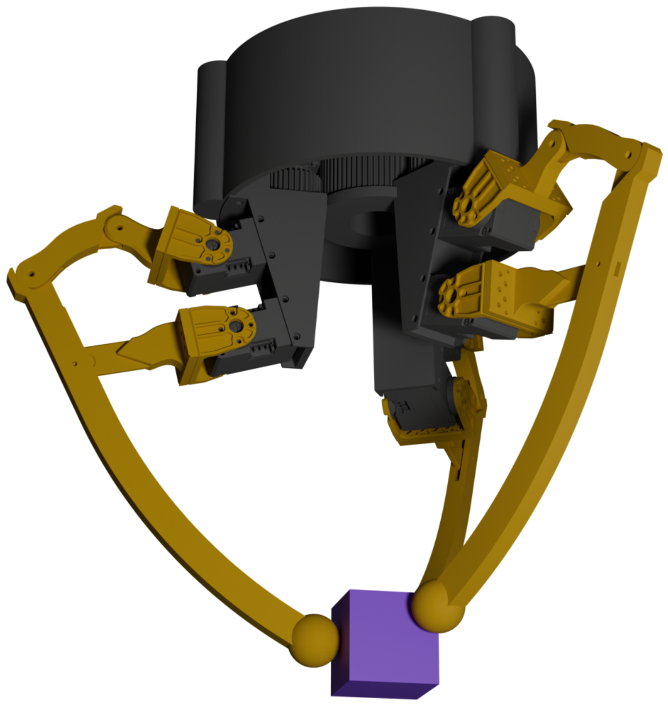
Virtual Design History File: D-PALI 3D In-Hand Manipulation with Reinforcement Learning

ELEC60014 - Group Consultancy Project

27th June 2025

Submitted to: Dr Ad Spiers



Maximilian Adam CID: 02286647
Akin Falase CID: 0219938
Tony Zeng CID: 02214416
Zakariyyaa Chachai CID: 02217311
Yueming Wang CID: 02061452

IMPERIAL

1 Project Brief

The objective of this project was to extend the D-PALI gripper, designed by Arunansu Patra [Arunansu Patra(2023)], capable of in-hand manipulation (IHM) with 2 degrees-of-freedom (2DOF) in a plane, to achieve full 3D IHM. An additonal design goal was to accomplish this using a minimal number of additional actuators and keeping cost of manufacturing low.

2 Specifications

2.1 Mechanical and Simulation Specifications

The 3D manipulation is achieved by applying a three-finger configuration arranged radially around a central base. Each finger is mounted on a satellite gear engaging with a central ring gear, enabling coordinated radial motion. Furthermore, each finger is equipped with an independent actuator, allowing precise control of its angular position relative to the base, enabling full 3D manipulation capability. Breakdown model pictures of the re-designed structure is listed in STL fabrication
The physical and simulated design of the gripper is defined across three core XML files.

- **Gripper3D.xml:** Defines the main kinematic tree of the robot. It consists of a root body and three identical fingers designated Left, Right, and Up (*Base_L*, *Base_R*, *Base_U*). Each finger is a multi-link mechanism including a *Back_Link*, *Short_Link*, *Long_Link* *Front_Link* and *End_Effector*.
- **Shared3D.xml:** Contains shared assets and default parameters for the simulation. This includes:
  - Mesh files for the gripper's components (e.g., `Main_Base.stl`, `Finger_Base.stl`, `Long_Link.stl`).
  - Default classes for visual and collision geometries.
  - Equality constraints that mechanically connect the links of each finger, which are not directly connected in the kinematic tree.
  - Actuator definitions for each joint.
- **DPALI3D.xml:** The main world file that assembles the simulation environment. It includes the gripper from `Gripper3D.xml`, the information from `Shared3D.xml`, a floor, a platform, the manipulable object (*object*), and the target marker (*target*).

2.2 Reinforcement Learning Environment Specifications

The learning environment is defined in `d_pali_hand.py`, which creates a custom Gymnasium environment.

- **Action Space:** The action space is a continuous, normalized box of shape (9,), representing the 9 actuators in the model. Actions are scaled from [-1, 1] to [$-\pi$, π] within the simulation.
- **Observation Space:** The observation is a high-dimensional vector containing the following information:
  - Joint positions and velocities ('qpos', 'qvel').
  - Cartesian positions of the three end-effectors.
  - Position and orientation (quaternion) of the cube and the target.
  - Relative vector from the cube to the target.
  - Relative vectors from the cube to each end-effector.
  - A boolean array indicating which end-effectors are in contact with the cube.
  - The angular difference between the cube's and target's orientation.
- **Reward Function:** Our reward is a carefully shaped function designed to guide the agent. It is a weighted sum of several components, using sparse and dense rewards:
  - **Approach Reward:** Encourages fingertips to move close to the cube.
  - **Manipulation Reward:** An exponential reward based on the distance between the cube and the target, given by $R_{\text{manipulation}} = e^{-\alpha \cdot d_{\text{cube-target}}}$.
  - **Orientation Reward:** Encourages alignment of the cube's orientation with the target's orientation.
  - **Contact Reward:** Provides positive reward for having 2 or 3 fingers in contact with the cube.
  - **Centering Reward:** Once 3 fingers are in contact, this encourages the cube to be at the centroid of the fingertips.
  - **Penalties and Bonuses:** A large success bonus is given for achieving the goal state, and penalties are applied for dropping the cube or for penetrating the cube's geometry.

3 Relevant Code

3.1 Key Code files

The key code files for our project are as follows.

- **MuJoCo XML Files (`*.xml`):** The set of XML files that define the robot model and the simulation world for the MuJoCo physics engine.
- **`d_pali_hand.py`:** Defines the `DPALI_Hand` class, a custom `MujocoEnv` for the gripper. It manages the simulation state, observation vector, action application, and reward computation.
- **`train_TD3.py`:** The main script for training the reinforcement learning agent. It uses the Stable Baselines 3 library to implement the TD3 (Twin-Delayed Deep Deterministic Policy Gradient) algorithm. It also contains functions for setting up the environment, continuing training, and testing a trained model. Key hyperparameters such as learning rate, batch size, and network architecture get defined here before training a model.
- **`TD3_tune.py`:** The script for the hyperparameters searching. It use the Ray-Tune library which provides parallel computing architecture and a customisable searching algorithm. The key hyperparameters mentioned above are founded by this script.
- **`callbacks.py`:** A file containing customised callbacks during training, mainly for additional tensorboard logging.

3.2 Code Structure

The code is seperated into three main folders, `assets/`, `src/`, `training/`. With a few others also contain useful actions and information.

- `assets/` folder includes all MuJoCo descriptions (XML files) and CAD models (stl files).
- `src/` folder included Gymnasium register file and D-Pali hand definition class, `d_pali_hand.py`.
- `training/` folder includes all training scripts and logs.
- `docs/` folder includes documents that reflect our development history.
- `scripts/` folder contains scripts used for testing the environment, producing observations for imitation learning, and other small useful scripts.
- `models/` folder contains RL models with corresponding normalization statistics

The Setup guides and instructions are in the README file under the repository root directory.

4 Materials Sourced and Used

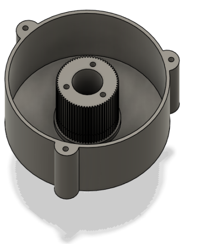
4.1 Tools & Resources Used

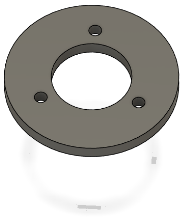
- **Software and Libraries:**
  - Python 3
  - MuJoCo Physics Simulator
  - Gymnasium
  - Stable Baselines 3 for RL algorithms (TD3)
  - Ray[tune] for hyperparameters searching
  - PyTorch (as the backend for Stable Baselines 3)
  - NumPy
- **3D Assets (Meshes):** The following STL files are used for the gripper's geometry:
  - `Main_Base.stl`
  - `Finger_Base.stl`

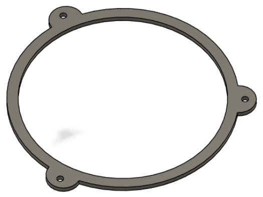
  - Back_Link.stl
  - Short_Link.stl
  - Front_Link.stl
  - Long_Link.stl

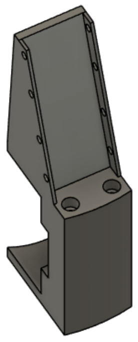
- **Intellectual Sources:**
  - The project builds upon the original D-PALI gripper design by Arunansu Patra (2023).
  - It also considers the work on using Reinforcement Learning to train the original D-PALI by Yanzhou Jin (2024).

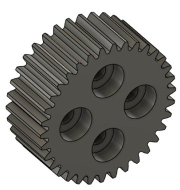
4.2 STL Designs

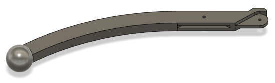

(a) `Main_Base.stl`: Redesigned base consisting centre gear for finger radial movement

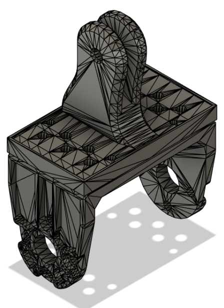

(b) Cap mounted on the centre of the base to lock the satellite part in place while providing inner rotary trajectory

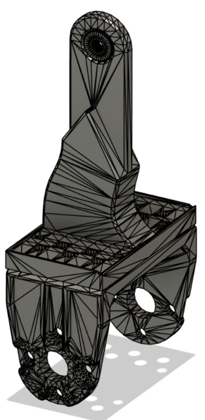

(c) Cap mounted on base to lock the satellite part in place while providing outer rotary trajectory


(d) `Finger_Base.stl`: Satellite structure rotates radially, equipped with D-PALI finger structure


(e) Gear mounted on satellite actuators for rotary movement


(f) `Long_Link.stl`: Reworked Long-link structure from Yanzhou's D-Pali version to accompany workspace


(g) `Back_Link.stl`: Backlink structure from Yanzhou's modified D-Pali design


(h) `Front_Link.stl`: Front link structure from Yanzhou's modified D-Pali design

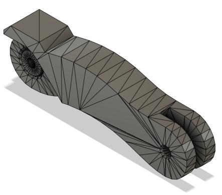

(i) `Short_Link.stl`: Short link structure from Yanzhou's modified D-Pali design

Figure 1: All STLs used in Project

5 Record of meetings held

Date(DD/MM/YY)	Format	Purpose
30/04/25	In-person(408)	Introduction of Group Members to each other
01/05/25	In-person(1004)	Initial client briefing with Ad Spiers
05/05/25	Online(Teams)	Initial distribution of labour of researching different aspects of the project
06/05/25	In-person(505)	Brainstorming of initial ideas for the hand structure design
07/05/25	In-person(505)	Distribution of labour into learning XML,
–	–	designing the modified linkages and base for the new D-PALI model
–	–	and researching machine learning algorithms for training
09/05/25	Online(Teams)	Assembling of First Design in Mujoco
12/05/25	Online(Teams)	Importing model into OpenAI Gymnasium for 1st set of training
14/05/25	In-person(402)	Meeting with academic supervisor, Sarim Baig
19/05/25	Online(Teams)	Catchup with group members to compare training results
28/05/25	Online(Teams)	Catchup with academic supervisor, Sarim Baig
29/05/25	In-person(505)	Design of first leaflet draft
05/06/25	Online(Teams)	Final Design of leaflet
11/06/25	Online(Teams)	Catchup meeting with client, Ad Spiers
17/06/25	Online(Teams)	Design of first Poster draft

6 Decisions Taken with Rationale

- **Decision:** Add a third 'finger' to the existing 2D D-PALI design and allow the fingers to translate radially.
  - **Rationale:** This was determined to be the most efficient method to extend the established stable grasp of the 2D gripper into full 3D in-hand manipulation. Doing this by enabling more complex behaviours such as pinch and drag for rotation about the Y & X axis.

- **Decision:** Revisit and redesign the gripper parts after initial unsuccessful training attempts.
  - **Rationale:** A workspace analysis was performed by using an orthographic camera in MuJoCo to visualize the reachable workspace of each end-effector. The analysis showed that there was very little overlap between the workspaces of the end-effectors due to the large distance between them. This physical limitation meant the gripper was incapable of effectively manipulating the test cube, regardless of the reward function's quality.

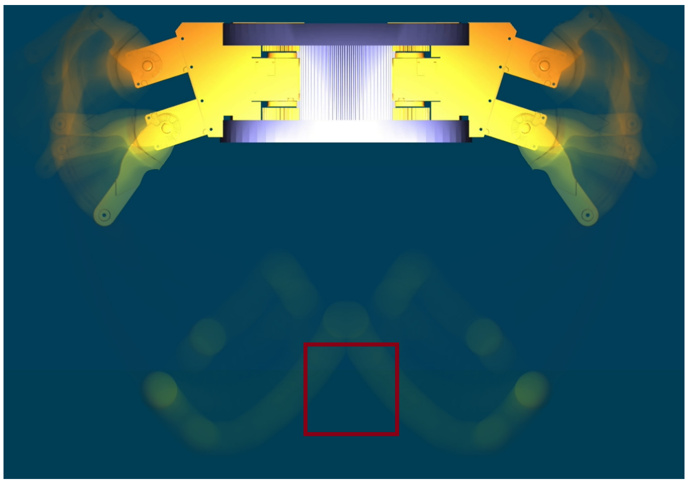


Figure 2: Initial Design end-effector workspaces

- **Decision:** Utilize the TD3 (Twin-Delayed Deep Deterministic Policy Gradient) algorithm for training.
  - **Rationale:** TD3 is excellent for continuous control tasks, which is appropriate for the defined action space of our gripper. Our implementation also uses Ornstein-Uhlenbeck action noise for improved exploration.

- **Decision:** Implement a complex, multi-component shaped reward function instead of a simple sparse reward.
  - **Rationale:** To effectively guide the agent towards the complex goal of 3D manipulation, a shaped reward provides intermediate feedback on its performance. We found this necessary to accelerate learning and decrease the very large search space for our task.

- **Decision:** Utilise imitation learning to assist reinforcement learning.
  - **Rationale:** Due to the complexity of the task for the RL model to complete we decided to utilise imitation learning to explicitly reinforce desirable behaviours for our model to learn on top of. This helped reduced training time needed for a good model to be reached and directly improved overall model performance.

7 Suggestions

7.1 RL work flow

If you would like to continue our project, here is a suggested work flow.

1 Learn different models of robot agents. For example PPO, TD3, SAC.

2 Design a reward function and observation space based on your task.

3 Do hyperparameter searches. Train the model with sufficient steps. Based on our experience, the model needs to be trained with 1M-5M steps. The hyperparameters are specific to the exact reward function; therefore, we do not suggest that you do hyperparameter searches before you found a good reward function.

4 Examine the performance of the reward function using found hyperparameters. Based on our experience, if the reward function does not work after 500K steps of training, you should optimise the reward function or the observation space (physics of model). Repeat this step until you are confident with the reward function and your observation space.

5 Migrate the model to other tasks. If you can't find a good result with unsupervised learning, try to do imitation learning.

6 The method of reward and observation normalisation (VecNormalise) also means this code is easily adaptable to running multiple instances of training at once, which you may wish to consider to speed up the training process.

7.2 Hyperparameters

These hyperparameters are specific to reward function. These are highly task dependant parameters.

- **learning_rate**: $1e-4$ to $7e-4$.
- **batch_size**: 512.
- **buffer_size**: $1e6$.
- **tau**: 0.005.
- **gamma**: 0.98.
- **actor network arch**: $[1024, 512, 256]$
- **critic network arch**: $[512, 256, 64]$
- **activation function**: torch.nn.ReLU

References

[Arunansu Patra(2023)] A. J. S. Arunansu Patra. (2023) D-pali: A low-cost open source robotic gripper platform for planar in-hand-manipulation. [Online]. Available: https://imperiallondon.sharepoint.com/sites/D-PALIInhandManipulationin3DRL-EE/Shared%20Documents/General/D-PALI_A_Low-Cost_Open_Source_Robotic_Gripper_Platform_for_Planar_In-Hand-Manipulation.pdf?CT=1750356561694&OR=ItemsView&wdOrigin=TEAMSFILE.FILEBROWSER.DOCUMENTLIBRARY