The M&M Hand: a modular and minimal hand for four-fingered dexterous manipulations

Andrina Frischknecht-Grimm[1] Shubham Gupta[1] Zeno Hamers[1] Deepana Ishtaweera[1] Robert Jomar Malate[1]

***Abstract*—A wide variety of robotic hands have been developed for object manipulation. Recently there has been a trend towards biomimetic design. 4-fingered designs offer the potential to be simpler and cheaper than their 5-fingered counterparts. However, the performance of 4-fingered hands relative to their 5-fingered counterparts has not been fully explored. This work analyses the performance of a 4-fingered, tendon-driven, biomimetic hand design by deploying computer vison-based teleoperation and zero-shot transfer reinforcement learning policies to perform a range of object grasping and manipulation tasks. Our work shows that despite being a overall simper design, the 4-fingered design can perform object manipulation tasks similar to a that of a 5-fingered hand.**

***Index Terms*—Robotic hand, Teleoperation, Reinforcement learning, Biomimetic design**

I. Introduction

A. Motivation

Grasping and manipulation, especially in the human environment, have been at the forefront of robotics research and challenges. To address this, there has been a shift in the field towards developing robotic manipulators that are increasingly bio-mimetic. As argued by the developers of the Faive Hand system, in order to be able to perform robotic manipulation tasks in human environments, mimicking the human form is optimal due to humans' influence on constructing those very environments [1]. Additionally, this provides the benefit of being able to learn from the plethora of human-object manipulation examples in combination with learning-based methods to perform the task [1]. This is further supported by the fact that their ability to grasp a wider range of objects, notably objects with soft and deformable properties, is superior to traditional robotic grippers, such as in the case of fruit picking, where adaptive force control is needed [2].

Because of this, many robotic hands modelled after the human hand have been developed, most having either five or four fingers ([3] [4] [5] [6]).

However, little work has been done that explores and compares the performance and limitations of a 4-fingered design, especially with biomimetic designs. A 4-fingered manipulator offers a potentially simpler mechanical design compared to a 5-fingered manipulator due to requiring fewer parts. This potentially reduces the cost and complexity of the hardware design and manufacturing, which is a common bottleneck in robotics applications. Although most people have five digits on each hand, some people lead their normal and independent lives using only four fingers. On a high level, it shows that it is possible to operate in a human environment with one less digit, and therefore, a 4-fingered robotic hand may be able to as well. Given all of this, we explored and investigated the performance of a biomimetic, 4-fingered robot hand in this work. We have done this by designing a low-cost, tendon-driven based hardware platform. To control and showcase the performance of the hand, we developed and deployed a computer vision (CV) - based teleoperation. Additionally, we deploy two reinforcement learning (RL) policies to the hand that performs in-hand manipulation of a sphere rotation and cube rotation.

[1]Dept. of Mechanical and Process Eng., ETH Zürich, Switzerland. (grimman, shugupta, zhamers, dishtaweera, rmalate)@ethz.ch

B. Related Work

As was mentioned in the previous section, a variety of hand-like manipulators have been developed. One of the most popular platforms is the Shadow Hand [5]. To achieve a more anthropomorphic design, the team at the Soft Robotics Lab (SRL) at ETH Zürich developed the Faive Hand [1].

There have been other 4-fingered robotic hands developed and used for dexterous manipulation research. In 2012, Sandia National Laboratories developed a robotic hand designed to perform a variety of tasks in hazardous environments [4]. However, this hand does not seem to be available to non-governmental organizations. A widely used 4-fingered design among the research community is the Allegro hand [6]. In response to this, the LEAP Hand was developed as an open-source, low-cost, and accessible version of the Allegro hand [3].

C. Contributions

The following are the contributions of this work:

- A modular, low-cost, 3D printed design of a soft, tendon-driven 4-fingered hand.
- Development of a graphical user interface (GUI) and computer vision-based (CV) teleoperation.
- Training and deployment of reinforcement learning (RL) policies for in-hand sphere rotation and in-hand cube rotation from the virtual world to the physical robot hand.
- A flexible prototyping platform enabling further exploration of tendon-driven, 4-fingered manipulator designs.

II. Hardware Design

A. System Overview

The hand is composed of four subsystems: fingers, thumb, palm and motorrack. The skeletal parts of the hand were

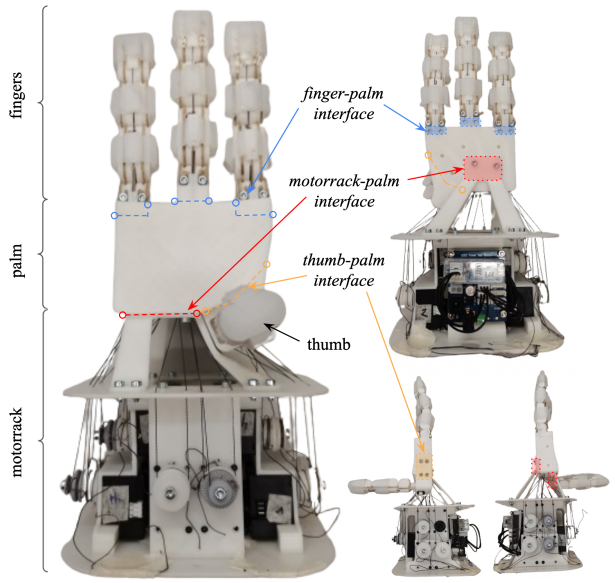

Fig. 1: The hand has four subsystems: fingers, palm, thumb and motorrack. The interfaces (marked in color) enable the components to be replaced and exchanged quickly and easily.

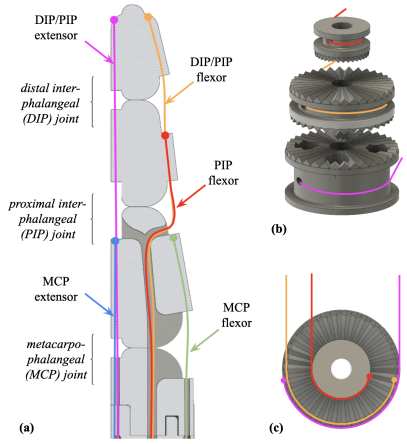

Fig. 2: Detailed view of the finger and spools: (a) Each finger has two independent DoFs: flexion/extension of the MCP joint and a coupled flex/ext of the DIP & PIP joint. (b) The spools have a ribbed surface to help against slippage. (c) Top view of the spools and tendon attachments of the DIP & PIP flex/ext.

printed with a PRUSA Printer with Polylactic Acid (PLA) filament. Threaded heat-set inserts and M3 screws were used to ensure a secure fastening in all of the interfaces. For quick prototyping purposes, the interfaces were placed at the subsystem borders (see the coloured markings in Fig. 1).

B. Finger Design

Each finger has 2 independent Degrees of Freedom (DoF). The metacarpophalangeal (MCP) joint at the base of the finger can be flexed and extended. To minimize the amount of actuators, the movement of the distal interphalangeal (DIP) joint and the proximal interphalangeal (PIP) joint were coupled. The finger segments can move with respect to each other with Rolling Contact Joints (RCJ). The length of the finger segments was based on Boislcair et al.'s work on the optimal design of underactuated RCJ fingers [7]. Fig. 2 (a) shows the tendon routings of the finger. The DIP/PIP flexor and PIP flexor are routed to the same actuator (see spool design in Fig. 2 (b) and II-D). To avoid a change of the length of the PIP flexor, DIP/PIP flexor and DIP/PIP extensor during the movement of the MCP joint, the respective tendons were routed through a channel in the middle of the finger.

C. Thumb Design

The design of the thumb was based on the design of the fingers explained above. It contains the MCP joint and PIP joint but not the finger segment after the DIP joint. However, the thumb has an additional DoF that for adduction and abduction. This was implemented using a pin joint. In total, the thumb has a total of 3 DoF, with each joint being routed to its own actuator. The placement of the thumb was done in a way that be nearly opposite of the other fingers. Furthermore, it is also able to touch the tips of the other three fingers, which is relevant for any sort of pinch grasps.

D. Motorrack & Spool Design

The motorrack holds 9 Dynamixels XC330-T288-T servo motors and a driver-board needed to actuate the hand. It has a spare interface for a 10th motor i.e. for a 4th DoF of the thumb. For the experiments, an additional interface for the Franka Emrika Panda robot arm was mounted to the bottom of the motorrack. The tendons are connected over spools to the actuators. Every actuator has two spools, one for the flexion tendon and one for the extension tendon. The actuators for the DIP/PIP joint have an additional 3rd spool for the PIP flexor. To avoid slippage between the spools, a rigged texture was added on the bottom and top of the spools as shown in Fig. 2 (b) and (c). They are tightened to each other with the help of a screw.

E. Joint/Tendon mapping

The hardware design was accompanied by a low-level software controller design, which maps the desired angles for the finger joints to the required rotations in the individual servo motors. The kinematics of the rolling contact joint [8] were used to investigate the relative change in the extensor and flexor tendon upon joint movement. The two lengths seemed to change with almost a constant ratio with 5% variation. This calculation was then used to iteratively select the design and parameters for the spools used in conjunction with the motor to control the joints.

III. Manipulator Control

Two control strategies were developed for achieving the target tasks on the hardware design, which will be discussed in Sec. IV-A. Firstly, a GUI controller was developed to help in quick testing of the hardware design and make improvements. To improve dexterity, a computer vision-based teleoperation

control was then implemented. A reinforcement learning-based controller is also implemented on the hardware to further improve the manipulation abilities.

A. GUI controller

The GUI controller was developed to speed up hardware prototype testing and for tuning the low-level controller. The GUI could be used to pass individual joint commands to the robot hand, which are converted to motor positions as described in Sec. II-E. Some basic grips and hand poses were also hard-coded for testing purposes.

B. Teleoperation

The teleoperation controller extracts the landmarks of the hand of the operator using the MediaPipe Hands algorithm [9] as in Fig 3. These landmarks were subsequently used to define hand key-vectors as shown in Fig 4.

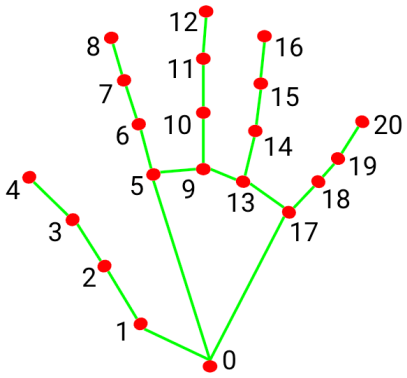

Fig. 3: MediaPipe Hand landmarks [9]

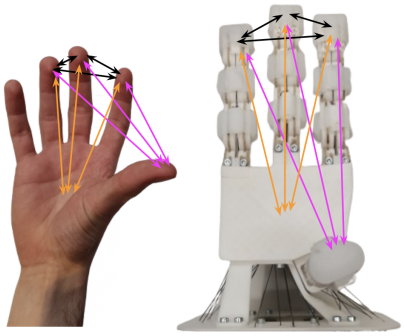

Fig. 4: Hand pose key-vectors as defined on the operator and robot hand

The operator's hand was then retargeted to the robot hand by minimisation of an energy function $E(\cdot)$ defined in Eq. 1.

$$E((\beta_h, \theta_h), q_a) = \sum_{i=0}^{K} \|(v_{i,h} - c_i \cdot v_{i,a})\|_2^2 \qquad (1)$$

(β_h, θ_h) define the MANO pose for the operator hand extracted using the MediaPipe algorithm [9]. q_a denotes the robot hand pose, $v_{i,h} = v_{i,h}(\beta_h, \theta_h)$ denotes the i^{th} key-vector for the operator hand pose, $v_{i,a} = v_{i,h}(q_a)$ denotes the i^{th} key-vector for the robot hand pose and c_i denotes the scaling factor for the robot hand key-vector. The scaling factor c_i normalises differences in size and dimensions between the operator and robot hand. Tuning these scaling factors for a specific operator improved performance during experimentation. The robot hand key-vectors are defined using Pytorch Kinematics library [10], and the energy function Eq. 1 is optimised using Adam optimiser in PyTorch. This outputs the retargeted joint angles for the robot hand, which are converted to motor positions as described in Sec. II-E.

C. Reinforcement Learning control

Proximal Policy Optimization [11] was used as the learning method for the reinforcement learning (RL) controller. It is an *off-policy* algorithm that uses two Neural Networks as Actor and Critic networks and optimises a surrogate problem for neural network training.

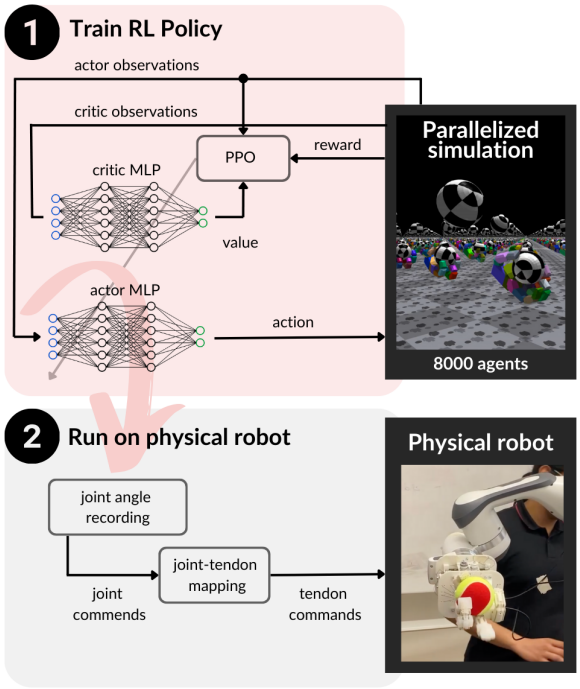

Fig. 5: Overview of the RL training framework for achieving dexterous manipulation on the M&M hand. After training the policy within a simulation environment, the policy is run within the simulation environment. The recorded joint measurements are played at 20Hz in the real robot.

The controller was trained in a parallelised manner using IsaacGym Envs [12] and faive_gym [1], with multiple agents training in-parallel the actor and critic networks on an Amazon Web server. Domain randomisation was performed while training the agents to reduce the Sim-to-Real gap. The reward function for the algorithm was designed for each task specifically, as described in Sec. IV-B2. The algorithm policy learns the hand joint angles to achieve the task. The trained

networks were used to generate policy recordings. Afterwards, they were then deployed to the simulation environment, which were then played back for a zero-shot transfer to the physical robot hand. The joint/tendon mapping was used to transfer the learned joint angles to be applied to the physical robot.

IV. Results and Discussion

A. Experiment setup

The ability of the robotic hand design was tested using a series of target tasks using one of the control strategies discussed in Sec. III. These tasks were chosen to test the hardware as well as the controller design in the areas of adaptability, dexterity, strength, and manoeuvrability and to gauge the overall performance. The robot hand was mounted as an end-effector on a Franka Emrika Panda robot arm. The robot arm was controlled separately using a joystick to send velocity commands for the arm end-effector movement in the task space.

TABLE I: List of tasks performed by the robotic hand and the control strategy applied

No.	Tasks	Controller
1	Pick and place multiple objects	Teleoperation
2	Lifting a piece of paper using slide and pinch grip	Teleoperation
3	Pickup and operate a drill	Teleoperation
4	Hold and rotate a fidget spinner in hand	Teleoperation
5	Picking up and pouring from a cup	Teleoperation
6	Rotate a tennis ball in hand in 2 different directions	Reinforcement Learning
7	Rotate a Rubix cube 90 degrees in hand in 2 different directions	Reinforcement Learning

B. Results

1) Teleoperation: The camera teleoperation was implemented using the Luxonis DepthAI OAK-D Lite camera hardware. The camera hardware was used to track and extract the landmarks, which were then further processed as described in Sec. III-B. Figs. 6-10 display the various tasks performed using this controller and the results.

This performance and success on the variety of tasks performed by the hand demonstrate the capability of the manipulator, resonating with our original motivation to prove the four-fingered manipulator's capabilities are as good as a five-fingered one. The tasks encompass diversity in both the types of grasps that are needed for achieving it as well as the overall manipulation control of the hand, proving variety in the objects being manipulated and giving a well-rounded assessment of the hand.

2) RL controller: To accomplish the in-hand ball and cube rotation task, reward functions had to be designed to incentivise the correct behaviour of the RL agent. The rewards in Eq. 2 were designed for the hand to move smoothly and to

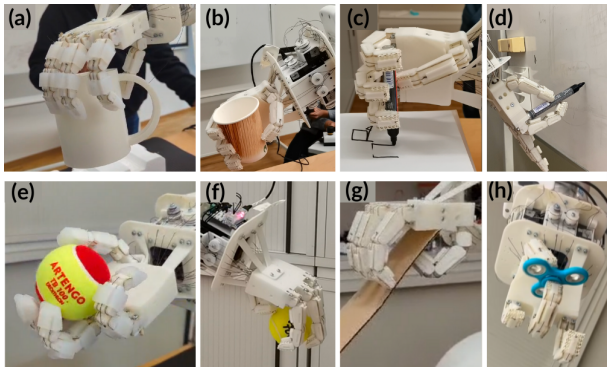

Fig. 6: Different types of grasping capabilities of the robot hand: (a) grasping a cup from the top, (b) grasping a cup from the side using a power grasp, (c) holding a pen using all fingers, (d) grasping the pen using a pinch grasp, (e) grasping a tennis ball, (f) grasping a ball from the top, (g) holding a thin cardboard sheet and (h) holding fidget spinner using pinch grasp.

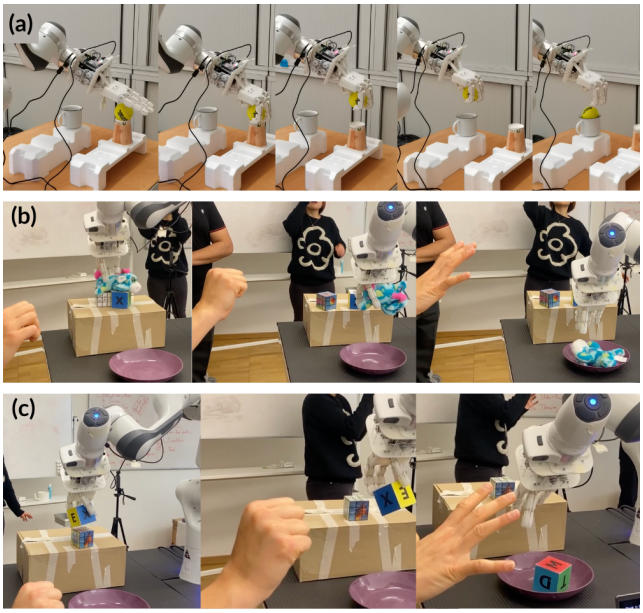

Fig. 7: Different teleoperation tasks of picking and placing objects. These objects have widely varying sizes, shapes and deformability to test the hand: (a) pick and place a tennis ball (sphere object), (b) pick and place a plush toy (deformable object), (c) pick and place a rigid cube.

prevent dropping the ball or cube. The reward function can be decomposed into three terms, as shown below:

$$\begin{aligned} R_{\text{acceleration}} &= w_a \left\| a_{joints} \right\|_2 \\ R_{\text{torque}} &= w_t \left\| T \right\|_2 \\ R_{\text{drop}} &= w_d \cdot \delta \\ R_{total} &= R_{\text{acceleration}} + R_{\text{torque}} + R_{\text{drop}} \end{aligned} \tag{2}$$

$R_{\text{acceleration}}$ penalizes joint accelerations, R_{torque} penalizes joint torques and R_{drop} is the drop penalty, summed up for

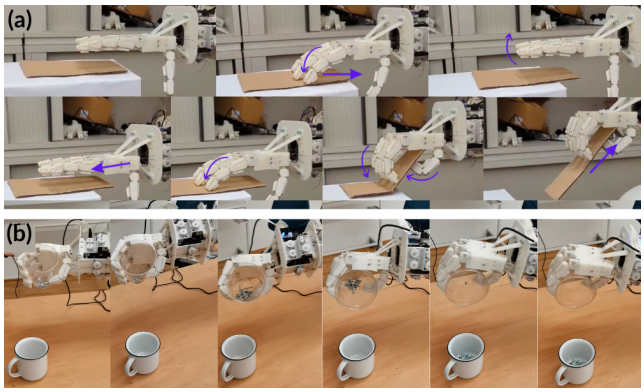

Fig. 8: Manipulations tasks using teleoperation: (a) lifting a piece of paper using slide and pinch grip, (b) picking up and pouring from a cup.

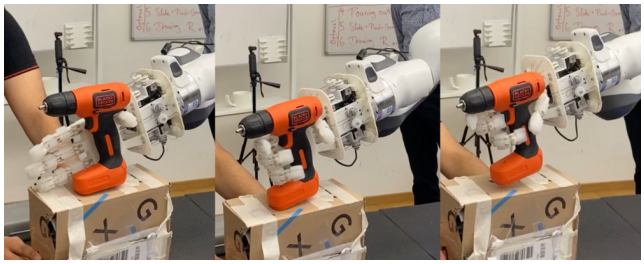

Fig. 9: Performance demonstration of a drill (approximately 700g weight) pick up and displacement using the robot hand

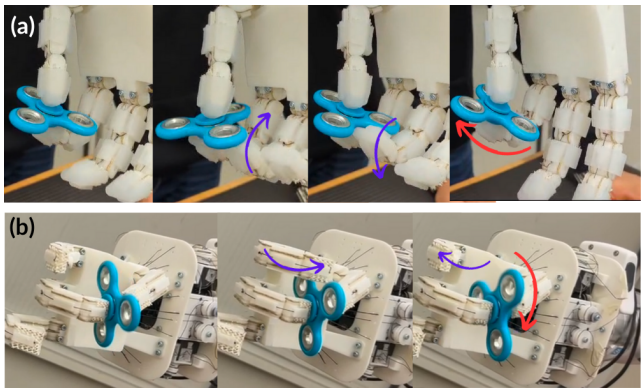

Fig. 10: Hold and rotate a fidget spinner. (a) using the thumb and index finger to hold the fidget spinner. The middle finger is used to push the fidget spinner, and (b) the thumb and middle finger are used to hold the fidget spinner. The index finger is used to push the fidget spinner.

R_{total} denoting the complete reward function. a_{joints} is the joint acceleration vector, T is the force tensor, δ is a boolean for the ball dropping out of the palm of the robot hand, and (w_a, w_t, w_d) are the weights for each of these terms.

After an extensive parameter search, the parameters of the objective function (Eq. 2) (p_a, p_t, p_d) listed in table III were found to perform the best in simulation.

TABLE II: Tuned Parameter Values, optimized for smooth movement of the hand and elimination of object dropping.

Parameter	Tuned Values
p_a	-0.0005
p_t	-0.01
p_d	-5.0

To incentivise the RL agent to perform the ball rotation task, the reward functions listed in Eq. 3 were added to the previous objective function (Eq. 2). The reward given for the ball rotation task is proportional to the rotational velocity of the ball around the x-axis, shown below:

$$R_{\text{xrotvel}} = p_v \min(d \cdot v_x + 1, 2, v_x + 4) \tag{3}$$

Here, R_{xrotvel} is the reward proportional to the rotational velocity of the ball around the x-axis v_x (this is bounded, max result when the rotational velocity greater or equal than 1 radiant per second), d is the parameter describing the desired rotational direction of the ball (+1 for the negative rotation direct and -1 for the positive rotation direction), p_v is the reward parameter (is tuned).

For the cube rotation task, the reward function in Eq. 3 was replaced by three new rewards described in Eq. 4. To incentivise the PPO algorithm to learn how to rotate the cube 90 degrees (counter)clockwise, the concept of a goal object (with position and orientation) is introduced. A success reward R_{success} is released to the agent once the object pose and orientation are identical to the set goal (with a 10% tolerance). Two other rewards are added to the total objective function R_{total}, the object distance reward $R_{\text{obj\_dist}}$, which rewards the agent based on the distance between the object pose (P_{obj}) and the goal pose (P_{goal}), and $R_{\text{obj\_rot}}$, which rewards the agent based on the orientation alignment of the object with the goal orientation (represented with the quaternions q_{obj} and q_{goal}, $q^{\star}_{goal}$ depicts the conjugate). The formulas for R_{success}, $R_{\text{obj\_dist}}$ and $R_{\text{obj\_rot}}$ can be found in Eq. 4. The parameters (w_d, w_r, w_s) are the weights for the respective reward functions. σ is a boolean for the object reaching the desired goal state.

$$\begin{aligned} R_{\text{obj\_dist}} &= w_d \left\| P_{obj} - P_{goal} \right\|_2 \\ R_{\text{obj\_rot}} &= w_r \frac{1}{2|\arcsin\left\| q_{obj} \cdot q^{\star}_{goal} \right\|_2| + 0.1} \\ R_{\text{success}} &= w_s \cdot \sigma \end{aligned} \tag{4}$$

Here, the variables are as defined above.

TABLE III: Tuned Parameter Values, optimized for the ball rotation task (top) and the cube rotation task (below).

Ball Rotation	**Value**
p_v	0.1
Cube Rotation	**Value**
p_d	-0.05
p_r	-0.03

For the ball rotation task (in both directions), as well as for the positive rotation direction for the cube rotation task, relative control was used. Absolute control was used for the cube rotation task around the negative rotation direction since this showed better results in the simulation. Fig. 11 showcases the performance of the sphere rotation task deployed both in simulation and on the physical robot. A rotational velocity of -1.573 rad/s and 1.9 rad/s was observed in the simulation for the negative and positive rotational directions, respectively (see Fig. 13). Fig. 13 also depicts the acquired total average reward function by the RL agents, plotted against the training iteration steps. A clear positive and converging trend can be observed. When deployed on the physical robot, visible rotation of the ball in both directions was observed.

Concerning the cube rotation task, Fig. 12 showcases the performance of the trained RL policy, deployed both in simulation and on the physical hand. The policy performed well in simulation, showcasing a very fast and efficient reorientation of the cube in both directions. Fig. 14 depicts the general trend of the consecutive successes (consecutive successful reorientation and displacements of the object state to the goal state) and the object reorientation reward $R_{\text{obj\_rot}}$. A clear positive trend and convergence is observed in both. When deployed on the physical robot, the robotic hand was able to rotate the cube in both directions, but a significant difference in performance could be observed. The hand rotated the cube in a very swift and efficient manner around the negative z-axis ($\pm$ 4 seconds). Around the positive z-axis, the rotation of the cube was significantly slower and less smooth ($\pm$ 10 seconds). The authors hypothesize that the increased reachability of the thumb and pinky required for the manipulation in the positive z-axis is a likely cause of the difference in performance. For both directions, a slight displacement of the cube from the starting position could be observed.

V. Future Work

Building upon our modular design approach in 4-fingered robotic manipulation, the authors propose several key areas for future research. These include exploring different joint types to enhance the dexterity and manipulation capabilities of the robotic hand (e.g. pin joints), incorporating sensors to improve environmental interaction and task performance, and implementing a closed-loop RL controller (instead of playback of the recorded policy in simulation) to improve adaptability and efficient task execution on the robot. Finally, we aim to focus on techniques that allow the robotic hand to apply learned skills across different tasks, enhancing its autonomy and versatility. However, some improvements still need to be made to the current design, both on the hardware and software front. The system for tendon routing between the joints and fingers needs to be made more robust such that each tendon keeps a constant tension. With regards to the silicone grip, these could be redesigned such that they are more adaptable and easier to put on the system. The output of the vision-based teleoperation and the low-level controller would need to be more synchronized, increasing the accuracy of the pipeline. This could be attributed to the fact that the low-controller does not have any feedback control mechanism.

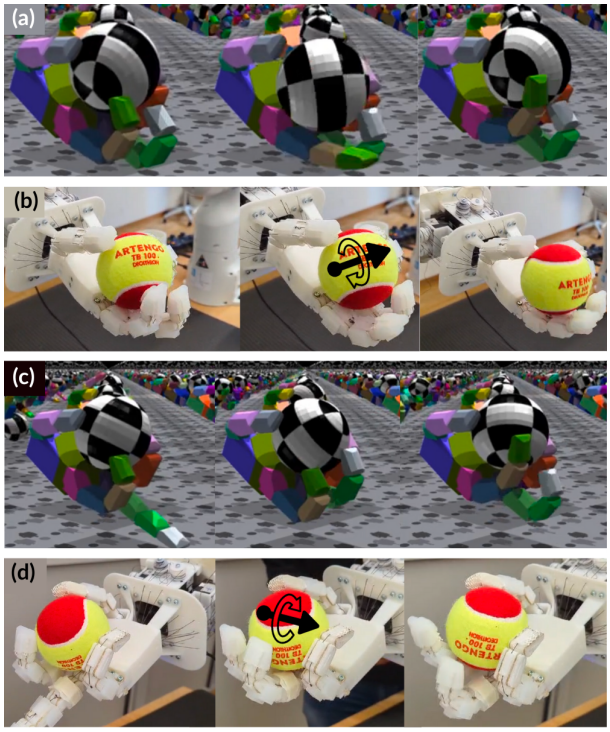

Fig. 11: Performance of the RL policy for the sphere rotation task, deployed both in simulation (a, c) as in real life (b, d). Photo series (a) and (b) show the negative x-axis rotation and (c) and (d) show the positive x-axis rotation.

VI. Conclusion

In conclusion, this work focused on creating a design platform to explore and evaluate the performance of a 4-fingered dexterous anthropomorphic robot manipulator. After developing a modular hardware platform, it was then tested over a multitude of tasks using two different control strategies: vision-based teleoperation and RL-based controller. Based on the experiments of performing different object grasping and manipulation tasks, the hand was able to achieve satisfactory performance for all of the tasks. This helps us show that a 4-fingered hand is as capable as a 5-fingered hand. The work also peers into learning and simulation aspects of designing and testing a robotic hand and helps highlight the capabilities as well as the current limitations of these methods. In conclusion, future work will focus enhancements in joint design and sensor integration, alongside the development of robust, close-loop

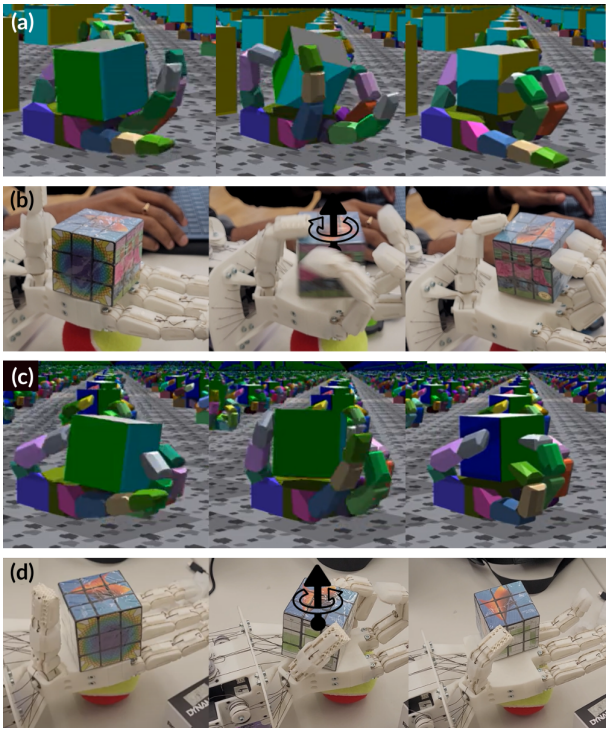

Fig. 12: Performance of the RL policy for the cube rotation task, deployed both in simulation (a, c) as in real life (b, d). Photo series (a) and (b) show the negative rotation direction (around the z-axis of the hand) and (c) and (d) show the positive rotation direction.

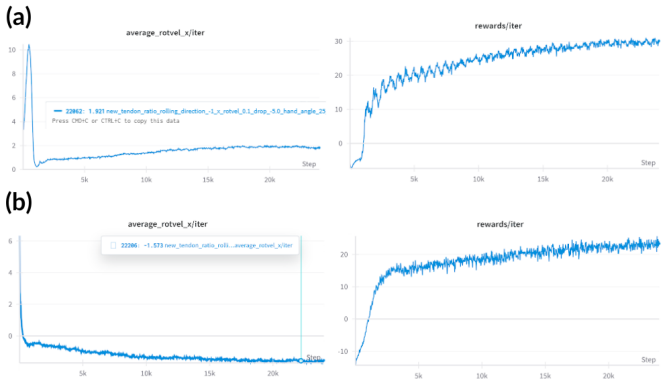

Fig. 13: Results of the RL policy for the sphere rotation task captured in Weights & Biases. The left figures include the average rotational velocity per iteration. The right figures include the total rewards per iteration. (a) contain the results for the rotation of the sphere in the positive x direction. The final average x rotational velocity achieved is $1.9 rad/s$, (b) similarly rotates in the negative x direction. The rotational velocity achieved is $-1.573 rad/s$. The formatting of these figures is meant to show the general trend and convergence of the reward functions. Detailed graphs can be obtained by contacting the authors.

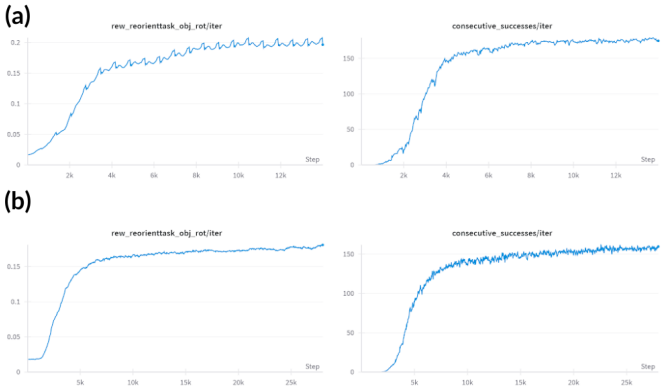

Fig. 14: Results of the RL policy for the cube rotation task captured in Weights & Biases. The left figures include the rewards for the reorientation task per iteration. The right figures include the consecutive successes per iteration. (a) contain the results for the rotation of the cube in 90 degrees in the positive z direction. (b) similarly rotates in the negative z direction.

reinforcement learning pipelines, to markedly improve the precision and adaptability of 4-fingered robotic manipulation.

Acknowledgment

The Soft Robotics Lab, ETH Zürich provided the funding and resources for the work. The authors extend thanks to Prof. Dr Robert Katzschmann and his team for guiding us in the work.

References

[1] Y. Toshimitsu, B. Forrai, B. G. Cangan, U. Steger, M. Knecht, S. Weirich, and R. K. Katzschmann, "Getting the ball rolling: Learning a dexterous policy for a biomimetic tendon-driven hand with rolling contact joints," 2023.

[2] S. K. Sampath, N. Wang, H. Wu, and C. Yang, "Review on human-like robot manipulation using dexterous hands," *Cognitive Computation and Systems*, vol. 5, p. 14–29, Feb 2023.

[3] K. Shaw, A. Agarwal, and D. Pathak, "Leap hand: Low-cost, efficient, and anthropomorphic hand for robot learning," 2023.

[4] S. N. Laboratories, 2012.

[5] S. R. Company, "Shadow dexterous hand series - research and development tool," 2023.

[6] W. Robotics, "Allegro hand: Highly adaptive robotic hand for r&d," 2023.

[7] J.-M. Boisclair, T. Laliberté, and C. Gosselin, "On the optimal design of underactuated fingers using rolling contact joints," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4656–4663, 2021.

[8] C. L. Collins, "Kinematics of robot fingers with circular rolling contact joints," *J. Robot. Syst.*, vol. 20, pp. 285–296, June 2003.

[9] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann, "Mediapipe hands: On-device real-time hand tracking," 2020.

[10] S. Zhong, T. Power, A. Gupta, and M. Peter, "PyTorch Kinematics," 3 2023.

[11] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.

[12] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State, "Isaac gym: High performance gpu-based physics simulation for robot learning," 2021.