”. In: *International Journal of Social Robotics* 3 (2011), pp. 395–404.

[83] Hyeonjun Park and Donghan Kim. “An open-source anthropomorphic robot hand system: HRI hand”. In: *HardwareX* 7 (2020), e00100.

[84] Ji-Hun Bae et al. “Development of a low cost anthropomorphic robot hand with high capability”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 4776–4782.

[85] Naoki Fukaya et al. “Design of the TUAT/Karlsruhe humanoid hand”. In: *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)(Cat. No. 00CH37113)*. Vol. 3. IEEE. 2000, pp. 1754–1759.

[86] openhandproject. *Dextrus V1.1 Robotic Hand*. 2016. URL: https://www.instructables.com/Dextrus-v11-Robotic-Hand/ (visited on 05/27/2024).

[87] Jörg Butterfaß et al. “DLR-Hand II: Next generation of a dextrous robot hand”. In: *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*. Vol. 1. IEEE. 2001, pp. 109–114.

[88] Nianfeng Wang, Kunyi Lao, and Xianmin Zhang. “Design and myoelectric control of an anthropomorphic prosthetic hand”. In: *Journal of Bionic Engineering* 14.1 (2017), pp. 47–59.

[89] Inspire Robotics. *RHC-1*. 2024. URL: https://en.inspire-robots.com/product-category/the-dexterous-hands?gad_source=1&gclid=EAIaIQobChMIhqfQg4q3hgMVXJVQBh0o3AGcEAAYASAAEgJgFvD_BwE (visited on 05/27/2024).

[90] Stefan Schulz, Christian Pylatiuk, and Georg Bretthauer. “A new ultralight anthropomorphic hand”. In: *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*. Vol. 3. IEEE. 2001, pp. 2437–2441.

[91] CS Lovchik and Myron A Diftler. “The robonaut hand: A dexterous robot hand for space”. In: *Proceedings 1999 IEEE international conference on robotics and automation (Cat. No. 99CH36288C)*. Vol. 2. IEEE. 1999, pp. 907–912.

[92] Lyndon B Bridgwater et al. “The robonaut 2 hand-designed to do work with tools”. In: *2012 IEEE International Conference on Robotics and Automation*. IEEE. 2012, pp. 3425–3430.

[93] Naoki Fukaya et al. “Development of a five-finger dexterous hand without feedback control: The TUAT/Karlsruhe humanoid hand”. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2013, pp. 4533–4540.

[94] Morgan Quigley et al. “Mechatronic design of an integrated robotic hand”. In: *The International Journal of Robotics Research* 33.5 (2014), pp. 706–720.

[95] Maxime Chalon et al. “Dexhand: a space qualified multi-fingered robotic hand”. In: *2011 IEEE international conference on robotics and automation*. IEEE. 2011, pp. 2204–2210.

[96] Ivo Boblan et al. “A human-like robot hand and arm with fluidic muscles: Biologically inspired construction and functionality”. In: *Embodied Artificial Intelligence: International Seminar, Dagstuhl Castle, Germany, July 7-11, 2003. Revised Papers*. Springer. 2004, pp. 160–179.

[97] Matthew S Johannes et al. “An overview of the developmental process for the modular prosthetic limb”. In: *Johns Hopkins APL Technical Digest* 30.3 (2011), pp. 207–216.

[98] Charalambos Konnaris et al. “Ethohand: A dexterous robotic hand with ball-joint thumb enables complex in-hand object manipulation”. In: *2016 6th IEEE international conference on biomedical robotics and biomechatronics (BioRob)*. IEEE. 2016, pp. 1154–1159.

[99] Immanuel Gaiser et al. “A new anthropomorphic robotic hand”. In: *Humanoids 2008-8th IEEE-RAS International Conference on Humanoid Robots*. IEEE. 2008, pp. 418–422.

[100] Prensilla. *IH2 Azzurra*. 2016. URL: `https://www.prensilia.com/ih2-azzurra-hand/` (visited on 05/27/2024).

[101] Goldman Sachs. *Humanoid robots: Sooner than you might think*. 2022. URL: `https://www.goldmansachs.com/intelligence/pages/humanoid-robots.html` (visited on 05/27/2024).

[102] Jean-Pierre Gazeau et al. “The LMS hand: force and position controls in the aim of the fine manipulation of objects”. In: *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*. Vol. 3. Ieee. 2001, pp. 2642–2648.

[103] Active Robots. *AR10 Humanoid Robotic Hand*. 2013. URL: `https://www.active-robots.com/ar10-humanoid-robotic-hand.html` (visited on 05/27/2024).

[104] Linda Resnik, Frantzy Acluche, and Matthew Borgia. “The DEKA hand: A multifunction prosthetic terminal device—patterns of grip usage at home”. In: *Prosthetics and Orthotics International* 42.4 (2018), pp. 446–454.

[105] Nina Robson, Jong-Seob Won, and Vanessa Audrey. “Towards Modeling Finger Joint Coordination for Natural Motion”. In: *USCToMM Symposium on Mechanical Systems and Robotics*. Springer. 2022, pp. 54–64.

[106] Tom Tyler. “The rule of thumb”. In: *JAC* (2010), pp. 435–456.

[107] Jiang Hua et al. “Learning for a robot: Deep reinforcement learning, imitation learning, transfer learning”. In: *Sensors* 21.4 (2021), p. 1278.

[108] OpenAI: Marcin Andrychowicz et al. “Learning dexterous in-hand manipulation”. In: *The International Journal of Robotics Research* 39.1 (2020), pp. 3–20.

[109] Steffen Puhlmann, Jason Harris, and Oliver Brock. “RBO hand 3: A platform for soft dexterous manipulation”. In: *IEEE Transactions on Robotics* 38.6 (2022), pp. 3434–3449.

[110] Volodymyr Mnih et al. “Human-level control through deep reinforcement learning”. In: *nature* 518.7540 (2015), pp. 529–533.

[111] Dong Han et al. “A survey on deep reinforcement learning algorithms for robotic manipulation”. In: *Sensors* 23.7 (2023), p. 3762.

[112] Bharat Singh, Rajesh Kumar, and Vinay Pratap Singh. "Reinforcement learning in robotic applications: a comprehensive survey". In: *Artificial Intelligence Review* 55.2 (2022), pp. 945–990.

[113] Csaba Szepesvári. *Algorithms for Reinforcement Learning*. Vol. 4. Jan. 2010. DOI: 10.2200/S00268ED1V01Y201005AIM009.

[114] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347 [cs.LG].

[115] Timothy P. Lillicrap et al. *Continuous control with deep reinforcement learning*. 2019. arXiv: 1509.02971 [cs.LG].

[116] David Silver et al. "Deterministic policy gradient algorithms". In: *International conference on machine learning*. Pmlr. 2014, pp. 387–395.

[117] Scott Fujimoto, Herke Hoof, and David Meger. "Addressing function approximation error in actor-critic methods". In: *International conference on machine learning*. PMLR. 2018, pp. 1587–1596.

[118] Volodymyr Mnih et al. *Asynchronous Methods for Deep Reinforcement Learning*. 2016. arXiv: 1602.01783 [cs.LG].

[119] Tuomas Haarnoja et al. *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor*. 2018. arXiv: 1801.01290 [cs.LG].

[120] Tuomas Haarnoja et al. "Reinforcement learning with deep energy-based policies". In: *International conference on machine learning*. PMLR. 2017, pp. 1352–1361.

[121] Justin Fu, Katie Luo, and Sergey Levine. *Learning Robust Rewards with Adversarial Inverse Reinforcement Learning*. 2018. arXiv: 1710.11248 [cs.LG].

[122] Sungjoon Choi et al. "Density matching reward learning". In: *arXiv preprint arXiv:1608.03694* (2016).

[123] Jonathan Ho and Stefano Ermon. "Generative adversarial imitation learning". In: *Advances in neural information processing systems* 29 (2016).

[124] Justin Youney. "A comparison and evaluation of common PID tuning methods". In: (2007).

[125] Ultraleap. *Leap Concepts*. 2024. URL: https://docs.ultraleap.com/api-reference/tracking-api/leapc-guide/leap-concepts.html (visited on 05/27/2024).

[126] Mark Towers et al. *Gymnasium*. Mar. 2023. DOI: 10.5281/zenodo.8127026. URL: https://zenodo.org/record/8127025 (visited on 07/08/2023).

[127] Antonin Raffin et al. "Stable-baselines3: Reliable reinforcement learning implementations". In: *Journal of Machine Learning Research* 22.268 (2021), pp. 1–8.

[128] Adam Gleave et al. *imitation: Clean Imitation Learning Implementations*. arXiv:2211.11972v1 [cs.LG]. 2022. arXiv: 2211.11972 [cs.LG]. URL: https://arxiv.org/abs/2211.11972.

[129] Liangliang Lu et al. “A method of robot grasping based on reinforcement learning”. In: *Journal of Physics: Conference Series*. Vol. 2216. 1. IOP Publishing. 2022, p. 012026.

[130] Laurel D Riek et al. “Cooperative gestures: Effective signaling for humanoid robots”. In: *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE. 2010, pp. 61–68.

[131] Sylvain Calinon and Aude Billard. “Learning of gestures by imitation in a humanoid robot”. In: *Imitation and social learning in robots, humans and animals: behavioural, social and communicative dimensions* (2007), pp. 153–177.

[132] Paul Bremner et al. “Conversational gestures in human-robot interaction”. In: *2009 IEEE international conference on systems, man and cybernetics*. IEEE. 2009, pp. 1645–1649.

[133] Aude Billard and Danica Kragic. “Trends and challenges in robot manipulation”. In: *Science* 364.6446 (2019), eaat8414.

[134] Teppei Tsujita et al. “Humanoid robot motion generation for nailing task”. In: *2008 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*. IEEE. 2008, pp. 1024–1029.

[135] Erhan Oztop, Thierry Chaminade, and David W Franklin. “Human-humanoid interaction: is a humanoid robot perceived as a human?” In: *4th IEEE/RAS International Conference on Humanoid Robots, 2004.* Vol. 2. IEEE. 2004, pp. 830–841.

[136] Vignesh Prasad, Ruth Stock-Homburg, and Jan Peters. “Human-robot handshaking: A review”. In: *International Journal of Social Robotics* 14.1 (2022), pp. 277–293.

[137] Kerstin Dautenhahn et al. “KASPAR–a minimally expressive humanoid robot for human–robot interaction research”. In: *Applied Bionics and Biomechanics* 6.3-4 (2009), pp. 369–397.

[138] Berk Calli et al. “Benchmarking in manipulation research: The ycb object and model set and benchmarking protocols”. In: *arXiv preprint arXiv:1502.03143* (2015).

[139] Kayla Matheus and Aaron M Dollar. “Benchmarking grasping and manipulation: Properties of the objects of daily living”. In: *2010 IEEE/RSJ international conference on intelligent robots and systems*. IEEE. 2010, pp. 5020–5027.

[140] Pieter Wolfert et al. “Should beat gestures be learned or designed?: A benchmarking user study”. In: *ICDL-EPIROB 2019 Workshop on Naturalistic Non-Verbal and Affective Human-Robot Interactions*. IEEE conference proceedings. 2019.

[141] SM Mizanoor Rahman. “Evaluating and benchmarking the interactions between a humanoid robot and a virtual human for a real-world social task”. In: *Advances*

in Information Technology: 6th International Conference, IAIT 2013, Bangkok, Thailand, December 12-13, 2013. Proceedings 6. Springer. 2013, pp. 184–197.

[142] Amir Aly, Sascha Griffiths, and Francesca Stramandinoli. "Metrics and benchmarks in human-robot interaction: Recent advances in cognitive robotics". In: *Cognitive Systems Research* 43 (2017), pp. 313–323.

[143] Arunachalam Kumar. "The Pollex–Index complex and the Kinetics of Opposition". In: *Journal of Health and Allied Sciences NU* 2.04 (2012), pp. 80–87.

[144] Pierre Lemelin and Daniel Schmitt. "On primitiveness, prehensility, and opposability of the primate hand: the contributions of Frederic Wood Jones and John Russell Napier". In: *The evolution of the primate hand: Anatomical, developmental, functional, and paleontological evidence* (2016), pp. 5–13.

[145] Rickye S Heffner and R Bruce Masterton. "The role of the corticospinal tract in the evolution of human digital dexterity". In: *Brain, behavior and evolution* 23.3-4 (1983), pp. 165–183.

[146] Eram Neha et al. "Grasp analysis of a four-fingered robotic hand based on matlab simmechanics". In: *Journal of Computational & Applied Research in Mechanical Engineering (JCARME)* 9.2 (2020), pp. 169–182.

[147] Julio Fajardo et al. "Galileo hand: An anthropomorphic and affordable upper-limb prosthesis". In: *IEEE access* 8 (2020), pp. 81365–81377.

[148] Uikyum Kim et al. "Integrated linkage-driven dexterous anthropomorphic robotic hand". In: *Nature communications* 12.1 (2021), pp. 1–13.

Appendix A

Robotic-Hand Literature Review

This appendix expands upon Section 3.1.2, by providing an in-depth description of our systematic literature review methodology. Section A.1 describes the process of generating queries to identify relevant papers in the field. Following this, we compile a list of robotic hands mentioned throughout these papers, which we filter further based upon criteria outlined in Section A.2. The comprehensive results of our literature review are presented in Section A.3.

A.1 Queries

Our literature review aims to identify physical and/or simulated dexterous robotic hands. To achieve this, we generate a set of keywords which help us identify relevant literature. After a brief review of the field, we compile the following list of keywords which frequently appeared in the context of dexterous robotic hands.

- Dexterous, Manipulation
- Human, Humanoid, Human-like, Biomimetic
- Robot, Robotic
- Hand, Hands, Arm, Arms
- Simulated, Simulation
- Learning, Reinforcement Learning, Imitation Learning

We include keywords such as *reinforcement learning* and *imitation learning*, as it is not uncommon for researchers to develop new robotic hands for the sole purpose of RL/IL training. In order to target highly-dexterous robotic hands, we include keywords such as *dexterous* and *biomimetic*. Humanoid robotic hands may be referred to as *arms* when connected to an arm component, so we also include this word in our search.

From these keywords, we select the following **22 queries**, which we believe covers a substantial portion of the research space.

- robot hand
- robotic hand
- robot arm
- robotic arm
- human robot hand
- humanoid robotic hand
- biomimetic robotic hand
- biomimetic robotic arm
- dexterous humanoid robotic hand
- dexterous humanoid robotic arm
- dexterous manipulation humanoid robotic hand
- dexterous manipulation humanoid robotic arm
- humanoid robotic hand survey
- humanoid robotic hand review
- dexterous manipulation robotic hand survey
- dexterous manipulation robotic hand review
- robotic hand simulation
- dexterous robotic hand simulation
- dexterous robotic arm simulation
- simulated robotic hand
- dexterous robotic hand reinforcement learning
- dexterous robotic hand imitation learning

We conducted our review on 27th May 2024. We searched for each query on Google Scholar[1] and selected the top 20 results, yielding **280 unique papers**.

[1] `http://scholar.google.com`

A.2 Filtering

Given the initial 280 papers, we filter out papers whose name and abstract clearly indicate that the paper is not relevant. We exclude papers for the following reasons:

- The paper does not refer to robotic hands or arms (perhaps referring instead to human or animal hands).
- The paper focuses purely on soft robots. Our investigation is only concerned with robots built primarily from rigid components.
- The paper focuses purely on robotics software, such as ML and control, without any application to robotic hands.
- The paper refers to only specific components of humanoid hands, such as fingertips and skin.

After this process we are left with 220 papers, which we review in detail to compile a list of all **254** hands mentioned.

We apply a second filtering process, to identify dexterous robotic hands relevant to our investigation. We use the following filtering criteria:

- We remove soft robotic hands (56 hands removed).
- We remove robotic hands which did not attempt to mimic the biology of a human hand, such as pincer-gripper robots (53 hands removed).
- We remove robotic hands that lack sufficient dexterity, defined as having less than 15 DOF (9 local DOF). This criterion eliminates robotic hands that do not aim to mimic the dexterity of the human hand (45 hands removed).
- Finally, we exclude robotic hands from our review if their respective papers (or sources) do not provide sufficient information for us to determine the robot's number of DOF (54 hands removed).

From this, we identify **45** relevant robotic hands which met all our criteria.

A.3 Results

Table A.1 presents the details of all relevant dexterous robotic hands identified in our review. The hands are listed in descending DOF order, followed by descending DOA order. Where ambiguous or unspecified, we assume the robot's DOA to be equal to their DOF (as is the case for most rigid-body robots).

Robotic Hand	Year	Real-world Based	Simulation Based	Number of Digits	DOF	DOA
Shadow Dexterous Hand [55]	2013	✓	✓	5	23	27
ADROIT Hand [59]	2013	✓	✓	5	23	27
Shadow EDC Hand [60]	2002	✓	✓	5	23	23
NTU Hand [61]	1998	✓	✗	5	23	23
Unnamed [62]	2016	✓	✗	5	23	23
Unnamed [63]	2024	✓	✗	5	23	23
Unnamed [64]	2005	✓	✗	5	23	23
CEA Dexterous Robot Hand [65]	2013	✓	✗	5	23	23
Asada Hand [66]	2007	✓	✗	5	23	23
Unnamed [67]	2005	✓	✗	5	22	22
Allegro Hand [56]	2015	✓	✓	4	22	22
Gifu hand I [68]	1999	✓	✗	5	22	22
Gifu hand II [69]	2002	✓	✗	5	22	22
Gifu hand III [70]	2004	✓	✗	5	22	22
UTAH/MIT [53]	1985	✓	✓	4	22	22
Blackfingers [71]	2000	✓	✗	5	22	22
Dist Hand [72]	1998	✓	✗	4	22	22
BUAA [73]	2001	✓	✓	4	22	22
KITECH Hand [74]	2016	✓	✗	4	22	22
DART Hand [75]	2011	✓	✗	5	22	22
SKKU Hand II [76]	2006	✓	✓	4	22	22
PUCP hand [77]	2022	✓	✗	5	22	22
SmartHand [78]	2011	✓	✗	5	22	22
SCHUNK S5FH [79]	2017	✓	✗	5	21	26
TH-3R Hand [80]	2009	✓	✗	5	21	21
DLR/HIT Hand II [81]	2008	✓	✗	5	21	21
GCUA [82]	2011	✓	✗	5	21	21
HRI Hand [83]	2020	✓	✗	5	21	21
KYTECH HAND [84]	2012	✓	✗	4	20	20
TUAT/Karlsruhe [85]	2000	✓	✗	5	20	20
Dextrus Hand [86]	2016	✓	✗	5	20	20
DLR Hand II [87]	2001	✓	✗	4	19	23
Unnamed [88]	2017	✓	✗	5	19	19
RHC-1 [89]	2013	✓	✗	5	19	19

Ultralight Anthropomorphic Hand [90]	2001	✓	✗	5	19	19
DLR Hand I [54]	1993	✓	✗	4	18	22
Robotnaut Hand [91]	1999	✓	✓	5	18	18
Robotnaut 2 Hand [92]	2012	✓	✓	5	18	18
Karlsruhe Dextrous Hand II [93]	2013	✓	✗	5	18	18
Sandia Hand [94]	2014	✓	✓	4	18	18
DexHand [95]	2011	✓	✗	4	18	18
ZAR3 Hand [96]	2004	✓	✗	5	18	18
MPL [97]	2010	✓	✗	5	17	26
EthoHand [98]	2016	✓	✗	5	17	24
FRH4 [99]	2008	✓	✗	5	17	18
IH2 Azzurra [100]	2016	✓	✗	5	17	17
Unnamed [146]	2020	✓	✓	5	17	17
AR10 Robot Hand [103]	2013	✓	✗	5	16	16
LMS Hand [102]	2001	✓	✗	4	16	16
Deka Hand [104]	2018	✓	✗	5	16	16
Galileo Hand [147]	2020	✓	✗	5	16	16
ILDA hand [148]	2021	✓	✗	5	15	20

Table A.1: The results of our systematic literature review, identifying **45** relevant physical and/or simulated dexterous robotic hands across **220 papers**. The results are listed in descending DOF order, followed by descending DOA order.

Appendix B

Robotic-Hand URDF Descriptions

This appendix describes our robotic-hand simulation's URDF description (Section 4.1.1), as well as changes made to the alternate, less-dexterous models (Section 3.1.4). Section B.1 highlights the key components of the URDF format, and describes the `xacro` macro system built on top of URDF. Sections B.3–B.5 describe the different components in the hand's URDF model, and Section B.6 lists the hand's functional dimensions. Finally, we discuss the changes made to the other (less-dexterous) models in Section B.7

B.1 URDF & Xacro

URDF is an XML format primarily consisting of *links* and *joints*. A joint connects a *child link* to a *parent link*; in the human hand, parent links are positioned closer to the heart, and child links are positioned farther from the heart. This can be implemented in URDF as:

```
<robot xmlns:xacro="http://www.ros.org/wiki/xacro">
  <joint name="<joint_name>" type="<joint_type>" />
    <origin xyz="0 0 0" rpy="0 0 0" />
    <parent link="<parent_link_name>" />
    <child link="<child_link_name>" />
    <axis xyz="1 0 0" />
    <limit lower="0" upper="1" velocity="100" effort="100" />
    <xacro:damping_and_friction />
  </joint>
</robot>
```

In our robotic hand, the `<joint_type>` is either *prismatic* (for translations) or *revolute* (for rotations). The `origin` component tells the model where to place this joint with respect to the parent link's origin. This contains both the relative *positional coordinates* (`xyz`) and the relative *orientation* in the form of roll-pitch-yaw angles (`rpy`).

We then specify the names of the `parent` and `child` links which this joint connects. The `axis` component tells us which axis the joint should slide or rotate along (for prismatic

and revolute joints respectively). For example, a revolute joint with the axis `xyz = "1 0 0"` will rotate along the X axis.

The joint's limits refer to the minimum and maximum bounds on its movement/rotation (`lower` and `upper`), as well as the maximum velocity and effort that can be exerted onto the joint. We also add damping and friction to the joint, so that physics interactions are well-behaved. To save having to write the damping and friction coefficients each time, we can wrap them in a `xacro` macro, which expands to the full URDF code when compiled. Hence, we can define URDF code once, and reuse it multiple times (potentially with different arguments passed in at the macro's call-site).

B.2 Hand Description

```
<robot xmlns:xacro="http://www.ros.org/wiki/xacro">

  <xacro:include filename="wrist.xacro" />
  <xacro:include filename="thumb.xacro" />
  <xacro:include filename="finger.xacro" />

  <xacro:wrist />

  <xacro:empty_link name="root_link" />

  <xacro:thumb />
  <xacro:finger finger_id="II" />
  <xacro:finger finger_id="III" />
  <xacro:finger finger_id="IV" />
  <xacro:finger finger_id="V" />

</robot>
```

The wrist acts as the root of the hand, from which all other components stem. We explicitly define a `root_link` with no length to act as the hand's anchor point. The hand's wrist (Section B.3), currently wrapped as a macro, controls the global DOF of the `root_link`. The remaining hand components (the thumb and four fingers, also wrapped as macros) are attached as children to this `root_link`, so that they inherent the global position and orientation of the wrist.

B.3 Wrist Description

```
<robot xmlns:xacro="http://www.ros.org/wiki/xacro">

  <xacro:macro name="wrist">

    <joint name="wrist_x_translation_joint" type="prismatic">
      <origin xyz="0 0 0" rpy="0 0 0" />
      <parent link="base_link" />
      <child link="wrist_y_translation_link" />
      <axis xyz="1 0 0" />
      <limit lower="-1000" upper="1000" velocity="100" effort="100" />
      <xacro:damping_and_friction />
```

```
    </joint>

    <xacro:empty_link name="wrist_y_translation_link" />

    <joint name="wrist_y_translation_joint" type="prismatic">
      <origin xyz="0 0 0" rpy="0 0 0" />
      <parent link="wrist_y_translation_link" />
      <child link="wrist_z_translation_link" />
      <axis xyz="0 1 0" />
      <limit lower="-1000" upper="1000" velocity="100" effort="100" />
      <xacro:damping_and_friction />
    </joint>

    <xacro:empty_link name="wrist_z_translation_link" />

    <joint name="wrist_z_translation_joint" type="prismatic">
      <origin xyz="0 0 0" rpy="0 0 0" />
      <parent link="wrist_z_translation_link" />
      <child link="wrist_z_rotation_link" />
      <axis xyz="0 0 1" />
      <limit lower="-1000" upper="1000" velocity="100" effort="100" />
      <xacro:damping_and_friction />
    </joint>

    <xacro:empty_link name="wrist_z_rotation_link" />

    <joint name="wrist_z_rotation_joint" type="revolute">
      <origin xyz="0 0 0" rpy="0 ${pi/2} 0" />
      <parent link="wrist_z_rotation_link" />
      <child link="wrist_x_rotation_link" />
      <axis xyz="1 0 0" />
      <limit lower="${-pi/2}" upper="${pi/2}" velocity="100" effort="100" />
      <xacro:damping_and_friction />
    </joint>

    <xacro:empty_link name="wrist_x_rotation_link" />

    <joint name="wrist_x_rotation_joint" type="revolute">
      <origin xyz="0 0 0" rpy="0 ${-pi/2} 0" />
      <parent link="wrist_x_rotation_link" />
      <child link="wrist_y_rotation_link" />
      <axis xyz="1 0 0" />
      <limit lower="${-pi/2}" upper="${pi}" velocity="100" effort="100" />
      <xacro:damping_and_friction />
    </joint>

    <xacro:empty_link name="wrist_y_rotation_link" />

    <joint name="wrist_y_rotation_joint" type="revolute">
      <origin xyz="0 0 0" rpy="0 0 0" />
      <parent link="wrist_y_rotation_link" />
      <child link="root_link" />
      <axis xyz="0 1 0" />
      <limit lower="0" upper="${pi}" velocity="100" effort="100" />
      <xacro:damping_and_friction />
    </joint>

  </xacro:macro>

</robot>
```

Here we define the `wrist` macro as six joints connected by dimensionless links. These six joints control the wrist's six global DOF: three for position and three for orientation. The positional DOF are implemented as prismatic links, allowing the hand to move freely in 3D space (upper and lower bounded by $1000m$). The orientation DOF are implemented as three separate yaw, pitch, and roll rotations. These are bounded by $[-\pi/2, \pi/2]$ for the *up-down* and *left-right* hand motions, and $[0, \pi]$ for the *wrist-twist* motion.

B.4 Finger Description

```
<robot xmlns:xacro="http://www.ros.org/wiki/xacro">

    ...

    <joint name="${finger_id}_root_joint" type="fixed">
        <origin xyz="0 0 0" rpy="${-1 * root_angle} 0 0" />
        <parent link="root_link" />
        <child link="${finger_id}_root_to_mcp_link" />
    </joint>

    <xacro:segment name="${finger_id}_root_to_mcp_link" x_len="${depth}" y_len="${width}"
        z_len="${root_to_mcp_len}" />

    <joint name="${finger_id}_mcp_joint" type="fixed">
        <origin xyz="0 0 ${root_to_mcp_len}" rpy="${root_angle} 0 0" />
        <parent link="${finger_id}_root_to_mcp_link" />
        <child link="${finger_id}_mcp_flexion_link" />
    </joint>

    <xacro:empty_link name="${finger_id}_mcp_flexion_link" />

    <joint name="${finger_id}_mcp_flexion_joint" type="revolute">
        <origin xyz="0 0 0" rpy="0 0 0" />
        <parent link="${finger_id}_mcp_flexion_link" />
        <child link="${finger_id}_mcp_abduction_link" />
        <axis xyz="0 1 0" />
        <limit lower="${(-pi / 180) * mcp_extension}" upper="${(pi / 180) * mcp_flexion}" velocity="10000"
            effort="10000" />
        <xacro:damping_and_friction />
    </joint>

    <xacro:empty_link name="${finger_id}_mcp_abduction_link" />

    <joint name="${finger_id}_mcp_abduction_joint" type="revolute">
        <origin xyz="0 0 0" rpy="0 0 0" />
        <parent link="${finger_id}_mcp_abduction_link" />
        <child link="${finger_id}_mcp_to_pip_link" />
        <axis xyz="1 0 0" />
        <limit lower="0" upper="${(pi / 180) * mcp_abduction}" velocity="10000" effort="10000" />
        <xacro:damping_and_friction />
    </joint>

    <xacro:segment name="${finger_id}_mcp_to_pip_link" x_len="${depth}" y_len="${width}"
        z_len="${mcp_to_pip_len}" />

    <joint name="${finger_id}_pip_flexion_joint" type="revolute">
        <origin xyz="0 0 ${mcp_to_pip_len}" rpy="0 0 0" />
        <parent link="${finger_id}_mcp_to_pip_link" />
        <child link="${finger_id}_pip_to_dip_link" />
```

```
49        <axis xyz="0 1 0" />
50        <limit lower="0" upper="${(pi / 180) * pip_flexion}" velocity="10000" effort="10000" />
51        <xacro:damping_and_friction />
52    </joint>
53
54    <xacro:segment name="${finger_id}_pip_to_dip_link" x_len="${depth}" y_len="${width}"
55        z_len="${pip_to_dip_len}" />
56
57    <joint name="${finger_id}_dip_flexion_joint" type="revolute">
58        <origin xyz="0 0 ${pip_to_dip_len}" rpy="0 0 0" />
59        <parent link="${finger_id}_pip_to_dip_link" />
60        <child link="${finger_id}_dip_to_tip_link" />
61        <axis xyz="0 1 0" />
62        <limit lower="${(pi / 180) * dip_extension}" upper="${(pi / 180) * dip_flexion}" velocity="10000"
              effort="10000" />
63        <xacro:damping_and_friction />
64    </joint>
65
66    <xacro:segment name="${finger_id}_dip_to_tip_link" x_len="${depth}" y_len="${width}"
67        z_len="${dip_to_tip_len}" />
68
69    </xacro:macro>
70
71 </robot>
```

The finger comprises three joints: MCP, PIP, and DIP. The MCP joint (located at the knuckle) can perform *abduction-adduction* (finger splay) or *flexion-extension* (bending-forward) motions. Abduction-adduction is a revolute rotation along the X axis, whereas flexion-extension is a revolute rotation along the Y axis. The PIP and DIP joints can only perform *flexion-extension* movements, which are similarly implemented as a revolute rotations along the Y axis. Note that the lower limit represents the finger's hyperextension limit (bending behind the knuckle), whereas the upper limit represents the finger's flexion limit (how far forward it can bend).

The `X_to_Y` links represent the bones connecting together joints `X` and `Y`. These are implemented as `segment` macros, which refers to pre-defined 3D boxes parameterised on the bone's dimensions.

The `$X` operation retrieves the value of the variable `X`, defined at the parent macro's call-site. In this case, we can generate all four fingers at once by calling the `finger` macro four times, but passing in a different `finger_ids` each time. We can also retrieve hand dimensions in this way, by fetching values from our functional-dimensions YAML file (Section B.6).

B.5 Thumb Description

```
1 <robot xmlns:xacro="http://www.ros.org/wiki/xacro">
2
3   <xacro:macro name="thumb">
4
5     ...
6
7     <joint name="thumb_root_joint" type="fixed">
```

```
    <origin xyz="0 0 0" rpy="${-1 * root_angle} 0 0" />
    <parent link="root_link" />
    <child link="thumb_root_to_cmc_link" />
  </joint>

  <xacro:segment name="thumb_root_to_cmc_link" x_len="${depth}" y_len="${width}"
    z_len="${root_to_cmc_len}" />

  <joint name="thumb_cmc_joint" type="fixed">
    <origin xyz="0 0 ${root_to_cmc_len}" rpy="${root_angle} 0 0" />
    <parent link="thumb_root_to_cmc_link" />
    <child link="thumb_cmc_abduction_link" />
  </joint>

  <xacro:empty_link name="thumb_cmc_abduction_link" />

  <joint name="thumb_cmc_flexion_joint" type="revolute">
    <origin xyz="0 0 0" rpy="0 0 0" />
    <parent link="thumb_cmc_abduction_link" />
    <child link="thumb_cmc_flexion_link" />
    <axis xyz="1 0 0" />
    <limit lower="${(-pi / 180) * cmc_extension}" upper="${(pi / 180) * cmc_flexion}" velocity="10000"
        effort="10000" />
    <xacro:damping_and_friction />
  </joint>

  <xacro:empty_link name="thumb_cmc_flexion_link" />

  <joint name="thumb_cmc_abduction_joint" type="revolute">
    <origin xyz="0 0 0" rpy="0 0 0" />
    <parent link="thumb_cmc_flexion_link" />
    <child link="thumb_cmc_to_mcp_link" />
    <axis xyz="0 -1 0" />
    <limit lower="${(-pi / 180) * cmc_abduction}" upper="0" velocity="10000" effort="10000" />
    <xacro:damping_and_friction />
  </joint>

  <xacro:segment name="thumb_cmc_to_mcp_link" x_len="${depth}" y_len="${width}"
    z_len="${cmc_to_mcp_len}" />

  <joint name="thumb_mcp_abduction_joint" type="revolute">
    <origin xyz="0 0 ${cmc_to_mcp_len}" rpy="0 0 0" />
    <parent link="thumb_cmc_to_mcp_link" />
    <child link="thumb_mcp_flexion_link" />
    <axis xyz="1 0 0" />
    <limit lower="${(-pi / 180) * mcp_abduction}" upper="0" velocity="10000" effort="10000" />
    <xacro:damping_and_friction />
  </joint>

  <xacro:empty_link name="thumb_mcp_flexion_link" />

  <joint name="thumb_mcp_flexion_joint" type="revolute">
    <origin xyz="0 0 0" rpy="0 0 0" />
    <parent link="thumb_mcp_flexion_link" />
    <child link="thumb_mcp_to_ip_link" />
    <axis xyz="1 0 0" />
    <limit lower="0" upper="${(pi / 180) * mcp_flexion}" velocity="10000" effort="10000" />
    <xacro:damping_and_friction />
  </joint>

  <xacro:segment name="thumb_mcp_to_ip_link" x_len="${depth}" y_len="${width}"
    z_len="${mcp_to_ip_len}" />
```

```
69
70    <joint name="thumb_ip_flexion_joint" type="revolute">
71      <origin xyz="0 0 ${mcp_to_ip_len}" rpy="0 0 0" />
72      <parent link="thumb_mcp_to_ip_link" />
73      <child link="thumb_ip_to_tip_link" />
74      <axis xyz="1 0 0" />
75      <limit lower="${(-pi / 180) * ip_extension}" upper="${(pi / 180) * ip_flexion}" velocity="10000"
            effort="10000" />
76      <xacro:damping_and_friction />
77    </joint>
78
79    <xacro:segment name="thumb_ip_to_tip_link" x_len="${depth}" y_len="${width}"
80      z_len="${ip_to_tip_len}" />
81
82  </xacro:macro>
83
84 </robot>
```

We implement the thumb's CMC, MCP, and IP joints in a similar way to the finger's joints. However, due to the thumb's opposable movement, the *flexion-extension* movements in the CMC and MCP joints instead rotate along the X axis, while the *abduction-adduction* movements rotate along the Y axis. However, this does not apply to the IP joint, which is implemented the same as the finger's DIP and PIP joints.

B.6 Functional Dimensions

We use the functional dimensions measured by Cobos et al. [25] throughout our thesis. These are listed below in the YAML format, so that the dimensions can be imported into our hand's URDF description.

```yaml
thumb:
  ip_to_tip_len: 0.032
  mcp_to_ip_len: 0.033
  cmc_to_mcp_len: 0.037
  root_to_cmc_len: 0.028
  root_angle: 1.3
  cmc_flexion: 15
  cmc_extension: 90
  cmc_abduction: 60
  mcp_flexion: 80
  mcp_abduction: 60
  ip_flexion: 80
  ip_extension: 10
  width: 0.019
  depth: 0.015

II_finger:
  dip_to_tip_len: 0.025
  pip_to_dip_len: 0.025
  mcp_to_pip_len: 0.037
  root_to_mcp_len: 0.076
  root_angle: 0.420
  mcp_flexion: 90
  mcp_extension: 40
  mcp_abduction: 60
  pip_flexion: 110
  dip_flexion: 90
  dip_extension: 5
  width: 0.019
  depth: 0.015

III_finger:
  dip_to_tip_len: 0.027
  pip_to_dip_len: 0.029
  mcp_to_pip_len: 0.034
  root_to_mcp_len: 0.077
  root_angle: 0
  mcp_flexion: 90
  mcp_extension: 40
  mcp_abduction: 45
  pip_flexion: 110
  dip_flexion: 90
  dip_extension: 5
  width: 0.019
  depth: 0.015

IV_finger:
  dip_to_tip_len: 0.027
  pip_to_dip_len: 0.027
  mcp_to_pip_len: 0.032
  root_to_mcp_len: 0.073
  root_angle: -0.384
  mcp_flexion: 90
  mcp_extension: 40
  mcp_abduction: 45
  pip_flexion: 120
  dip_flexion: 90
  dip_extension: 5
  width: 0.019
  depth: 0.015

V_finger:
  dip_to_tip_len: 0.024
  pip_to_dip_len: 0.020
  mcp_to_pip_len: 0.028
  root_to_mcp_len: 0.071
  root_angle: -0.810
  mcp_flexion: 90
  mcp_extension: 40
  mcp_abduction: 50
  pip_flexion: 135
  dip_flexion: 90
  dip_extension: 5
  width: 0.019
  depth: 0.015
```

B.7 Alternate Model Descriptions

Hand Model: `finger-no-abd`

To remove the finger's abduction-adduction motions, we can simply put a fixed limit on the finger's second MCP joint. This would look like:

```
...

<joint name="${finger_id}_mcp_abduction_joint" type="revolute">
  <origin xyz="0 0 0" rpy="0 0 0" />
  <parent link="${finger_id}_mcp_abduction_link" />
  <child link="${finger_id}_mcp_to_pip_link" />
  <axis xyz="1 0 0" />
  <limit lower="0" upper="0" velocity="10000" effort="10000" />
  <xacro:damping_and_friction />
</joint>

...
```

Noting that the joint's `lower` and `upper` limits are now both fixed to 0.

Hand Model: `finger-interdep`

Since this model joins together different joint movements, it is implemented at the controller level rather than at the model description level. There is no easy way to describe linearly-correlated joint rotations in URDF.

Hand Model: `thumb-no-mcp`

This can be implemented in a similar fashion to the `finger-no-abd` case. Fixing both rotations at this joint gives us:

```

...

<joint name="thumb_mcp_abduction_joint" type="revolute">
  <origin xyz="0 0 ${cmc_to_mcp_len}" rpy="0 0 0" />
  <parent link="thumb_cmc_to_mcp_link" />
  <child link="thumb_mcp_flexion_link" />
  <axis xyz="1 0 0" />
  <limit lower="0" upper="0" velocity="10000" effort="10000" />
  <xacro:damping_and_friction />
</joint>

<xacro:empty_link name="thumb_mcp_flexion_link" />

<joint name="thumb_mcp_flexion_joint" type="revolute">
  <origin xyz="0 0 0" rpy="0 0 0" />
  <parent link="thumb_mcp_flexion_link" />
  <child link="thumb_mcp_to_ip_link" />
  <axis xyz="1 0 0" />
  <limit lower="0" upper="0" velocity="10000" effort="10000" />
  <xacro:damping_and_friction />
```

```
</joint>

...
```

Hand Model: `thumb-no-flex`

Here we fix only the *flexion-extension* URDF joints in the thumb's CMC and MCP joints:

```
...

<joint name="thumb_cmc_flexion_joint" type="revolute">
  <origin xyz="0 0 0" rpy="0 0 0" />
  <parent link="thumb_cmc_abduction_link" />
  <child link="thumb_cmc_flexion_link" />
  <axis xyz="1 0 0" />
  <limit lower="0" upper="0" velocity="10000" effort="10000" />
  <xacro:damping_and_friction />
</joint>

...

<xacro:empty_link name="thumb_mcp_flexion_link" />

<joint name="thumb_mcp_flexion_joint" type="revolute">
  <origin xyz="0 0 0" rpy="0 0 0" />
  <parent link="thumb_mcp_flexion_link" />
  <child link="thumb_mcp_to_ip_link" />
  <axis xyz="1 0 0" />
  <limit lower="0" upper="0" velocity="10000" effort="10000" />
  <xacro:damping_and_friction />
</joint>

...
```

Hand Model: `three-fingers`

We can remove the pinky finger by simply removing its URDF macro from the hand description as follows:

```
<robot xmlns:xacro="http://www.ros.org/wiki/xacro">

  <xacro:include filename="wrist.xacro" />
  <xacro:include filename="thumb.xacro" />
  <xacro:include filename="finger.xacro" />

  <xacro:wrist />

  <xacro:empty_link name="root_link" />

  <xacro:thumb />
  <xacro:finger finger_id="II" />
  <xacro:finger finger_id="III" />
  <xacro:finger finger_id="IV" />
</robot>
```

Appendix C

PID Controller Values

Recalling from Section 2.3.2, the values K_p, K_i, K_d parameterise a PID controller. Below, we list the PID parameters $d = K_d, i = K_i, p = K_p$ for each DOF in our robotic-hand model.

```
wrist_x_translation_joint:
  angle_wraparound: false
  d: 80.0
  ff_velocity_scale: 0.0
  i: 0.0
  i_clamp: 0.0
  normalize_error: false
  p: 500.0
wrist_y_translation_joint:
  angle_wraparound: false
  d: 80.0
  ff_velocity_scale: 0.0
  i: 0.0
  i_clamp: 0.0
  normalize_error: false
  p: 500.0
wrist_z_translation_joint:
  angle_wraparound: false
  d: 80.0
  ff_velocity_scale: 0.0
  i: 0.0
  i_clamp: 0.0
  normalize_error: false
  p: 500.0
wrist_x_rotation_joint:
  angle_wraparound: false
  d: 0.0
  ff_velocity_scale: 0.0
  i: 0.0
  i_clamp: 0.0
  normalize_error: false
  p: 100.0
wrist_y_rotation_joint:
  angle_wraparound: false
  d: 0.0
  ff_velocity_scale: 0.0
  i: 0.0
  i_clamp: 0.0
  normalize_error: false
  p: 100.0
wrist_z_rotation_joint:
  angle_wraparound: false
  d: 0.0
  ff_velocity_scale: 0.0
  i: 0.0
  i_clamp: 0.0
  normalize_error: false
  p: 100.0
thumb_cmc_flexion_joint:
  angle_wraparound: false
  d: 0.0
  ff_velocity_scale: 0.0
  i: 0.0
  i_clamp: 0.0
  normalize_error: false
  p: 150.0
thumb_cmc_abduction_joint:
  angle_wraparound: false
  d: 0.0
  ff_velocity_scale: 0.0
  i: 0.0
  i_clamp: 0.0
  normalize_error: false
  p: 150.0
thumb_mcp_flexion_joint:
  angle_wraparound: false
  d: 0.0
  ff_velocity_scale: 0.0
  i: 0.0
  i_clamp: 0.0
  normalize_error: false
  p: 100.0
thumb_mcp_abduction_joint:
  angle_wraparound: false
  d: 0.0
  ff_velocity_scale: 0.0
  i: 0.0
  i_clamp: 0.0
  normalize_error: false
  p: 100.0
thumb_ip_flexion_joint:
  angle_wraparound: false
  d: 0.0
  ff_velocity_scale: 0.0
  i: 0.0
  i_clamp: 0.0
  normalize_error: false
  p: 50.0
II_mcp_flexion_joint:
  angle_wraparound: false
  d: 0.0
  ff_velocity_scale: 0.0
  i: 2.0
  i_clamp: 2.0
  normalize_error: false
  p: 50.0
II_mcp_abduction_joint:
  angle_wraparound: false
  d: 2.0
  ff_velocity_scale: 0.0
  i: 0.0
  i_clamp: 0.0
  normalize_error: false
  p: 500.0
II_pip_flexion_joint:
  angle_wraparound: false
  d: 0.0
  ff_velocity_scale: 0.0
  i: 2.0
  i_clamp: 2.0
  normalize_error: false
  p: 50.0
II_dip_flexion_joint:
  angle_wraparound: false
  d: 0.0
  ff_velocity_scale: 0.0
  i: 2.0
  i_clamp: 2.0
  normalize_error: false
  p: 50.0
III_mcp_flexion_joint:
  angle_wraparound: false
  d: 0.0
  ff_velocity_scale: 0.0
  i: 2.0
  i_clamp: 2.0
  normalize_error: false
  p: 50.0
III_mcp_abduction_joint:
  angle_wraparound: false
  d: 2.0
  ff_velocity_scale: 0.0
  i: 0.0
  i_clamp: 0.0
  normalize_error: false
  p: 500.0
III_pip_flexion_joint:
  angle_wraparound: false
  d: 0.0
  ff_velocity_scale: 0.0
  i: 2.0
  i_clamp: 2.0
  normalize_error: false
  p: 50.0
III_dip_flexion_joint:
  angle_wraparound: false
  d: 0.0
  ff_velocity_scale: 0.0
  i: 2.0
  i_clamp: 2.0
  normalize_error: false
  p: 50.0
IV_mcp_flexion_joint:
  angle_wraparound: false
  d: 0.0
  ff_velocity_scale: 0.0
  i: 2.0
  i_clamp: 2.0
  normalize_error: false
  p: 50.0
IV_mcp_abduction_joint:
  angle_wraparound: false
  d: 2.0
  ff_velocity_scale: 0.0
  i: 0.0
  i_clamp: 0.0
  normalize_error: false
  p: 500.0
IV_pip_flexion_joint:
  angle_wraparound: false
  d: 0.0
  ff_velocity_scale: 0.0
  i: 2.0
  i_clamp: 2.0
  normalize_error: false
  p: 50.0
IV_dip_flexion_joint:
  angle_wraparound: false
  d: 0.0
  ff_velocity_scale: 0.0
  i: 2.0
  i_clamp: 2.0
  normalize_error: false
  p: 50.0
V_mcp_flexion_joint:
  angle_wraparound: false
  d: 0.0
  ff_velocity_scale: 0.0
  i: 2.0
  i_clamp: 2.0
  normalize_error: false
  p: 50.0
V_mcp_abduction_joint:
  angle_wraparound: false
  d: 2.0
  ff_velocity_scale: 0.0
  i: 0.0
  i_clamp: 0.0
  normalize_error: false
  p: 500.0
V_pip_flexion_joint:
  angle_wraparound: false
  d: 0.0
  ff_velocity_scale: 0.0
  i: 2.0
  i_clamp: 2.0
  normalize_error: false
  p: 50.0
V_dip_flexion_joint:
  angle_wraparound: false
  d: 0.0
  ff_velocity_scale: 0.0
  i: 2.0
  i_clamp: 2.0
  normalize_error: false
  p: 50.0
```

Appendix D

Ethics Committee Approval

Ethics Application - Approved

TITLE: Investigating How Dexterity Affects Imitation Learning in Humanoid Robotic-Hand Simulations

APPLICANTS: Dylan Moss, Neil Lawrence, Chapa Hewa Pelendage

EMAIL: dm894@cam.ac.uk, ndl21@cam.ac.uk, csh66@cam.ac.uk

DATES: 30/04/2024 ==> 21/05/2024

STUDY TYPE: controlled experiment

REFERENCE: #1840

DESCRIPTION

In this study, we analyse how the dexterity of a simulated robotic hand affects its ability to learn how to complete tasks by mimicking human behaviour. The study will involve participants controlling the hand simulation to complete grasping tasks, which we will use as training data to train a machine learning algorithm to complete these tasks autonomously. The user will control the simulation through a hand-tracking camera, which maps their hand motions onto the robotic-hand simulation. Note that the grasping tasks are only performed in the simulated environment (with participants mimicking grasping actions in the real-world with no physical objects). There is no physical interactions in this experiment: the participant is not directly touching anything and there are no intrusions / harmful materials in direct contact with the participant. The only exception is that a hand-tracking camera may be worn on the participant's head using a headband.

We expect to perform our experiments on 5-10 participants. The participant will be asked to control the hand simulation to pick up a variety of simulated shapes (such as spheres and cubes). For each object, the process ends when the participant has picked up the simulated object, with a 1 minute cutoff threshold. We vary the hand's dexterity throughout the

experiment, repeating each grasp for a specific object and hand dexterity 3-5 times. We will collect the following data: which objects the participant was able to grasp, how long it takes them to grasp the objects, and the trajectory the hand followed to grasp the objects (to use as training data). The data will be fully anonymised, and a subset of this data will possibly submitted as part of the report. Before the experiment, all participants will be provided with the written guidelines for the experiments (with an optional verbal explanation). At this point, participants will sign written consent before performing any experiments — each individual participant should follow the same procedure and sign the consent form personally. A demonstration of the task will then be provided, following a script and predefined demonstration environment to ensure demonstrations are near-identical for each participant. Our participants will primarily consist of undergraduate or PhD students, who we will recruit by sending messages on relevant group chats and mailing lists asking for participants. If we are lacking participants, we may add a small token gift (for example chocolates) to encourage participation.

PRECAUTIONS

We will conduct at least one pilot study, whose results will be discarded from our final analysis. We do not expect there to be significant bias caused by our participant recruitment process: our experiments are more concerned with how the hand simulation is affected by human control, rather than how well humans perform at completing the assigned tasks. Participants will be selected from those who have no prior experience with robotic applications, to avoid data bias. We will treat participants with respect, and ensure that they are aware we are not testing or judging their ability to control the simulated robotic hand. If we use a token gift to encourage participation, this gift will not be dependent on task performance, and will still be awarded if the participant chooses to withdraw from the experiment.

We will prepare a consent form stating the nature of the experiment and rights of the participant. Before starting the experiment, we will ensure that the participants have read and signed the consent form. Precautions will be taken to ensure the demonstrations are near-identical each time, such as following a predefined script and demonstration environment. We will conduct a brief interview after the experiment, and ask questions such as: how difficult was it to pick up the objects of different shapes, how easy was it to control the simulated hand, and their opinion regarding our project's hypothesis. This will allow us to engage with participants to garner further insights into the experiment's results. No personal or private data is collected during this experiment, and all data collected will be fully anonymised. We plan to retain and possible publish the data collected in these experiment.

Appendix E

Participant Experiment Results

This appendix presents the complete results from our teleoperation experiments (Section 4.2). Seven users participated in these experiments. However, due to time constraints, we could not conduct all 360 experiments on each person. Therefore, we performed as many experiments as we could in the participant's available time.

Participant 1

Grasping Task: `med-cyl`

Robotic-Hand Model	Attempt 1	2	3	4	5	6	7	8	9	10	Total
`full`	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	9
`finger-no-abd`	✗	✗	✓	✗	✗	✗	✓	✗	✓	✓	4
`finger-interdep`	✓	✓	✗	✓	✗	✗	✗	✓	✓	✓	6
`thumb-no-mcp`	✗	✓	✗	✗	✓	✓	✓	✗	✓	✗	5
`thumb-no-flex`	✗	✗	✓	✓	✗	✓	✓	✓	✗	✓	6
`three-fingers`	✓	✗	✓	✓	✓	✓	✓	✓	✓	✗	8

Grasping Task: `med-sphere`

Robotic-Hand Model	Attempt 1	2	3	4	5	6	7	8	9	10	Total
`full`	✓	✓	✗	✓	✓	✓	✓	✗	✗	✓	7
`finger-no-abd`	✓	✗	✓	✓	✓	✓	✓	✗	✓	✓	8
`finger-interdep`	✓	✓	✓	✓	✗	✓	✗	✓	✓	✗	7
`thumb-no-mcp`	✓	✓	✗	✗	✓	✗	✗	✓	✓	✗	5
`thumb-no-flex`	✓	✗	✗	✗	✓	✓	✗	✓	✗	✗	4
`three-fingers`	✗	✗	✗	✓	✗	✓	✗	✗	✓	✓	4

Participant 2

Grasping Task: `thin-sheet`

Robotic-Hand Model	Attempt 1	2	3	4	5	6	7	8	9	10	Total
`full`	✗	✗	✓	✗	✗	✓	✓	✓	✓	✓	6
`finger-no-abd`	✓	✓	✓	✗	✓	✓	✗	✓	✓	✓	8
`finger-interdep`	✓	✓	✗	✓	✓	✓	✗	✓	✗	✓	7
`thumb-no-mcp`	✗	✓	✗	✗	✗	✗	✓	✗	✓	✓	4
`thumb-no-flex`	✓	✓	✗	✗	✓	✓	✓	✓	✗	✗	6
`three-fingers`	✓	✗	✗	✓	✓	✓	✓	✗	✗	✓	6

Participant 3

Grasping Task: `small-sphere`

Robotic-Hand Model	Attempt 1	2	3	4	5	6	7	8	9	10	Total
`full`	✓	✗	✓	✓	✓	✓	✓	✗	✓	✓	8
`finger-no-abd`	✗	✗	✗	✓	✗	✓	✓	✗	✗	✓	4
`finger-interdep`	✗	✓	✓	✓	✗	✓	✓	✓	✗	✗	6
`thumb-no-mcp`	✓	✗	✓	✓	✗	✓	✗	✗	✓	✓	6
`thumb-no-flex`	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	2
`three-fingers`	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	1

Grasping Task: `thin-cyl`

Robotic-Hand Model	Attempt 1	2	3	4	5	6	7	8	9	10	Total
`full`	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	9
`finger-no-abd`	✗	✓	✓	✗	✗	✗	✗	✗	✗	✓	3
`finger-interdep`	✗	✓	✗	✓	✓	✓	✓	✗	✗	✓	6
`thumb-no-mcp`	✓	✓	✓	✓	✗	✓	✗	✓	✓	✓	8
`thumb-no-flex`	✓	✓	✗	✗	✓	✓	✓	✓	✓	✗	7
`three-fingers`	✗	✓	✓	✓	✓	✓	✓	✓	✓	✗	8

Participant 4

Grasping Task: `thick-cyl`

Robotic-Hand Model	Attempt 1	2	3	4	5	6	7	8	9	10	Total
`full`	✓	✓	✓	✗	✓	✓	✓	✓	✗	✓	8
`finger-no-abd`	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	9
`finger-interdep`	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	10
`thumb-no-mcp`	✗	✓	✗	✗	✓	✗	✓	✗	✗	✓	4
`thumb-no-flex`	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	4
`three-fingers`	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	10

Grasping Task: `thin-cyl`

Robotic-Hand Model	Attempt 1	2	3	4	5	6	7	8	9	10	Total
`full`	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	10
`finger-no-abd`	✓	✗	✓	✓	✗	✓	✓	✗	✓	✓	7
`finger-interdep`	✗	✗	✓	✗	✗	✓	✓	✓	✓	✓	6
`thumb-no-mcp`	✓	✓	✗	✓	✓	✗	✓	✓	✓	✗	7
`thumb-no-flex`	✗	✗	✓	✓	✗	✓	✗	✓	✗	✓	5
`three-fingers`	✓	✓	✗	✗	✗	✓	✓	✗	✓	✗	5

Grasping Task: `med-cyl`

Robotic-Hand Model	Attempt 1	2	3	4	5	6	7	8	9	10	Total
`full`	✓	✓	✗	✓	✗	✗	✗	✓	✗	✓	5
`finger-no-abd`	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	2
`finger-interdep`	✗	✗	✓	✗	✓	✗	✗	✗	✓	✗	3
`thumb-no-mcp`	✗	✗	✗	✗	✗	✓	✓	✗	✗	✓	3
`thumb-no-flex`	✗	✗	✗	✓	✓	✓	✗	✓	✗	✓	5
`three-fingers`	✗	✗	✗	✓	✗	✗	✓	✓	✗	✓	4

Participant 5

Grasping Task: `thick-cyl`

Robotic-Hand Model	Attempt 1	2	3	4	5	6	7	8	9	10	Total
`full`	✓	✓	✗	✓	✗	✓	✓	✓	✓	✓	8
`finger-no-abd`	✓	✗	✗	✓	✗	✓	✓	✗	✗	✓	5
`finger-interdep`	✗	✓	✗	✓	✗	✗	✓	✓	✗	✓	5
`thumb-no-mcp`	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	2
`thumb-no-flex`	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗	2
`three-fingers`	✗	✓	✓	✗	✗	✓	✗	✗	✓	✗	4

Grasping Task: `small-sphere`

Robotic-Hand Model	Attempt 1	2	3	4	5	6	7	8	9	10	Total
`full`	✗	✓	✗	✓	✗	✓	✓	✓	✓	✓	7
`finger-no-abd`	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	9
`finger-interdep`	✗	✓	✓	✗	✗	✗	✓	✗	✓	✓	5
`thumb-no-mcp`	✗	✗	✗	✗	✗	✓	✗	✓	✓	✓	4
`thumb-no-flex`	✗	✗	✓	✗	✓	✓	✓	✗	✓	✗	5
`three-fingers`	✗	✓	✓	✗	✗	✓	✗	✓	✓	✗	5

Participant 6

Grasping Task: `medium-sphere`

Robotic-Hand Model	Attempt 1	2	3	4	5	6	7	8	9	10	Total
`full`	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓	8
`finger-no-abd`	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	9
`finger-interdep`	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	8
`thumb-no-mcp`	✓	✗	✓	✓	✓	✗	✓	✓	✓	✗	7
`thumb-no-flex`	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	9
`three-fingers`	✓	✗	✓	✓	✓	✗	✓	✗	✗	✗	5

Grasping Task: `thin-sheet`

Robotic-Hand Model	Attempt										Total
	1	2	3	4	5	6	7	8	9	10	
`full`	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	10
`finger-no-abd`	✓	✓	✓	✓	✓	✗	✓	✗	✓	✓	8
`finger-interdep`	✗	✓	✗	✗	✗	✓	✓	✓	✓	✓	6
`thumb-no-mcp`	✓	✓	✓	✓	✓	✓	✗	✓	✓	✗	8
`thumb-no-flex`	✓	✓	✗	✓	✓	✓	✓	✓	✗	✓	8
`three-fingers`	✓	✗	✓	✓	✓	✓	✗	✓	✓	✓	8

Participant Interviews

Three out of the seven participants agreed to a post-experiment interview. However, none consented for the full (anonymised) transcript of their interview to be recorded in this thesis. Paraphrased excerpts from their interviews can be found in Section 5.1.