UNIVERSITY OF BIRMINGHAM

Empowering Vision-Guided Robotic Manipulation Tasks with Machine Learning - with Example Applications to Electric Vehicle Battery Dismantling

PhD Thesis

Student Name:	Ali Aflakian
Supervisors:	Prof Rustam Stolkin
	Dr Alireza Rastegarpanah
Student ID:	[redacted]
Date:	27th August, 2024
Academic Year:	2024/2025
Word Count:	46936
Total Pages:	141

Contents

List of Figures

List of Tables

Acronyms

ACF	Aggregate Channel Features
AI	Artificial Intelligence
AORLD	Action Optimizer Reinforcement Learning from Multi-demonstrations
AORLD-CL	hypercube with modified loss function
AORLD-HF	convex hull with filtering actions outside the hull
AORLD-HL	convex hull with modified loss function
AORLD-HP	convex hull with projecting actions onto the hull
BC	Behavioral Cloning
DDPG	Deep Deterministic Policy Gradient
DLS	Damped Least Squares
DOF	Degrees of Freedom
DR	Domain Randomization
EE	End-effector
EOL	end-of-life
EV	Electric Vehicle
FDM	Virtual Forward Dynamics Mode
FOV	Field of View
GAIL	Generative Adversarial Imitation Learning
HDVS	Hybrid Decoupled Visual Servoing
HVS	Hybrid VS
IBVS	Image-based VS
IRL	Inverse Reinforcement Learning
ITRA	Interfacing Toolbox for Robotic Arms
LBMPC	Learning-based MPC
LfD	Learning from Demonstration
LoLiMoT	Local Linear Model Tree

MPC	Model Predictive Control
PBVS	Position-based VS
RL	Reinforcement Learning
RLFD	Reinforcement Learning from Demonstration
RMSE	Root Mean Square of Errors
ROS	Robot Operating System
SMD	Standardized Mean Difference
TD3	Twin Delayed DDPG
VS	Visual Servoing

Nomenclature

Chapter 2: List of Variables

Symbol	**Description**
f	Focal length of the camera.
$\mathbf{s} = (u, v)$	Coordinates of a point in the image plane.
$\mathbf{L}_i$	Interaction matrix (Image Jacobian) for the ith feature.
$\mathbf{e}_i$	Difference between the current and desired feature in the image.
$\mathbf{v}_{cam} = (\mathbf{v}_c, \mathbf{w}_c)$	Camera velocity vector.
$\mathbf{v}_c = (v_x, v_y, v_z)$	Camera linear velocity vector.
$\mathbf{w}_c = (w_x, w_y, w_z)$	Camera angular velocity vector.
k_i	Positive proportional gain.
$\mathbf{L}_i^+$	Pseudo-inverse of the interaction matrix $\mathbf{L}_i$.
$\bar{m}_i = [x_i, y_i, z_i]^\mathrm{T}$	Euclidean coordinate of the features in the camera frame.
$\bar{m}_i^* = [x_i^*, y_i^*, z_i^*]^\mathrm{T}$	Euclidean coordinate of the features in the desired camera frame.
λ_p	Positive controller gain in PBVS method.
$\mathbf{L}_p$	6×6 matrix in PBVS method.
$\mathbf{R}$	Rotation matrix.
$\mathbf{L}_\omega(u(t), \theta(t))$	3×3 matrix for rotation components.
$u(t) \in \mathbb{R}^3$	Rotation axis vector.
$\theta(t) \in \mathbb{R}$	Rotation angle.
$[u]_\times$	Skew-symmetric matrix associated with the vector u.
$\mathbf{e}_v$	Translational error signal.
$\mathbf{e}_\omega = u\theta$	Rotational error signal.
$\mathbf{L}_v$	3×3 matrix for translational components.
$\mathbf{L}_{v\omega}$	3×3 matrix relating translational and rotational velocities.
α	Product of the camera scaling factor.
$\mathbf{L}_{xy}$	Decoupled interaction matrix for X and Y translational velocities.
$\mathbf{L}_r$	Decoupled interaction matrix for Z translational and rotational velocities.
$\mathbf{L}_{Pxy}$	Matrix generated by the first and second columns of $\mathbf{L}_p$ in PBVS method.
$\mathbf{L}_{Pr}$	Matrix generated by the last four columns of $\mathbf{L}_p$ in PBVS method.
$\mathbf{J}^{\dagger\lambda}$	Damped Least Squares (DLS) inverse of the Jacobian matrix.
$\boldsymbol{\xi}_c^e$	Transformation matrix from the EE frame to the camera frame.
$\mathbf{t}_c^e$	Translation vector between the EE frame and the camera frame.
$\mathbf{R}_c^e$	Rotation matrix between the EE frame and the camera frame.
$sk(\mathbf{t}_c^e)$	Skew-symmetric matrix of the translation vector $\mathbf{t}_c^e$.
λ	Damping factor in DLS inverse method.
$\mathbf{I}$	Identity matrix.

Symbol	Description
$\dot{\mathbf{q}}$	Joint velocity vector.
$\dot{\mathbf{q}}_0$	Joint velocity vector for secondary task.
k_0	Positive gain for secondary task.
$w(q)$	Cost function for secondary task (robot manipulability).
$\|\|e\|\|$	Norm of the feature error.
$k(\|\|\mathrm{e}\|\|)$	Adaptive gain dependent on feature error.
$\hat{y}_i$	Output of the i-th linear local model in the LoLiMoT neural network.
$\omega_{i0}, \omega_{i1}, \ldots, \omega_{ip}$	Parameters associated with the i-th neuron in the LoLiMoT neural network.
$x_1, x_2, \ldots, x_p$	Input features to the neural network, corresponding to feature errors in the image screen.
M	Total number of neurons in the LoLiMoT neural network.
$\phi_i(x)$	Membership function for the i-th neuron, used to weigh the contribution of each local model to the final output.
$\hat{y}$	Final output of the LoLiMoT neural network, calculated as a weighted sum of the outputs of all local models.
$\mu_i(x)$	Gaussian function representing the unnormalized membership value for the i-th neuron.
c_{ij}	Center of the i-th neuron's receptive field in the j-th input dimension.
σ_{ij}	Standard deviation of the i-th neuron's Gaussian membership function in the j-th input dimension.
P	Number of input features to the LoLiMoT neural network.
m	Number of neurons in the LoLiMoT neural network.
$u_1, u_2, \ldots, u_p$	Input variables in the Gaussian membership function for each neuron.
n_{obs}	Number of observations used for LoLiMoT NN.
xp_i	Predicted value for the i-th observation.
xa_i	Actual (reference) value for the i-th observation.

Chapter 3: List of Variables

Symbol	Description
a	Action taken by the reinforcement learning agent.
A	Action space from which the agent selects actions.
s	State observed from the environment.
S	State space from which the states are sampled.
$\pi(a\|s)$	Policy function that maps state s to action a.
r	Scalar reward received after taking action a in state s.
s'	Preceding state after taking action a in state s.
$R(s, a)$	Reward function mapping state-action pairs to rewards.
$T(s, a, s')$	State transition probability, $P(s'\|s, a)$.
$Q^*(s, a)$	Optimal value function providing the maximum expected reward for state-action pair (s, a).
γ	Discount factor used in the Bellman equation.
$\mathbb{E}$	Expectation operator in the Bellman equation.
$Q^\pi(s, a)$	Action-value function under policy π.
$\mathcal{L}_a$	Loss function for the actor in the DDPG algorithm.
$\mathcal{L}$	Loss function for the critic in the DDPG algorithm.

Symbol	Description
y_t	Target value used to compute the critic's loss function.
$\mathbf{a}_{bound}$	Bounds for the action space, defined by the minimum and maximum values from different approaches.
r_1	First reward function focused on minimizing feature errors.
$r_2(\mathbf{q})$	Second reward function for avoiding joint limits.
$r_3(\mathbf{q})$	Third reward function for avoiding singularities.
$\mathbf{J}$	Jacobian matrix of the robot.
Q_{ϕ_1}	First Q-function used in the TD3 algorithm.
Q_{ϕ_2}	Second Q-function used in the TD3 algorithm.
$\mu_{\theta_{\text{targ}}}$	Target policy function in the TD3 algorithm.
ϵ	Noise term added to target actions in the TD3 algorithm.
c	Constant for clipping the noise in TD3 algorithm.
$y(r, s', d)$	Target value in the TD3 algorithm's Q-learning update.
θ	Parameters of the actor network in the TD3 algorithm.
$\pi(s_t\|g)$	Policy function used to select actions in the TD3 algorithm.
G	Set of additional goals sampled from the replay buffer.
B	Mini-batch sampled from the replay buffer.
n	Number of features or joints in the reward functions or action space.
q_j	Position of the jth joint.
q_{jM}	Maximum angle of the jth joint.
q_{jm}	Minimum angle of the jth joint.
$\dot{\mathbf{q}}_{ibvs}$	Joint velocities from the IBVS approach.
$\dot{\mathbf{q}}_{pbvs}$	Joint velocities from the PBVS approach.
$\dot{\mathbf{q}}_{hdvs}$	Joint velocities from the HDVS approach.
$\mathbf{a}_{min}$	Minimum action values determined from the action bounds.
$\mathbf{a}_{max}$	Maximum action values determined from the action bounds.
k	Indices defining the convex hull of action vectors.
$a_{min,i}$	Minimum value for the ith dimension of actions in the convex hull.
$a_{max,i}$	Maximum value for the ith dimension of actions in the convex hull.

Chapter 4: List of Variables

Symbol	Description
$\boldsymbol{P}$	Tool centre point (TCP) pose comprising position $\boldsymbol{p}$ and ZYX Euler angle orientation $\boldsymbol{R}$
$\boldsymbol{p}$	Position vector of the TCP
$\boldsymbol{R}$	ZYX Euler angle orientation of the TCP
$\boldsymbol{f}_e$	External measured wrench
$\boldsymbol{r}(h)$	Position vector of a path parameterized by arc length h
$\hat{\boldsymbol{c}}(h)$	Path direction vector at a point on the path, unit vector in the direction of the tangent
$\boldsymbol{r}'(h)$	First derivative of the position vector with respect to arc length h
$\\|\boldsymbol{r}'(h)\\|$	Magnitude of the first derivative of the position vector
$\boldsymbol{p}_{start}$	Start-point position of the path
$\boldsymbol{p}_{end}$	End-point position of the path
s	Scalar distance from the endpoint of the path
d	Deviation from the path

Symbol	Description
$\Delta \boldsymbol{p}$	Differential position over the sampling time T_s
$\boldsymbol{x}$	State vector in the LBMPC approach or in the RL approach, which includes $(d, s, \Delta \boldsymbol{p}, \boldsymbol{f}_e, \boldsymbol{R})$
$\boldsymbol{u}$	Action vector in the LBMPC approach or in the RL approach
$\boldsymbol{x}_{k+1}$	State vector at the next time step in LBMPC
$\boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k)$	State transition function in LBMPC
L	Cost metric in the LBMPC optimization problem
$\boldsymbol{J}(\boldsymbol{U})$	Cost function to be minimized in LBMPC
$u_{\max}$	Maximum allowed control input
$\mathbf{H}$	Mass matrix of the robot in the FDM approach
$\mathbf{J}$	Jacobian matrix of the robot in the FDM approach
$\ddot{\boldsymbol{q}}$	Joint accelerations in the FDM approach
$\boldsymbol{f}$	External force applied to the robot in the FDM approach
$\mathbf{K}_p$	Stiffness gain matrix in the FDM approach
$\mathbf{K}_d$	Damping gain matrix in the FDM approach
$\boldsymbol{e}$	Distance error between the target and the current end-effector position
$\boldsymbol{e}_d$	Derivative of the distance error
w_d	Weighting term for deviation in the reward function
w_s	Weighting term for slicing in the reward function
w_u	Weighting term for effort in the reward function
w_c	Weighting term for contact reward in the reward function
w_f	Weighting term for force penalty in the reward function
$\boldsymbol{q}$	Joint position commands
T_s	Sampling time
$\mathbf{J}^+$	Moore-Penrose pseudo-inverse of the manipulator Jacobian
$g(N)$	Envelope function specifying the evolution of the randomisation distribution over training
F_N	Fraction of the limits at episode N in curriculum-based domain randomisation
F_0	Fraction of the limits at episode zero in curriculum-based domain randomisation
$l_{N\pm}$	Maximum and minimum limits for episode N in curriculum-based domain randomisation
l_+	Maximum value of a variable for domain randomisation
l_-	Minimum value of a variable for domain randomisation
N_{max}	Maximum number of episodes in curriculum-based domain randomisation
f_{max}	Target threshold for the ramping force penalty
f_{min}	Minimum force threshold for establishing contact
C	Discrete reward for establishing contact with the environment
r_{term}	Terminal penalty for early episode termination

Chapter 5: List of Variables

Symbol	Description
$\mathbf{M}(\boldsymbol{q})$	Joint-space inertia matrix of a rigid N-link manipulator
$\mathbf{C}(\boldsymbol{q}, \boldsymbol{q})$	Coriolis and centrifugal matrix

Symbol	Description
$\boldsymbol{g}(\boldsymbol{q})$	Gravitational torques vector
$\boldsymbol{\tau}_{\text{ext}}$	External torques vector acting on each link
$\boldsymbol{\tau}$	Control torques vector
$\boldsymbol{q}_l$	Joint configuration of the master robot (Phantom Omni)
$\boldsymbol{q}_f$	Joint configuration of the slave robot (Franka Panda)
${}^{P}\mathbf{T}_{\Delta}$	Delta transformation matrix for the Phantom Omni
${}^{F}\mathbf{T}_{\Delta}$	Delta transformation matrix for the Franka Panda
${}^{P}_{F}\mathbf{T}$	Homogeneous transformation matrix from the Franka arm's base frame to the Phantom Omni's base frame
$\boldsymbol{x}_f$	End-effector pose of the Franka Panda
$\boldsymbol{x}_t$	End-effector pose of the Phantom Omni
$\boldsymbol{e}_x$	Task space pose error, computed as $\boldsymbol{x}_f - \boldsymbol{x}_t$
$\mathbf{J}$	Jacobian matrix of the slave manipulator
$\mathbf{K}_p$	Controller stiffness matrix
$\mathbf{K}_d$	Controller damping matrix
$\boldsymbol{e}_q$	Joint space error, computed as $\boldsymbol{q}_f - \boldsymbol{q}_l$
$\boldsymbol{e}_q$	Derivative of the joint space error
$\boldsymbol{\tau}_f$	Control torque for the slave Franka Panda
$\boldsymbol{\tau}_l$	Control torque for the master Phantom Omni
$\boldsymbol{F}_{\text{ext}}$	External force vector experienced at the Franka end-effector
$\boldsymbol{F}_l$	Force feedback vector received by the Phantom Omni
G	Scaling factor for force feedback, empirically determined to be 0.1
$\boldsymbol{\Lambda}$	Task space inertia matrix
$\mathbf{K}_{d,l}$	Damping matrix for the master Phantom Omni

Abstract

This thesis is focused on the integration of advanced Artificial Intelligence (AI) techniques with computer vision methods to enhance the capabilities of robotic manipulation systems. The research application focuses on the disassembly of Electric Vehicle (EV) battery packs using robotic systems. As the demand for EVs continues to rise, the need for sustainable, and autonomous battery recycling grows, due to the limited lifespan of these batteries. Employing robots in this process is crucial as it guarantees not only cost-effectiveness and precision but also mitigates hazardous exposure for human workers. However, the disassembly of EV batteries is a challenging task for the robots. Firstly because of the diversity in battery types, shapes, and sizes and the absence of standardized formats across manufacturers. Secondly, robotics and automation in manufacturing typically operate within structured environments, doing repetitive tasks on known objects in fixed positions. However, adapting robots to handle different objects and unpredictable situations is challenging, necessitating innovative AI-based robotic solutions that can adapt to the diverse nature of battery packs while ensuring efficient and safe disassembly operations. To achieve these objectives, this study aims to develop control strategies that enable robots to perform precise and efficient manipulation tasks in dynamic and complex environments. The thesis is organized into several chapters, each addressing specific aspects of this comprehensive goal. The thesis begins by explaining the problem statement and difficulties related to the safe recycling of Lithium-ion batteries. It is certainly the case that challenges with disassembly procedures require automated solutions, with robots being a key component. Thereafter, a general introduction to the field, highlighting the need to enhance adaptability, precision, and efficiency in robotic systems through the integration of AI and vision-integrated techniques is presented. The objectives of the research are also provided, preparing the setting for the following chapters. The second chapter focuses on optimizing hybrid Visual Servoing (VS) approaches. Conventional methods in 2D, 3D, and hybrid VS are known to have limitations such as convergence issues, sub-optimal trajectories, and singularities. We carefully studied the behaviour of each approach in controlling specific components and proposed a decoupled hybrid VS approach. This approach integrates the functionalities of both 2D and 3D VS methods, minimizing their shortcomings while leveraging their distinct strengths. Adaptive gains, task sequencing, damped least squares, and projection operator techniques are integrated into the proposed method, each addressing a specific challenge to enhance the overall efficacy of the VS. Since the proposed method introduced computational complexities, we used neuro-fuzzy neural networks to predict and model the behaviour of the proposed VS approach. Then the proposed method is compared with traditional approaches, demonstrating improved efficiency and robustness. In the third chapter, the project investigates the integration of Reinforcement Learning (RL) techniques with VS. This involves expanding the AI intervention beyond image spaces to joint spaces of the robot. The main contribution of this chapter explores the incorporation of deep RL algorithms with the data of several controllers during training to enhance the performance of the trained policy. The proposed approach improves training by dynamically constraining the agent's action spaces using several controller demonstrations, allowing for the learning of robust policies adaptable to various scenarios, and reducing the risk of sub-optimal solutions during training by utilizing the knowledge of mathematically proven control methods. Additionally,

the proposed strategy can be combined with other techniques to enhance the training process. This approach results in a case study demonstrating a 51% reduction in training time, improved convergence rates, and enhanced controllability of the trained agent. Thereafter, the performance of the controller was demonstrated through comparisons with traditional VS methods. The fourth chapter extends the project's scope to contact-rich manipulation tasks. A framework was introduced that employs RL to address the challenges posed by uncertain and complex environments. By utilizing a curriculum-based domain randomization approach, the robotic system's robustness to uncertainties is improved, enabling the successful execution of compliant path-following in a range of conditions. To accelerate the RL training and reduce the problem of getting stuck in sub-optimal solutions, we proposed a strategy using human demonstrations. The data from human demonstrations in completing the desired task was gathered across various surfaces with various friction and stiffness. Thereafter, this data is used to create a 3D shape that includes all the demonstrated trajectories. This shape helps to limit where the RL algorithm should search for solutions. Notably, this strategy differs from imitation learning in that the agent is not required to imitate any particular behaviour. Therefore, it can accommodate even imperfect demonstrations, as the RL policy will correct the agent's behaviour during the training. Chapter five underscores the practicality and adaptability of the developed techniques across a range of scenarios. This chapter presents a comprehensive exploration of various experiments and supplementary works carried out during the PhD project. These include tasks such as sorting, utilizing different grippers and robots, teleoperation, employing haptic devices, and applying computer vision methods such as deep learning for object detection, a model-based tracker for tracking predefined models, and the proposed VS approaches. The final chapter, the conclusion, summarizes the key findings, contributions, limitations, and implications of the research. It reflects on how the integration of AI and VS has led to advancements in robotic manipulation capabilities. This chapter also discusses potential future directions for research and emphasizes the broader impact of the project's outcomes on robotics and automation. Overall, this PhD aims to bridge the gap between advanced computer vision, AI, and traditional robotic manipulation techniques, ultimately contributing to more efficient, adaptable, and robust manipulation systems capable of performing in complex real-world environments.

Acknowledgements

I am profoundly grateful to my supervisors, Prof. Rustam Stolkin and Dr. Alireza Rastegarpanah for their unwavering guidance, invaluable insights, and constant encouragement throughout the journey of this thesis. Your expertise, patience, and dedication have been instrumental in shaping the direction and quality of this work. Your feedback and constructive criticism have pushed me to reach new heights, and I am truly fortunate to have had the opportunity to learn under your supervision.

I would also like to extend my heartfelt appreciation to my parents for their unconditional love, endless support, and belief in my abilities. Your encouragement has been the driving force behind my aspirations, and I am deeply grateful for the sacrifices you have made to see me succeed.

I am indebted to my friends and colleagues who provided valuable insights and encouragement during this endeavor. Your discussions, brainstorming sessions, and cooperation have added depth and dimension to my work.

List of Papers

Ali Aflakian, Rastegarpanah, Alireza, and Rustam Stolkin. "Optimized hybrid decoupled visual servoing with supervised learning." Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering 236.2 (Published 2022): 338-354.

Ali Aflakian, Rastegarpanah, Alireza, and Rustam Stolkin. "Improving the manipulability of a redundant arm using decoupled hybrid visual servoing." Applied Sciences 11.23 (Published 2021): 11566.

Aflakian, Ali, Alireza Rastegharpanah, and Rustam Stolkin. "Boosting Performance of Visual Servoing Using Deep Reinforcement Learning From Multiple Demonstrations." IEEE Access 11 (Published 2023): 26512-26520.

Aflakian, Ali, et al. "An Online Hyper-volume Action Bounding Approach for Accelerating the Process of Deep Reinforcement Learning from Multiple Controllers." Journal of Field Robotics, (Published 2024).

Ali Aflakian, Rustam Stolkin, Alireza Rastegarpanah, "Integrating Multi-Demonstration Knowledge and Bounded Workspaces for Efficient Deep Reinforcement Learning", IEEE-RAS International Conference on Humanoid Robots, (Published 2023).

Scott, S., Islam, Z., Allen, J., Yingnakorn, T., **Aflakian, A.**, Hathaway, J., Rastegarpanah, A., Harper, G.D., Kendrick, E., Anderson, P.A. and Edge, J., 2023. Designing lithium-ion batteries for recycle: The role of adhesives. Next Energy, 1(2), (Published 2023), p.100023.

Hathaway, J., Shaarawy, A., Akdeniz, C., **Aflakian, A.**, Stolkin, R. and Rastegarpanah, A., 2023. Towards Reuse and Recycling of Lithium-ion Batteries: Tele-robotics for Disassembly of Electric Vehicle Batteries.(Published 2023) :2304.01065.

Ali Aflakian, Jamie Hathaway, Rustam Stolkin, and Alireza Rastegarpanah, "A curriculum-based domain randomisation approach for learning contact-rich tasks with parametric uncertainties", Journal of IEEE Access, (Published 2024).

Rastegarpanah, A., Mineo, C. , Contreras, C.A, **Aflakian, A.**, Paragliola, G., Stolkin, R. "Electric Vehicle Battery Disassembly Using Interfacing Toolbox for Robotic Arms", Journal of Batteries (MDPI), (Published 2024).

CHAPTER 1

Introduction

Whether it is in manufacturing, healthcare, logistics, space exploration, or other domains, the demand for versatile and intelligent robotic systems continues to rise. To this aim, this research focuses on the convergence of two critical and rapidly evolving fields: Vision-guided manipulations and Artificial Intelligence (AI). Traditional vision-guided robotic manipulation methods, while effective, often encounter challenges when confronted with complex, dynamic, or uncertain environments. The advent of AI has opened up possibilities for addressing these challenges. By fusing the power of AI with established computer vision and robotic control strategies, this research strives to push the boundaries of what robots can achieve, enabling them to perform tasks more autonomously, efficiently, and adaptively.

1.1 | Industrial challenge that motivates this program

The primary focus of this study is to integrate vision and AI-proposed algorithms in the example application of battery disassembly with robots, aiming to enhance efficiency and precision in this critical process. It is predicted that yearly sales of Electric Vehicles (EV) will rise from 2 million in 2020 to over 14 million in 2030 (without taking into account plug-in hybrid cars) [1]. Therefore, due to the limited lifespan of EVs (10-15 years [2]), a substantial number of EV batteries are anticipated to reach the end of their operational life. This indicates that tons of costly and essential raw materials, like as graphite, nickel, cobalt, and lithium, may be wasted in the ensuing decades [3]. Consequently, there has been an increasing focus on the disassembly and recycling of end-of-life (EOL) battery components in both industry and academia. Recycling materials from existing batteries reduces the demand for new materials in battery production, leading to a reduced overall environmental cost of manufacturing. Moreover, considering that material costs constitute 75-80$\%$ of battery manufacturing expenses [4], recycling holds the potential to lower material prices. This decrease in manufacturing expenses makes electric vehicles a more attractive option, providing benefits for both manufacturers and consumers. During the disassembly of EV battery packs, some materials can be reused, others recycled, while some inevitably become waste. Among these, reusing stands out as the most favourable choice, prioritizing sustainability and resource conservation. If reusing is not feasible, recycling becomes the next best alternative, ensuring that valuable materials are repurposed rather than discarded. Ultimately, minimizing waste generation and maximizing resource utilization are key considerations in the disassembly process [5].
Methods for disassembling batteries can be broadly categorized into three main groups: fully manual, semi-autonomous, and fully autonomous techniques. Previous studies such as [6] have emphasized the importance of autonomous disassembly considering cost savings, time efficiency, and environmental impact reduction. However, there are limitations, including the high degree of variation in EV battery models, the absence of design standardization, and the lack of dexterity in unstructured dynamic environments [7]. Therefore, for many years the

disassembly of batteries remains a primarily manual process for trained experts, demanding a high degree of attention and precision [8, 9]. In Figure 1.1, three distinct battery cell designs and the corresponding packs from existing electric vehicles are shown. It is evident that the three vehicles have significantly diverse designs, necessitating various methods of disassembly, especially in terms of automation. These varied component formats and sizes also limit the applications for re-use.

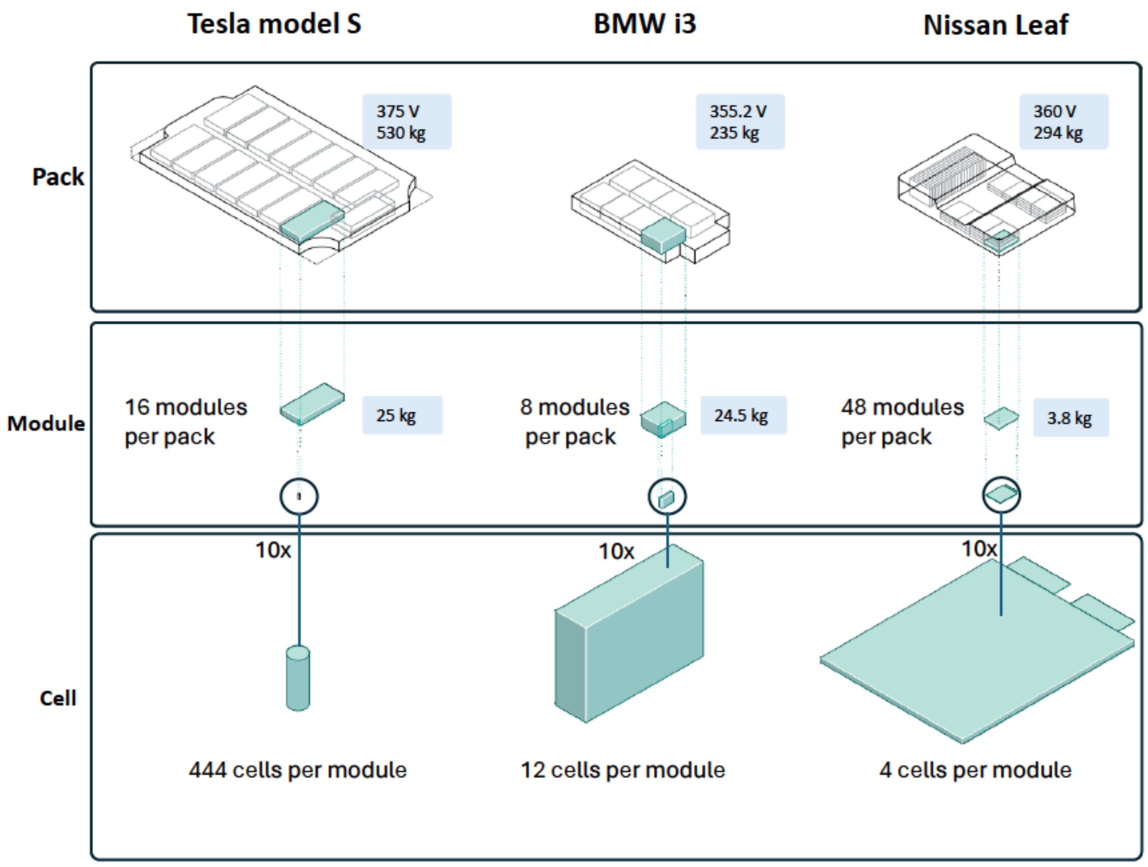


Fig. 1.1. Comparison of different battery packs and components for three car manufacturers made in 2014 [5].

In [9], hybrid frameworks for dismantling EV batteries have been suggested in which robots and humans work collaboratively close to each other. However, EV batteries cause many thermal and chemical risks due to residual charge and the risk of thermal runaway. The related hazards are further increased for damaged batteries, which are difficult to disassemble autonomously due to additional uncertainty in component condition [8, 9]. As a result of the potential hazards faced by human operators in EV battery disassembly settings, recent research has predominantly focused on the development of autonomous methods with the aim of enhancing both efficiency and safety. Additionally, automation might enhance the mechanical separation of components and materials, improving the purity of the materials that are separated and increasing the efficiency of downstream separation and recycling operations [5].
Figure 1.2, illustrates various risks and challenges that exist in the disassembly and recycling of battery materials in different scales. Utilizing intelligent robotic approaches in this context has the potential to reduce these risks and make the disassembly process more cost-efficient. Autonomous methods seek to improve disassembly efficiency by enabling robots to independently plan and execute repetitive tasks in unstructured environments, leveraging visual and tactile feedback. Examples of such research can be found in references such as [10–13]. A common

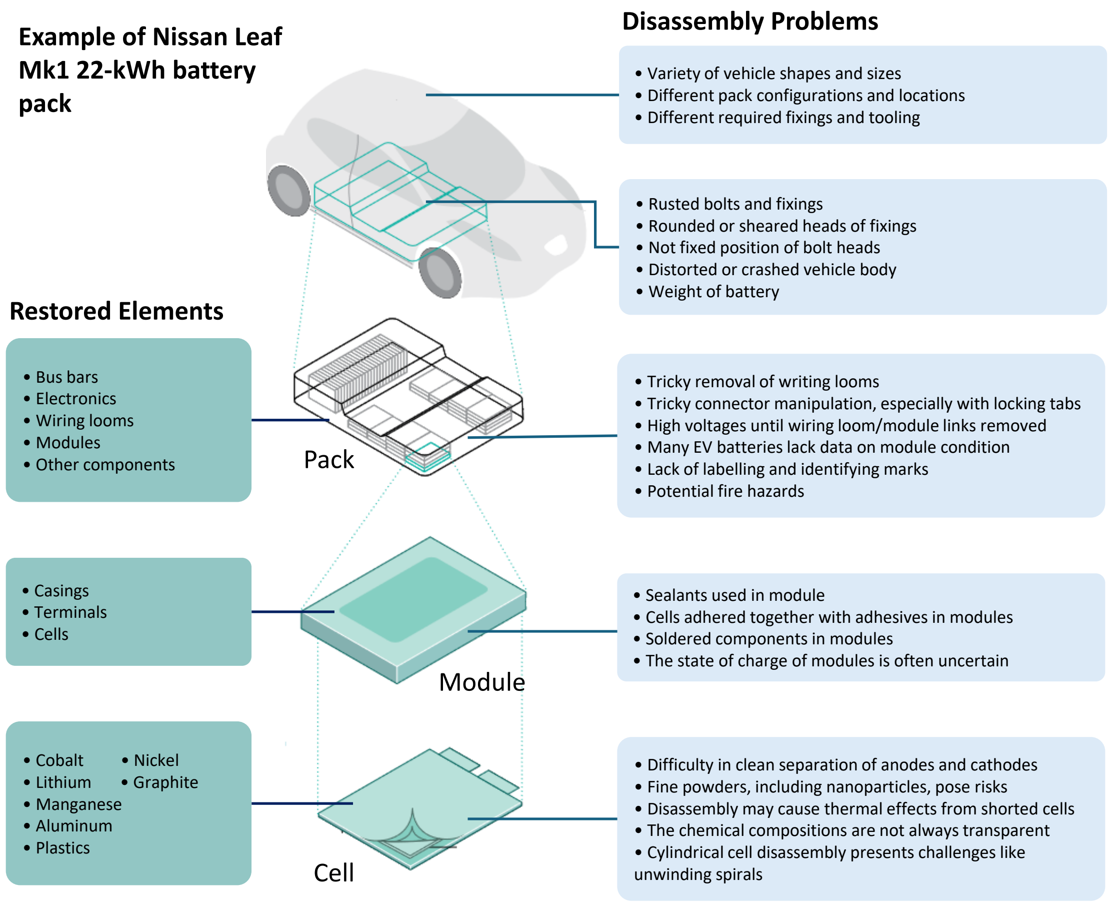

Example of Nissan Leaf Mk1 22-kWh battery pack
Disassembly Problems
• Variety of vehicle shapes and sizes
• Different pack configurations and locations
• Different required fixings and tooling
• Rusted bolts and fixings
• Rounded or sheared heads of fixings
• Not fixed position of bolt heads
• Distorted or crashed vehicle body
• Weight of battery
Restored Elements
• Bus bars
• Electronics
• Wiring looms
• Modules
• Other components
Pack
• Tricky removal of writing looms
• Tricky connector manipulation, especially with locking tabs
• High voltages until wiring loom/module links removed
• Many EV batteries lack data on module condition
• Lack of labelling and identifying marks
• Potential fire hazards
• Casings
• Terminals
• Cells
Module
• Sealants used in module
• Cells adhered together with adhesives in modules
• Soldered components in modules
• The state of charge of modules is often uncertain
• Cobalt
• Lithium
• Manganese
• Aluminum
• Plastics
• Nickel
• Graphite
Cell
• Difficulty in clean separation of anodes and cathodes
• Fine powders, including nanoparticles, pose risks
• Disassembly may cause thermal effects from shorted cells
• The chemical compositions are not always transparent
• Cylindrical cell disassembly presents challenges like unwinding spirals

Fig. 1.2. Complex disassembly challenges in electric vehicle battery packs at different levels of scale [5].

element in these approaches involves the utilization of labelling and detection techniques to autonomously recognize components and fasteners, subsequently constructing appropriate disassembly plans. To securely dismantle battery packs, clever and flexible robotic solutions are required due to the enormous range of battery models and constructions, as well as uncertainties during the disassembly. In other words, unlike assembly processes where component positions and hierarchy of tasks are typically known, disassembly involves uncertain and changing configurations. This is especially true for contact-rich tasks, where interactions with the environment can lead to changes in the positions of components and robots. Not to mention that dealing with wire and loose components adds more challenges, as their shapes and positions may vary case by case. Utilizing vision systems plays a crucial role in such applications, just as our human eyes assist us in correcting our actions to achieve goals. Another vital aspect of having real-time vision feedback is the possibility of collisions, especially in activities involving multiple robots and potentially humans. AI strengthens the robustness, applicability, and generalizability of vision algorithms. This research, titled is primarily focused on the integration of AI and vision algorithms in robotic manipulations.
The project involves multiple stages, including tasks such as unbolting, cutting, grasping, manipulating, and sorting. In this specific phase of the research, our main focus is on the use of vision systems to track and locate battery components in relation to the robots, which is referred to as Visual Servoing (VS). This approach introduces several challenges, including

dealing with the limitations of stationary robots, losing objects from the camera's Field of View (FOV), addressing image-related complexities, and finding physically accessible trajectories for the robot. The motivation for this research stems from the compelling need to empower robots with the ability to detect and manipulate objects effectively, precisely, and robustly in a wide range of scenarios. Section 1.3, explains the integration of robotics, AI, and computer vision within the framework of disassembly in this study.

1.2 | Background and Literature Review

Some content within this section is extracted from our accepted publications out of this research in [14–17].
Vision sensors are widely used in industry to provide contact-less knowledge of the workplace and adjust robot behaviour to deal with the uncertainties of unstructured settings [18]. This section will provide an introduction to the literature and background of generally used concepts in the field of vision-guided robotics, as well as the application of AI to it.

1.2.1 Visual servoing

Visual control, also known as VS or visual tracking, essentially consists of using data from one or more cameras as input to real-time closed-loop control schemes. It is crucial to define some terms before further delving into the VS:

- **Visual features:** Refer to distinctive elements within the visual data captured by cameras, such as corners, edges, or key points. These features serve as essential landmarks for the control system to interpret the surroundings and distinguish between objects.

- **Feature extraction:** Identifying relevant visual features in the camera images is called feature extraction. The defined target features for the robot's task are also called desired image features. The VS control law works by determining the difference between these current and desired features.

- **Computer vision:** Computer vision focuses on enabling computers to interpret, understand, and process visual information from the world, typically in the form of images or videos. It involves tasks such as image recognition, object detection, and image generation. The goal of computer vision is to give machines the ability to see and extract meaningful information from visual data, akin to human vision, enabling them to perform various tasks and make decisions based on what they perceive.

- **Convergence criteria:** Conditions that determine when the robot has successfully reached the desired state. Typically, these criteria involve the error signal reaching a small or acceptable value.

- **Camera calibration:** Camera calibration involves determining the intrinsic and extrinsic parameters of a vision sensor, allowing for accurate reconstruction of the 3D world from 2D images. More in-depth details in 7.1.

- **Eye-in-Hand and Eye-to-Hand:** In eye-in-hand configuration the camera is mounted on the robot's End-effector (EE), providing a direct view of the task space. Nevertheless, in eye-to-hand configuration the camera is fixed in the environment, offering a static viewpoint for broader perspective planning. Figure 1.3 illustrates these two configurations in which the robot uses the visual feedback from the cameras to adjust its movements continually. By doing so, it brings the current features closer to the desired ones, until they align together (the so-called visual servoing).

- **Image Jacobian:** Image Jacobian (Interaction matrix) relates the change in visual features in the image space to the cartesian velocity of the camera or robot's EE. It is a crucial component in the control law for computing the necessary robot motions.

Dynamic manipulation of a system to perform a task defined by a collection of visual constraints is the objective of VS [19]. Since the nature of VS is to correct the movement of a manipulator using online feedback of vision, it helps to compensate for the modelling inaccuracies and uncertainties. [19].

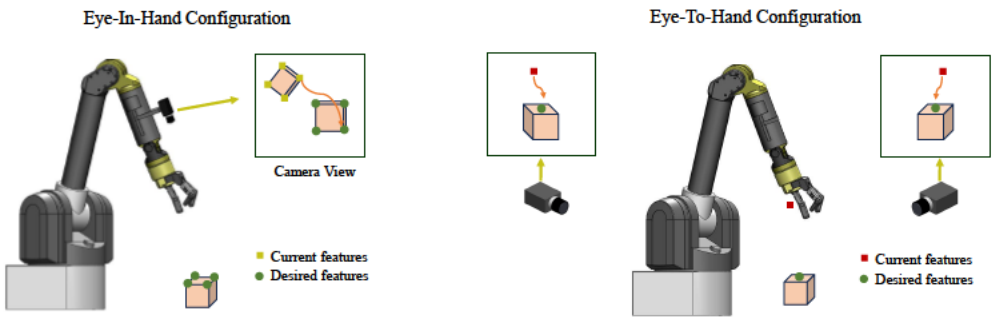
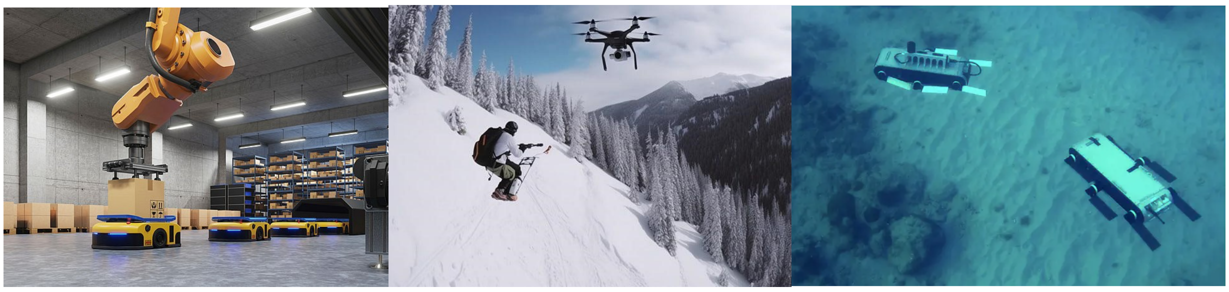


Fig. 1.3. An overview of eye-in-hand and eye-to-hand configurations used for VS.

Controlling a robot using image information has been the focus of various robotic applications, including mobile robots, flying robots (drones), marine robots, medical robots, industrial robots, and more. In mobile robot applications, VS helps robots navigate in dynamic environments by using visual information. It assists the robot in avoiding obstacles, following predefined paths, and tracking objects or targets, making them suitable for applications like warehouse automation, surveillance, smart toys, and autonomous driving cars [20]. Furthermore, robots equipped with VS can identify and harvest ripe fruits or vegetables with precision [21]. VS is also crucial for humanoid robots to interact with their environment, grasp objects, and perform tasks that require hand-eye coordination [22]. In flying robots, VS aids drones in autonomously navigating through complex and dynamic environments by using visual data for obstacle avoidance and path planning to achieve precision landing on designated places and tracking targets[23].
VS is also employed in underwater robots for tasks such as exploring the ocean floor, inspecting underwater structures, and tracking marine life [24]. Another application of VS is in Medical robots. It is applied in robotic-assisted surgery, allowing precise control of surgical instruments based on real-time visual feedback [25]. It enhances the surgeon's accuracy and minimizes errors. VS also has the potential to be employed in space exploration for robotic systems to enhance their capabilities and autonomy for precise maneuvers [26]. An example of that is using VS during spacecraft docking operations, enabling precise alignment and connection with other spacecraft or space stations. In industrial applications VS is used for guiding industrial robots in tasks such as assembly, disassembly, pick-and-place operations, and handling objects with varying positions and orientations. In Figures 1.4 and 1.5, some of the applications of robotic visual tracking are illustrated. There are a number of research studies that applied VS for solving industrial challenges and human-robot cooperation [15, 27–29].

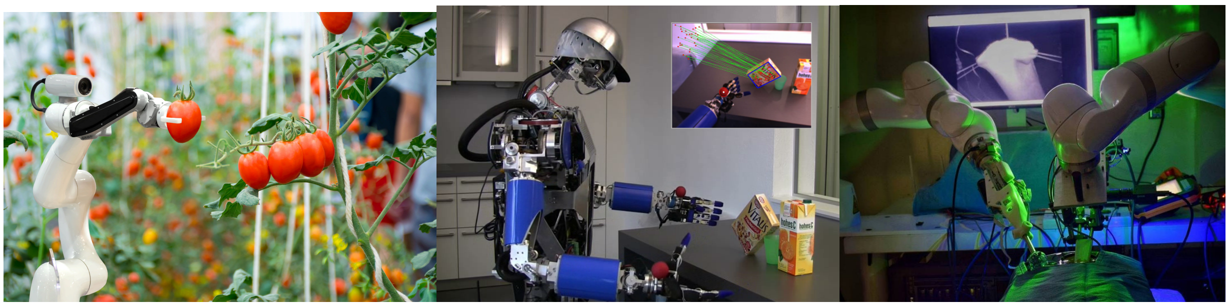
[†]https://www.mvtec.com/application-areas/intralogistics-and-automated-warehouses
[†]https://www.droneguru.net/8-best-drones-that-follow-you-follow-drones/


(a) Visual perception in Robotics for warehousing[*]

(b) Using vision for tracking a moving target with drones[†]

(c) Visual tracking with marine robots [24].

Fig. 1.4. Application of visual tracking for mobile robots, flying drones, and marine robotic systems showcasing their capabilities in navigating dynamic environments and tracking dynamic targets.


(a) Integration of visual feedback with agricultural robotics[‡]

(b) Humanoids dual arm manipulation with VS [30]

(c) Integration of visual feedback with surgical robots[25]

Fig. 1.5. The application of VS in agricultural robotics, humanoids, and medical robotics.

However, VS brings complexities within the 2D image and 3D robot spaces, as well as their intersections [31]. Image-based VS (IBVS) or 2D VS, Position-based VS (PBVS) or 3D VS, and Hybrid VS (HVS) or 2 & 1/2D VS are three primary categories of VS control approaches. The IBVS method computes the desired controller actions directly from extracted image-space features, offering greater resilience against camera calibration [32]. Moreover, image features are less likely to be lost from the camera field of view [33]. On the other hand, IBVS has certain limitations. For example, it may generate controller commands that are physically unattainable for the robot, such as being out of reach. Additionally, IBVS methods struggle with depth information ambiguity, leading to difficulties in handling scenes with occlusions or textureless regions [34]. Besides, the image Jacobian matrix (Interaction matrix) may cause problems with error convergence of features, like singularities and local minima [35]. IBVS failures are mostly for trajectories required rotations about the z-axis (the so-called Chaumette conundrum [19]) and translation in the z-axis (named as camera retreat [35]). In 3D VS or PBVS, the controller will do the visual tracking based on the 3D estimation of 2D features in the camera screen. Due to direct measurement of the camera velocities from the task space errors in the PBVS process, the interaction matrix problems (i.e., local minima and singularity) would be avoided, and thus feasible trajectories for the robot can be generated [36]. However, any error in the camera calibration may create an error in the 3D estimation of the target and consequently affect the entire tracking task [19]. Moreover, the possibility of losing the features in the image screen is higher than 2D VS, since the control law is designed using

[‡]https://dmexco.com/stories/smart-farming/

the 3D estimation of the workspace, To combine the advantages of 2D and 3D VS, hybrid methods were introduced. In such methods, the task function is expressed in a combination of the image space (2D) and the Cartesian space (3D) [37]. One of the limitations of these hybrid methods is their dependency on the co-planar features for real-time tracking, otherwise, it is computationally complex. Furthermore, Using hybrid methods, the control signal might suffer from discontinuity and they need more time to converge in comparison to the 2D and 3D VS [38]. The switching approach is a hybrid visual servoing technique in which the controller alternates between IBVS and PBVS based on their efficacy in different situations [39]. However, when switching occurs the controller suffers from discontinuities, particularly when the object is close to the image borders [40]. Task sequencing techniques provide a solution to fill such gaps [41], nevertheless, sequencing techniques increase the convergence time [19]. Furthermore, two failures in IBVS (i.e., camera retreat and Chaumette Conundrum) could not be easily identified because image-Jacobian is not ill-conditioned in those configurations [34, 35]. As a result, switching between IBVS and PBVS can not resolve these issues. On the other hand, even inducing rotational motions about the camera optic axis could not solve the Chaumette Conundrum, as rotational contributions cancel out one another [35]. In [35], a hybrid VS method is presented that separates the translation and rotation about Z-axis, from the interaction matrix to avoid the chaumette conundrum and the camera retreat issues. However, the expensive computation of the pseudo-inverse is a remaining challenge. Moreover, controlling rotations about the X and Y axes in the image space results in undesirable robot motions [39]. In another study [42], by decomposing translations from rotations in 2 and 1/2D visual servoing methods, unnecessary motions of the robot were reduced. These methods, however, are computationally intensive and necessitate homography construction, which makes them susceptible to image noises [35]. Another disadvantage of the 2 and 1/2D VS approach is the necessity of using co-planar features to estimate the homography matrix. Otherwise, at least 8 visual features are required to make this estimation, while 4 features are sufficient in most of the methods [43]. The 2 and 1/2D VS approach also decomposes homography to remove rotational parameters associated with non-unique solutions [44].
On another note in the case of using static arm manipulators for VS, issues like unfavourable configurations, joint limits, kinematic singularities, and occlusions should be considered. Redundant robots are preferable because adding redundancy to the robot increases the manipulability and versatility [45]. Many studies investigated the use of redundancy to define various types of constraints by integrating secondary tasks that express the constraints with the main task [46–48]. A global objective function is used in [49] to determine a balance between the main task and the secondary tasks by using the redundant Degrees of Freedom (DOF) of the robot with respect to the main task. However, significant perturbations may occur by the obtained motions which are generally incompatible with the main task. In another classical method [50], the authors employed a gradient projection method to solve the redundancy resolution; nonetheless, this approach necessitates that the main task will not constrain all DOFs of the robot. In such cases, the main Jacobian will gain full rank, and there would be no redundancy space left for projecting any constraint. This is a drawback of the traditional gradient projection technique. In [45], a projection operator for the redundant systems was proposed based on a task function specified as the norm of the usual error; in this approach, even if the main task is full rank, this projection operator allows the secondary tasks to be completed.

1.2.2 Artificial intelligence in robotic vision

The integration of visual feedback into robotic control systems has been instrumental in enhancing the capabilities of robots in various applications. However, traditional control systems, which rely on predefined rules and models, face challenges in adapting to dynamic environments and handling complex tasks and uncertainties. AI introduces a paradigm shift in robotics. Unlike traditional control systems, AI allows robots to learn from experience, adapt to changing conditions, and improve their performance over time. In [51], the authors discuss the advancements and challenges in vision-based autonomous systems and AI, including path-planning, localization, perception, and robot control. AI approaches can be broadly categorized into three types: supervised learning, unsupervised learning, and reinforcement learning.

- **Supervised learning:** In supervised learning, the labelled dataset is used to train the algorithm, where a corresponding output is linked to each input. The goal is for the model to learn the mapping between inputs and their corresponding outputs.

- **Unsupervised learning :** In unsupervised learning, the model is trained on unlabeled data, where the algorithm investigates patterns, relationships, or structures within the data without guidance. It aims to discover the inherent structure or distribution of the input data.

- **Reinforcement learning:** RL involves an agent that learns to find solutions by interacting with an environment. The agent observes the feedback in the form of rewards and penalties based on its actions, and the goal is to learn a model (policy) that maximizes the cumulative reward over time.

Regression-based Neural Networks have been extensively used in the literature to approximate the nonlinear behaviour of image-Jacobian [52–55]. For example, in [56], the authors discuss the use of convolutional neural networks for semantic segmentation in robotics, enhancing traditional tasks with an additional level of intelligence, crucial for VS. It should be mentioned that obtaining labelled data in a supervised learning approach can be challenging in numerous real-world scenarios, while unlabeled data are often abundant. To address this issue of limited labelled data, unsupervised learning methods and RL algorithms offer viable solutions.
Recent advancements in deep learning and RL research have empowered robots to tackle progressively more complex tasks, as evidenced by Hua et al.'s work in 2021 [57]. In an RL framework, an agent strives to maximize its expected reward through interactions with its environment. Some example of these are in [53, 58, 59], where researchers use RL to solve vision tracking challenges which are complex to model or computationally expensive. The practical implementation of RL has been constrained by the substantial exploration demands. To overcome this challenge, researchers use the data of demonstrations from human experts or controllers to reduce unnecessary explorations of the RL agent. Incorporating demonstrator or expert data in RL is beneficial for its ability to expedite learning, guide exploration, reduce sub-optimal solutions, and facilitate knowledge transfer to new tasks. There are two potential approaches for incorporating knowledge from expert demonstrations in RL: prior knowledge, comprising demonstrations prior to RL refinement; and online knowledge, in which case demonstrations are occasionally presented while the RL iterations are in progress [60]. The online method can significantly enhance the learned policy's convergence towards an expected performance level while lowering the likelihood of distributional mismatch compared to the prior knowledge approach [61]. However, several challenges remain with the online use of demonstrations while the agent is learning. These include the high cost of data collection; the inability to generalise to different scenarios; and the limitation of the agent's exploration to blindly following the demonstrator [62–64].

1.2.3 Reinforcement learning from demonstrations

RL provides a theoretical way for learning policies through the exploration of the action space. However, the amount of exploration required has limited its implementation in real-world applications. In traditional deep reinforcement learning approaches, the agent explores the entire state space to find the optimal solution. This can be a time-consuming process, and the agent may get stuck in local minima, which are sub-optimal solutions that appear to be optimal within a certain region of the state space [65]. There have been several previous attempts to accelerate the process of deep reinforcement learning using human demonstrations [66–69]. Early Reinforcement Learning from Demonstration (RLFD) algorithms are classified into three main groups: Behavioral Cloning (BC), Generative Adversarial Imitation Learning (GAIL), and Inverse Reinforcement Learning (IRL) [70]. BC was developed based on direct policy learning, which enables the distribution of the state/action trajectory to match the demonstration given by a supervisor. The agent has no capability to respond to environmental changes [62]. Therefore, in the case of using a small number of samples, the trained BC policy has little capability to generalize to different scenarios. IRL was developed to tackle the problem of reward function design and is more adaptable to new situations [63]. While BC and IRL methods gain experience from demonstrations, they have no capability to interact with experts during training to make the trained policy more optimized and robust. To enable the agent to better exploit the expert when optimizing a policy, the GAIL approaches were developed based on generative adversarial networks [64]. The GAIL approach is applied by making a comparison between generated and expert strategies and converging them as closely as possible. However, the GAIL method is susceptible to convergence on local minima [64]. Also, BC methods suffer from data mismatch and compounding error issues. Consequently, the DAgger algorithm was developed to tackle this problem [71]. DAgger is an iterative policy learning method that employs online learning as a reduction in which the main classifier will be retrained on all states encountered by the learner at each iteration. Despite this, the policy trained with DAgger will not be generalised to different scenarios, and the approach is limited to learning from the expert and cannot surpass its performance [72]. Interactive imitation learning methods (e.g. HG-DAgger and Thrifty DAgger) are variants of DAgger, and they are also introduced to address some robustness issues of DAgger [73, 74].
Typically, human demonstrations were employed for such interactive imitation techniques. Algorithms that can use other controllers as experts have been less well studied. Modern RLfD techniques incorporate aspects from imitation learning and push the agent to replicate the demonstrated behaviours when feedback from the environment is scarce or even missing [75, 76]. They specifically reshape the reward function in RL by adding another term to encourage expert exploration. While rewarding expert-like activities might assist in minimising unnecessary exploration, applying such rewards throughout the learning phase can be troublesome with imperfect demonstrations. There is no guarantee that limiting divergence from expert behaviour will result in an improved agent policy [77].

1.2.4 Contact-rich robotic

In recent years, modern robots with proper sensor sets have provided a versatile platform for automating various manual and repetitive jobs by interacting with their surroundings. However, a challenging problem in contact-rich applications is developing a robust control algorithm capable of modelling and generalising unknown contact dynamics and environment changes. To this end, many control issues may be formulated as optimum control problems with discrete-time dynamics and cumulative costs across time. One well-known technique to solve such optimal control problems is Model Predictive Control (MPC) which predicts the behaviour of a system based on its model. MPC employs an online optimization method to determine the best control action for delivering the anticipated output to the reference [78]. However, MPC suffers from several well-known disadvantages, chiefly, high computational complexity, inhibiting the online deployment capability [79]. Explicit MPC approaches furthermore require considerable domain expertise or large amounts of offline labelled training data. These data must have good domain coverage of the state/action space to ensure the accuracy of the model. Moreover, the precision of the model employed for predictions significantly impacts the controller performance [80]. RL approaches show promise for addressing such issues since they allow agents to acquire behaviours through interaction with their surroundings and generalise to new, previously unknown scenarios [81].

Several approaches exist for tackling the challenges presented by high sample complexity and the suitability of RL to real-world deployment, particularly concerning contact-rich tasks. One such approach is based on noting the complementary advantages and shortcomings of MPC and RL, using the former to act as an expert policy to assist the latter. This has been applied in [82], [83]. In the latter, a combination of MPC with RL providing worst-case performance guarantees was proposed, enabling online deployment with improved learning stability. However, the baseline model used neglects the parametric uncertainty in the environment. This is typical of most explicit MPC approaches, which require considerable domain expertise and prior knowledge, necessitating approaches with higher computational complexity. Learning-based MPC (LBMPC) is one such approach that has been used to address the problem of uncertain environments in robotic manipulation tasks [84–86]. In LBMPC, a model of the system is learned through interaction with the environment, and this model is then used to predict future system behaviour and generate control actions. For example, an LBMPC approach for contact-rich tasks with reduced sample complexity was considered in [87], exploring the capability of memory-augmented neural networks as a system model in an MPC framework incorporating visual feedback. However, the high computational complexity of recurrent network architectures, including MANNs, limits the applicability of online deployment.

In a similar vein, recent works have emphasised the selection of suitable action spaces to facilitate the learning of tasks in the real world. In [88], a task-frame formalism was employed to directly train a real-world policy for the task of vegetable cutting by exploiting the task-specific action-space constraints, however, this approach is naturally dependent on accurate prior knowledge of the task specification. In [89, 90], the selection of suitable action spaces for contact-rich manipulation tasks is explored, with related approaches presented based on a variable impedance control system in which the controller gains constitute the policy action space. Methods based on this approach separate the task of compliant path following, which is essential for many interaction tasks, into the distinct problems of trajectory generation and trajectory tracking. Accurate task planning is not always possible to obtain and may not be desired in many applications as explored in [91]. Besides [91], this has been explored extensively in the context of MPC [92, 93], though demonstrably remains an issue for RL.

Many RL-based methods are developed in simulation, owing to the difficulty of transferring learned behaviours to the real world, particularly for destructive tasks. To address this, alternative approaches based on reinforcement learning have been explored to narrow the sim-to-real transfer gap. The Domain Randomization (DR) approach aims to close the reality gap by exposing the agent to a large number of scenarios that may not individually reflect the real-world system. However, it results in a policy representation that is robust to system uncertainties such that it may adapt to the real-world case. Examples of this approach include [94] for path following using an industrial robot. Curriculum-based, or automatic DR has been employed with success in dexterous manipulation tasks as demonstrated by OpenAI in [95, 96], reducing the trial and error procedure of defining an environment with suitable variation to ensure a robust policy, while improving the capability to learn challenging tasks from scratch. An overview of the proposed method to address the aforementioned challenges in the literature will be given in section 1.3.

1.3 | Research objectives

In this section, the research objectives of each developed method will be explained. By breaking down the objectives, the section aims to provide an understanding of the purpose and intended outcomes of each methodological approach employed throughout the study.
To address the limitations inherent in classical VS methods, we have developed a novel approach known as Hybrid Decoupled Visual Servoing (HDVS). In this research, we thoroughly investigated the behaviour of the camera velocity vector components, individually, within the context of both 2D and 3D VS methods. Subsequently, we employed a decoupling strategy to isolate and control the problematic components in 2D and 3D spaces, respectively. This approach effectively allowed us to enhance the robustness and performance of the VS system. In addition to the decoupling mechanism, we integrated various algorithms, such as Damped Least Squares (DLS), task sequencing, adaptive gains, and projection operator techniques into our proposed HDVS method. DLS was used to mitigate discontinuities resulting from the decoupling process. Adaptive gains were used to accelerate the process of tracking. Task sequencing was introduced as a strategic measure to avoid velocity discontinuities. Additionally, using projection operator techniques the introduction of secondary tasks serves to enhance the controllability of the robot throughout the tracking process. Figure 1.6, illustrates the concise schematic overview of the proposed method at this particular stage.

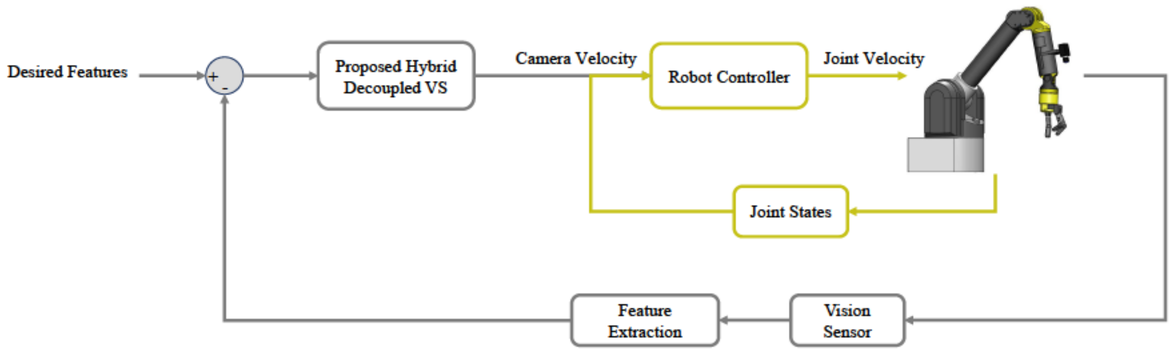


Fig. 1.6. Proposing a hybrid decoupled VS method to overcome traditional VS drawbacks.

However, some computational complexities were introduced by our methodology. To address such complexities, we have used a supervised learning technique to predict and model the behaviour of our proposed VS system. A simplified version of the proposed framework is depicted in Figure 1.7, and an in-depth description of the methodology would be found in Section 2.3.1.

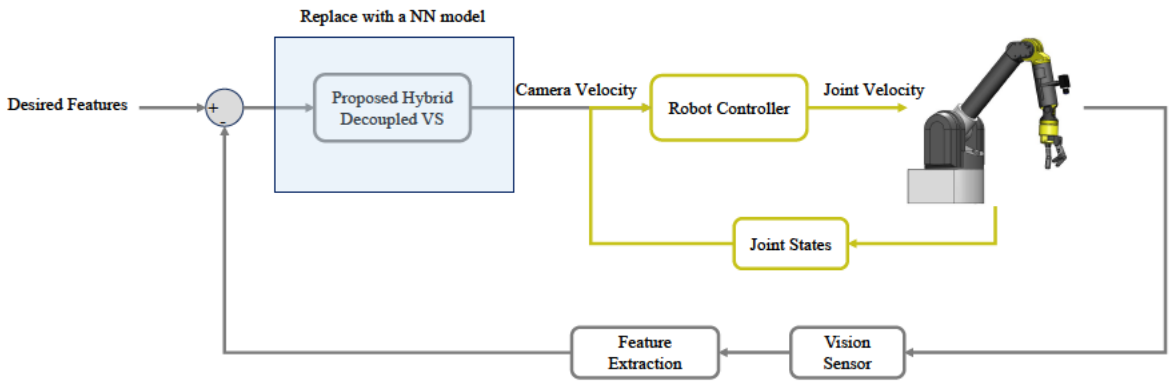


Fig. 1.7. Using supervised learning on image space to enhance the proposed hybrid VS method.

In our next advancement, we have elevated our proposed VS method by integrating RL techniques. This extended approach not only operates within the image space but also extends its influence to the robot's joint space. The incorporation of RL establishes a direct link between image features and the desired joint velocities of the robot's manipulator. This direct mapping serves to mitigate potential issues, including instability and robot singularities. Additionally, it eliminates the necessity for computationally demanding calculations, such as the robot's and the image's pseudo-inverse Jacobian, thereby enhancing the overall efficiency and effectiveness of the control strategy. In Figure 1.8, the adjustments that we applied to implement RL with our previous control loop in Figure 1.7 are illustrated.

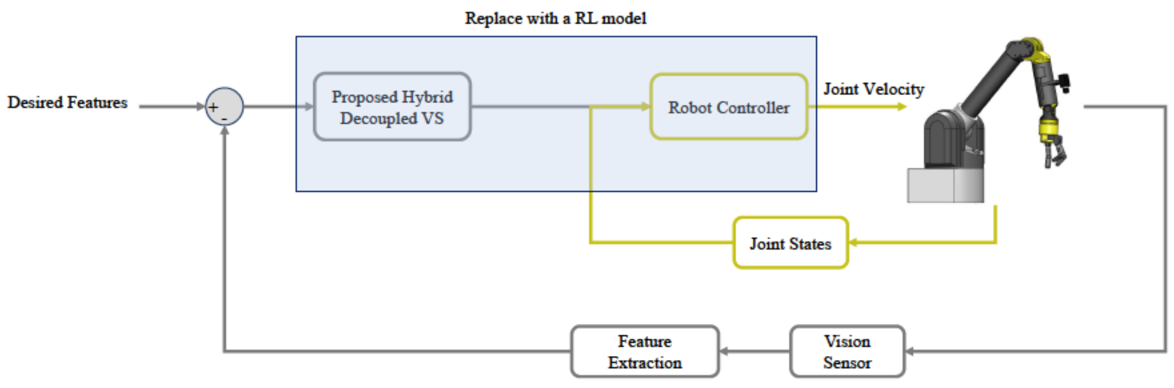


Fig. 1.8. Using RL on both image space and robot space to further improve the proposed VS method.

To avoid unnecessary exploration while also overcoming the problem of the agent overly following the expert behaviour, we present an online Action Optimizer Reinforcement Learning from Multi-demonstrations (AORLD) (detailed in Section 3.2). Figure 1.9, illustrates the proposed AORLD method in summary. As depicted in Figure 1.9, we employed the control error derived from the desired features and current features (the Feedback block) to generate a range of actions using different controllers. Subsequently, we employed these actions to define a convex hull or boundary encompassing these action possibilities. Then, we instructed our RL policy to focus its search space for optimal solutions within the confines of these action bounds.

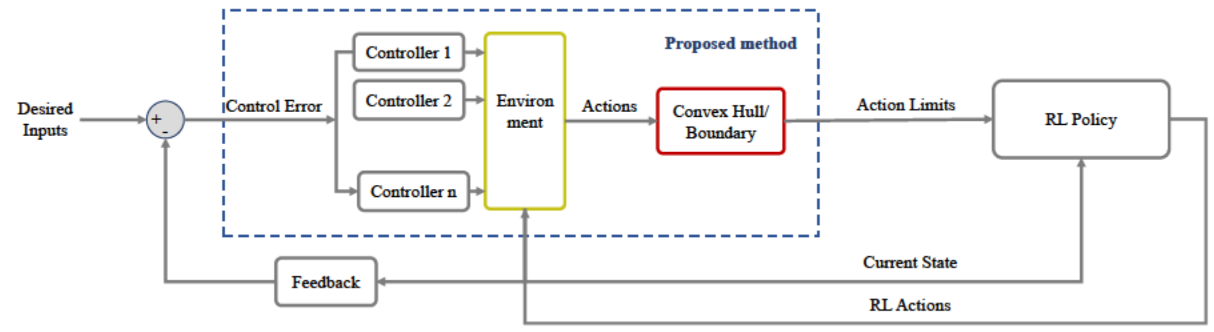


Fig. 1.9. Proposing a Reinforcement Learning from Demonstrations method to accelerate the process of learning and boost the RL policy.

In our previous contributions, we explored how AI could improve tasks involving contact-less vision-guided manipulation. In Chapter 4 we explore further and develop a framework for handling challenging tasks that involve physical contact. A significant challenge in contact-rich applications is developing a robust control algorithm capable of modelling and adapting to unknown contact dynamics and environmental changes. Many of these control issues can be framed as optimal control problems with discrete-time dynamics and cumulative costs over time.

To address these challenges, RL approaches show promise by enabling agents to acquire behaviours through interactions with their environment and generalize their capabilities to new and previously unseen scenarios [81]. In our research, we explore the application of vision-guided RL to tackle contact-rich tasks with parametric uncertainties. We also compare its performance with that of MPC and the Virtual Forward Dynamics Mode (FDM). In traditional deep reinforcement learning methods, agents explore the entire state space to identify optimal solutions. However, this process can be time-consuming and may lead the agent into local minima, which are sub-optimal solutions within specific regions of the state space [65].

To expedite RL and mitigate local minima issues, we proposed a novel method. We gathered data from human demonstrations performed on various surfaces with varying friction and stiffness properties, focusing on a contact-rich path following manipulation. Subsequently, we computed a 3D convex hull that encapsulated all the paths demonstrated by individuals. This convex hull was then employed to define a bounded workspace for RL, thereby reducing the agent's search space to a region more likely to contain optimal solutions, ultimately enhancing sample efficiency. Importantly, our approach does not require the agent to mimic any specific behaviour, distinguishing it from imitation learning. Moreover, it can accommodate even imperfect demonstrations, as the RL policy will adapt the agent's behaviour accordingly. Figure 1.10, shows a simple diagram illustrating our approach, which involves using human demonstrations and limiting the area the agent explores. Chapter 4 of this thesis provides a deeper exploration of the specific details and the outcomes of this research effort.

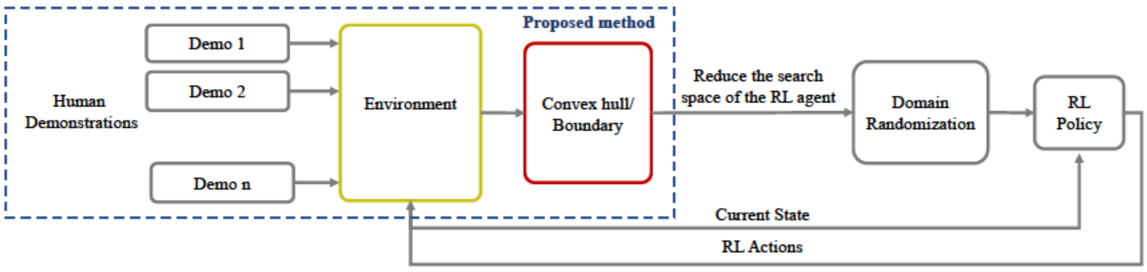


Fig. 1.10. Propose a method to effectively reduce the search space of the RL agent from human demonstrations.

In Chapter 5 of this thesis, the techniques and approaches developed in previous chapters, along with deep learning, model-based tracking, and various other robotic control methods have been applied to the process of battery disassembling. Chapter 5 offers a thorough exploration of the experiments and tasks undertaken within the realm of battery disassembly, employing a range of robotic systems and technologies. The chapter commences with an overview of the experimental setups, which include details about the robots used, the types of grippers employed, haptic devices, vision systems, and the incorporation of deep learning techniques and model-based pick-and-place approaches. It essentially sets the stage by describing the equipment and technology involved. Subsequently, the chapter delves into a thorough exploration of specific disassembly tasks. These tasks encompass sorting, unbolting or unscrewing, cutting, and teleoperation. Each task is elaborated upon, offering insights into how they were executed and what they entailed. To provide a clear and visual understanding of the setups and disassembly processes, a collection of images accompanies each task, offering a visual context for readers to better comprehend the experimental procedures and the equipment used. This visual component enhances the overall clarity and accessibility of the chapter's content. The upcoming subsection outlines publications and submitted papers from this study and elaborates on the objectives and contributions of each one.

1.3.1 Publications out of this thesis

The thesis builds upon a base of peer-reviewed contributions and includes some materials which are submitted to Journals and Conferences. The published papers associated with all authors involved are listed below, along with a brief description connecting each contribution to a chapter.

1. **Ali Aflakian**, Rastegarpanah, Alireza, and Rustam Stolkin. "Improving the manipulability of a redundant arm using decoupled hybrid visual servoing." Applied Sciences 11.23 (Published 2021): 11566.
   Authors contributions: Conceptualization, A.R., A.A. and R.S.; methodology, A.A.; software, A.A; validation, A.R. and A.A.; formal analysis, A.A.; investigation, A.R. and A.A.; resources, A.R., A.A. and R.S.; data curation, A.A.; writing—original draft preparation, A.R. and A.A.; writing—review and editing, A.R., A.A. and R.S.; visualization, A.A.; supervision, A.R. and R.S.; project administration, A.R.; funding acquisition, A.R and R.S.
   Brief description: This study presents an innovative hybrid visual servoing technique

designed to address issues in classical 2D, 3D, and hybrid approaches. The method incorporates adaptive gains, and the Damped Least Square approach to enhance performance and mitigate challenges like convergence issues and robot singularities. Comparisons with traditional methods show that the proposed approach results in more efficient robot trajectories, shorter camera paths, and improved object tracking. Overall, the technique offers enhanced robot controllability, outperforming alternative methods. Chapter 2 of the thesis details the theory, mathematical formulation, and the experimental section of the study. In Chapter 5, the practical utility of the method is demonstrated through its application in the disassembly of a Nissan Leaf battery pack. This chapter serves as a real-world illustration of how the developed method is employed in a specific context, showcasing its effectiveness in the practical domain of disassembling the mentioned battery pack.

2. **Ali Aflakian**, Rastegarpanah, Alireza, and Rustam Stolkin. "Optimized hybrid decoupled visual servoing with supervised learning." Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering 236.2 (Published 2022): 338-354.
   Authors contributions: Conceptualization, A.R., A.A. and R.S.; methodology, A.A.; software, A.A; validation, A.R. and A.A.; formal analysis, A.A.; investigation, A.R. and A.A.; resources, A.R., A.A. and R.S.; data curation, A.A.; writing—original draft preparation, A.R. and A.A.; writing—review and editing, A.R., A.A. and R.S.; visualization, A.A.; supervision, A.R. and R.S.; project administration, A.R.; funding acquisition, A.R and R.S.
   Brief description: This study represents an advancement of a previous publication, detailed in paper 1, by incorporating supervised learning techniques into the development of the proposed decoupled hybrid VS method. A neuro-fuzzy neural network called the local linear model tree approximates the pseudo-inverse of the proposed interaction matrix. This helps avoid singularities, and complex calculations, handle ill-conditioning, and enhance robustness to image noise and camera calibration. Chapter 2 in the thesis delves into the theory, mathematical formulation, and experimental components of the study. In Chapter 5, the method's practical application is illustrated through its use in disassembling a Nissan Leaf battery pack. This chapter exemplifies how the developed method is effectively employed in a specific context, demonstrating its practical effectiveness.

3. **Aflakian, Ali**, Alireza Rastegharpanah, and Rustam Stolkin. "Boosting Performance of Visual Servoing Using Deep Reinforcement Learning From Multiple Demonstrations." IEEE Access 11 (Published 2023): 26512-26520.
   Authors contributions: Conceptualization, A.A., A.R. and R.S.; methodology, A.A.; software, A.A; validation, A.A.; formal analysis, A.A.; investigation, A.A. and A.R.; resources, A.A., A.R. and R.S.; data curation, A.A.; writing—original draft preparation, A.A.; writing—review and editing, A.A., A.R., and R.S.; visualization, A.A.; supervision, A.R. and R.S.; project administration, A.R.; funding acquisition, A.R and R.S.
   Brief description: This study represents an advancement of previously proposed methods in papers 1 and 2 by incorporating RL into the VS framework, leading to improved performance and efficiency in training. The proposed method employs knowledge from various controllers, integrating them with deep reinforcement learning to train a VS technique. The aim is to address the data insufficiency issue in RL methods by

developing a strategy that generates online hyper-volume action bounds from demonstrations by multiple controllers (experts). The agent then explores within these bounds to discover optimized solutions and gain more rewards, reducing unnecessary explorations and improving both training time and performance of the trained policy. Chapter 3 in the thesis extensively examines the theory, mathematical formulation, and experimental components of the study. Transitioning to Chapter 5, the practical application of the method is demonstrated through its implementation in the disassembly of a Nissan Leaf battery pack.

4. **Aflakian, Ali**, et al. "An Online Hyper-volume Action Bounding Approach for Accelerating the Process of Deep Reinforcement Learning from Multiple Controllers." Journal of Field Robotics, (Published 2024).
   Authors contributions: Conceptualization, A.A., A.R. and R.S.; methodology, A.A.; software, A.A; validation, A.A.; formal analysis, A.A.; investigation, A.A. and A.R.; resources, A.A., A.R. and R.S.; data curation, A.A.; writing—original draft preparation, A.A.; writing—review and editing, A.A., A.R., J.H., and R.S.; visualization, A.A.; supervision, A.R. and R.S.; project administration, A.R.; funding acquisition, A.R and R.S.
   Brief description: This study explores different methods for bounding the actions of the RL agent during training, comparing their effectiveness. This publication extends the previous method, explained in paper 3, for broader applicability across various tasks, presenting alternative approaches for constraining action space and demonstrating improved training progress in VS. Chapter 3 of the thesis conducts a thorough exploration of the study, the theoretical foundations, mathematical formulations, and algorithms. This chapter provides a comprehensive examination of the principles and methodologies that form the basis of our proposed AORLD approach.

5. **Ali Aflakian**, Jamie Hathaway, Rustam Stolkin, and Alireza Rastegarpanah, "A curriculum-based domain randomisation approach for learning contact-rich tasks with parametric uncertainties", Journal of IEEE Access, (Published 2024).
   Authors contributions: Conceptualization, A.A., J.H., A.R. and R.S.; methodology, A.A., J.H.; software, A.A, J.H.; validation, A.A., J.H.; formal analysis, A.A., J.H.; investigation, A.A. and A.R.; resources, A.A., J.H., A.R. and R.S.; data curation, A.A., J.H.; writing—original draft preparation, A.A., J.H.; writing—review and editing, A.A., J.H., A.R., and R.S.; visualization, A.A.; supervision, A.R. and R.S.; project administration, A.R.; funding acquisition, A.R and R.S.
   Brief description: In this study, we expand our application scope from contact-less approaches to contact-rich scenarios. We introduce a framework for contact-rich path following using RL with a blend of visual and tactile feedback, enabling path following in unknown environments. Our approach incorporates a curriculum-based domain randomization strategy with a time-varying sampling distribution, ensuring robustness to uncertainties in the robot-environment system. Through simulation evaluations in compliant path-following scenarios with random uncertain environments, we demonstrate the policy's robustness across a range of stiffness and friction values. Additionally, we extend the concept to unknown surfaces with varying curvatures to enhance the trained policy's robustness in adapting to surface changes. The results were compared with the Learning-based Model predictive controller (LBMPC) and Virtual Forward Dynamics Model (FDM) approaches. The details of this method, its underlying principles, and the outcomes of simulations are presented in chapter 4 for a thorough understanding of the

proposed approach. We suggest the potential application of this method for learning more challenging tasks, such as milling, which are difficult to model and depend on a wide range of process variables.

6. **Ali Aflakian**, Rustam Stolkin, Alireza Rastegarpanah, "Integrating Multi-Demonstration Knowledge and Bounded Workspaces for Efficient Deep Reinforcement Learning", IEEE-RAS International Conference on Humanoid Robots, (Accepted 2023).
   Authors contributions: Conceptualization, A.A., A.R. and R.S.; methodology, A.A.; software, A.A; validation, A.A.; formal analysis, A.A.; investigation, A.A. and A.R.; resources, A.A., A.R. and R.S.; data curation, A.A.; writing—original draft preparation, A.A.; writing—review and editing, A.A., A.R., and R.S.; visualization, A.A.; supervision, A.R. and R.S.; project administration, A.R.; funding acquisition, A.R and R.S.
   Brief description: In this study, as an extension of our previous approach in paper 5, we present a novel method to enhance deep RL by incorporating human demonstrations and offline workspace bounding. Our approach involves gathering data from human demonstrations on diverse surfaces with varying friction and stiffness properties. A 3D convex hull is computed to encompass all demonstrated paths. Defining task and desired parameters as reward functions allows the RL agent to learn within this bounded space, reducing the required search space. Comparing our approach with a baseline, results indicate accelerated learning, improved policy performance, and enhanced resilience to local minima. Combining our method with RL also enables the refinement of imperfect demonstrators' behaviour during learning. In Chapter 4 of this thesis, we provide a theoretical and mathematical explanation of our approach, accompanied by simulation results. Furthermore, in Chapter 5, we elaborate on the potential application of our approach in automating and optimizing teleoperation examinations through the use of RL.

7. Hathaway, J., Shaarawy, A., Akdeniz, C., **Aflakian, A.**, Stolkin, R. and Rastegarpanah, A., 2023. Towards Reuse and Recycling of Lithium-ion Batteries: Tele-robotics for Disassembly of Electric Vehicle Batteries.(Published 2023) :2304.01065.
   Authors contributions: Conceptualization, A.R.; methodology, J.H., A.S. and A.R.; software, A.S., J.H. and A.A.; investigation, A.S., J.H., A.A. and A.R.; validation, J.H, A.S and A.R; data curation, J.H. and A.S.; formal analysis, J.H. and A.S.; visualisation, J.H. and A.S.; writing—original draft, C.A., J.H, A.S and A.R; writing—review & editing, A.R., C.A. and J.H, A.A; resources, A.R. and R.S.; supervision A.R. and R.S.; funding acquisition, A.R. and R.S.; project administration, A.R.
   Brief description: In this study we explained that Telerobotics offers a solution for semi-autonomous robotic disassembly, particularly emphasizing the importance of realistic haptic interactions for precision and safety. To investigate, we conducted a comparative study on the disassembly of a Nissan Leaf 2011 module stack using both traditional haptic-cobot master-slave framework and identical master and slave cobots. Chapter 5 of this thesis provides more detailed insights into this study.

8. Rastegarpanah, A., Mineo, C., Contreras, C.A, **Aflakian, A.**, Paragliola, G., Stolkin, R. "Electric Vehicle Battery Disassembly Using Interfacing Toolbox for Robotic Arms", Journal of Batteries (MDPI), (Published 2024).
   Authors contributions: Conceptualization, A.R., C.M.; methodology, A.A., C.M., G.P. ; software, C.M, A.A; validation, A.R., C.M., C.A.C., A.A., and G.P.; investigation, A.R.,

C.M., C.A.C., A.A., and G.P.; resources, A.R., C.M., C.A.C., A.A., G.P., and R.S.; data curation, C.A.C., and A.A.; writing—original draft preparation, C.A.C.; writing—review and editing, A.R., C.M., C.A.C., A.A., and G.P.; visualization, C.A.C., and A.A.; supervision, A.R., C.M. and R.S.; project administration, A.R.; funding acquisition, A.R. and R.S.
Brief description: This paper presents an innovative approach to automating the disassembly of electric vehicle (EV) batteries by integrating the Interfacing Toolbox for Robotic Arms (ITRA) with vision and ROS capabilities. To this aim, we have integrated and tested our proposed hybrid decoupled visual serving approach with industrial robots (KUKA KR500, KUKA KR10, KUKA AGILUS).

This thesis was supported in part by the project called "Research and Development of a Highly Automated and Safe Streamlined Process for Increase Lithium-ion Battery Repurposing and Recycling" (REBELION) under Grant 101104241 and in part by the UK Research and Innovation (UKRI) project "Reuse and Recycling of Lithium-Ion Batteries" (RELiB) under RELiB2 Grant FIRG005 and RELiB3 Grant FIRG057.

1.4 | Thesis structure

The structure of this thesis is visually represented in Figure 1.11. This figure conveys the central theme of the thesis, which revolves around the utilization of vision and artificial intelligence in the context of manipulation tasks. These tasks are broadly categorized into two main types: contact-less and contact-rich tasks. Moreover, the specific application chosen for in-depth study in this thesis is the disassembly of battery packs.
In Chapter 2, the primary focus is on introducing the proposed VS method. This chapter explores the VS method and the incorporation of supervised learning to enhance its performance. The findings gained in this chapter establish the foundation for the subsequent chapters for further investigation and improvements. Chapter 3 introduces the integration of Reinforcement Learning (RL) with the previously proposed VS method. This chapter discusses general RL challenges and our strategies for mitigating them, including leveraging data from other systematic controllers and incorporating domain randomization and adaptation in our learning framework. The investigation continues in Chapter 4, with a focus on using vision in contact-rich applications and RL. This investigation is built on a case study, with an emphasis on using domain randomization within a curriculum-based randomization framework. The chapter concludes by demonstrating the advantages of incorporating human demonstrations to improve RL performance and reduce the agent's search space and sub-optimal solutions. The possibility of using model-based tracker and deep learning in the disassembly loop is explored in Chapter 5. The chapter also details various setups and practical demonstrations conducted during the study. This section provides an in-depth overview of the various ways used for the disassembly process, highlighting the benefits of these approaches. Finally, Chapter 5 serves as the conclusion, providing a cohesive summary of the key findings and limitations of each method explored in the earlier chapters. The chapter also identifies possible future directions and applications for further research and concludes the thesis by providing a comprehensive overview of the contributions made.
Each chapter, excluding the conclusion and index, follows a consistent structure. Starting with an abstract and introduction outlining the challenges and gaps, followed by a methodology, results, explanations, details on the setups that were used, and a final summary. The goal is to make sure that the research material is presented in the thesis in a coherent and logical form.

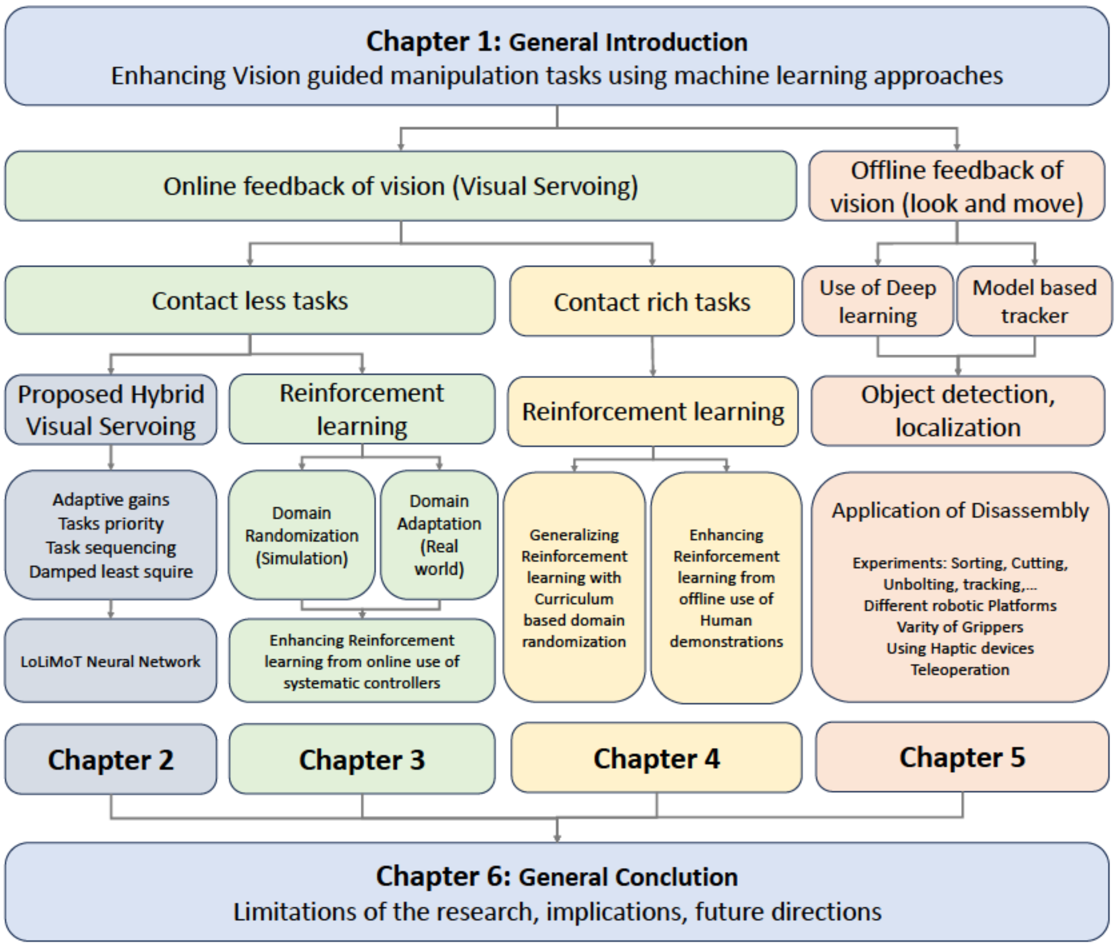

Chapter 1: General Introduction
Enhancing Vision guided manipulation tasks using machine learning approaches
Online feedback of vision (Visual Servoing)
Offline feedback of vision (look and move)
Contact less tasks
Contact rich tasks
Use of Deep learning
Model based tracker
Proposed Hybrid Visual Servoing
Reinforcement learning
Reinforcement learning
Object detection, localization
Adaptive gains
Tasks priority
Task sequencing
Damped least squire
Domain Randomization (Simulation)
Domain Adaptation (Real world)
Generalizing Reinforcement learning with Curriculum based domain randomization
Enhancing Reinforcement learning from offline use of Human demonstrations
Application of Disassembly
Experiments: Sorting, Cutting, Unbolting, tracking,...
Different robotic Platforms
Varity of Grippers
Using Haptic devices
Teleoperation
LoLiMoT Neural Network
Enhancing Reinforcement learning from online use of systematic controllers
Chapter 2
Chapter 3
Chapter 4
Chapter 5
Chapter 6: General Conclution
Limitations of the research, implications, future directions

Fig. 1.11. Overview of chapters and their interconnections

CHAPTER 2

Hybrid-Decoupled Visual Servoing with Supervised Learning

Content from this chapter has been published in the following:

Ali Aflakian, Rastegarpanah, Alireza, and Rustam Stolkin. "Improving the manipulability of a redundant arm using decoupled hybrid visual servoing." Applied Sciences 11.23 (Accepted 2021): 11566.

Ali Aflakian, Rastegarpanah, Alireza, and Rustam Stolkin. "Optimized hybrid decoupled visual servoing with supervised learning." Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering 236.2 (Accepted 2022): 338-354.

2.1 Introduction

To address the explained convergence and performance issues of VS in literature 1.2, we proposed an optimized VS method called Hybrid Decoupled Visual Servoing (HDVS). In the proposed method, all three rotations, and translation in the Z-axis have been decoupled from the image-Jacobian. These four components' errors will be regulated from the 3D reconstruction of the visual features. Consequently, the controller has independent control over translation in Z-axis, and rotations. Thereafter, the LoLiMoT neural network has been used to approximate the pseudo-inverse of the proposed interaction matrix. Using LoLiMoT avoids singularities that could be happened in the interaction matrix, and reduces the computational complexities effectively. Accordingly, the controller becomes robust to camera calibration errors and image noises. LoLiMoT is a fast, effective neuro-fuzzy neural network that learns a huge number of nonlinear models [97]. LoLiMoT method guarantees a global optimization solution which implies the generalization ability of the method [97]. Regression approaches are blind in the detection of global minima, but LoLiMoT is axis orthogonal and operates by errors, thus it will not stuck in local minima [98]. The locality of this method provides online learning in one region without forgetting the other operating regions [99]. Besides, the number of required trial-and-error steps will be reduced in the LoLiMoT approach. In the following sections, the method will be discussed in detail and its efficacy will be validated in both simulation and the real world. The contributions of this chapter are itemized in the following:

- The proposed method produces an optimized trajectory, both in the image space and the joint space in terms of control effort and convergence time. Moreover, HDVS is robust to camera calibration and image noises.
- The proposed approach produces more controllable trajectories (higher manipulability) for the robot than IBVS when tracking objects. In comparison to PBVS and HVS

approaches, the HDVS approach is less likely to lose the object from the camera's Field of View (FOV). The method's effectiveness will be compared with the other methods in both simulation and the real world (2.5).

- The VS process has been boosted by using adaptive gains and The functionality of the manipulator has been increased by defining the manipulability as a secondary task. This contribution has been explained explicitly in Section 2.3.1.

- HDVS benefits from Damped-least squire (DLS) inverse instead of pseudo-inverse to generate the joint velocities. Therefore, HDVS reduces the effect of the robot's singularities and it assists in smoothing the discontinuities created by the decoupling process and adaptive gains. This contribution has been explained explicitly in Section 2.3.1.

- A set of LoLiMoT NNs have been trained in the presence of image noise to approximate the interaction matrix. Consequently, singularities of the interaction matrix would be avoided and computational complexities would be reduced effectively. Nevertheless, using LoLiMoT NN makes the controller robust to image noises. This contribution has been explained explicitly in Section 2.3.3.

Figure 2.1 illustrates the critical problems in 2D, 3D, and Hybrid methods and contributions of the proposed HDVS approach in the image space, joint space, and the interaction of these two. The rest of this chapter is as follows: A brief overview of various visual servoing controllers

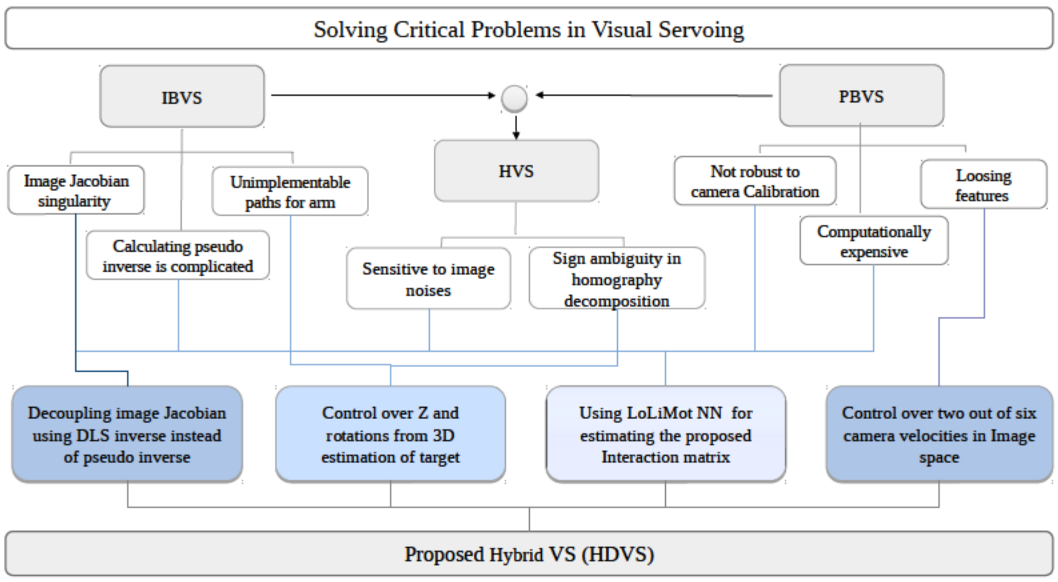

Solving Critical Problems in Visual Servoing
IBVS
PBVS
HVS
Image Jacobian singularity
Unimplementable paths for arm
Calculating pseudo inverse is complicated
Sensitive to image noises
Sign ambiguity in homography decomposition
Not robust to camera Calibration
Computationally expensive
Loosing features
Decoupling image Jacobian using DLS inverse instead of pseudo inverse
Control over Z and rotations from 3D estimation of target
Using LoLiMot NN for estimating the proposed Interaction matrix
Control over two out of six camera velocities in Image space
Proposed Hybrid VS (HDVS)

Fig. 2.1. Problem domains in classical visual servoing and contributions of the proposed HDVS method.

has been investigated in Section 2.2. Section 2.4 explains the simulation and experimental setups designed to evaluate the performance of the proposed VS method. In Section 2.5, the behaviour and effective parameters of four VS methods have been investigated. Finally, this work is concluded in Section 2.6 and future works are discussed.

2.2 | Background

In this section, a background of the classical visual servoing methods is given first. Thereafter, our proposed HDVS method will be explained in detail.

2.2.1 Image-Based visual servoing

In the IBVS method, the feedback from the image features will be directly used and the image-Jacobian (Interaction matrix $\mathbf{L}_i$) will be used to relate the pixel velocity to the camera velocity [100]. The Interaction matrix for the ith feature would be defined as follows [100]:

$$\mathbf{L}_i = \begin{bmatrix} \frac{f}{Z} & 0 & -\frac{u}{Z} & -\frac{uv}{f} & \frac{f^2+u^2}{f} & -v \\ 0 & \frac{f}{Z} & -\frac{v}{Z} & -\frac{f^2+v^2}{f} & \frac{uv}{f} & u \end{bmatrix} \tag{2.1}$$

where f denotes the focal length of the camera and $\mathbf{s} = (u, v)$ denotes the coordinates of a point in the image plane. Let's consider $\mathbf{e}_i$ is difference between current and desired positions of each feature in the image plane, and $\mathbf{v}_{cam} = (\mathbf{v}_c, \mathbf{w}_c)$ is the camera velocity vector (where $\mathbf{v}_c = (v_x, v_y, v_z)$ is camera linear velocity vector, and $\mathbf{w}_c = (w_x, w_y, w_z)$) is its angular velocity vector. The exponential decoupled decrease of the error can be obtained when the Interaction matrix at the desired pose is not singular (i.e., $\mathbf{s}_i(t) - \mathbf{s}_{id}(t) = \mathbf{e}_i(t) = 0$). Therefore, the appropriate camera velocity vector will be determined using the following control law [100]:

$$\begin{bmatrix} \mathbf{v}_c \\ \mathbf{w}_c \end{bmatrix} = -k_i \mathbf{L}_i^+ \mathbf{e}_i \tag{2.2}$$

where k_i represents a positive proportional gain and $\mathbf{L}_i^+$ represents the pseudo-inverse of $\mathbf{L}_i$.

2.2.2 Position-based visual servoing

In position-based visual servoing, the feedback is obtained from the pose reconstruction of the environment. The pose estimation will be calculated with the help of Euclidean methods and camera parameters, from the camera image.

The Euclidean coordinate of the features in the camera frame is $\bar{m}_i \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}^{\mathrm{T}}$ and the Euclidean coordinate of the features in the desired camera frame is $\bar{m}_i^* \begin{bmatrix} x_i^* & y_i^* & z_i^* \end{bmatrix}^{\mathrm{T}}$. The controller in this method is defined as [43]:

$$\begin{bmatrix} \mathbf{v}_c \\ \mathbf{w}_c \end{bmatrix} = -\lambda_p \mathbf{L}_p^{-1} \mathbf{e}_p \tag{2.3}$$

Where λ_p is a positive controller gain and $\mathbf{L}_p(t)$ is a 6×6 matrix obtained from the following equation:

$$\mathbf{L}_p = \begin{bmatrix} \mathbf{R} & 0 \\ 0 & \mathbf{L}_\omega \end{bmatrix} \tag{2.4}$$

Where $\mathbf{L}_\omega(u(t), \theta(t))$ is a 3×3 matrix defined as:

$$\mathbf{L}_\omega = I_3 - \frac{\theta}{2}[u]_\times + \left(1 - \frac{\operatorname{sinc}(\theta)}{\operatorname{sinc}^2\left(\frac{\theta}{2}\right)}\right)[u]_\times^2 \tag{2.5}$$

Where $u(t) \in R^3$ and $\theta(t) \in R$ are the rotation axis and rotation angle, decomposed from the rotation matrix. $[u]_\times$ is the skew-symmetric matrix associated to the vector u.

2.2.3 Homography-based visual servoing

The Homography-based visual servo control approaches mostly decompose the six-DoF motion of the camera in two separate controllers to achieve the convergence goal; one for the translational components and the other one for the rotational components. The error signal will be selected by the estimated position from the Euclidean information and directly from the image information [101]:

$$\mathbf{e}_v = \begin{bmatrix} m_{xi} - m_{xi}^* & m_{yi} - m_{yi}^* & \ln\left(\frac{z_i}{z_i^*}\right) \end{bmatrix}^{\mathrm{T}} \tag{2.6}$$

$$\mathbf{e}_\omega = u\theta \tag{2.7}$$

$\dot{\mathbf{e}}_v = \mathbf{L}_v \mathbf{v}_c + \mathbf{L}_{v\omega}\mathbf{w}_c$ and $\dot{\mathbf{e}}_\omega = \mathbf{L}_\omega \mathbf{w}_c$ could be defined as corresponding transitional and rotational errors. In which $\mathbf{L}_\omega(t)$ was presented in Eq.2.5 and $\mathbf{L}_v(t), \mathbf{L}_{v\omega}(t)$ are 3×3 matrices that are obtained from:

$$\mathbf{L}_v = -\frac{\alpha_i}{z_i^*}\begin{bmatrix} 1 & 0 & -m_{xi} \\ 0 & 1 & -m_{yi} \\ 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

$$\mathbf{L}_{v\omega} = \begin{bmatrix} m_{xi}m_{yi} & -1 - m_{xi}^2 & m_{yi} \\ 1 + m_{yi}^2 & -m_{xi}m_{yi} & -m_{xi} \\ -m_{yi} & m_{xi} & 0 \end{bmatrix} \tag{2.9}$$

Where α is the product of the camera scaling factor. The translation and the rotation controllers could be defined as:

$$\mathbf{w}_c = -k\mathbf{L}_\omega^{-1}\mathbf{e}_\omega = -k\mathbf{e}_\omega \tag{2.10}$$

$$\mathbf{v}_c = -\mathbf{L}_v^{-1}\left(k\mathbf{e}_v - k\mathbf{L}_{v\omega}\mathbf{e}_\omega\right) \tag{2.11}$$

2.3 | Proposed approaches

In the following, our proposed approaches to enhance the performance of the visual serving are explained in more detail.

2.3.1 Proposed hybrid visual servoing

The proposed HDVS method decouples the Z-axis of translational velocity and the three rotational velocities (the components that create the IBVS singularity and unnecessary motions for the robot) from the image-Jacobian matrix. The translational velocities of X and Y components will be measured in 2D, while the error of the other four parameters will be calculated by 3D estimation of the target. The control rule in the classic IBVS system is defined below:

$$\dot{\mathbf{s}} = \mathbf{L}_i \mathbf{v}_{cam} \tag{2.12}$$

By decoupling the interaction matrix, the control law would be amended as follows:

$$\dot{\mathbf{s}} = \mathbf{L}_{xy}\mathbf{v}_{xy} + \mathbf{L}_r \mathbf{v}_r \tag{2.13}$$

where $\mathbf{v}_{xy} = [\mathbf{v}_x \mathbf{v}_y]^T$ and $\mathbf{v}_r = [\mathbf{v}_z w_x w_y w_z]^T$. In addition, $\mathbf{L}_{xy}$ and $\mathbf{L}_r$ are calculated as follows:

$$\mathbf{L}_{xy} = \begin{bmatrix} \frac{f}{Z} & 0 \\ 0 & \frac{f}{Z} \end{bmatrix} \tag{2.14}$$

$$\mathbf{L}_r = \begin{bmatrix} -\frac{u}{Z} & -\frac{uv}{f} & \frac{f^2+u^2}{f} & -v \\ -\frac{v}{Z} & -\frac{f^2+v^2}{f} & \frac{uv}{f} & u \end{bmatrix} \tag{2.15}$$

Hence:

$$\mathbf{v}_{xy} = \mathbf{L}_{xy}^{+}(\dot{\mathbf{s}} - \mathbf{L}_r \mathbf{v}_r) \tag{2.16}$$

since the time variation of the features is related to the feature errors $\dot{\mathbf{s}} = -ke$, therefore (2.16) will change to:

$$\mathbf{v}_{xy} = \mathbf{L}_{xy}^{+}(-k(||\mathrm{e}||)\mathbf{e} - \mathbf{L}_r \mathbf{v}_r) \tag{2.17}$$

To reduce the convergence time, the following adaptive representation of the controller gain has been implemented [102]:

$$k(||\mathrm{e}||) = (k(0) - k(\infty))e^{\frac{-k(0)}{k(0)-k(\infty)}||\mathrm{e}||} + k(\infty) \tag{2.18}$$

In (2.18), for small amounts (less than 0.005 m) of $||e||$ the positive amount of gain is $k_0 = k(0)$, while for the high amounts (more than 0.005 m) of $||e||$ the gain is $k_\infty = k_{||e||\to\infty}, k(||\mathrm{e}||)$, and the slope of k at $||e|| = 0$ is k'_0.
The term $\mathbf{L}_r\mathbf{v}_r$ will be calculated during each iteration, and the outcome of this term will be placed in (2.17). To determine $\mathbf{v}_r$ in the PBVS method the same scenario for the IBVS method will happen as follows:

$$\dot{\mathbf{s}_p} = \mathbf{L}_{Pxy}\mathbf{v}_{xy} + \mathbf{L}_{Pr}\mathbf{v}_r \tag{2.19}$$

where $\mathbf{L}_{Pxy}$ is a matrix generated by the first and second columns of $\mathbf{L}_P$ in (2.4), and $\mathbf{L}_{Pr}$ is a matrix created by the last four columns of $\mathbf{L}_P$ in (2.4). Therefore:

$$\mathbf{v}_r = \mathbf{L}_{Pr}^{+}(-k(||\mathrm{e}||)\mathbf{e}_p - \mathbf{L}_{Pxy}\mathbf{v}_{xy}) \tag{2.20}$$

Finally, the camera velocity vector is calculated by solving (2.17) and (2.20), simultaneously.

2.3.2 Robot kinematics with task priority

Joint velocities ($\dot{\mathbf{q}}$) are computed after the end-effector (EE) velocities are calculated using the kinematics of the robot:

$$\dot{\mathbf{q}} = \mathbf{J}^{\dagger\lambda}\boldsymbol{\xi}_c^e\mathbf{v}_{cam}^c \tag{2.21}$$

The transformation matrix $\boldsymbol{\xi}_c^e$ is used to map the velocities represented in the robot end-effector (EE) frame to the camera frame [103]:

$$\boldsymbol{\xi}_c^e = \begin{bmatrix} \mathbf{R}_c^e & sk\left(\mathbf{t}_c^e\right)\mathbf{R}_c^e \\ 0 & \mathbf{R}_c^e \end{bmatrix} \tag{2.22}$$

where $\mathbf{t}_c^e$ is the translation vector between the EE frame and the camera frame. $\mathbf{R}_c^e$ is the rotation matrix between the EE and the camera frame, and $sk(\mathbf{t}_c^e)$ is the skew-symmetric matrix of the translation vector. It is worth mentioning that $\boldsymbol{\xi}_c^e$ is constant in such a scenario (Eye-in-Hand configuration). By using this approach the effect of robot singularity has been reduced, and discontinuities caused by the decoupling process have been greatly smoothed, thanks to the DLS inverse [104]:

$$\mathbf{J}^{\dagger^{\lambda}} = \mathbf{J}^T\left(\mathbf{J}\mathbf{J}^T + \lambda^2\mathbf{I}\right)^{-1} \tag{2.23}$$

where λ is a positive scalar known as the damping factor. Using DLS inverse minimizes the term $||\mathbf{J}\dot{\mathbf{q}} - \dot{\mathbf{x}}||^2 + \lambda^2||\dot{\mathbf{q}}||^2$. Choosing λ will ensure that the solution norm stays within an assigned range [105]. It is worth noting that regularisation techniques help with reducing the effect of singularity configurations. In addition, they increase the convergence time [106]. In this study, the task priority for the given tasks (i.e., feature error convergence and robot manipulability) is calculated [104], and given by:

$$\dot{\mathbf{q}} = \mathbf{J}^{\dagger\lambda}\xi_c^e \mathbf{v}_{cam}^c + (\mathbf{I} - \mathbf{J}^{\dagger\lambda}\mathbf{J})\dot{\mathbf{q}}_0 \tag{2.24}$$

where $\mathbf{I} - \mathbf{J}^{\dagger\lambda}\mathbf{J}$ is the Null space projection matrix. Therefore, the closest point in the Jacobian matrix null space will be identified, satisfying both tasks. $\dot{\mathbf{q}}_0$ is defined as follows [45]:

$$\dot{\mathbf{q}}_0 = k_0 \left(\frac{\partial w(q)}{\partial q}\right)^T \tag{2.25}$$

In (2.25), k_0 is a positive gain and w is the cost function for another task. w should be maximized to consider the other objectives. As mentioned earlier, the manipulability of the robot has been considered as the second task.

$$w(q) = \sqrt{\det\left(\mathbf{J}(q)\mathbf{J}^T(q)\right)} \tag{2.26}$$

Using the classical projection operator defined in [45], the secondary task will be computed and will be added to the joint velocity vector. The value of $w(q)$ is called manipulability value and shows the functionality of the robot in each configuration. The more the amount of m, the better adjustment in the workspace is possible (greater range of possible motions) [107]. The proposed visual servoing control schema before using LoLiMOT NN is depicted in Figure 2.2. The red blocks represent the image-based controller, the blue blocks represent the position-based controller, and the grey blocks represent algorithms in the task space. As shown in the control block diagram in Figure 2.2, the camera velocities have been decoupled; in 2D, two of them (translations in X and Y) were considered by using the features created from the image screen, as feedback. The remaining components were modelled in 3D (computed by partial 3D reconstruction of the environment attained by the extracted features). Following that, the computed velocities will be given to the robot. The controller will exchange the desired camera velocities with the desired joint velocities using the Jacobian of the robot.

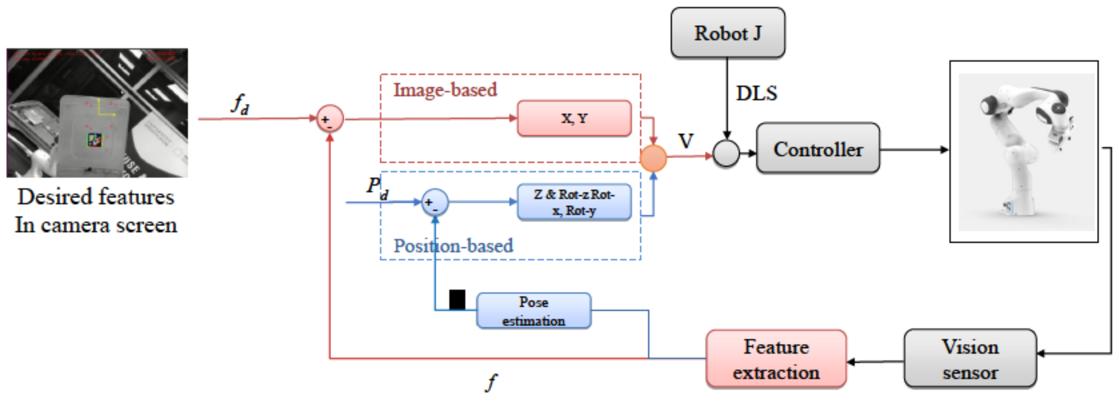


Fig. 2.2. The control schema of the proposed visual servoing approach before applying LoLiMOT NN.

To minimise the effects of robot singularity, the DLS inverse method was used instead of the pseudo-inverse approach. The 3D calculation of the visual features is used to regulate the errors of rotations and translation in the Z-axis. As a result, feasible trajectories for the robot will be generated, and image singularities caused by these four components in the interaction matrix will be eliminated. In Algorithm 1, the pseudocode of the proposed approach has been illustrated.

Algorithm 1: Hybrid Decoupled Visual Servoing.

```
1  Inputs: $\mathbf{e}, \mathbf{e}_p$;
2  Outputs: $\dot{\mathbf{q}}$;
3  Initialization;
4  Define desire points;
5  Modify task as Eye-in-Hand;
6  while Not converged do
7      for i=1 to i=features.size() do
8          Compute $\mathbf{L}_{xy}$, $\mathbf{L}_r$, $\mathbf{L}_{Pr}$ and $\mathbf{L}_{Pxy}$
9      end
10     for j=1 to j=camera-velocity.size() do
11         compute camera velocities $\mathbf{v}_{cam} = (\mathbf{v}_{xy}, \mathbf{v}_r)$;
12         use adaptive gain;
13         IBVS update visual features;
14         $\mathbf{v}_{xy} = \mathbf{L}_{xy}^{+}(-k(||\mathbf{e}||)\mathbf{e} - \mathbf{L}_r \mathbf{v}_r)$;
15         PBVS update visual features;
16         $\mathbf{v}_r = \mathbf{L}_{Pr}^{+}(-k(||\mathbf{e}||)\mathbf{e}_p - \mathbf{L}_{Pxy}\mathbf{v}_{xy})$;
17     end
18     for k=1 to k=joint-velocity.size() do
19         calculate joint velocities $\dot{\mathbf{q}}$;
20         define manipulability as a secondary task;
21         use damped least square method;
22         command robot;
23         $\dot{\mathbf{q}} = \mathbf{J}^{\dagger\lambda}\xi_c^e \mathbf{v}_{cam}^c + (\mathbf{I} - \mathbf{J}^{\dagger\lambda}\mathbf{J})\dot{\mathbf{q}}_0$;
24     end
25 end
```

The inputs of the algorithm are the feature errors in the image screen and their counterparts in the 3D space. The output of the algorithm will be the vector of joint velocities. In line 8 of Algorithm 1, the decoupled matrices will be determined from (2.14), (2.15), and (2.4). Camera velocity would be estimated in lines 14 and 16 using the calculated decoupled matrices in line 8. An adaptive representation of the controller gains was used in this calculation to increase the VS task speed. Eventually, the robot joint velocities will be measured and commanded to the robot velocity controller in line 23 of Algorithm 1. Not to mention that the controller would use manipulability as a secondary task in measuring the joint velocities, and DLS inverse would be used instead of pseudoinverse to convert EE velocities to the joint velocities.

2.3.3 Estimation of camera velocities using Neural Networks

A trained neural network is employed to provide an accurate estimation of the camera velocities from the feature errors (NN replaced with lines 7 to 17 in Algorithm 1). To this end, feature errors in the image screen were deployed as input of the neural network and the calculated

camera velocities were defined as the outputs. A Local Linear Model Tree (LoLiMoT) neural network was used for this purpose.
The approximated equations by the NN are the combination of the right-hand side of the Equation 2.17 (including pseudo-inverse of L_{xy}, to calculate translational velocities in X and Y-axis from 2D image errors) and the right-hand side of the Equation 2.20 (including pseudo-inverse of L_{Pr} to calculate the translational velocity in Z-axis and three rotational velocities from estimated 3D errors of the image features). To this end, 76000 sets of unique feature errors with their relevant camera velocities (calculated from the achieved Equations number 2.17 and 2.20) all over the image screen (480*640 pixel) for the test and training data have been gathered. These results were taken by moving the camera manually and trying to include the feature's positions all around the image screen as best as possible. In Figure 2.6, the positions of the current features have been shown in the camera screen with green. To train and validate the neural network, the data was split into training, validation, and test sets, with 70% used for training and 30% for testing. This division was done to ensure that the model could generalize well to unseen data. In comparison to other neuro-fuzzy NNs, LoLiMoT is more efficient in learning non-linear systems with fewer neurons [108].

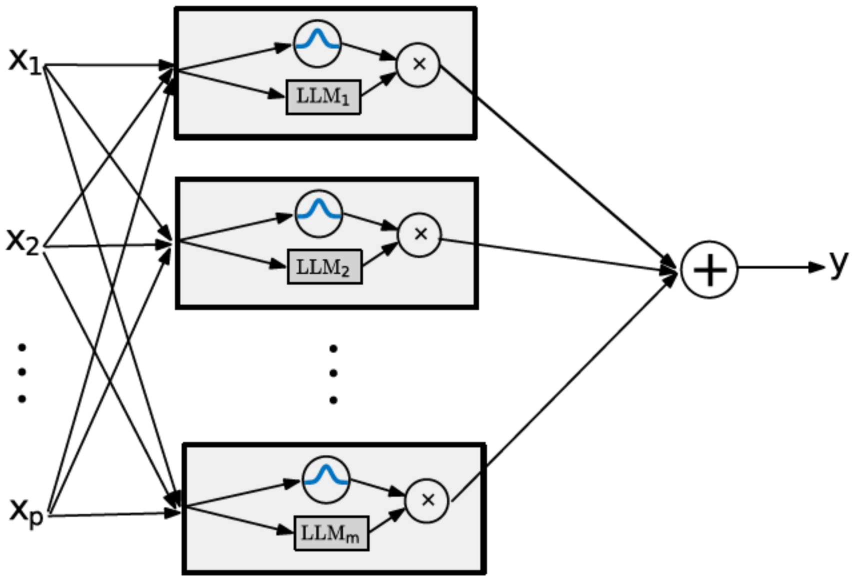


Fig. 2.3. The structure of LoLiMoT NN with m neuron and P input.

The LoLiMoT algorithm operates by the errors and it is axis-orthogonal. Therefore, the network divides the space in a properly optimized way w.r.t the errors. Other types of neural networks, which work by the use of gradient descent, operate blindly and can not guarantee global optimization [109]. The structure of the LoLiMoT network is depicted in Figure 2.3. A linear local model alongside a membership function is assigned to each neuron. For the allocated linear model, the validated area would be determined with an assigned membership function. The linear model is formulated as follows [110]:

$$\hat{y}_i = \omega_{i0} + \omega_{i1}x_1 + \ldots + \omega_{ip}x_p \tag{2.27}$$

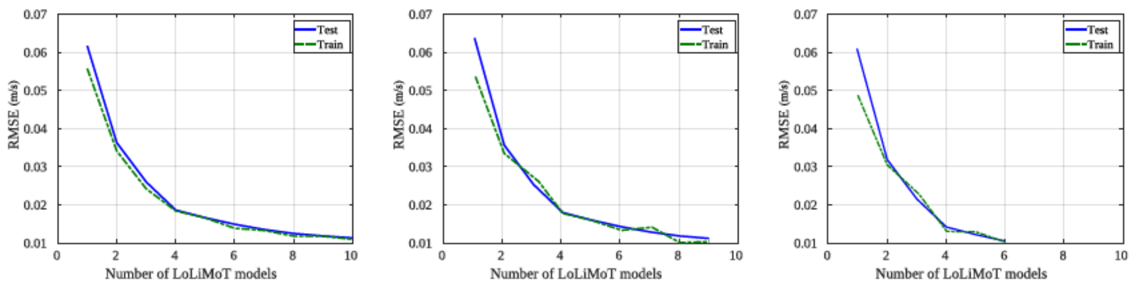
In equation 2.27, ω_{ij} is the associated parameters to i_{th} neuron. The final output is calculated as follows:

$$\hat{y} = \sum_{i=1}^{M} \hat{y}_i \phi_i \left(\mathrm{x}\right), \qquad \sum_{i=1}^{M} \phi_i \left(\mathrm{x}\right) = 1 \tag{2.28}$$

Where $\phi_i\,(\mathrm{x})$ stands for the membership function which is assumed as a normalized Gaussian function in Eq. 2.28. The corresponding member function is derived by the following equation:

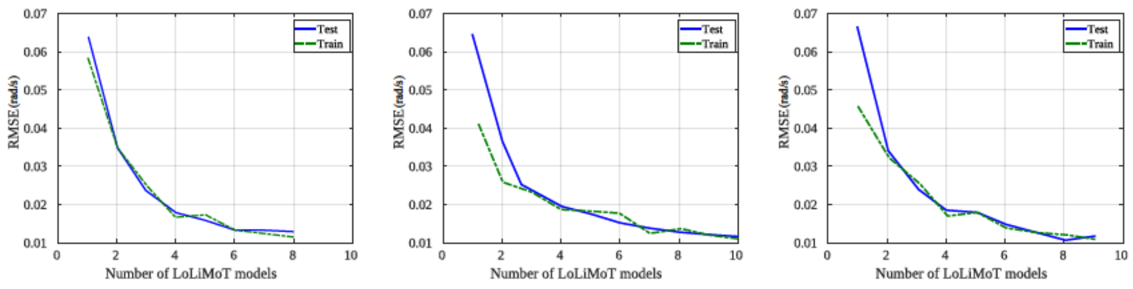
$$\begin{aligned} \phi_i\,(x) &= \frac{\mu_i(x)}{\sum_{j=1}^{M}\phi_j(x)} \\ \mu_i\,(x) &= e^{\frac{(u_1-c_{i1})^2}{-2\sigma_{i1}^2}} \times \ldots \times e^{\frac{(u_p-c_{ip})^2}{-2\sigma_{ip}^2}} \end{aligned} \tag{2.29}$$

In equation 2.29, c_{ij} and σ_{ij} are the centre of the area and the standard deviation, respectively. The NN has been trained for each output separately (6 camera velocities).



(a) RMSE of V_x **(b)** RMSE of V_y **(c)** RMSE of V_z

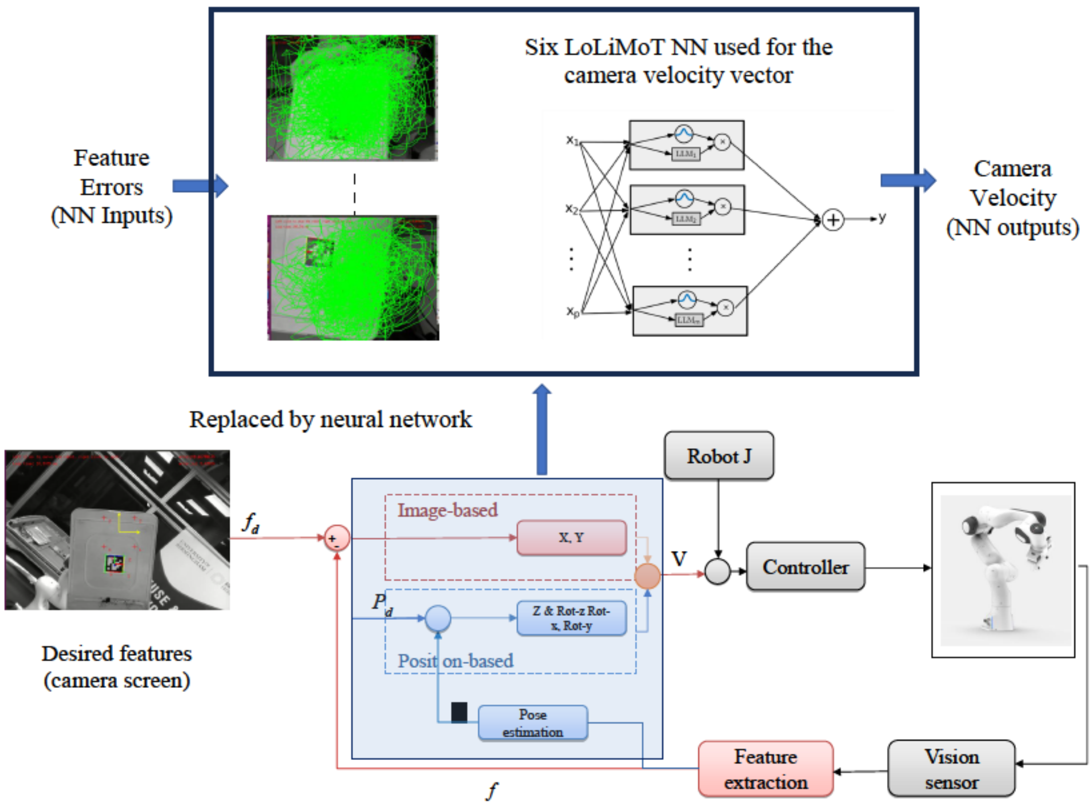
Fig. 2.4. RMSE of test and train samples of LoLiMoT neural network for translational camera velocity outputs.



(a) RMSE of ω_x **(b)** RMSE of ω_y **(c)** RMSE of ω_z

Fig. 2.5. RMSE of test and train samples of LoLiMoT neural network for rotational camera velocity outputs.

As illustrated in Figure 2.6, the pseudo-inverse of the decoupled image-Jacobian and pose estimation are replaced with six NNs to predict the camera velocity vector at each iteration. The feature error vector will be computed by subtraction of current and desired features. Control input has been defined by eight components of the error vector, consequently, the output would be the camera velocity vector with six components. This camera velocity vector will be transformed to the EE frame of the robot using Eq. 2.22. Eventually, joint velocities were computed using the robot's Jacobian, and they were commanded to the actuators of each joint. The control loop will be continued until feature errors converge to zero.
Decoupling the image-Jacobian and using the LoLiMOT method would solve the image singularity and local minima problem in the image-Jacobian. Since the networks are trained with the calibrated parameters, therefore the controller will be highly robust in terms of calibration errors. Moreover, supplementary noise has been added to the neural network input data set during training to improve its robustness to the image noises. The added Gaussian noise has a standard deviation of one and a mean of zero (white noise), generated by the



Six LoLiMoT NN used for the camera velocity vector
Feature Errors (NN Inputs)
Camera Velocity (NN outputs)
Replaced by neural network
Robot J
Image-based
X, Y
V
Controller
f_d
P_d
Z & Rot-z Rot-x, Rot-y
Position-based
Pose estimation
Desired features (camera screen)
Feature extraction
Vision sensor
f

Fig. 2.6. The control schema of the proposed visual servoing (HDVS) integrated with LoLiMOT NN.

pseudo-random number generator. Using NN, the complexity of 3D estimation of the target position and pseudo-inverse of the decoupled image-Jacobian have been relaxed, as described in subsection 2.3.1.
For each output, one LoLiMoT network has been used. Therefore, six neural networks have been trained, coming after six camera velocities. Each network has a different number of LoLiMoT models. In LoLiMoT, the performance of the proposed model is evaluated using the Root Mean Square of Errors (RMSE). RMSE is a commonly used metric that quantifies the difference between values predicted by the model and the actual observed values:

$$\text{RMSE} = \sqrt{\frac{1}{n_{obs}} \sum_{i=1}^{n_{obs}} (xp_i - xa_i)^2} \tag{2.30}$$

where, n_{obs} is the number of observations, xp_i is the predicted value for the i-th observation, and xa_i is the actual (reference) value for the i-th observation. In our analysis, RMSE is computed separately for both the training data and the testing data. The RMSE for the training data indicates how well the model fits the data it was trained on. On the other hand, the RMSE for the testing data assesses the model's generalization ability to new, unseen data. The number of neurons in each output component is determined through trial and error. The approach involves training the network with an initially assigned number of neurons. If the RMSE of the test process decreases as the network trains, the training is considered successful. The number of neurons is then adjusted through multiple trials, and the process is repeated until the configuration that results in the lowest RMSE is found. It was also observed that

increasing the number of neurons sometimes caused the RMSE to jump suddenly at certain points. These jumps indicate that the neural network is starting to overfit, or "overlearn," the training data. This is a key sign that the network has too many neurons for the task and that the number should be reduced to avoid overfitting. This method ensures that the network is neither underfitted (too few neurons) nor overfitted (too many neurons). The learning process stops when the test and train samples acquire a predefined accuracy. This accuracy will be checked by the RMSE in both test and train samples (RMSE of 0.01m/s for translational velocities and RMSE of 0.01rad/s for rotational velocities). Such behaviour will effectively decrease the number of trials and errors.
The number of allocated LoLiMoT models (membership functions alongside linear local model) and RMSE of the trained Neural network for the test and the train samples have been depicted in Figure 2.4 and 2.5. Each LoLiMoT model consists of a linear local model alongside a Gaussian membership function.
As Figure 2.4a suggested, having 10 models is optimal to have the RMSE of 0.01 in test and train samples. Having more than 10 models is redundant and could lead the network to over-fitting. The number of LoLiMoT models is 9 for the prediction of velocity controller in Y direction (Figure 2.4b). This number is 6 for velocity in Z (Figure 2.4c).
In Figure 2.5a, it can be seen that 8 LoLiMoT models are required for rotational velocity about X-axis, 10 models for rotational velocity about Y-axis (Figure 2.5b) and 9 LoLiMoT models for rotational velocity about Z-axis (Figure 2.5c). The trained set of these 6 networks will be used to predict the camera velocity required to perform the visual servoing task. The updated set of feature errors will be utilized for the networks at each iteration.

2.4 | Simulation and experimental setup

In this study, two different setups were introduced to evaluate the efficacy of the proposed HDVS method in comparison with other VS methods. In the first setup, different behaviours of the proposed method have been studied such as singularity, performance, and manipulability (Figure 2.7). The second setup was also designed to demonstrate the capability of the proposed DHVS method for performing an industrial application (i.e., sorting) (Figure 2.9). It is worth mentioning that all the case studies were performed with the same adaptive and DLS gains ($k_0 = 4, k_\infty = 0.4, k'_0 = 30,\ \lambda = 0.1$).

2.4.1 Design of setup 1

In the first setup, the HDVS method has been tested in simulation and then validated in the real world. The simulation platform includes two Franka manipulators; one robot is equipped with an RGB-D camera, and another robot arm holds an object (with a printed tag marker). Four corners of the marker are used as points of interest/visual features in this study. Tracking the object under this condition has been tested via different VS methods. The experimental and simulation setups are identical. Franka robots have 7 degrees of freedom across 7 axes, with a 3 kg payload, and positioning accuracy of $+/-$ 0.1 mm in all directions. The proposed method was modeled in simulation using ROS/Gazebo. ROS Melodic on Linux 18 was used for both the simulation and the experiment. The joint state controller was used to publish the joint state (at a rate of 1 kHz) and the joint velocity group controller was used to set the joint velocities computed from the VS approaches. A system with the following CPU specification was used for the visual servoing operation: AMD Ryzen 7 3700x, 8-core with 16 CPU Threads, 3.6 GHz base clock, and 36 MB total cache. Figure 2.7a depicts an snapshot of the developed simulation environment in Gazebo, and Figure 2.7b shows the identical experimental setup in the real world.

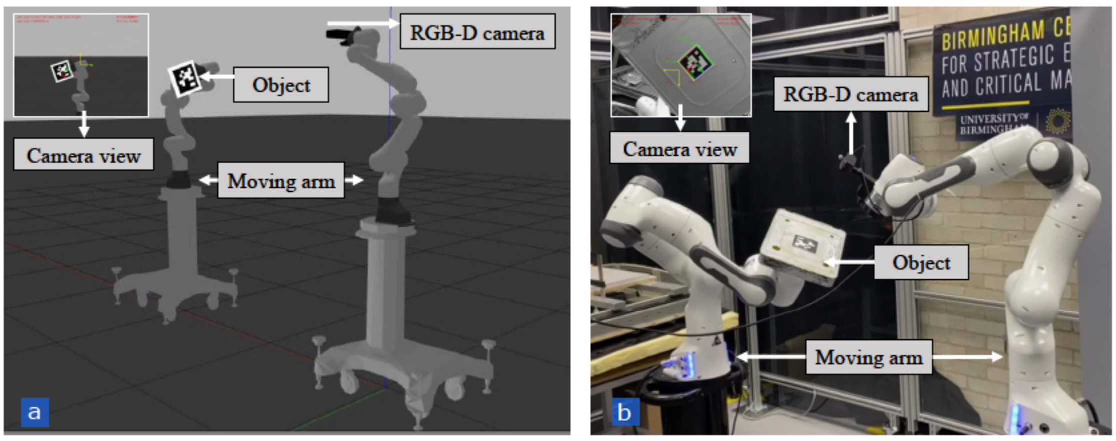


Fig. 2.7. The modelled setup 1 in (**a**) the simulation environment, and (**b**) the real world.

Figure 2.8 also illustrates the ROS schematic graph, including ROS nodes and ROS topics to communicate between these nodes. The joint-states topics provide the position and velocity of each joint from the robot models simulated in Gazebo. Using a ROS Handler this information will be published to subscribe and use in each node (vs-node and moving-marker). The provided information will be used as feedback in robots' controllers. The velocity input will be commanded to the simulated robots by using the panda-arm-controller/command topic. Not to mention that the camera information will be acquired from the Gazebo or real robot node to the visual servoing node by camera/image-raw topic.

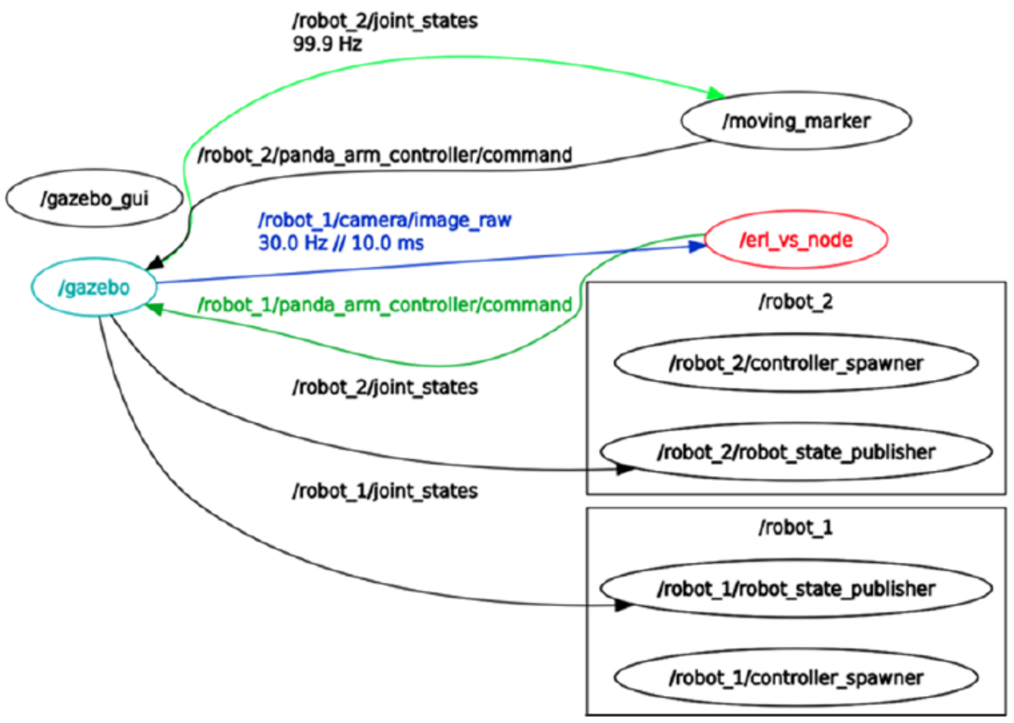

/robot_2/joint_states
99.9 Hz
/moving_marker
/robot_2/panda_arm_controller/command
/gazebo_gui
/robot_1/camera/image_raw
30.0 Hz // 10.0 ms
/erl_vs_node
/gazebo
/robot_1/panda_arm_controller/command
/robot_2
/robot_2/controller_spawner
/robot_2/joint_states
/robot_2/robot_state_publisher
/robot_1/joint_states
/robot_1
/robot_1/robot_state_publisher
/robot_1/controller_spawner

Fig. 2.8. Considered ROS topics to communicate between ROS nodes.

By using Setup 1, three different case studies have been designed to compare different behaviours of the HDVS method such as singularity, performance, and manipulability with other VS methods.

Case Study 1

To demonstrate the effectiveness of the proposed method, this case study has been designed in which the robot with the attached marker moved to a pre-defined position, and another robot, equipped with the camera, tracked the visual features. This case study shows scenarios in which the proposed method could complete the task while other methods are unsuccessful to do so. The camera calibration was intentionally degraded by at least 20% in all of the case studies to evaluate the performance of HDVS in an imperfect calibrated condition.

Case Study 2

In this case study, a comprehensive comparison between the effective parameters in VS is carried out. Ten random positions were defined for the object (with an attached marker) in order to be tracked by another robot (with the attached camera) using different VS methods. The experiments are performed under the same conditions for all four VS approaches (IBVS, PBVS, HVS, HDVS). Thereafter, the performance of the robot and the VS methods in the image space and Cartesian space were compared quantitatively based on the RMSE, range of feature error, required number of iterations for convergence, and the manipulability of the robot.

Case Study 3

In this case study, the object is not fixed (the opposite of Case Study 1 and Case Study 2) and the controller will track a dynamic object to demonstrate how HDVS could improve the manipulability of the robot. The selected trajectory includes a wide range of rotations and translations.

2.4.2 Design of setup 2

The second setup was designed to evaluate the efficacy of the proposed HDVS method for sorting applications. Figure 2.9 depicts the experimental setup used for sorting the dismantled components of an EV battery pack (Nissan Leaf 2011). The VS has been used to guide the suction gripper toward the battery module (i.e., the object with an attached marker). Then, the battery module will be lifted and will be placed in the corresponding basket. The rationale for performing this case study is that in industry, the robot and the object that the robot is interacting with must be precisely positioned, otherwise, the robot might fail to complete the task due to uncertainties in the environment.

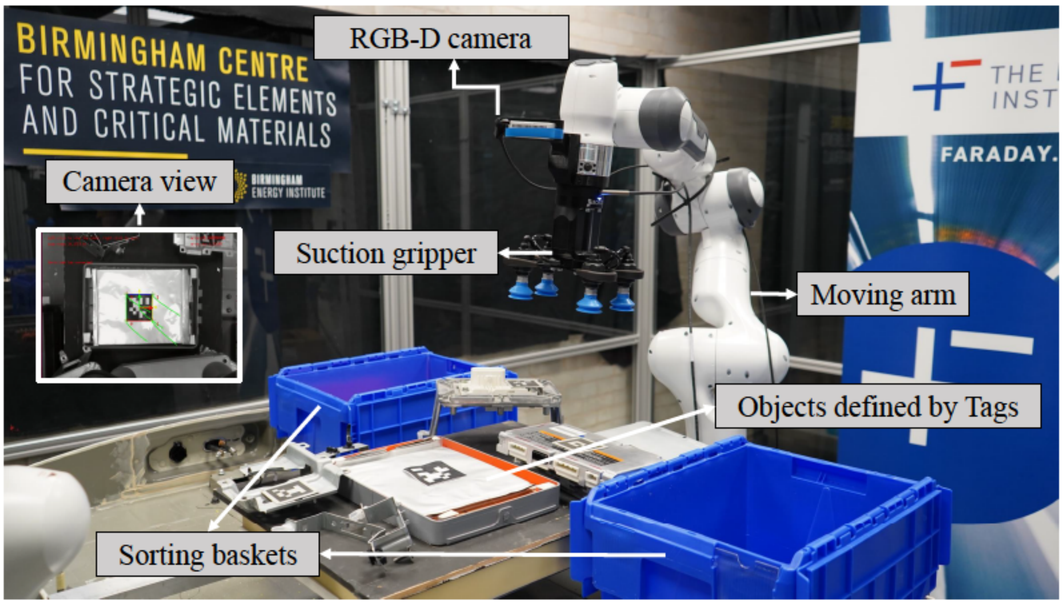


Fig. 2.9. The automated process of sorting the dismantled EV battery components introduced in Setup 2. The object tracking is carried out by the proposed VS method (HDVS) and the object is manipulated using a redundant manipulator arm.

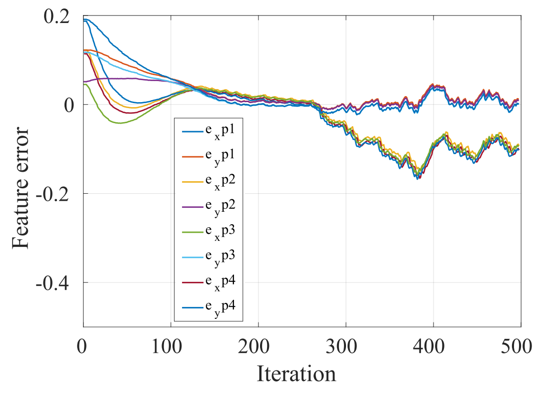
2.5 | Results and discussion

Various scenarios have been studied and compared the proposed HDVS approach with HVS, IBVS, and PBVS approaches. To have a logical comparison, the same adaptive gains have been used for every method ($k_0 = 4, k_\infty = 0.4, k'_0 = 30$). The λ gain in DLS inverse is set to 0.1. Moreover, each iteration of the control loop was designed to be completed within a fixed time interval (0.025s), representing the maximum allowable time for all necessary computations. This fixed iteration time was chosen to reflect the constraints of real-time control systems, where maintaining a consistent update rate is essential for ensuring the robot's responsiveness to environmental changes. Using a fixed iteration time, we ensured all methods were tested under the same time conditions. This way, the number of iterations needed to reach a solution shows how fast each method is, given the same amount of time per iteration.

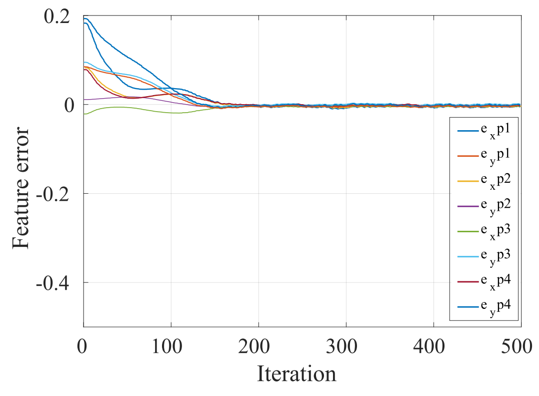
2.5.1 Case 1: Analysis of unsuccessful VS methods, setup 1

Comparing the performance of PBVS with HDVS

A case has been defined to follow a pre-defined position of the target with an uncalibrated camera. The performance of HDVS and PBVS in regulating the errors is illustrated in Figure 2.10. In Figure 2.10a, it is shown that the position-based controller has failed to track the features, and the controller could not follow the desired features from iteration 350. This failure comes from the fact that there is an error in the 3D estimation of the target generated by an uncalibrated camera.
The reason that PBVS is not robust to the camera calibration is that the errors in camera parameters propagated to errors in the 3D estimation of the target. However, HDVS is robust in terms of camera parameters. Figure 2.10b illustrates the feature errors on the y-axis against the number of iterations on the x-axis. Iteration in the plots refers to a single execution cycle of the main control loop. In the VS, each iteration represents one complete pass through the loop, during which the system acquires an image, detects features, computes errors, applies the control law, and updates the robot's velocities. Not to mention that the feature errors



(a) Uncalibrated camera in PBVS



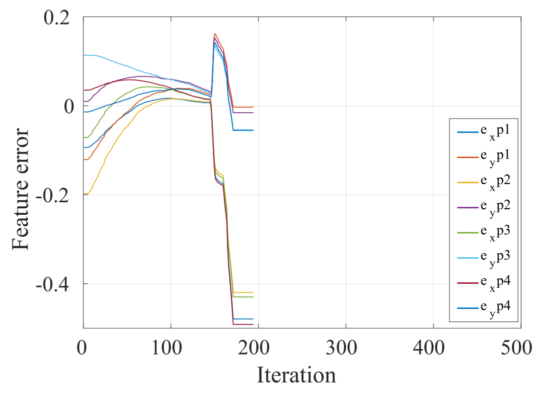
(b) Uncalibrated camera in HDVS

Fig. 2.10. Result of tracking desired features with an uncalibrated camera for PBVS and HDVS.

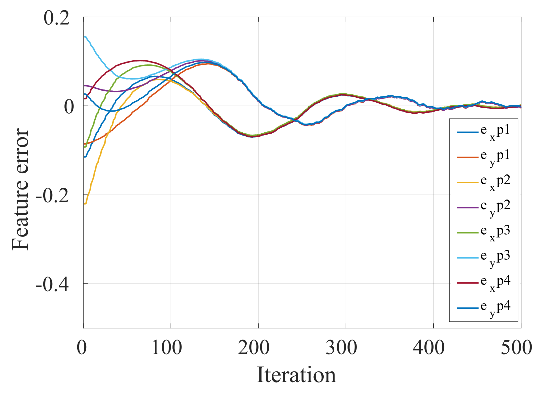
were represented in normalized pixels, scaling the pixel coordinates relative to the 640×480 image dimensions. This normalization ensures consistent error measurements across different image resolutions. Figure 2.10b suggested that HDVS tracked the desired features successfully. This is because in HDVS camera velocities are generated directly from the trained LoLiMoT NN. Collecting data for the NN has been done with an accurate camera calibration. Therefore, it is not important how inaccurate is the camera calibration in online mode (i.e. robot during visual servoing), the controller will work well and is more robust to camera calibration errors compared to PBVS.

Comparing the performance of IBVS with HDVS

IBVS failures are mostly associated with the rotations about Z-axis (Chaumette Conundrum) [34] and Camera retreat [35]. Furthermore, camera ambiguities might happen between translations and rotations in the image plane; this implies that different camera velocities could produce the same motion of points in the image and some camera motions lead to no change in the image [111]. A case study has been defined in such that the desired features rotated 90° about the Z-axis. The performance of HDVS and IBVS in converging feature errors to zero is depicted in Figure 2.11.



(a) IBVS



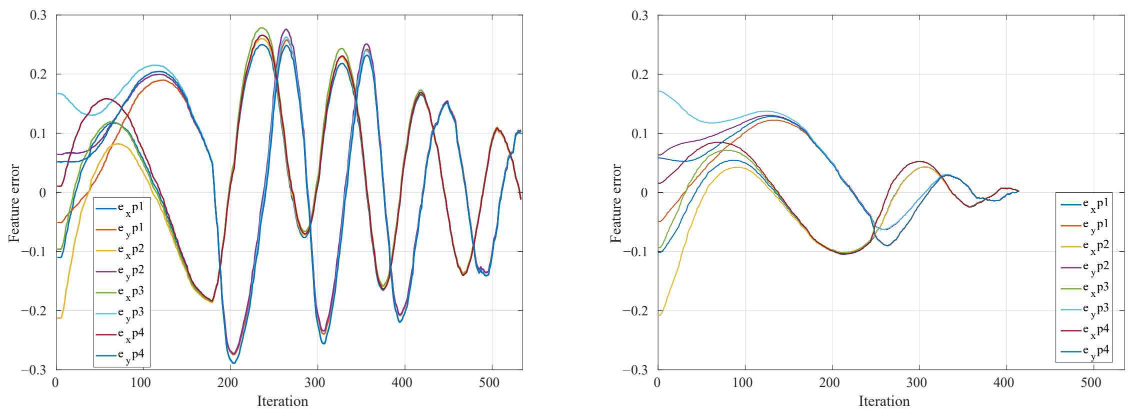
(b) HDVS

Fig. 2.11. Result of tracking desired features with an uncalibrated camera for IBVS and HDVS.

As shown in Figure 2.11, the controller will end up with unpleasant behaviour in IBVS. Figure 2.11a shows that the errors could not converge to zero as the controller produces a large velocity in Z which makes the robot go to its joint limits. The main reason associated with this limitation is that reducing the rotation error about the Z-axis in the image screen could be achieved by moving the camera away from the target (the so-called camera retreat phenomenon). Such a wrong decision in 2D could produce a large motion in the Z axis in 3D. An extreme version of this behaviour is when there is a pure π rad rotation between features and their desired positions in the image. In such a case, the features will be driven towards the origin by mistake and cause image singularity [34]. In this configuration, the controller would servo forever, and features can not reach the desired ones [35]. The proposed HDVS method will provide the most straightforward way of compensating errors of Z and rotations in task space. This claim comes from the fact that the error of these four parameters (translation in Z-axis and three rotations) is directly compensated by 3D estimation of the target position w.r.t the camera position. In Figure 2.11b, it is illustrated that using the proposed controller the errors easily converge (in 509 iterations) without causing singularity. Moreover, HDVS avoids camera ambiguity while the controller distinguishes the camera rotations from translations. Camera ambiguity could be explained by the structure of the image-Jacobian matrix in Eq. 2.1. When the camera focal length is large or when the pixel coordinates are small, columns 1 and 4 become very similar. The same ambiguity could happen for columns 2 and 5 (large focal length dominates u squared). Therefore, transnational velocities in the X axis and Y axis could not be easily detected from rotational velocities about the Y and X axis, respectively. These components are decoupled from the image-Jacobian in HDVS, and the controller avoided these kinds of ambiguities.

Comparing the performance of HVS with HDVS

In the traditional HVS approach, the controller compensates for translation in Z-axis from the 2D information given by the camera. As a result, the controller continues to generate a non-optimized trajectory for the camera, which results in generating robot singularity. Figure 2.12a clearly shows that the controller using the HVS method failed to converge the errors to zero with an uncalibrated camera. However, HDVS could successfully finish the tracking in Figure 2.12b. Not to mention that the overshoots in Figure 2.12b are caused by the uncalibrated camera parameters, but the key point is that the controller still managed to complete the task successfully. Another advantage of HDVS in comparison with traditional



(a) HVS **(b)** HDVS

Fig. 2.12. Result of tracking desired features with an uncalibrated camera for HVS and HDVS.

HVS methods is the significant reduction in computation costs. The computing and modelling process (Eq. 2.10 and 2.11) in traditional HVS for each iteration is time-consuming and complex. However, in HDVS neuro-fuzzy NNs are used for estimating the required velocities. Since the NNs are trained offline and used online as a predictor of required velocities, the convergence time will be reduced considerably. Moreover, HVS methods need homography construction and decomposition. Homography construction is sensitive to image noises and there exists sign ambiguity in homography decomposition (non-unique solutions). However, HDVS is robust to image noises, and globally optimized solutions will be achieved from the LoLiMoT network.
Figure 2.13 compares the success rate of four VS methods for 10 different random paths in simulation and their counterparts in the real world. From Figure 2.13, it can be concluded that

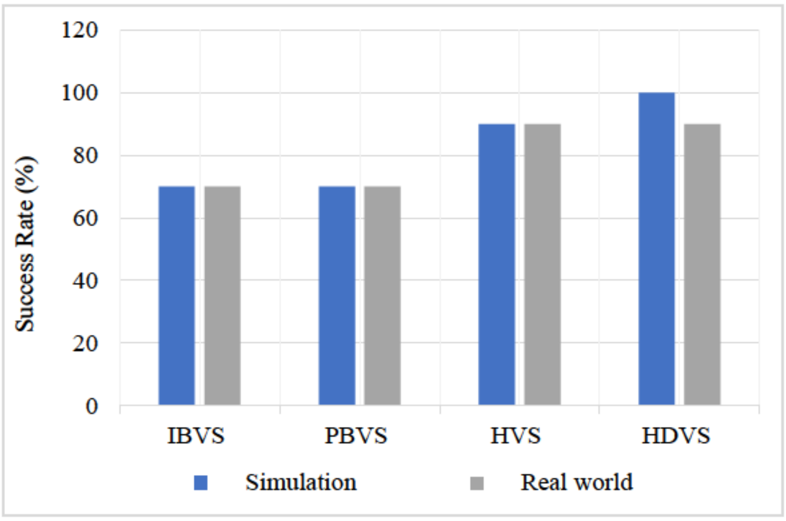


Fig. 2.13. The success rate of four VS methods over 10 trials in simulation and real world

HDVS outperformed with its success rate in both simulations (with 10 out of 10 successful trials) and the real world (with 9 out of 10 successful trials) compared to other VS methods. In summary, Figures 2.10, 2.11, and 2.12 suggest that the proposed HDVS approach could avoid the singularity, mitigate the discontinuities, and complete the VS tasks without the use of complex and time-consuming methods, while the other VS methods failed to complete the task successfully.

2.5.2 Case 2: Performance analysis of VS methods, setup 1

In this section, the effective parameters during tracking 10 positions by four different VS methods are compared with each other quantitatively and the results are tabulated in Tables 2.1–2.3.
According to Table 2.1, the HDVS method has a lower mean RMSE than PBVS and HVS, and the range of feature errors in PBVS and the classical HVS are greater than that value in the HDVS. As a result, using the HDVS, the object is less likely to be missed from the camera screen than in the other two approaches. Not to mention that the proposed method is quicker than traditional HVS, and needs fewer iterations to complete the same mission. Table 2.1 also illustrates that the IBVS method not only has the smallest range of feature error but also the smallest RMSE as compared to other approaches. Without a doubt, IBVS performs better in

Table 2.1
Performance of visual servoing methods in the image space.

Method	RMSE	Feature Error Range	Iteration	Mean of Error	Mean Standard Deviation of Error
IBVS	0.0222	[−0.36, 0.310]	453	0.0152	0.0095
PBVS	0.0383	[−0.445, 0.507]	487	0.0204	0.0164
HVS	0.0273	[−0.448, 0.486]	624	0.0168	0.0141
HDVS	0.0249	[−0.419, 0.407]	505	0.0155	0.0101

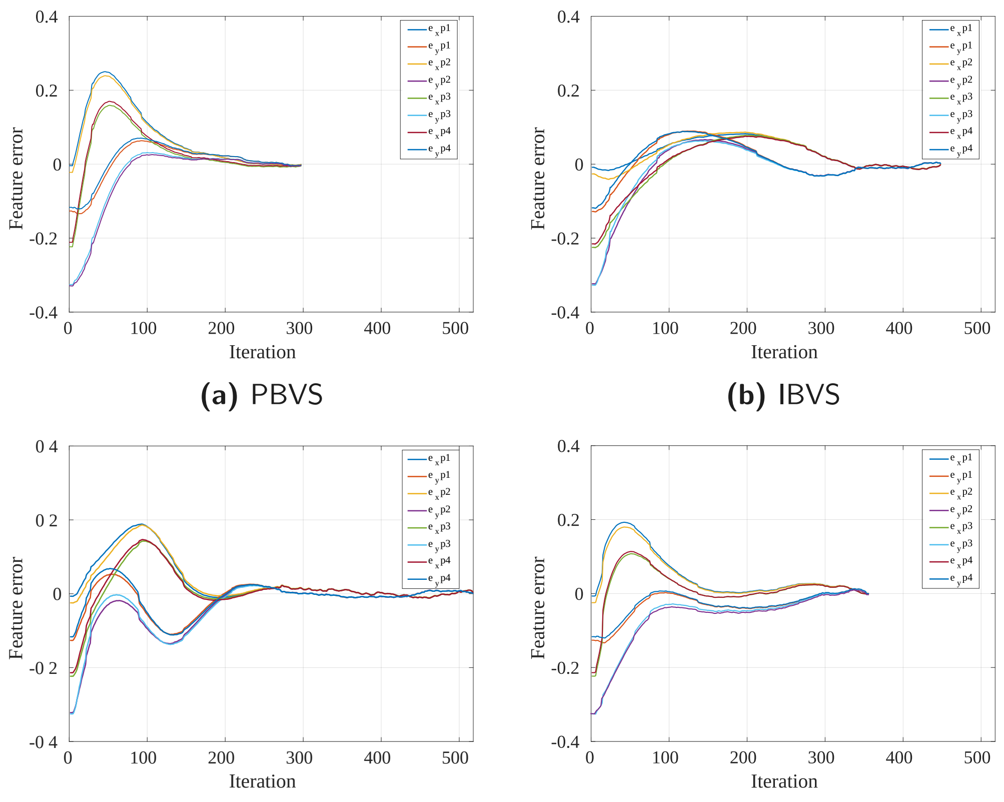
Table 2.2
Performance of visual servoing methods in the Cartesian space.

Method	RMSE of Position (m)	RMSE of Orientation (°)	Camera (or EE) Travelled Distance (m)
IBVS	0.036	9.43	0.942
PBVS	0.022	6.54	0.722
HVS	0.034	8.41	0.917
HDVS	0.027	7.11	0.746

the image-plane if there is no singularity or local minima. However, the controller operates blindly in the Cartesian space (with the highest RMSE for location and orientation as shown in Table 2.2). Large camera motions are common in the IBVS approach.
As shown in Table 2.2, in Cartesian space, the HDVS method performed better than IBVS and HVS, as one would expect. However, the mean RMSE amounts of position and orientation in the PBVS method were less than their counterparts in the other three methods. As a consequence, in Cartesian space, the PBVS method had better outcomes, followed by HDVS. Ultimately, the HDVS method indicates more optimised efficiency in the Cartesian space than IBVS and HVS (based on the amounts of mean RMSE shown in Table 2.2). Furthermore, HDVS outperforms PBVS and HVS in image space (based on the amounts of RMSE and feature error ranges, shown in Table 2.1).
In Table 2.3, a quantitative comparison of manipulability has been presented as well. When using the HDVS approach for VS, the mean of manipulability across the entire path is higher than when using the other three methods. The mean of manipulability with our proposed HDVS method after 10 different trials was 0.0486. However, for the same number of trials and the same initial position of the robot and the marker, this amount was 0.0407 for the IBVS method, 0.0446 for the PBVS method, and 0.0396 for the classical hybrid method. In conclusion, the proposed HDVS technique had advantages in terms of controllability and the ability to select a wider range of joint positions, compared to PBVS, IBVS, and HVS approaches.
As a prime example, in Figures 2.14 and 2.15, the behaviours of different VS methods for one of the experiments (i.e., tracking one of the ten positions) have been depicted in this case study.
According to Figures 2.14 and 2.15, the proposed hybrid approach will not inherently have the best performance in tracking the features in the image-space and Cartesian space (robot space), but it has an optimised performance in both.
It stems from the fact that translation in the X-axis and Y-axis of the camera velocities are computed directly from the image space, while the rest are computed from the 3D

Table 2.3
Comparison of manipulability in different VS methods.

Method	RMSE	Manipulability Mean	Manipulability Range	Iteration
IBVS	0.0222	0.0407	[0.0140, 0.0810]	153
PBVS	0.0383	0.0446	[0.0245, 0.0807]	187
Hybrid VS	0.0273	0.0396	[0.0208, 0.0806]	224
HDVS	0.0247	0.0486	[0.0278, 0.0816]	203

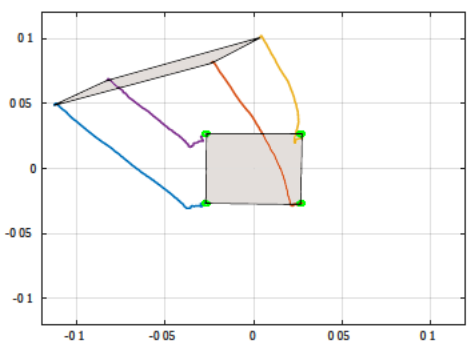


(a) PBVS **(b)** IBVS

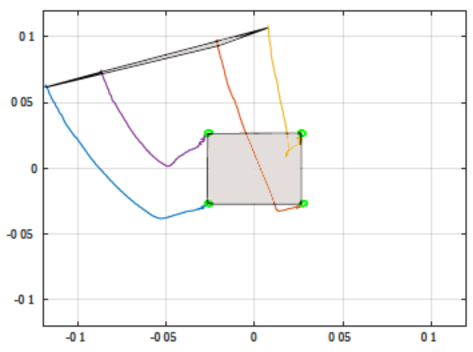
(c) HVS **(d)** HDVS

Fig. 2.14. The real world result of the feature errors in different methods for the same scenario.

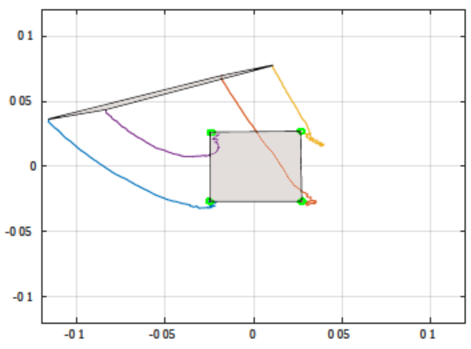
reconstruction of the environment. Furthermore, as compared to HVS and PBVS, the object in HDVS is less likely to be lost from the camera FOV. This conclusion was reached by comparing the maximum feature error in Figure 2.14d to the one in Figure 2.14a and 2.14c, which is less. The larger the error, the more likely the feature might be lost from the camera FOV.
According to Figure 2.15e, HDVS (the blue path) has a shorter camera path than HVS (the purple path) and IBVS (the red path). To elaborate, the camera (or the robot EE) travelled 0.843 m distance in HDVS; however, this value is 0.942 m and 0.917 m in IBVS and HVS, respectively. In PBVS, the camera travelled distance is 0.722 m, which provides the most optimised Cartesian trajectory of the robot EE, as predicted.
The IBVS reflects the most optimised path in the camera frame as shown in Figure 2.14b, followed by the HDVS method. IBVS has a lower RMSE than the other three VS approaches. In Figure 2.14, this amount is 0.036 in PBVS, 0.021 in IBVS, 0.032 in HVS, and 0.028 in



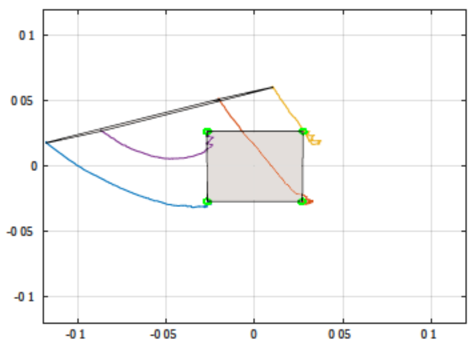
(a) IBVS in camera frame



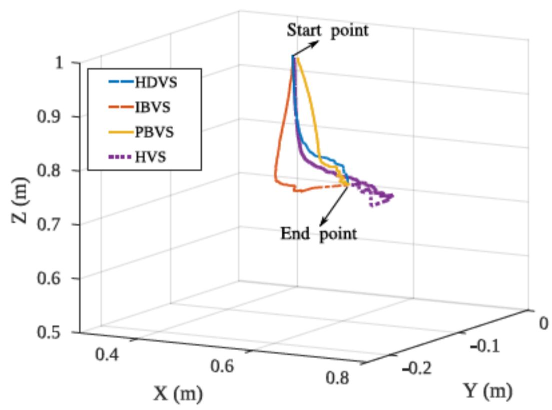
(b) PBVS in camera frame



(c) Hybrid-VS in camera frame



(d) HDVS in camera frame



(e) Comparison of EE trajectory in different methods.

Fig. 2.15. Image frame and Cartesian trajectory of EE for the same scenario in Figure 2.14.

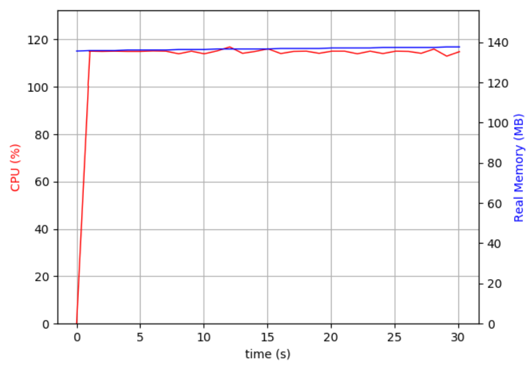
HDVS. As shown in Figure 2.14b, the maximum feature error along the entire path is 0.098, indicating that there is a very low chance of losing the object from the camera FOV. Since the PBVS controller performs blindly in the image screen, it has the most undesirable RMSE (0.036) in the camera screen. As illustrated in Figure 2.14a, PBVS has the highest feature error (0.24) compared to other approaches, therefore, it is more likely to lose the object from the camera FOV. Comparison of Figures 2.14c, and 2.14d also shows that the HDVS method is faster (converged in 300 iterations) with less RMSE amount than the HVS method (converged within 443 iterations).
A qualitative comparison of the aforementioned results for all different methods is presented in Table 2.4.

Table 2.4
A qualitative comparison of visual servoing schemes.

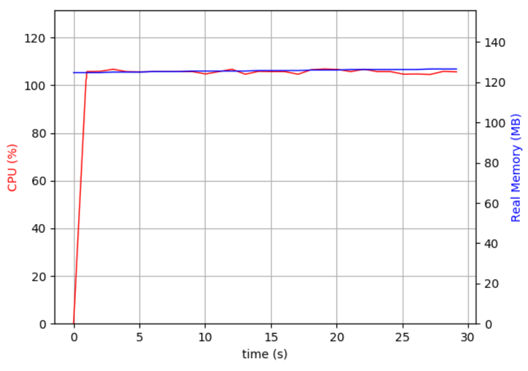
	IBVS	PBVS	HVS	HDVS
Optimized Cartesian Trajectory	low	high	med*	high
Optimized Feature Trajectory	high	low	med	med
Image Singularities	yes	no	no	no
Robustness to Camera Calibration	high	low	med	high
Convergence Speed	high	low	low	med
Computationally Complex	med	med	high	Low
Robustness to Image Noises	high	med	low	high

* med = medium

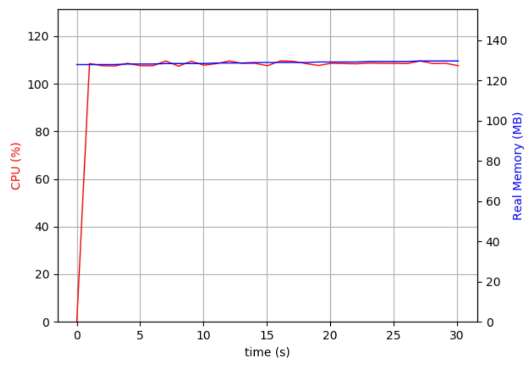
Table 2.4 illustrates that the proposed HDVS method offers an optimized trajectory both in Cartesian and image space. Furthermore, the controller is highly robust in terms of camera calibration (PBVS problem), image singularities (IBVS problem), and image noises (HVS problem). Not to mention that the proposed method has significantly relaxed the calculation, and the convergence speed is better than the classical HVS method.



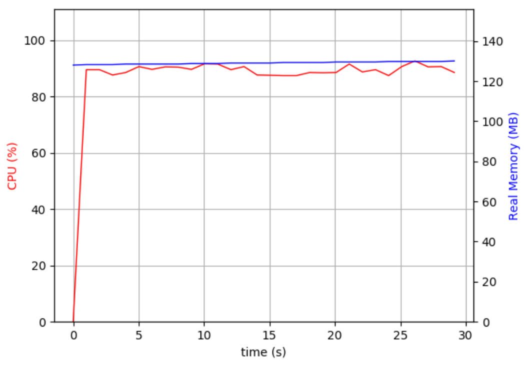
(a) PBVS



(b) IBVS



(c) HVS



(d) HDVS

Fig. 2.16. Comparison of CPU and RAM usage while each method is running in the real world

.

In Figure 2.16, the CPU and the RAM usage while different techniques are running have been shown. The comparison between the graphs in Figure 2.16 clearly illustrates our claim that

HDVS is less computationally heavy compared to the other three methods. The CPU usage in HDVS is around 90 percent; However, this amount is above 105 percent for other techniques.

2.5.3 Case 3: Manipulability analysis of VS methods, setup 1

Four different visual servoing methods (HDVS, IBVS, HVS, PBVS) have been used to track a dynamic object with a Tag (Figure 2.17).

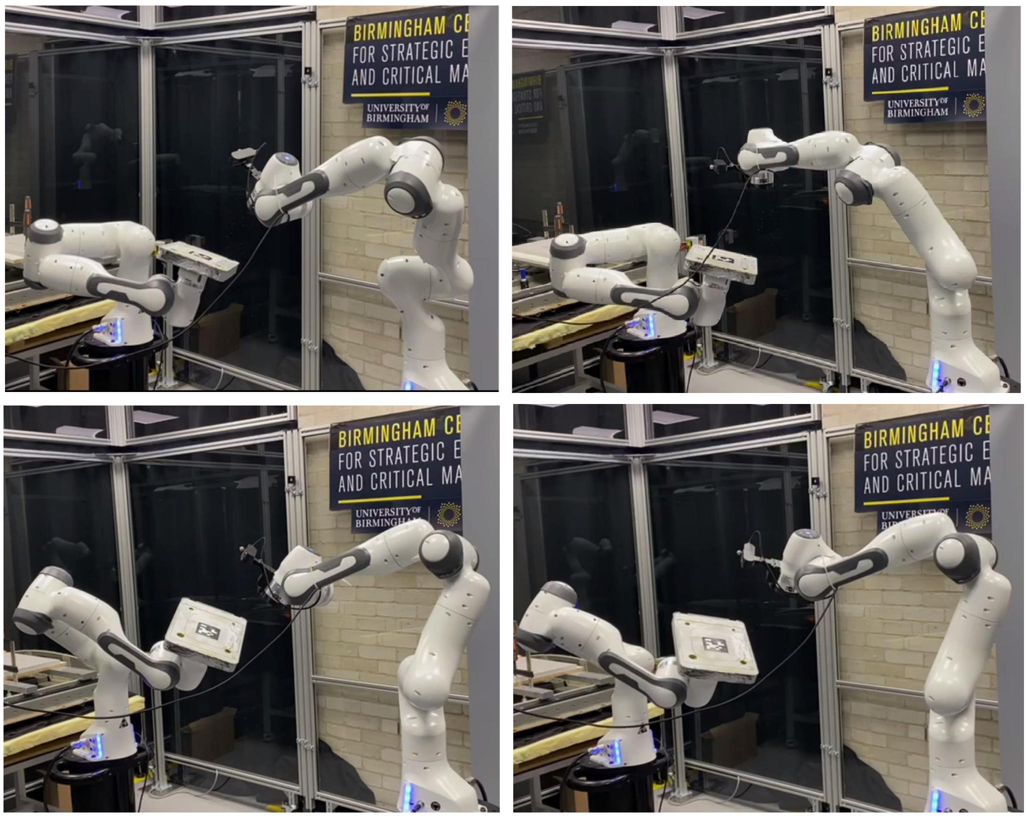


Fig. 2.17. Evaluating the manipulability performance of several visual servoing techniques in tracking a dynamic object.

Figure 2.18 depicts the amount of manipulability for different VS methods. As shown in Figure 2.18, the manipulability of the HDVS method was mostly greater than the other three methods. Not to mention that the manipulability value at the time zero is the same since the robot started moving from the same position with an identical configuration during all trials. The minimum amount of manipulability for the HDVS method is 0.0684, while this amount is 0.0588 for IBVS, 0.0613 for PBVS, and 0.0628 for the HVS method.
In Figure 2.19, the manipulability ellipsoid of the robot in its minimum amount (lowest amount of controllability for the robot) for different VS approaches has been illustrated. Since the manipulability ellipsoid is a hyper ellipsoid in six dimensions and plotting this in 3D space is a complex task, the first three elements of the ellipsoid have been plotted (i.e., translational velocities). The more isotropic the ellipsoid, the higher the controllability of the robot. From the plots in Figure 2.19, it is obvious that the proposed hybrid method has better controllability on the robot movements, compared to IBVS, PBVS, and HVS methods.

2.5.4 Case 4: Application of sorting with HDVS, setup 2

In industry, the robot and the object on which the robot is working must be precisely positioned for the task to be completed successfully. This is the most common procedure for industrial

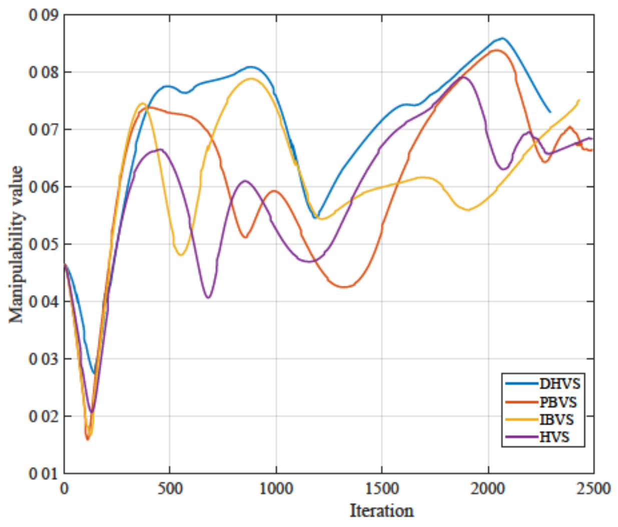
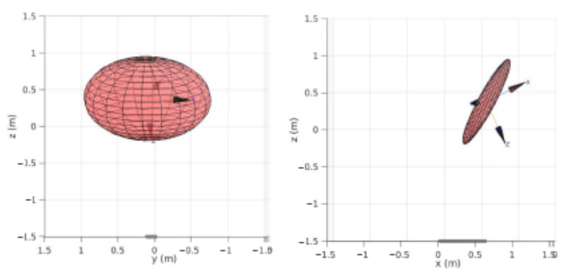


Fig. 2.18. Analysing the manipulability of different VS methods during tracking the trajectory (introduced in Case study 3) of a dynamic object.

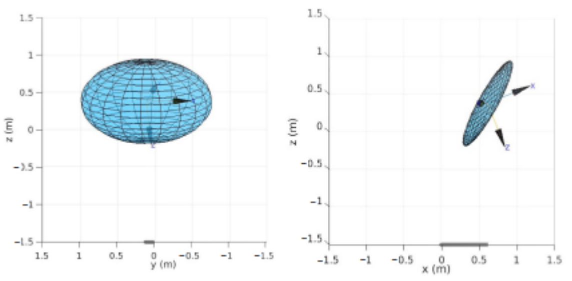
applications. Several things can go wrong during this process. There could be sensory errors, as well as robot kinematic model errors. Alongside, one might not be aware of the precise location of the robot base and the object on which the robot will perform the task. These positions may change during the task and must be calibrated regularly. Not to mention that all positions could not be predicted in advance during the disassembly process, especially when loose and flexible objects are present in the workspace. Currently, the process of dismantling the EV battery packs is carried out manually and robotic disassembly is limited to minor tasks with the assistance of humans [112]. The manual operations also take time and require qualified workers to complete which makes it not economical [113]. A solution to this challenge is fully automating the dismantling process to reduce the cost and to increase safety [114].
As a proof of concept, we designed the fourth case study to demonstrate the capability of the developed HDVS method in automating the process of sorting the dismantled EV battery components. This demonstration is carried out by using Setup 2 introduced in Section 2.4.2 (Figure 2.20).
The manipulator's arm tracks the object by converging the current features in the camera screen to the desired ones (red dots in Figure 2.20a). The trajectory of each feature is shown in green in the camera screen (Figure 2.20a). The convergence threshold for tracking is set to 0.00005m. By using the transformation matrix which links the camera to the vacuum suction gripper, the robot moves to a position above the object. Then, the robot moves in the Z-axis till makes contact with the object (Figure 2.20b, Figure 2.20f). The external force is calculated using the Jacobian of the robot and the joint force sensors (Figure 2.20e). The next step is to lift the object using the vacuum suction gripper (Figure 2.20c) and place it in the desired basket, depending on whether they are reusable or should be discarded (Figure 2.20d).

2.6 | Conclusion

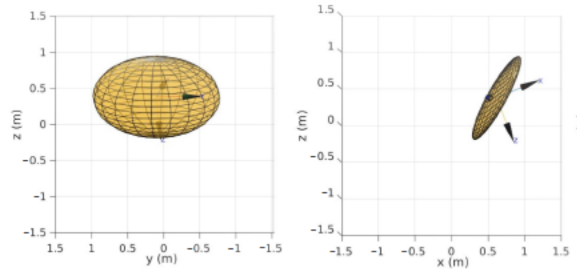
In this chapter, a hybrid decoupled visual servoing (HDVS) method has been proposed. This method has been developed to overcome the drawbacks of classical IBVS and PBVS methods and to improve the classical HVS method. In HDVS, all three rotations, and translation in the Z-axis have been decoupled from the image-Jacobian. These four components' errors will be regulated to zero by the 3D reconstruction of the visual features. Thereafter, a neuro-



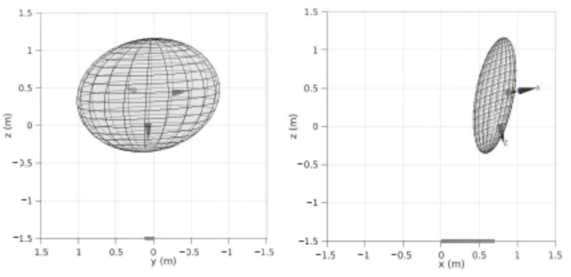
(a) PBVS



(b) IBVS



(c) HVS



(d) HDVS

Fig. 2.19. Manipulability ellipsoid in its minimum amount for three translation degrees of freedom in different VS approaches.

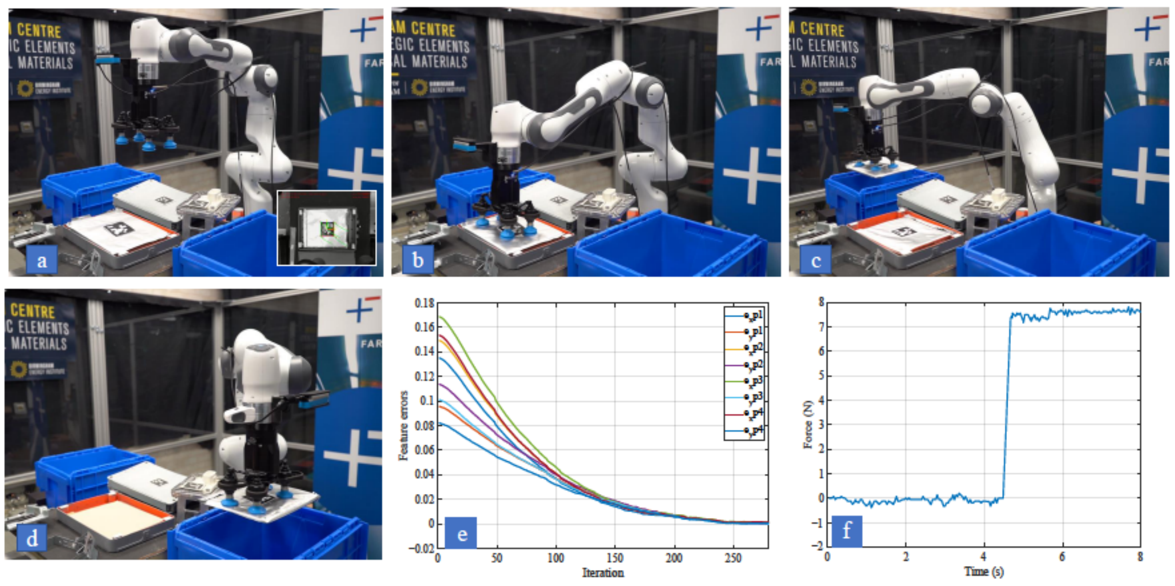


Fig. 2.20. HDVS method is used for tracking the visual features attached to a Lithium-ion battery. Manipulability is considered a secondary task for the controller. By using the proposed HDVS, the robot arm performs the task of sorting the battery; (**a**) the robot will follow the visual features of the object online; the tracked path of each feature is shown in the camera screen. (**b**) The robot goes down straight to detect the surface by the force feedback. (**c**,**d**) The object has been lifted by the vacuum suction gripper and then released the battery in the corresponding basket. (**e**) The feature errors converged to zero during Visual Servoing. (**f**) The force value in the Z-axis for detecting the object surface.

fuzzy LoLiMoT neural network was used to approximate the pseudo-inverse of the proposed interaction matrix. Moreover, the convergence time was reduced by the use of adaptive gains rather than a constant gain. The damped least square method was applied in order to reduce the effect of robot singularities and to smooth the discontinuities.
The method not only has an optimized solution for the robot's EE, but it also considers the optimized trajectories of features in the image space. Furthermore, HDVS has improved the performance of classical HVS in both image-plane and task space. The method is robust in terms of camera parameters and image noises to a good extent, and it avoids singularities of the image-Jacobian effectively. In addition, it is less likely to lose the object from the camera fields of view than the PBVS and HVS methods. Results obtained from 40 random experiments (10 experiments per method) suggested 100% and 90% success rates in executing the VS tasks in simulation and real-world, respectively. In the next chapter, the use of Reinforcement learning with the help of demonstrations for the application of VS has been studied.

Data availability

The supporting data and clips for the findings discussed in this chapter are available in the following repositories:
https://github.com/aaflakiyan/HDVS_Franka
https://github.com/aaflakiyan/DHVS
https://figshare.com/articles/media/Improving_the_manipulability_of_a_redundant_arm_using_Decoupled_Hybrid_Visual_Servoing/17040620
https://figshare.com/articles/journal_contribution/Optimized_Hybrid_Decoupled_Visual_Servoing_simulation_and_code/12980009

CHAPTER 3

Reinforcement Learning from Demonstrations with Visual Servoing

Content from this chapter has been published in the following:

Aflakian, Ali, Alireza Rastegharpanah, and Rustam Stolkin. "Boosting Performance of Visual Servoing Using Deep Reinforcement Learning From Multiple Demonstrations." IEEE Access 11 (Accepted 2023): 26512-26520.

Aflakian, Ali, et al. "An Online Hyper-volume Action Bounding Approach for Accelerating the Process of Deep Reinforcement Learning from Multiple Controllers." Journal of Field Robotics, (Submitted, under second round of review, 2023).

3.1 | Introduction

In this chapter, we have advanced our proposed VS method by integrating RL techniques. Building upon the developments made in the preceding chapter, where we employed LoLiMoT NN to estimate camera velocities based on feature errors, this chapter expands the scope of AI. Using Reinforcement learning, our approach encompasses not only image spaces but also extends to the joint space of the robot. Through this integration, we establish a direct mapping from image features to the desired joint velocities of the manipulator. This direct mapping helps to mitigate potential issues such as instability and robot singularity. Furthermore, it avoids the need for computationally expensive calculations like the robot's pseudo-inverse Jacobian and enhances the overall efficiency and effectiveness of the control strategy. However, deep RL algorithms generally require a large amount of data before they achieve an acceptable performance. We developed a method that uses demonstrations of multiple controllers to address the issue of insufficient data in RL. Unlike the RLfD approaches mentioned in section 1.2.3 that aimed to replace existing learning from demonstration strategies, our proposed method is complementary to established RLfD methods and we fuse ideas from RL, Learning from Demonstration (LfD), and Ensemble Learning into a single paradigm. In other words, knowledge from a mixture of control algorithms (experts) is used to constrain the action space of the agent, enabling faster RL refining of a control policy, by avoiding unnecessary explorative actions. Domain-specific knowledge of each expert is exploited, resulting in a robust policy against errors of individual experts, since it is refined by a reward function without copying any particular demonstration. We have established two distinct case studies to demonstrate the efficacy of our approach. In the first case study, we used the Deep Deterministic Policy Gradient (DDPG) to train the RL algorithm and employed a hypercube to constrain the action space of the agent by integrating insights from previous VS methodologies. More specifically,

the joint velocity data (RL actions) of three experts; image-based VS (IBVS), position-based VS (PBVS), and hybrid-decoupled VS (HDVS), are used in the proposed method to limit the action region of the agent and avoid inapplicable actions. Furthermore, the agent investigates the experts' achieved action bounds in greater depth to find even more ideal candidates who can increase the cumulative rewards. This adjustment served to curtail unnecessary agent explorations, resulting in a notable reduction in training time and an enhancement in the performance of the trained policy. Defining these limitations has nothing to do with the reward function. As a result, the procedure could still be applied even without an explicit reward function using an inverse RL method. Throughout this training process, we employed domain randomization and domain adaptation techniques, reinforcing the robustness of the VS approach in real-world scenarios. Remarkably, our efforts yielded a 51% reduction in training time to achieve the desired performance level, compared to the scenario where RL was applied alone.
Figure 3.1 illustrates the graphical overview of the proposed method.

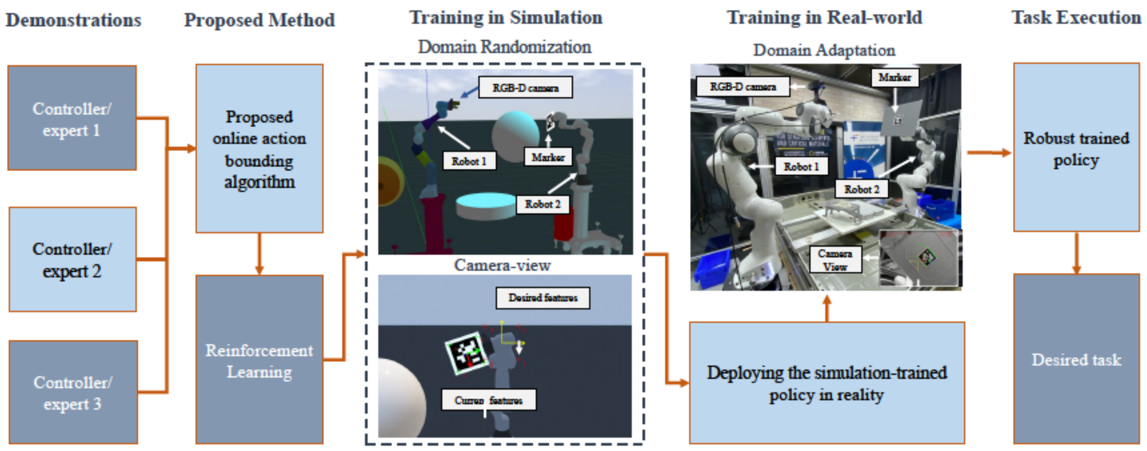


Fig. 3.1. The outline of the proposed strategy combined with the RL method for a VS application in case study 1. The Deep Deterministic Policy Gradient (DDPG) agent will be trained by using the DR method. The proposed method employs a combination of three different methods (PBVS, IBVS, and HDVS) as demonstrators to accelerate training and enhance VS performance. Following that, the policy will be further trained with the real robots to adapt to the real world, and the RL agent will be deployed to complete the visual servoing task.

In the second case study, we explore four methods for bounding the actions of a Twin Delayed DDPG (TD3) RL agent during training, based on the knowledge of several controllers. The first method involves constraining the action space to an online convex hull generated from the knowledge of controllers (AORLD-HL). We modify the loss function to penalize the agent for taking actions outside the generated hypervolume. The second method involves generating an online hypercube from the knowledge of experts to constrain the action space (AORLD-CL). We again modify the loss function to penalize the agent for taking actions outside the hypercube.
The third method involves filtering the actions suggested by the RL policy that lie outside the generated convex hull (AORLD-HF). We use the standard loss function for the RL agent. This method does not modify actions directly but instead filters out undesirable actions.
The fourth method involves projecting the actions outside the convex hull onto the convex hull (AORLD-HP). This method modifies the action space and ensures that the agent always takes actions within the convex hull. We compare the performance of these methods to that

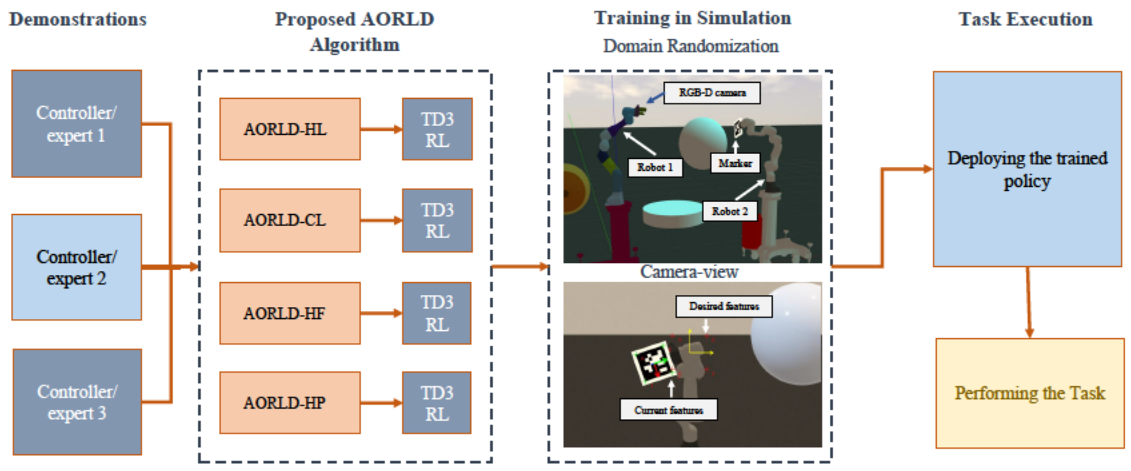


Fig. 3.2. The outline of the proposed online action-optimizer combined with the RL method for a VS application in case study 2. The Twin Delayed Deep Deterministic Policy Gradient (TD3) agent was trained by using the Domain Randomization (DR) method. The AORLD method employs a combination of three different methods (PBVS, IBVS, and HDVS) as demonstrators to accelerate training and enhance VS performance. Four different approaches have been used to constrain the action space of the agent and their results are compared together.

of a standard RL algorithm without any bounding method. Figure 3.2 outlines the AORLD method for a visual servoing application.
In Figure 3.3, the adjustments that we applied to implement RL with our previous control loop in Figure 2.6 are illustrated.
The proposed AORLD approach is generic, in that it can be applied to a wide variety of RL scenarios. However, to demonstrate and validate the method, we implement AORLD in the context of a Visual Servoing (VS) task. The highlights of this study are summarized as follows:

- AORLD adaptively constrains the agent's action spaces by exploiting demonstrations from different "expert" controllers, improving training efficiency.
- AORLD enables learning of policies that do not directly conform to any single demonstration, hence they are robust against errors or imperfections in any individual demonstration.
- The AORLD approach is generic, and can be incorporated into a wide variety of multi-agent and other RLfD algorithms, for many different applications.
- Since the action space is achieved using mathematically proven control methods, the possibility of stacking in the local minima while training the agent in the RL algorithm is reduced.

3.2 | Methodology

The next section gives an overview of the RL algorithms and the proposed strategies used in the two case studies.

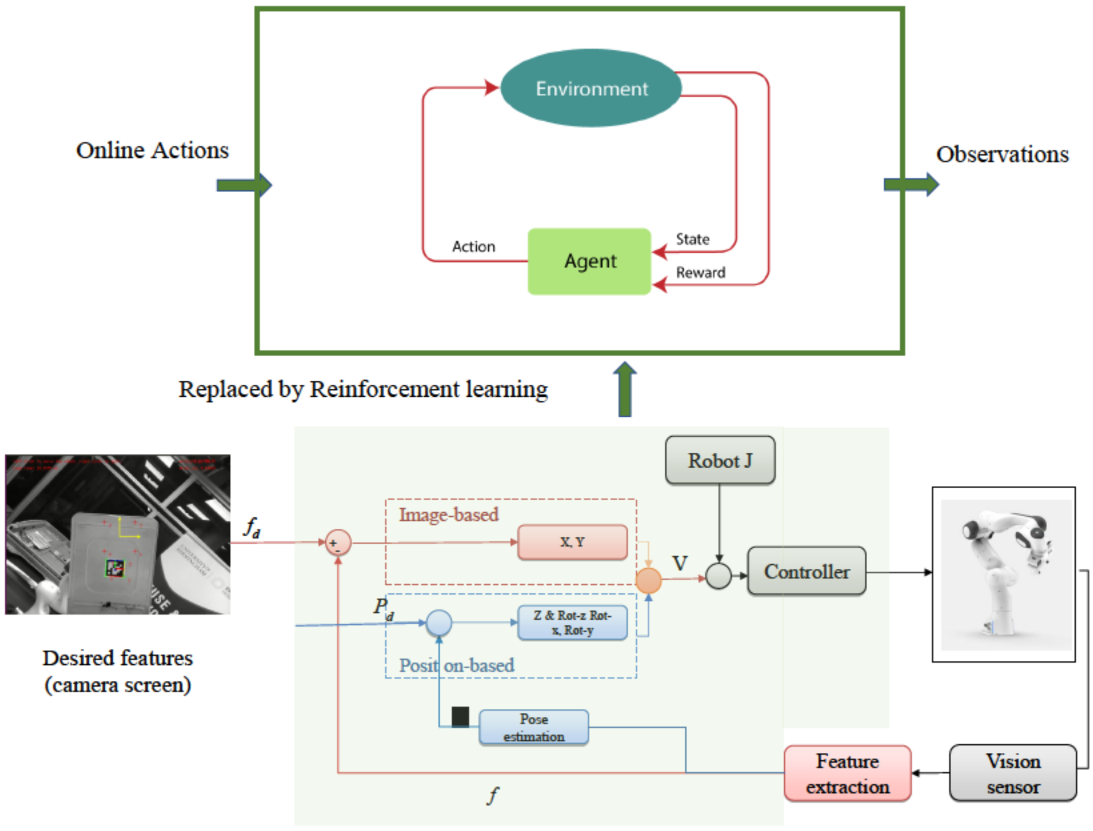


Fig. 3.3. Using RL in control loop introduces in Figure 2.2

3.2.1 Case 1: Integrating AORLD with DDPG

Reinforcement learning is well known for its applications in controlling complex, potentially non-linear systems. The policy is a function that connects observations with actions [115]. The policy parameters are continuously updated by the reinforcement learning algorithm. The goal is to find an optimal policy that is able to maximise the cumulative rewards. The reward indicates how successful an action is in terms of fulfilling the task goal. An RL agent interacts with the actual environment to perform an action a from an action space A and receive state s from a state space S, both of which are determined by a policy known as $\pi(a|s)$. A scalar reward r and the preceding state s' are the results of mapping from state s to actions a. The rewards function $R(s,a)$, the environment model, and the state transition probability $T(s,a,s') = P(s'|s,a)$ provide the basis for this mapping between states and actions. This process is continued in an episodic problem until the agent reaches a terminal state.
The value function $Q^*(s,a)$, which delivers maximal values in all states, is determined using the Bellman equation [72]:

$$Q^*(s,a) = \mathbb{E}\left[R(s,a) + \gamma \sum_{s'} P\left(s' \mid s,a\right) \max_{a'} Q^*\left(s',a'\right)\right] \tag{3.1}$$

where $\gamma \in (0,1]$ is the discount factor, $\mathbb{E}$ is the Bellman expectation, and $Q^\pi(s,a)$ is an estimate of the expected future reward.
DDPG is an algorithm that learns a Q-function and a policy at the same time. The Q function is used to learn the policy using the Bellman equation and off-policy data. Two different NNs

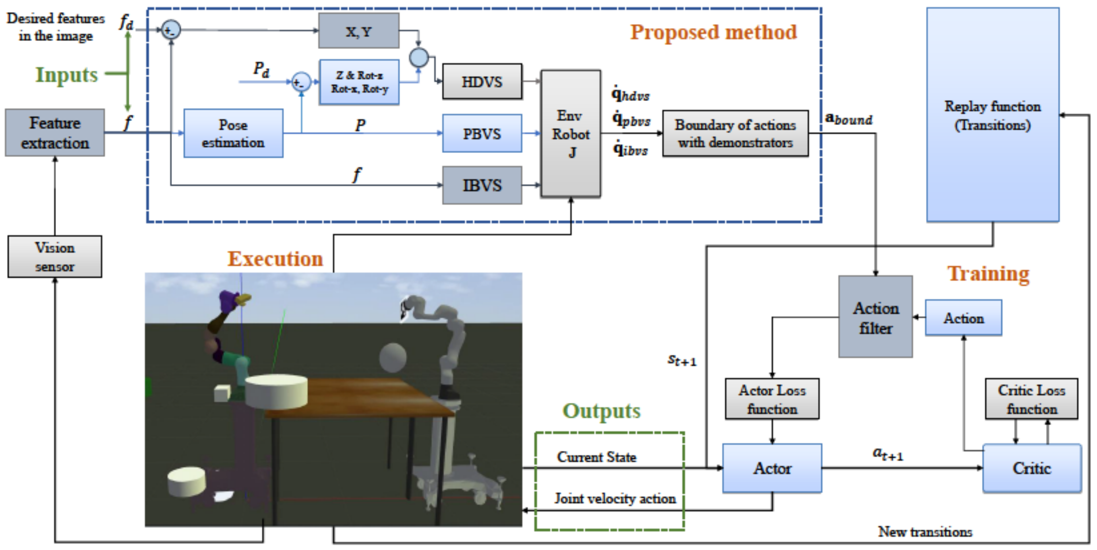


Fig. 3.4. The proposed method block receives current and desired features that have been extracted from the vision sensor as inputs. Consequently, the knowledge of HDVS, PBVS, and IBVS approaches is used at each episode to constrain the action space. Actions outside the created bounds are filtered using the Action filter block in the DDPG policy. The joint velocity actions will be applied to the training environment, and average rewards will be calculated accordingly.

are used in DDPG: An actor (target policy) $\pi : S \to A$, and a critic (action value function approximator) $Q : S \times A \to R$. Approximating the action-value function Q^π of the actor is the duty of the critic [116].
The actor trains using the following loss function:

$$\mathcal{L}_a = -\mathbb{E}_s Q(s, \pi(s)) \tag{3.2}$$

in which s is sampled from the replay buffer. Furthermore, mini-batch gradient descent on the loss $\mathcal{L}$ is used to train the critic network, which promotes the estimated Q-function to meet the Bellman equation:

$$\mathcal{L} = \mathbb{E}\left(Q\left(s_{\underline{t}}, a_t\right) - y_t\right)^2 \tag{3.3}$$

In (3.3), y_t is computed using the action which is the output of the actor network:

$$y_t = r_t + \gamma Q\left(s_{t+1}, \pi\left(s_{t+1}\right)\right) \tag{3.4}$$

Back-propagation through the combined critic and actor networks is used to calculate the gradient of $\mathcal{L}_a$ with respect to the actor parameters [117].
In Figure 3.4, the structure of the proposed technique for the visual servoing task is outlined. As illustrated in Figure 3.4, the proposed method takes in current and desired features extracted from a vision sensor as inputs. These features are used to constrain the action space using knowledge from different approaches, namely HDVS, PBVS, and IBVS (explained in 2.2). This helps to ensure that the actions taken by the system are within acceptable bounds. To enforce these bounds, an action filter block is employed in the DDPG policy. The DDPG policy comprises two blocks: the actor and the critic. The actor block maps the current state to an action, while the critic block evaluates the quality of the action based on the expected

reward. The actor and critic blocks are trained using actor and critic loss functions, respectively. During training, if the action generated by the actor block falls outside the bounds set by the knowledge of HDVS, PBVS, and IBVS approaches, the action filter block will remove the action from consideration. This ensures that only actions within acceptable bounds are considered, leading to better learning outcomes. The joint velocity actions that pass through the action filter block are then applied to the training environment, and the resulting average rewards are calculated.

Proposed method: action constrained strategy

As described in Section 1.2.3, our proposed method uses implemented control algorithms to limit the action space of the RL agent rather than letting the AI agent learn from scratch through their exploration. Joint velocities are defined as RL agent actions (VS commands). The observations are the positions of image feature points, camera poses, and the Jacobian of the robot. Three reward functions are combined to provide the agent reward. The feature errors are driven to zero in the first reward, r_1:

$$r_1 = -\sum_{i=1}^{4} \sqrt{(u_i - u_{id})^2 + (v_i - v_{id})^2} \tag{3.5}$$

where (u, v) denotes the coordinates of a point in the picture and (u_d, v_d) is the desired coordinates of that point. $i = 1$ to n is the number of features (in this case $n = 4$ features). The second reward function (r_2) is defined to avoid joint limits [118]:

$$r_2 = -\frac{1}{2n} \sum_{j=1}^{n} \left(\frac{q_j - \bar{q}_j}{q_{jM} - q_{jm}} \right)^2 \tag{3.6}$$

where $\bar{q}_j$ is center of j^{th} joint range. q_{jM} and q_{jm} are the maximum and minimum angles of the j^{th} joint respectively, and $n = 7$ is the number of joints. Singularity avoidance is introduced in the final reward component [118]:

$$r_3 = \sqrt{\det \left(\mathbf{J}(\mathbf{q}) \mathbf{J}^T(\mathbf{q}) \right)} \tag{3.7}$$

where $\mathbf{J}$ is the robot Jacobian. The overall procedure of the proposed learning method is detailed in Algorithm 2. The final reward function will be derived as:

$$r = -w_f r_1 - w_q r_2 - w_m r_3 \tag{3.8}$$

The terms w_f, w_q, and w_j are weighting factors that are manually adjusted. For each reward, values of $w_f = 10$, $w_q = 2$, and $w_j = 4$ were chosen to determine the weighting contribution. In the proposed acceleration method, the robot achieves the desired joint velocities at the beginning of each step from IBVS, PBVS, and HDVS methods. Thereafter, from a combination of these demonstrators, a set of bounds for the action will be defined $(\mathbf{a}_{bound})$: the lower limits of the bound for the i_{th} joint would be $min(\dot{\mathbf{q}}_{ibvs}[i], \dot{\mathbf{q}}_{pbvs}[i], \dot{\mathbf{q}}_{hdvs}[i])$ and the upper limit of the bound would be: $max(\dot{\mathbf{q}}_{ibvs}[i], \dot{\mathbf{q}}_{pbvs}[i], \dot{\mathbf{q}}_{hdvs}[i])$. In this way, the actions which are out of bounds and created from the critic network would be filtered, and the agent continues exploring by trial and error, within the created bounds. We defined a hypercube at each time step (i.e. a subset of the whole space), as restrictive bounds for action space. Not to mention that the hypercube is one of the simplest spaces which encapsulate all of the action vectors.

Algorithm 2: Action constrained approach

```
Inputs:
• Joint velocities from IBVS ($\dot{\mathbf{q}}_{ibvs}$), PBVS ($\dot{\mathbf{q}}_{pbvs}$), HDVS ($\dot{\mathbf{q}}_{hdvs}$)
Outputs:
• Optimized joint velocities $\mathbf{a}_t$ (actions)
Given:
• RL algorithm DDPG
• The strategy for sampling goals from replay
• The reward function
Initialize actor and critic weights randomly ;
Initialize replay buffer R ;
while VS error not converged do
    for episode i=1 to M do
        Sample g (goal) and initial $s_0$ (state);
        for t=0 to T-1 do
            for k=1 to 7 do
                Get $\mathbf{L}_{xy}$, $\mathbf{L}_r$, $\mathbf{L}_{Pr}$ and $\mathbf{L}_{Pxy}$, $\mathbf{e}_p$, $\mathbf{e}_i$ ;
                Get joint velocities $\dot{\mathbf{q}}_{ibvs}$ from IBVS ;
                Get joint velocities $\dot{\mathbf{q}}_{pbvs}$ from PBVS ;
                Get joint velocities $\dot{\mathbf{q}}_{hdvs}$ from HDVS ;
                Make bounds: $\mathbf{a}_{bound} = [min(\dot{\mathbf{q}}_{ibvs}[i], \dot{\mathbf{q}}_{pbvs}[i], \dot{\mathbf{q}}_{hdvs}[i]),$
                 $max(\dot{\mathbf{q}}_{ibvs}[i], \dot{\mathbf{q}}_{pbvs}[i], \dot{\mathbf{q}}_{hdvs}[i])]$ ;
                Sample $a_t$ (action) using DDPG policy and filter actions out of the
                 $\mathbf{a}_{bound}$ bound: $\pi(s_t|g) \rightarrow a_t$ ;
                Execute $a_t$ and observe $s_{t+1}$ (new state) ;
                $r_t := r(s_t, a_t, g)$ ;
                Store $(s_t|g, a_t, r_t, s_{t+1}|g)$ (transition) in R ;
                Sample $g'$ (additional goal) for replay $G : S$(current episode) ;
            end
            for $g' \in G$ do
                $r' := r(s_t, a_t, g')$ ;
                Store $(s_t|g', a_t, r', s_{t+1}|g')$ in R ;
            end
            for t=1 to N do
                Sample B (mini-batch) from the R (replay buffer) ;
                Execute one step of optimization using DDPG and B ;
            end
        end
    end
end
```

3.2.2 Case 2: Integrating AORLD with TD3

TD3 is an effective off-policy actor-critic algorithm that uses delayed policy updates and target policy smoothing to improve stability and performance. The algorithm involves the use of two Q-functions, Q_{ϕ_1} and Q_{ϕ_2}, which are learned simultaneously by minimizing the mean square Bellman error [119]. The Bellman equation is a fundamental concept in reinforcement learning that helps predict the expected cumulative reward of being in a state and taking an action. To

learn the Q-functions, TD3 uses the mean square Bellman error, which measures the difference between the predicted Q-value and the actual Q-value for a state-action pair. By using two Q-functions, TD3 aims to reduce the overestimation bias that can occur in single Q-function methods. The use of target policy smoothing, which adds noise to the actions selected by the actor, further enhances the stability and performance of the learned policy.
To form the Q-learning target in the TD3 reinforcement learning algorithm, actions are generated based on the target policy, denoted as $\mu_{\theta_{\text{targ}}}$. However, clipped noise is added to each dimension of the action to enhance exploration. This means that the target action is obtained by adding clipped noise to the output of the target policy and then clipping the result to ensure that it lies within the valid action range ($a_{Low} \leq a \leq a_{High}$). Mathematically, the target actions can be expressed as [120]:

$$a'(s') = \text{clip}\left(\mu_{\theta_{\text{targ}}}(s') + \text{clip}(\epsilon, -c, c), a_{min}, a_{max}\right) \tag{3.9}$$

where ϵ is a noise term drawn from a normal distribution with zero mean and standard deviation σ, and c is a constant that determines the amount of noise added to the action.
Clipped double-Q learning is a method employed in the TD3 reinforcement learning algorithm to alleviate the overestimation issue in the Q-function. The approach trains two Q-functions, labelled as Q_{ϕ_1} and Q_{ϕ_2}, simultaneously by regressing towards a single target value. The target value is computed by choosing the Q-function that gives the lower target value, which is expressed mathematically as:

$$y(r, s', d) = r + \gamma(1 - d) \min_{i=1,2} Q_{\phi_{i,\text{targ}}}(s', a'(s')) \tag{3.10}$$

Here, r is the reward, s' is the next state, d is a binary indicator of whether the episode has ended, γ is the discount factor, and $a'(s')$ is the target action with clipped noise, as described earlier in (3.9).
Both Q-functions, Q_{ϕ_1} and Q_{ϕ_2}, are then trained by minimizing the squared difference between the predicted Q-value and the target value [117].
Finally, the TD3 policy with the modified loss function is learned by maximizing Q_{ϕ_1}. The actor network, denoted as μ_θ, selects actions that maximize the Q-value estimated by Q_{ϕ_1}. Mathematically, the policy is learned by solving the following optimization problem:

$$\max_{\theta} \underset{s \sim \mathcal{D}}{\mathrm{E}} \left[Q_{\phi_1}(s, \mu_\theta(s))\right] \tag{3.11}$$

While we used TD3 in this case, the proposed method is readily applicable to different RL algorithms by modifying their target action limits. The overall procedure of the proposed optimization learning method is detailed in Algorithm 3 and Figure 3.5.
The algorithm takes candidate actions from multiple expert controllers as input and optimized actions for a given task, considering the observations, are the outputs of the trained policy. The RL algorithm used in this approach is TD3 (Twin Delayed Deep Deterministic Policy Gradient). The AORLD approach has four different methods for constraining the actions generated by the RL algorithm: convex hull with modified loss function (AORLD-HL), hypercube with modified loss function (AORLD-CL), convex hull with filtering actions outside the hull (AORLD-HF), and convex hull with projecting actions onto the hull (AORLD-HP). Each method modifies the TD3 algorithm in a specific way to constrain the generated actions.
In each iteration of the training process, the AORLD approach samples a goal and an initial state and generates an action using the RL algorithm. The generated action is then constrained by one of the four above-mentioned methods, depending on the chosen method. The resulting

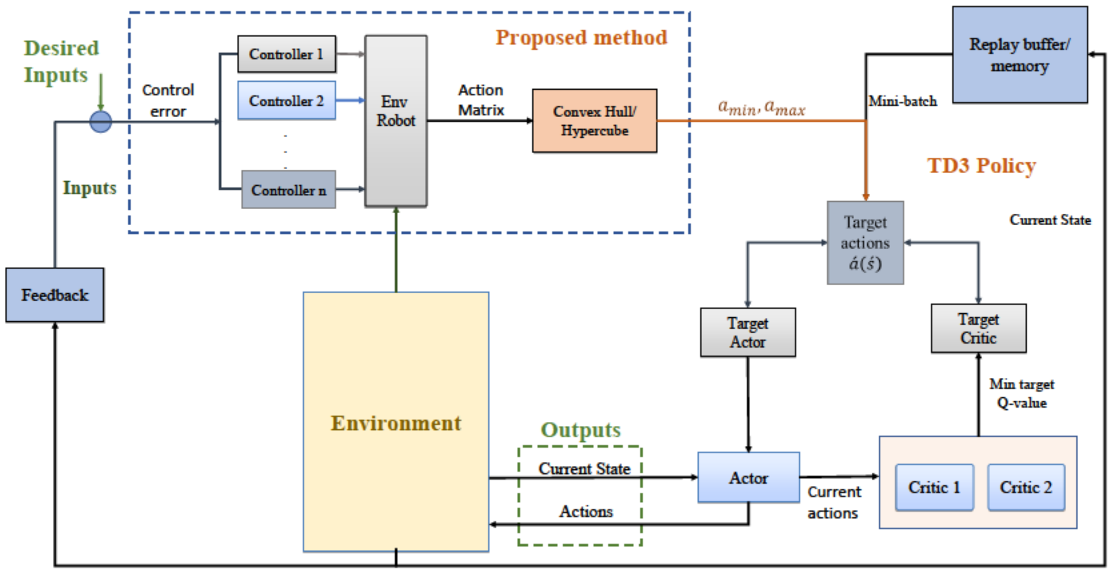


Fig. 3.5. Structure of AORLD integrated with TD3 RL. The proposed block diagram in this study takes in current and desired features extracted from the vision sensor as inputs. Then, during each episode, the knowledge from HDVS, PBVS, and IBVS approaches is utilized to restrict the action space. The joint velocity actions are then applied to the training environment, and the average rewards are computed accordingly.

action is executed in the environment, and the resulting state and reward are stored in a replay buffer.
The AORLD approach then samples additional goals from the replay buffer and generates actions for these goals. The resulting states and rewards are also stored in the replay buffer. The replay buffer is then sampled to create mini-batches for optimizing the policy using TD3. In the following, we will describe those four methods to constrain actions in detail and explain their respective pros and cons.
The AORLD-HL algorithm is introduced in Algorithm 4. To compute the a_{min} and a_{max} values in (3.9), the following procedure was adopted: data are collected from multiple controllers that generate candidate action vectors for each observation of the environment. Thereafter, the convex hull of action vectors is computed and the resulting convex hull is used to define the feasible action space for the reinforcement learning agent. Let k be the set of indices that define the convex hull of A, i.e., $k = \text{convhulln}(A)$. For each dimension i of A, minimum (a_{min}) and maximum (a_{max}) values over the vertices of the convex hull are computed as follows:

$$a_{min,i} = \min_{j\in k} a_{j,i} \quad \text{and} \quad a_{max,i} = \max_{j\in k} a_{j,i} \tag{3.12}$$

where $a_{j,i}$ is the ith component of the jth vertex of the convex hull. It should be noted that $\pi(s_t|g)$ in Algorithm 4 is the policy that maps the state s_t to an action a_t given the goal g, and $\rightarrow$ denotes the assignment of a_t to the output of the policy. The convex hull is a mathematical concept that defines the smallest convex set that contains all the given points in a higher dimensional space. In our case, the convex hull is a boundary that encloses most of the potentially desired action vectors. It should be mentioned that it requires at least n+1 unique points in n-dimensional space to create a n-dimensional convex hull. The algorithm uses each controller prediction to have at least n+1 data if there is not enough data to build the n-dimensional convex hull. Given the bounding convex hull, we can find the minimum and maximum values for each dimension by computing the minimum (a_{min}) and maximum

Algorithm 3: AORLD approach

```
Inputs:
• Candidate actions from expert 1 ($a_{ex1}$), expert 2 ($a_{ex2}$), ..., expert n ($a_{exn}$)
Outputs:
• Optimized actions $a_t$
Given:
• RL algorithm TD3
• The strategy for sampling goals from replay
• The reward function
Initialize actor and critic weights randomly;
Initialize a set of actions randomly;
Initialize replay buffer R;
while Controller stop criteria not achieved do
    for episode i=1 to M do
        Sample g (goal) and initial $s_0$ (state);
        for t=0 to T-1 do
            if AORLD-HL then
                Use Algorithm 2 to calculate $a_t$;
            else if AORLD-CL then
                Use Algorithm 3 to calculate $a_t$;
            else if AORLD-HF then
                Use Algorithm 4 to calculate $a_t$;
            else if AORLD-HP then
                Use Algorithm 5 to calculate $a_t$;
            end Execute $a_t$ and observe $s_{t+1}$ (new state);
            $r_t := r(s_t, a_t, g)$;
            Store $(s_t|g, a_t, r_t, s_{t+1}|g)$ (transition) in R;
            Sample $g'$ (additional goal) for replay $G : S(currentepisode)$ ;
            for $g' \in G$ do
                $r' := r(s_t, a_t, g')$;
                Store $(s_t|g', a_t, r', s_{t+1}|g')$ in R;
            end
            for t=1 to N do
                Sample B (mini-batch) from the R (replay buffer);
                Execute one step of optimization using TD3 and B;
            end
        end
    end
end
```

(a_{max}) values of each coordinate of the vertices of the convex hull. During training, the convex hull would be periodically updated using a new set of action vectors. As explained before, this approach aims to adaptively constrain the RL agent to choose actions within the feasible action space defined by the convex hull. Limiting the actions of the RL agent to lie within the convex hull can potentially simplify the learning problem and make it easier for the agent to converge to a good policy. Moreover, by limiting the actions to a smaller region of the action

Algorithm 4: AORLD-HL

```
Inputs:
• Candidate actions from expert 1 ($a_{ex1}$), expert 2 ($a_{ex2}$), ..., expert n ($a_{exn}$)
Outputs:
• Candidate actions ($a_t$) inside the generated convex hull
for j=1 to n do
    Generating convex hull around action vectors $A$:
    $k = convhulln(A)$;
    Find the minimum and maximum vectors for each dimension:
    $a_{min} = min(a(k(:), :), [], 1)$;
    $a_{max} = max(a(k(:), :), [], 1)$;
    Modify the TD3 algorithm to use the new bounded action space:
    $a'(s') = \text{clip}\left(\mu_{\theta_{\text{targ}}}(s') + \text{clip}(\epsilon, -c, c), a_{min}, a_{max}\right)$
    Sample actions ($a_t$) inside the convex hull using TD3 policy:
    $\pi(s_t|g) \longrightarrow a_t$ ;
end
```

space, the agent has fewer options to choose from and can more quickly learn which actions are likely to lead to good outcomes.

Algorithm 5: AORLD-CL

```
Inputs:
• Candidate actions from expert 1 ($a_{ex1}$), expert 2 ($a_{ex2}$), ..., expert n ($a_{exn}$)
Outputs:
• Candidate actions ($a_t$) inside the generated hyper cube
for j=1 to n do
    Make action bounds:
    $a_{bound} = [min(a_{ex1}[i], a_{ex2}[i], ..., a_{exn}[i]),\ \ max(a_{ex1}[i], a_{ex2}[i], ..., a_{exn}[i])]$ ;
    Modify the TD3 algorithm to use the new bounded action space:
    $a'(s') = \text{clip}\left(\mu_{\theta_{\text{targ}}}(s') + \text{clip}(\epsilon, -c, c), a_{min}, a_{max}\right)$
    Sample actions ($a_t$) inside the convex hull using TD3 policy:
    $\pi(s_t|g) \longrightarrow a_t$ ;
end
```

The AORLD-CL algorithm is established in Algorithm 5. In AORLD-CL, instead of generating a convex hull around the experts' outputs, a set of bounds for the action is defined based on the combination of demonstrators' data. These bounds represent the minimum and maximum values that each action can take. To create the lower limit of the i_{th} action, the minimum value of that action from all the demonstrators' data is used:

$$a_{min,i} = min(a_{ex1}[i], a_{ex2}[i], ..., a_{exn}[i] \tag{3.13}$$

Similarly, the upper limit of the i_{th} action is created using the maximum value of that action from all the demonstrators' data:

$$a_{max,i} = max(a_{ex1}[i], a_{ex2}[i], ..., a_{exn}[i] \tag{3.14}$$

This creates a hypercube in the action space that represents the feasible action space for the agent. Not to mention that the convex hull is the minimum bounding convex hypervolume that includes the actions from the controllers, which reduces the action and search space of the agent more than the hypercube. Convex hull also accounts for correlations between different actions, whereas the hypercube approach in AORLD-CL assumes each action dimension is independent. During training, the hypercube is updated periodically with the demonstrators' new set of action vectors. This ensures the feasible action space is updated as the agent learns from the demonstrators' data. The modified TD3 loss function in AORLD-CL enforces that the agent's predicted actions stay within these bounds. This simple approach to limiting the action space can be computationally less expensive than generating a convex hull and still helps to constrain the agent's actions to the feasible action space. Nevertheless, the agent's exploration space is not optimally condensed and is larger than when employing a convex hull.

Algorithm 6: AORLD-HF

```
Inputs:
• Candidate actions from expert 1 ($a_{ex1}$), expert 2 ($a_{ex2}$), ..., expert n ($a_{exn}$)
Outputs:
• Candidate actions ($a_t$) inside the generated convex hull
for j=1 to n do
    Generating convex hull around action vectors $A$:
    $k = convhulln(A)$;
    Filter actions outside the created bound:
    if Inhull(k,$a_t$)==1 then
        Sample actions ($a_t$) inside the convex hull using TD3 policy:
        $\pi(s_t|g) \rightarrow a_t$ ;
    else
        Repeat the action generation
    end
end
```

The AORLD-HL (Algorithm 4) and AORLD-HF (Algorithm 6) methods both use an online convex hull to constrain the action space of the RL agent, but they differ fundamentally in their approach and impact on the learning process. AORLD-HL modifies the agent's loss function to penalize actions outside the convex hull, thereby directly shaping the agent's policy during training to encourage inbound actions. This method actively discourages the agent from exploring out-of-bound actions by making them more costly. In contrast, AORLD-HF does not alter the loss function; instead, it acts as a post-processing filter, removing any actions that fall outside the convex hull before execution. This allows the agent to learn freely without penalties during training, with the constraints applied only when the agent is about to take an action.
The algorithm 6 follows the steps below:
(I) Generate a convex hull around action vectors A using the $convhulln$ function, (II) Find the minimum and maximum vectors for each dimension using the minimum and maximum functions, (III) For each action sample a_t, check if it is inside the generated convex hull using the "Inhull" function, (IV) If a_t is inside the hull, sample an action inside the convex hull using the TD3 policy, (V) If a_t is outside the hull, repeat the action generation.
Finally, AORLD-HP, introduced in Algorithm 7, differs from AORLD-HF in that AORLD-HF filters actions outside the generated bound; however, AORLD-HP uses a projection method explained in Algorithm 7. The projection method projects the candidate action onto the closest

Algorithm 7: AORLD-HP

```
Inputs:
• Candidate actions from expert 1 ($a_{ex1}$), expert 2 ($a_{ex2}$), ..., expert n ($a_{exn}$)
Outputs:
• Candidate actions ($a_t$) inside the generated convex hull
for j=1 to n do
    Generating convex hull around action vectors $A$:
    $k = convhulln(A)$;
    if Inhull(k,a)==1 then
        Sample action ($a_t$) inside the convex hull using TD3 policy:
        $\pi(s_t|g) \rightarrow a_t$ ;
    else
        Project the action vector onto the convex hull:
        for $l = 1 : size(k, 1)$ do
            $a_t = zeros(n, 1)$;
            $x = points(k(i, :), :)$;
            $u = x(1, :)'$;
            $v = x(2, :)' - u$;
            $proj = u + v * ((a - u)' * v)/norm(v)^2$;
            $a_t = a_t + proj$;
            $\pi(s_t|g) \rightarrow a_t$ ;
        end
    end
end
```

point on the boundary of the convex hull as follows: Let k be the set of indices of the convex hull points, $points$ be the set of points on the convex hull and a be the original action vector to be projected onto the convex hull. Let u be the starting point of a line segment on the convex hull, v be the direction of the line segment, and $proj$ be the projection of a onto the line segment.

$$proj = u + v\frac{(a-u)^T \cdot v}{|v|^2} \tag{3.15}$$

It should be mentioned that the hard constraint of modifying the loss function with new minimum and maximum actions in the AORLD-HL method is more effective than filtering and projecting actions on the hull (AORLD-HF and AORLD-HP, respectively), as it enforces the action constraints strictly. However, the algorithm in AORLD-HL assumes that the candidate actions from the expert controllers are sufficient to define the action space, which may not always be the case in complex environments. We will discuss VS as an application to test our suggested approaches in the following.

3.3 | Experimental setup

To train the policy, an environment was modelled in the simulation (i.e. ROS/Gazebo). Simulations are preferred over real-world trials because they provide inexpensive and fast experiments. In addition, using the simulation environment helps mitigate the risks of damaging the robot setup due to unexpected movements during training. Since the real system is assumed

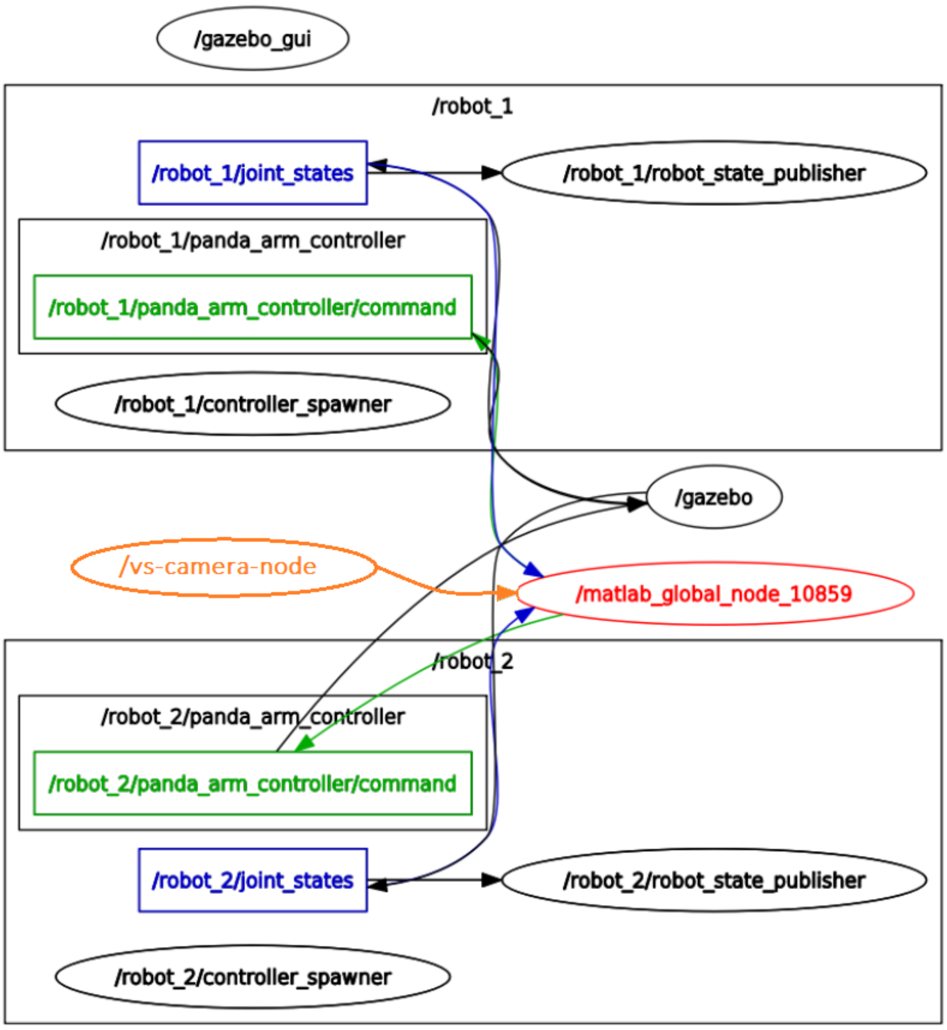

/gazebo_gui
/robot_1
/robot_1/joint_states
/robot_1/robot_state_publisher
/robot_1/panda_arm_controller
/robot_1/panda_arm_controller/command
/robot_1/controller_spawner
/gazebo
/vs-camera-node
/matlab_global_node_10859
/robot_2
/robot_2/panda_arm_controller
/robot_2/panda_arm_controller/command
/robot_2/joint_states
/robot_2/robot_state_publisher
/robot_2/controller_spawner

Fig. 3.6. Considered ROS nodes used in training of the RL agent.

to be one instance in a vast distribution of training variations, the trained model with DR can adapt to the real-world environment. As mentioned in 1.2.3, DR is a technique for training a model that works in a variety of simulated settings with randomized properties [121].
Figure 3.6, illustrates the ROS nodes used in this study. The RL algorithm was used as a ROS node $(/matlab-global-node)$, and policies were taught using Matlab reinforcement Learning Toolbox [122]. Control methods (IBVS, PBVS, and HDVS) and the camera information were employed as a distinct ROS node $(/vs-camera-node)$ to deliver the actions based on the respective observations. Joint velocity commands were applied to each robot with $/robot-n/panda-arm-controller/command$ $(n = 1, 2)$ topics. The joint-states topics provide the position and velocity of each joint from the robot models simulated in Gazebo. Using a ROS Handler this information will be published to subscribe and use in each node.
Figure 3.7 depicts the simulation environment in Gazebo. The simulation platform includes two Franka robot manipulators; one with an eye-in-hand configuration, and another one with a tag marker attached to its EE. The reason for using the second arm was to move the marker into different positions.
An Intel RealSense depth camera D435i was employed as a vision sensor. Two systems were linked together using an Ethernet connection in the same network. One system with the following specifications was utilized for the simulation: AMD Ryzen 7 3700x 8-core CPU with

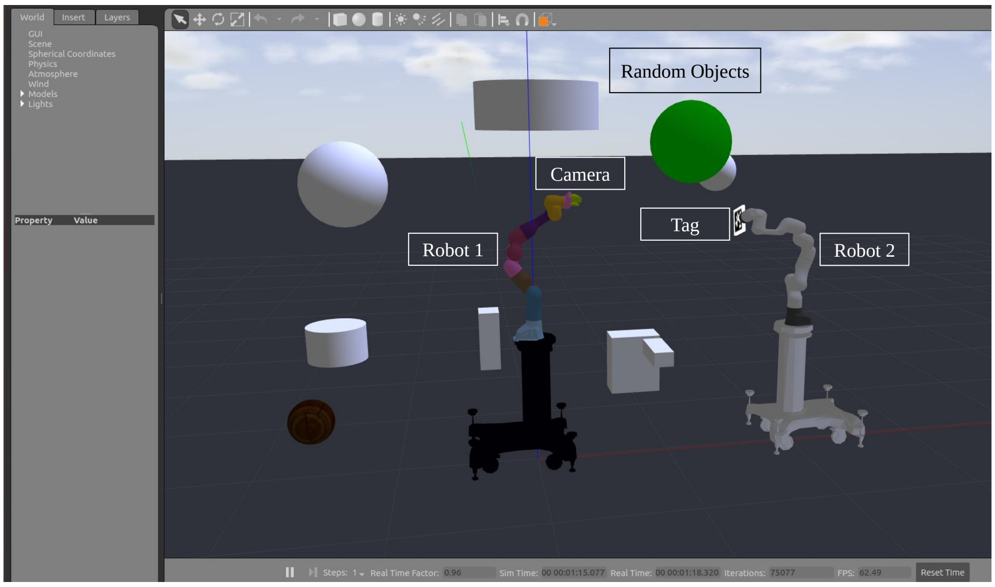


Fig. 3.7. Simulation environment in Gazebo

× 16 threads and a 3.6 GHz base clock. The graphics card (GPU) of another system with the following specifications was used for reinforcement learning and policy training: NVIDIA GTX 1080Ti GPU, Intel (R) Core (TM) i7-8086K 6-core CPU with × 12 threads and a 4GHz base clock with 32 GB installed RAM. Moreover, to accelerate the process of learning, parallel training was used with the help of Parallel Computing Matlab Toolbox [122]. In this study, 12 workers were deployed to create a simulation of the agent in the environment and send data back to the client.

3.4 | Results and discussions

3.4.1 Case 1: AORLD with DDPG policy results

To show the efficacy of our suggested strategy (detailed in Section 3.2.1), we compare the training progress and results of RL without the proposed strategy (agent-1) and with the proposed strategy (agent-2).
For the sim-to-real task, DR was used which has been identified as the most commonly used strategy for improving simulation realism. The training was carried out for the task of VS (i.e. tracking the features of a target in the camera screen). The initial position of the first robot (the robot with the camera mounted on its wrist) was randomized in each episode to generalize the trained policy better. The desired threshold of the average reward was defined to be -200 (determined from preliminary experiments). The agent restarted the episode in case of meeting one of the following criteria during training: (I) when the robot is close to the joint limits, (II) when the features are 10% close to the image boundary, (III) when the robot Jacobian manipulability is too small (less than 0.01), and finally (IV) when the number of steps in each episode surpasses 400. The used RL parameters are defined in Table 3.1.
As illustrated in Figure 3.8, the agents have learned to maximize the cumulative reward over time. According to Figure 3.8, it takes the agent approximately 57800 episodes and 6 million steps for the average reward to exceed the desired threshold (-200). However, it is shown that approximately 28400 episodes and 3 million steps are required to achieve the same average reward of -200 by combining the proposed method with RL. Thereafter, using the so-called domain adaptation approach, we let the trained policies train for 1500 more real-world episodes

Table 3.1
Employed RL and noise parameters in the training.

RL parameters		Noise options	
Target smooth factor	0.001	Mean	0
Target update freq	1	Mean Attraction constant	5
Sample time	0.025	Variance decay rate	0.00001
Discount factor	0.95	Variance	0.5

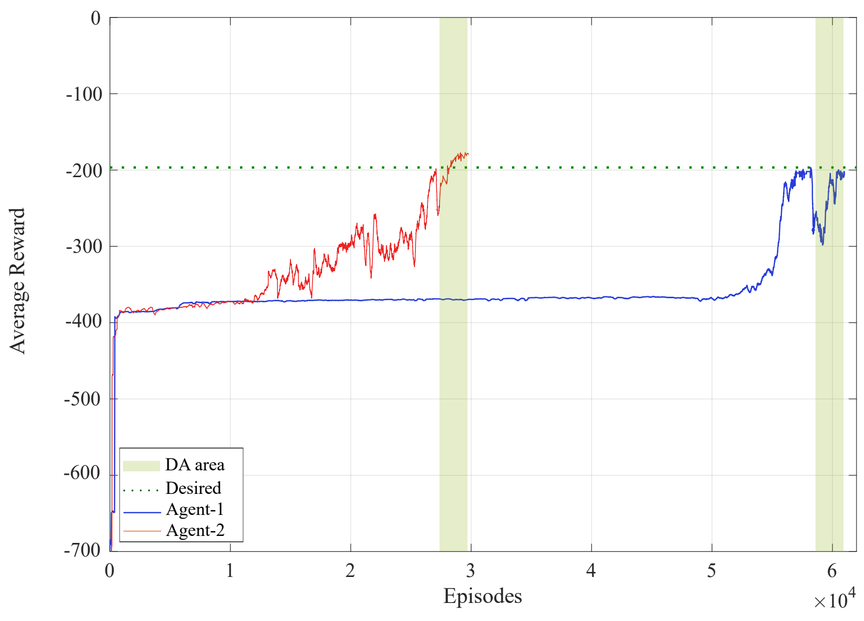


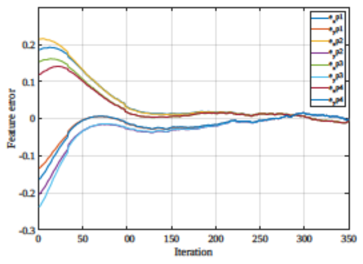
Fig. 3.8. Training graph without using any expert (agent-1, blue line), and with using three experts to constrain the action space (agent-2, red line).

using actual robots (green area in Figure 3.8). Both agent-1 and agent-2 reached the award of -200 before domain adaptation, however, agent-2 achieved a higher reward after domain adaptation. Due to the change of environment (i.e. sim-to-real), the reward values dropped for both agents after DA, as shown in Figure 3.8. Results suggest that agent-2 is faster and achieves a more effective policy (higher average reward) than agent-1.
Table 3.2 compares the effective parameters in the performance of individual VS methods and the trained policies. These parameters are derived by averaging 50 trials with 10 different initial positions of the robot equipped with the camera. These initial positions were chosen randomly with the condition of having all four features visible in the image frame. All experiments are duplicated under the same conditions for IBVS, PBVS, HDVS, agent-1, and agent-2. As shown in Table 3.2, IBVS shows an optimized behaviour in 2D image space, because it has the smallest Root Mean Square Error (RMSE) than other methods. Moreover, the smaller range of feature errors in IBVS confirms that the chance of losing the target object from the camera FOV in this method is lower than in the other four methods. However, agent-2 is still faster than IBVS based on the number of iterations. Not to mention that in VS both image space and robot space should be considered, and this is where the suggested method surpasses the IBVS. Following IBVS, the trained policy with agent-2 performs more optimally (smaller

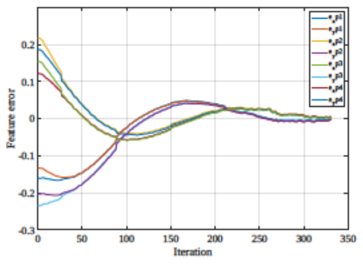
Table 3.2
The performance comparison of visual servoing techniques derived by averaging 50 trials (10 trails per method). The bold results represent the best candidate for each column.

Method	RMSE of 2D Errors	Feature Error Range	RMSE of Position (m)	RMSE of Orientation (deg)	Camera Travelled Distance (m)	Manipulability Mean	Manipulability Range	Iterations	Average Reward
IBVS	**0.0222**	**[-0.36, 0.31]**	0.036	9.43	0.942	0.0407	[0.014, 0.081]	292	-309.42
PBVS	0.0383	[-0.44, 0.51]	**0.022**	**6.54**	**0.722**	0.0446	[0.024, 0.080]	387	-316.23
HDVS	0.0273	[-0.45, 0.49]	0.034	8.41	0.917	0.0396	[0.021, 0.081]	424	-275.19
Agent-1	0.0458	[-0.52, 0.62]	0.046	15.3	1.140	0.0321	[0.013, 0.075]	383	-223.03
Agent-2	0.0258	[-0.34, 0.34]	0.030	7.82	0.839	**0.0492**	**[0.029, 0.082]**	**253**	**-184.21**

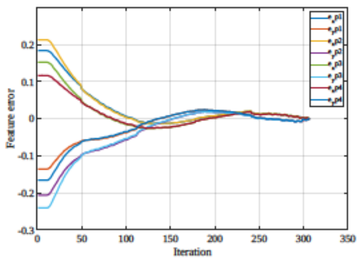
RMSE, smaller feature error range, and fewer iterations) than other methods. As a result, agent-2 outperformed PBVS, HDVS, and agent-1 in the image space.
From Table 3.2, the average mean value of manipulability was 0.0492 for agent-2, 0.0407 for IBVS, 0.0446 for PBVS, 0.0396 for HDVS, and 0.0321 for agent-1. Therefore, agent-2 performs better in terms of manipulability in comparison with the other four approaches. From Table 3.2, it would also be inferred that the robot has better controllability with agent-2 while tracking the target due to higher manipulability bounds created by agent-2.
Figure 3.10 depicts the commanded joint velocities (actions) in different VS methods for the same scenario in Figure 3.9. It should be noted that the joint velocities obtained by using RL



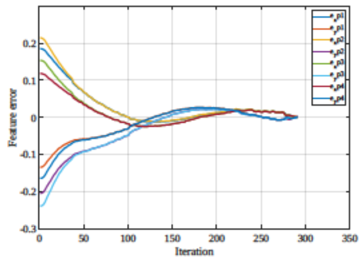
(a) IBVS



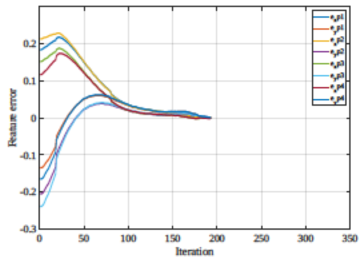
(b) PBVS



(c) HDVS



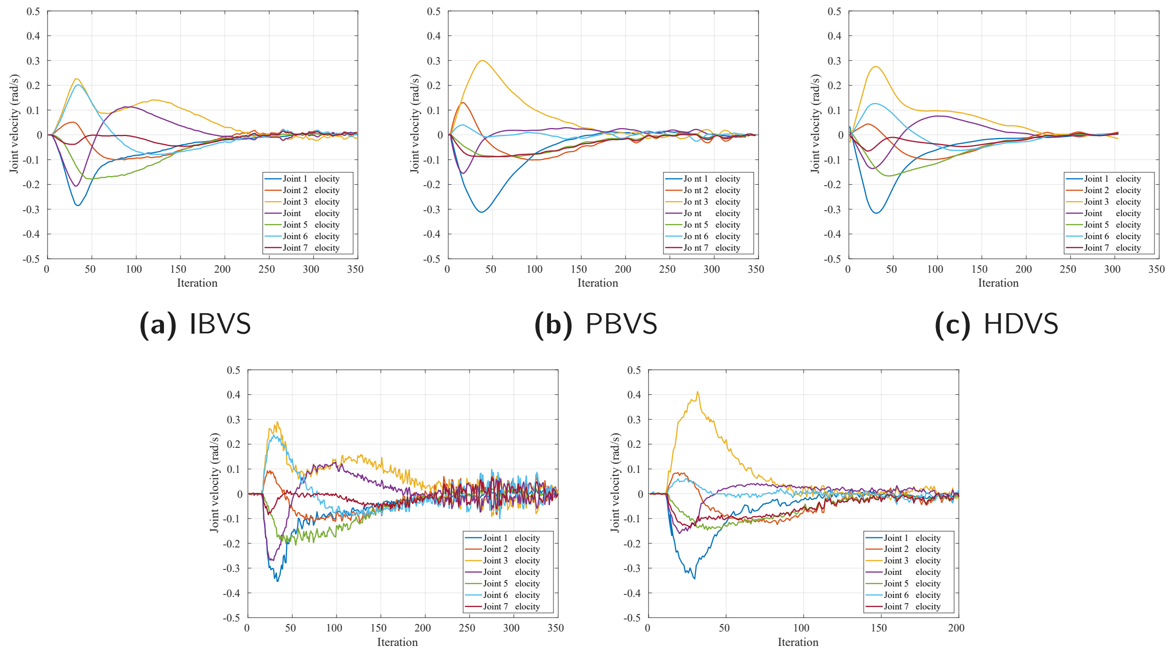
(d) Agent-1



(e) Agent-2 with AORLD

Fig. 3.9. Comparison of visual servoing feature errors with various approaches for one random trial. It should be noted that e_{xPi} and e_{yPi} represent the x and y components of the i^{th} feature, respectively.

methods are not as smooth as those obtained by using IBVS, PBVS, and HDVS (could be easily solved by using a Filter to smooth the commands). However, comparing Figure 3.10e to other subplots in Figure 3.10 shows that the performance of the controller, in terms of required iterations to complete the VS task, has been improved by using the proposed optimization method.
According to Table 3.2, the policy trained with agent-2 has a shorter camera path than IBVS, HDVS, and agent-1. The robot EE's travelled distance is 0.839m with agent-2. This value is 0.942m, 0.917m, and 1.14m, in IBVS, HDVS, and agent-1, respectively. In PBVS, the



(a) IBVS **(b)** PBVS **(c)** HDVS

(d) Agent-1 **(e)** Agent-2 with AORLD

Fig. 3.10. Comparison of visual servoing joint velocities with various approaches.

camera travelled distance was 0.722m, and thus provided the shortest EE trajectory. Agent-2 outperforms IBVS, HDVS, and agent-1 in the Cartesian (robot) space. However, still better results are achieved using the PBVS method in terms of EE travelled distance. It should be mentioned that the PBVS method has other drawbacks like losing the object from the camera FOV, high sensitivity to the camera parameters, and its sub-optimal performance in the image space, while the trained policy with agent-2 has solved these drawbacks. Another inference from Table 3.2 is that the RMSE of position and orientation with the trained policy of agent-2 is smaller than IBVS, HDVS, and agent-1 policy which indicates a more optimized trajectory created by agent-2.

In conclusion, agent-2 achieved an optimal overall performance over the image space and cartesian space, where PBVS and IBVS suffer respectively. Additionally, using our suggested approach the agent offers the best controllability among the other techniques. Furthermore, comparing the average rewards in Table 3.2, agent-2 performs 17.4% better than agent-1, 33.1% better than HDVS, 41.7% better than PBVS, and 40.5% better than IBVS.

We selected a random trial to demonstrate the error convergence and robot manipulability for all five VS methods. Figure 3.9 illustrates the convergence error for all VS methods. By comparing the RMSE for this trial, IBVS achieved an RMSE of 0.024; comparatively, this value was 0.041 for PBVS, 0.0276 for HDVS, 0.043 for agent-1, and 0.0224 for agent-2. The RMSE for agent-2 is even better than this value for the IBVS approach. Moreover, the task is completed in 350 iterations with IBVS, 341 iterations with PBVS, 308 iterations with HDVS, 292 iterations with agent-1, and 197 iterations with agent-2. As a result, agent-2 offers a faster solution than all other approaches, demonstrating that the trained policy with agent-2 not only inherits good IBVS performance in terms of RMSE but also learns to operate faster.

Furthermore, the manipulability of the robot arm for all five methods is plotted in Figure 3.11 for the same trial illustrated in Figure 3.9. In Figure 3.11, the manipulability of the RL methods with agent-1 and agent-2 is higher than other IBVS, PBVS, and HDVS methods. Agent-2 offers the highest manipulability compared to the other four methods in most robot

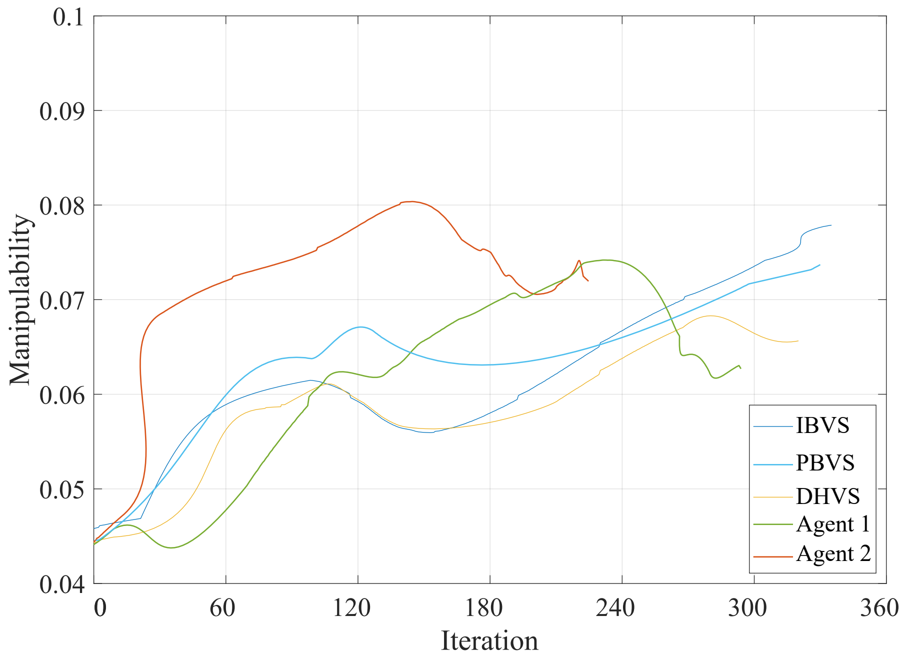


Fig. 3.11. The manipulability of five different VS methods

configurations. As a result, there would be more controllability for the robot joints using agent-2. In Figure 3.11, the mean manipulability for IBVS, PBVS, HDVS, agent-1, and agent-2 were 0.062, 0.0613, 0.642, 0.0662, and 0.0742, respectively.

3.4.2 Case 2: Extension of AORLD with TD3

Five different agents were defined and trained, and their training progress was compared. The first method, called AORLD-HL, involved constraining the agent's action space to an online convex hull generated from controller knowledge and modifying the agent's loss function to penalize actions outside the generated hyper-volume. The second method, AORLD-CL, involved generating an online hypercube to constrain the action space, and similarly modifying the loss function. The third method, AORLD-HF, involved filtering out actions suggested by the RL policy that lie outside the generated convex hull while using the standard loss function. The fourth method, AORLD-HP, involved projecting actions outside the convex hull onto the convex hull. Finally, the fifth policy was created using only RL without any demonstrator.
To make the policy robust to noise, calibration errors, and random objects in the scene, domain randomization was used. All five agents were trained for 25,000 episodes, with the initial position of the first robot randomized in each episode to generalize the trained policy. The agent restarted the episode if it met one of four criteria: (I) when the robot was close to joint limits, (II) when the features were very close to the image boundary, (III) when the robot's Jacobian manipulability was very small (less than 0.01), or (IV) when the number of steps in each episode exceeded 400. The parameters for the RL algorithm in the training were specified in Table 3.3 for all the agents.
The agents in the experiment have learned to maximize the cumulative reward over time, as shown in Figure 3.12. Among the tested methods, the TD3 agent with the AORLD-HL algorithm achieved the highest average reward of approximately -220, followed by the agent with AORLD-CL with an average reward of around -250. These two methods are effective in ensuring that the agent's actions are valid, as they enforce hard constraints. However, they require modifications to the RL algorithm, including changes to the loss function and

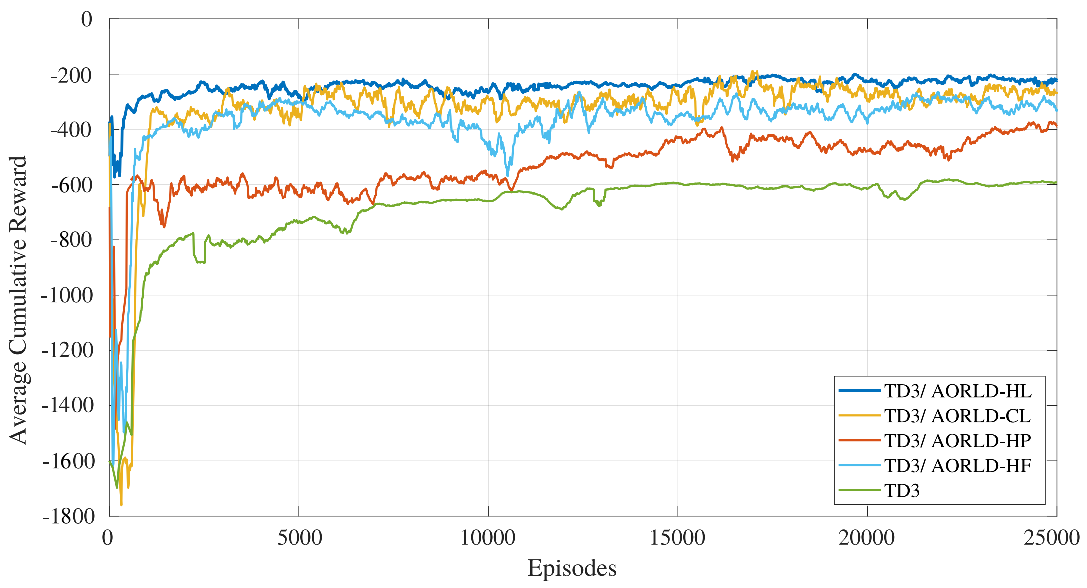


Fig. 3.12. Comparison of training progress across 5 different methods for action space constraint in reinforcement learning. The graph shows the average reward per episode for each method over the course of training. AORLD-HL and AORLD-CL show faster learning and higher average rewards, possibly due to the hard constraints on actions. AORLD-HF, AORLD-HP, and the agent without action constraints show slower learning and lower average rewards.

Table 3.3
RL and noise parameters employed in training

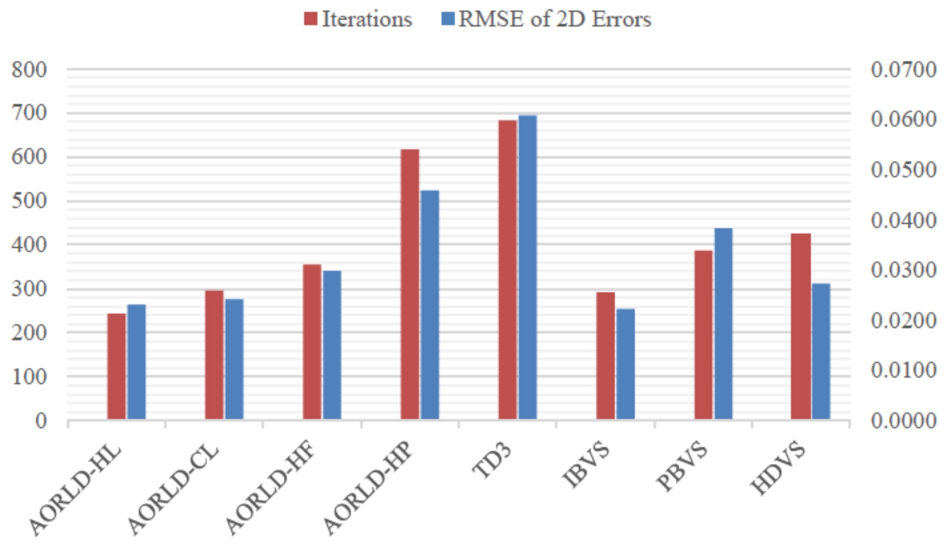
RL parameters		**Noise options**	
Target smooth factor	1e-03	Mean	0
Learning rate	[5e-04, 5e-04]	Mean Attraction constant	5
Sample time	2.5e-02	Variance decay rate	1e-05
Discount factor	0.95	Variance	0.5

target action. Furthermore, it can be inferred from the data presented in Fig. 3.12 that the AORLD-HL algorithm requires a smaller number of episodes to attain a satisfactory average reward compared to all other four methods. The less the agent must interact with the environment, the faster it will learn the task.

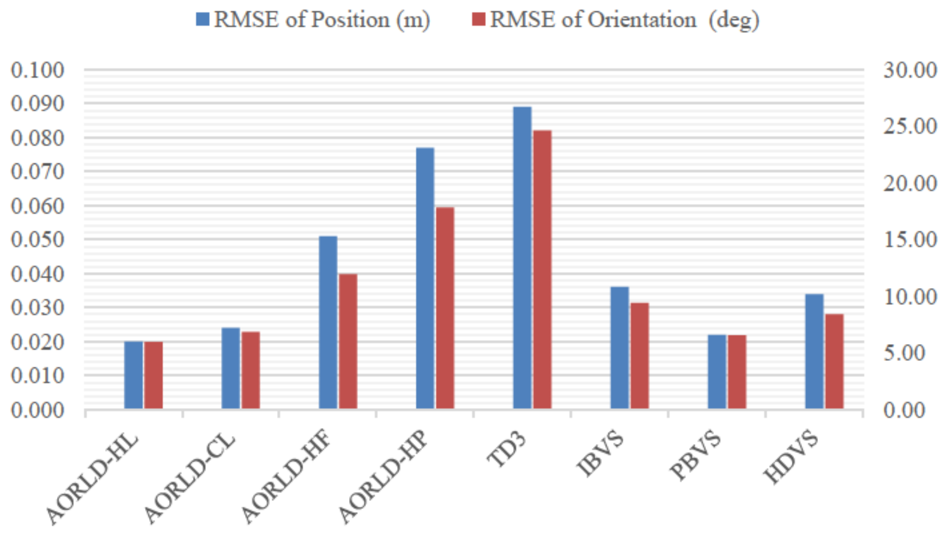
The AORLD-HF method, on the other hand, is simpler as it only allows the agent to choose valid actions without additional calculations. However, it resulted in a less effective agent, as it may limit the agent's ability to explore the state space and find optimal solutions. The average reward obtained by the agent in this method is approximately -300, as shown in Fig. 3.12. Therefore, the first two methods are more effective as they allow the agent to explore the state space while staying within the feasible action space.

The agent trained with the AORLD-HP algorithm had an average reward of around -400, which is lower than the first two methods (Fig. 3.12). This is because the projection method used in AORLD-HP may not always provide an accurate representation of the action space, especially in higher dimensions. Additionally, this method is computationally expensive since it requires projecting each action outside the hull onto the hull, and it may be less effective if the

hull is irregularly shaped or difficult to calculate. Finally, the agent without using any action constraints achieved an average reward of -600, indicating the effectiveness of the AORLD method.
In this study, we conducted a comparison between the performance of IBVS, PBVS, HDVS, and five trained policies using effective parameters. To obtain these parameters, we carried out 80 experiments with the robots starting from randomly selected initial positions, ensuring that all four features were visible in the image frame. The mean values of these parameters were then derived and reported in Figs. 3.13, 3.14, and 3.15.



(a) Iterations are taken to complete the VS task and RMSE of 2D errors in image space for eight different methods.



(b) RMSE of position and orientation errors for eight different methods in 3D task space.

Fig. 3.13. Comparison of VS performance parameters in 2D and 3D. All comparisons in these figures are based on the average of 10 trials for each method (overall 80).

Fig. 3.13a compares the number of iterations taken to complete the VS task and the root mean square (RMSE) of 2D errors in the image space for eight different methods, including AORLD-HL, AORLD-CL, AORLD-HF, AORLD-HP, IBVS, PBVS, HDVS, and the TD3. It is evident from the figure that the RMSE of errors in AORLD-HL, AORLD-CL, and IBVS are the smallest values among all the other methods, and AORLD-HL performs faster (with fewer iterations) than other methods. To have an optimised performance in VS, both image space and robot space should be taken into account.
Fig. 3.13b illustrates the RMSE of position and orientation for all eight methods in the 3D task space. From this figure, it is shown that the best performance in 3D task space is achieved by AORLD-HL and PBVS followed by AORLD-CL. This is because the RMSE of orientation and position is lower for AORLD-HL and PBVS compared to the other 6 methods.
Additionally, it is worth noting that the TD3 RL method offers the worst performance in both 2D and 3D tasks compared to the other 7 methods. This highlights the fact that the agent is more susceptible to getting stuck in local minima without using the knowledge of any other controllers or demonstrators. In other words, using the action space proposed by other controllers can significantly help the agent find its optimal solutions while avoiding unnecessary explorations.
Overall, the results presented in Figs. 3.13a and 3.13b demonstrate that the agent using AORLD-HL is highly effective for both 2D and 3D tasks, while the TD3 without using data of any demonstrators performs poorly in comparison.

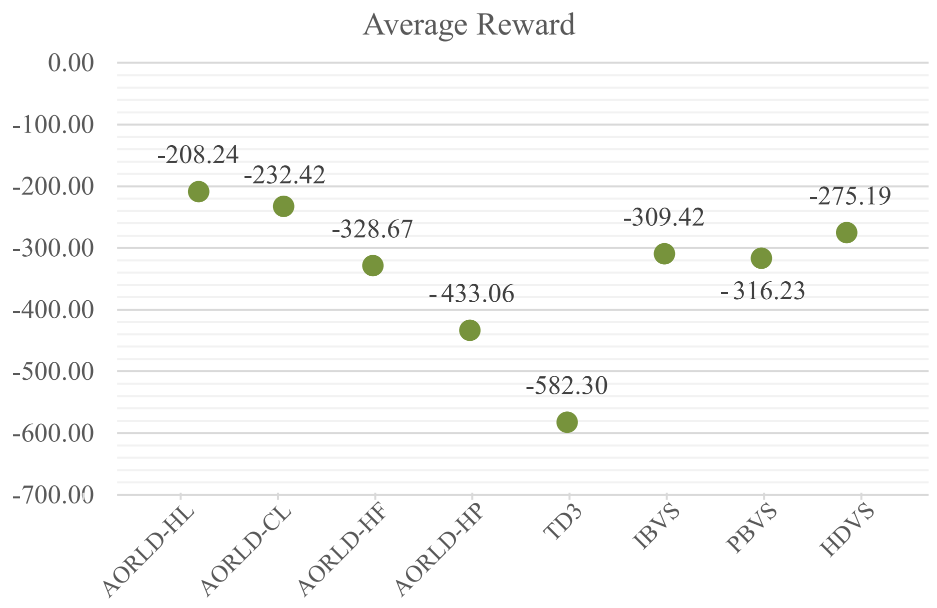


Fig. 3.14. Comparison of Average Reward for Different Methods: This figure shows a comparison of the average reward obtained by different methods. AORLD-HL offers the highest average reward followed by AORLD-CL, HDVS, IBVS, PBVS, AORLD-HF, AORLD-HP, and TD3, indicating the better performance of AORLD-HL compared to other methods.

Fig. 3.14, compared the mean average reward of different methods for the same 80 trials in Fig. 3.13. The data in Fig. 3.14 shows that AORLD-HL achieves the highest average reward of -208.24, outperforming all other methods. AORLD-CL comes in second place with an average reward of -232.42, followed by HDVS with an average reward of -275.19, IBVS with an average reward of -309.42, PBVS with an average reward of -316.23, AORLD-HF with an average reward of -328.67, AORLD-HP with an average reward of -433.06, and finally TD3 with an average reward of -582.30. The higher average reward indicates the outperforming of AORLD-HL in 2D and 3D space compared to the other methods.

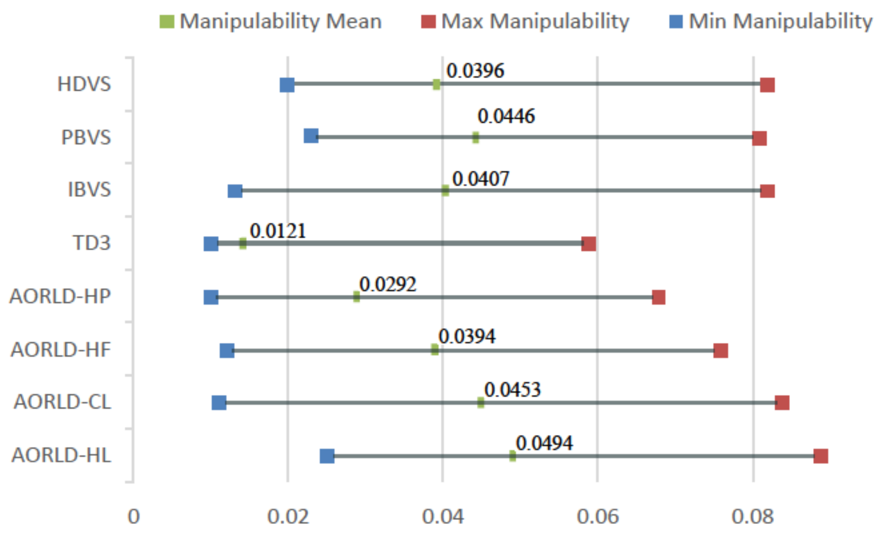


Fig. 3.15. Comparison of Manipulability values and ranges for eight methods over 80 trials.

The manipulability of a robot is a key factor in evaluating its performance in task execution. In this study, we compare the manipulability of eight different methods for robot control, as shown in Fig. 3.15. The manipulability values and ranges are calculated by averaging the tracking performance of desired features for the same 80 trials as in Fig. 3.13.
From the results presented in Fig. 3.15, it is observed that the AORLD-HL method provides the highest mean manipulability value of 0.0494, followed by AORLD-CL with 0.0453, PBVS with 0.0446, IBVS with 0.0407, HDVS with 0.0396, AORLD-HF with 0.0394, AORLD-HP with 0.0292, and TD3 with 0.0121. The higher manipulability values for AORLD-HL indicate its better performance compared to the other approaches.
Moreover, the results in Fig. 3.15 suggest that the robot has better controllability with the AORLD-HL method while tracking the desired features compared to other methods. This is evident from the fact that the lower and higher manipulability bound in agent-3 from the manipulability range is bigger than the other seven methods, indicating that the AORLD-HL method provides better control over the motion of the robot.
Overall, from the date of Fig. 3.13, 3.14 and 3.15, the TD3 agent which is trained with AORLD-HL achieves the best overall performance over the image space and Cartesian space, also suggests the best controllability compared to other approaches. In our earlier work [16], we integrated our method with domain adaptation to enable the agent to train in the real-world setting as well as simulation. However, the present study aims to compare and identify the most effective ways to constrain the action space and facilitate the exploration of potential action vectors by the agent in simulation. In future works, we plan to develop our methods with domain adaptation further and implement them on real robots.

3.5 | Conclusion

This chapter proposed a learning-based online action-policy optimizer named AORLD. The AORLD technique intelligently limits the action space of the agent, based on demonstrated actions from an ensemble of several different supervisory experts. Thereafter, the agent explores further within this constrained action space, refining its policy to become increasingly

optimal with respect to a reward function. The learning process is greatly accelerated because the policy search space has been reduced by the expert demonstrations.
We demonstrated AORLD in the context of a standard VS task, with TD3 algorithms to train the policy. IBVS, PBVS, and HDVS were defined as a set of expert supervisors for AORLD. We proposed and compared four methods to bound actions online while training. We found using the convex hull with modified loss function (AORLD-HL) is the most effective method for improving the exploration-exploitation trade-off in RL. Our experimental results demonstrate the effectiveness of these methods in improving the average reward progress during training, compared to using no bounding methods. Moreover, the agent trained with AORLD-HL achieves better overall performance in terms of feature trajectories in the 2D image plane, and also robot trajectories in the 3D task space, while also achieving higher Jacobian and manipulability of the robot throughout its motions.
Overall, our study highlights the importance of incorporating prior knowledge into the training process of RL policies to improve their performance, particularly in challenging environments with high-dimensional action spaces. The AORLD method can be used when there are multiple control methods available to serve as demonstrators (from two to arbitrarily many). AORLD finds a useful trade-off between these experts, while also incorporating the capabilities of RL to enable iterative optimising of policies with respect to a reward function. The methods presented in this study provide a promising approach for addressing this challenge and can be applied in various RL applications. In future work, we aim to introduce haptic experts in our proposed optimization method to correct and improve the control signals from a human operator during teleoperation tasks. Furthermore, AORLD could be integrated with multi-agent RL, to significantly reduce training episodes by intelligently limiting the exploration bounds of each agent.

Data availability

The data and clip supporting the findings presented in this chapter are available in the following repositories:
https://github.com/aaflakiyan/HDVS_Franka
https://github.com/aaflakiyan/RL-Visual-Servoing-Franka
https://figshare.com/articles/media/Reinforcement_Learning_from_Demonstrations_with_Visual_Servoing/25498297

CHAPTER 4

Vision Guided Contact-Rich Tasks Using Reinforcement Learning from Demonstrations

Content from this chapter has been published in the following:

Ali Aflakian, Rustam Stolkin, Alireza Rastegarpanah, "Integrating Multi-Demonstration Knowledge and Bounded Workspaces for Efficient Deep Reinforcement Learning", IEEE-RAS International Conference on Humanoid Robots, (Accepted 2023).

Ali Aflakian, Jamie Hathaway, Rustam Stolkin, and Alireza Rastegarpanah, "A curriculum-based domain randomisation approach for learning contact-rich tasks with parametric uncertainties", Journal of IEEE Access, (Submitted 2024).

4.1 | Introduction

In the previous chapters, we investigated the use of AI in improving contactless vision-guided manipulation tasks. In this chapter, we propose a framework for learning challenging contact-rich tasks with the RL algorithm. In our proposed RL method, we used a curriculum-based domain randomisation approach with a time-varying sampling distribution. As a result, the constructed policy would be robust to the parametric uncertainties in the robot-environment system. Based on evaluation in simulation for compliant path-following case studies with a random uncertain environment, and comparison with a Learning-based Model Predictive Controller (LBMPC) method and Virtual Forward Dynamics Model (FDM), the robustness of the obtained policy over a stiffness range 10^4–10^9 and friction range 0.1–1.2 is demonstrated. We furthermore trained the RL agent with various surface curvatures to enhance the robustness of the trained policy in terms of changes in surfaces. We demonstrate $\sim 15\times$ improvement in trajectory accuracy compared to the previous LBMPC method and $\sim 18\times$ improvement compared to using the FDM approach. We suggest the applications of the proposed method for learning more challenging tasks such as milling, which are difficult to model and dependent on a wide range of process variables. In another study, we proposed a novel approach for boosting deep RL using human demonstrations and offline workspace bounding. Our approach involves collecting data from human demonstrations on random surfaces with varying friction, stiffness, and surface curvatures. We then compute a 3D convex hull that encompasses all the paths taken by the demonstrators. By defining the task and the desired parameters as reward functions, we enable the RL agent to learn an optimal solution within the bounded space, significantly reducing the search space required for the agent. We compare the training progress and the behaviour of the trained policy of our approach with a baseline approach. The results demonstrate that our approach not only expedites learning but also improves the

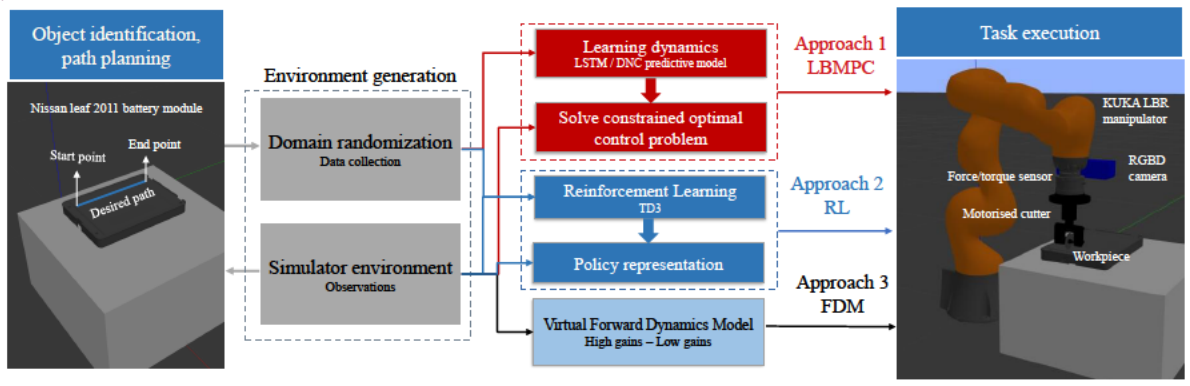


Fig. 4.1. Graphical overview of the suggested RL and comparison of path following in contact-rich tasks with MPC and FDM approaches.

policy's performance and resilience to local minima. Combining our approach with RL also enables the use of imperfect demonstrators as their behavior can be improved during the learning. Figure 4.1, illustrates the graphical overview of using three different approaches for path following in a contact-rich cutting task. Our work contributes to the growing body of research on using Deep RL for robotic manipulation tasks in uncertain environments and offers an approach for addressing the challenge of obtaining a robust policy representation with DR in complex tasks such as contact-rich manipulation tasks.
In another case study, we collect data from human demonstrations on random surfaces with random friction and stiffnesses for contact-rich path following. We then compute the 3D convex hull which encapsulates all the demonstrators' paths and use it to define a bounded workspace for RL. By doing this, we reduce the search space of the agent with an area that is more likely to contain the optimal solution and improve the sample efficiency. This approach does not require the agent to mimic any specific behaviour and is not considered imitation learning. Therefore, we can even use imperfect demonstrations since the RL policy will improve the agent's behaviour accordingly. Thanks to our proposed approach the agent would keep away from possible local minima outside the created hull. In our experiments, we compare the training progress and the behaviour of the trained policy using our method with a baseline method that uses pure RL without the bounded workspace. Our results demonstrate the effectiveness of our approach in reducing the search space of the agent and improving the sample efficiency. To ensure that the trained policy is robust and generalizable in terms of environmental parameters, we performed the training on a variety of surfaces with randomized friction and stiffness values. Specifically, we used a set of surface models and randomly varied their parameters within a certain range. This enabled the agent to learn a policy that is not only optimized for a specific environment but can also adapt to different environmental conditions. By training on a diverse set of environments, we aimed to reduce the likelihood of the agent over-fitting to specific surface properties, and instead, learn a more general and robust policy. Our approach has the potential to be combined with other methods to further boost the process of reinforcement learning, opening up new avenues for research and development in this field. Figure 4.2 provides a step-by-step process of our approach, starting from the collection of data from human demonstrations on random surfaces with varying friction and stiffness properties. It showcases the computation of a 3D convex hull that encompasses the paths taken by the demonstrators, which serves as the bounded workspace for the RL agent. The flowchart also highlights the use of domain randomization to generalize our approach to further unseen surfaces.

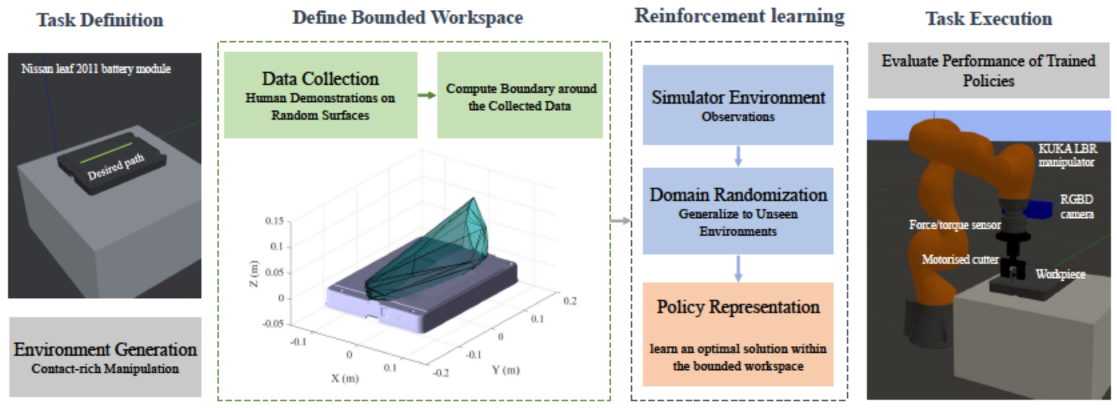


Fig. 4.2. This figure represents the sequential steps involved in the proposed approach, starting with data collection through human demonstrations on random surfaces. The data is then used to compute the boundary around the demonstrator trajectories, which defines the bounded workspace for reinforcement learning. Domain randomization is introduced to generate variations in the surface parameters, such as friction and stiffness, to enhance the agent's ability to generalize to different conditions. Reward functions are defined to guide the agent's learning process and RL is applied to learn an optimal policy within the bounded workspace.

4.2 | Materials and methods

We consider a reinforcement-learning-based approach to the case of learning contact-rich tasks for position-controlled manipulators in a simulation environment based on the twin-delayed deep-deterministic policy gradient (TD3) algorithm. For many contact-rich tasks, the desired behaviour is for the robot to track as closely as possible the desired path on the surface of an elastically compliant object with (potentially) unknown stiffness. Based on this specification, we consider the relevant raw measurements available as the tool centre point (TCP) pose, $\boldsymbol{P}$ comprising position $\boldsymbol{p}$ and ZYX Euler angle orientation $\boldsymbol{R}$, and the external measured wrench $\boldsymbol{f_e}$. Let $\boldsymbol{r}(h)$ be the position vector of a path parameterized by arc length h. Then, the path direction at any point on the path can be calculated as:

$$\hat{c}(h) = \frac{\boldsymbol{r}'(h)}{\|\boldsymbol{r}'(h)\|}$$

where $\boldsymbol{r}'(h)$ is the first derivative of the position vector with respect to arc length, and $\|\boldsymbol{r}'(h)\|$ is its magnitude. The path direction vector $\hat{c}$ is a unit vector that points in the direction of the tangent to the path at the point $\boldsymbol{r}(h)$. For simplicity, and due to the natural constraints imposed by the geometry of the tool, we consider the case of a linear reference path that will be modified to match the geometry of the surface. The path is defined by a start and end-point with position $\boldsymbol{p}_{start}$, $\boldsymbol{p}_{end}$ respectively. The path direction $\hat{c}$ is then defined as

$$\hat{c} = \frac{\boldsymbol{p}_{start} - \boldsymbol{p}_{end}}{\left\|\boldsymbol{p}_{start} - \boldsymbol{p}_{end}\right\|} \tag{4.1}$$

Although the TCP position is known directly, it is desirable to not expose its measurement directly to the agent to avoid over-fitting to specific tasks. Instead, the position is converted into a pair of task-specific features as the scalar distance (s) from the endpoint of the path:

$$s = (\boldsymbol{p} - \boldsymbol{p}_{end}) \cdot \hat{\boldsymbol{c}}, \tag{4.2}$$

and deviation from path d

$$d^2 = \left|\left|\boldsymbol{p} - \boldsymbol{p}_s\right|\right|^2 \tag{4.3}$$

The surface position estimate p_s was computed as a Gaussian weighted average of the sampled points in the depth image about the closest point on the desired path to the current TCP position [87].

4.3 | Case 1: Comparison of RL with MPC and FDM methods

The principle of the LBMPC approach [87] is to learn a model of the contact dynamics, given states, actions $\boldsymbol{x}$, $\boldsymbol{u}$ as:

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{4.4}$$

formulating the trajectory optimization as a constrained nonlinear optimization problem of some metric of cost, specified by L:

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{J}(\boldsymbol{U}) = \sum_{i=0}^{N-1} L\left(\boldsymbol{x}_{k+i}, \boldsymbol{u}_{k+i}\right) \\ \text{s.t.} \quad & \boldsymbol{x}_{k+i+1} = \boldsymbol{f}(\boldsymbol{x}_{k+i}, \boldsymbol{u}_{k+i}) \\ & \left|\left|\boldsymbol{u}_{k+i}\right|\right|_1 \leq u_{\max} \\ & i = 0, 1 \ldots N-1 \end{aligned} \tag{4.5}$$

where $\left|\left|\cdot\right|\right|_1$ denotes the ℓ^1 norm. In the LBMPC approach, the function $\boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k)$ is represented as an LSTM neural network which is trained from trajectories collected offline.
To include an additional comparison method, we have adopted the use of an FDM for contact-rich Cartesian robot control. We follow the approach described by the authors in [123], who omit the Coriolis term for simplification. The authors focus on solving Inverse Kinematics problems by assuming instantaneous motion (with zero joint velocities, $\dot{\boldsymbol{q}} = 0$) and neglecting the effects of gravity and velocity-dependent forces. This results in a simplified dynamic model which directly relates external forces to joint accelerations, given by:

$$\ddot{\boldsymbol{q}} = \mathbf{H}^{-1}\mathbf{J}^T\boldsymbol{f} \tag{4.6}$$

where $\mathbf{H}$ corresponds to the mass matrix of the robot, $\mathbf{J}$ is the Jacobian matrix, $\ddot{\boldsymbol{q}}$ represents the joint accelerations, and $\boldsymbol{f}$ is external force:

$$\boldsymbol{f} = \mathbf{K}_p\boldsymbol{e} + \mathbf{K}_d\boldsymbol{e}_d \tag{4.7}$$

while $\boldsymbol{e}$ is the distance error between the target and the current EE positions, $\boldsymbol{e}_d$ denotes the derivative of the distance error, $\mathbf{K}_p$ and $\mathbf{K}_d$ are positive definite diagonal stiffness and damping gains. As claimed by the authors in [123], omitting the Coriolis term is justified as it reduces computational complexity while still providing a practical and effective framework for solving IK problems. They have also demonstrated that the FDM approach is not only free from delays and noise but also inherently more stable in contact-rich applications compared to traditional Admittance controllers.

We explored two distinct sets of gain values, high-gain (FDM-H) and low-gain (FDM-L). We defined $\mathbf{K_p}$ as follows:

$$\begin{aligned}\text{FDM-H:}\quad \mathbf{K_p} &= \text{diag}([100, 100, 1000, 10, 10, 10])\\ \text{FDM-L:}\quad \mathbf{K_p} &= \text{diag}([10, 10, 200, 1, 1, 1])\end{aligned}$$

In both methods, we used the following value for damping gains: $\mathbf{K_d} = \text{diag}([1, 1, 1, 0.1, 0.1, 0.1])$
TD3 is an off-policy actor-critic learning algorithm. Its principle of operation is related to DDPG with key improvements in the introduction of twin critics, policy smoothing, Q-value clipping, and delayed actor updates [124]. It assumes the control problem can be modelled as a Markov decision process, in which the objective is to determine a policy that maximises an expected sum of rewards over time, weighted temporally by a discount factor. To ensure a fair comparison between the two methods, we design the reward function for the RL algorithm to be identical to the negated cost function used in the MPC approach. Hence:

$$L(\boldsymbol{x}, \boldsymbol{u}) = -r(\boldsymbol{x}, \boldsymbol{u}) \tag{4.8}$$

For the path following, based on the objectives defined in Section 4.2, we hence define the reward function r:

$$r(\boldsymbol{x}, \boldsymbol{u}) = -w_d d^2 - w_s \frac{|s|}{||\boldsymbol{c}||} - w_u \boldsymbol{u}^2 \tag{4.9}$$

where w_d, w_s, and w_u are manually tuned weighting terms. The deviation term, represented by the expression $w_d d^2$, is a scaling penalty that discourages excessive deviations from the desired path. The $w_s \frac{|s|}{||\boldsymbol{c}||}$ term, referred to as the slicing term, encourages the agent to progress along the path. The normalisation by $||\boldsymbol{c}||$ ensures the reward for path progression is independent of the path length. This reward also encodes desirable traits like productivity; as the cumulative path progress penalty is minimised by agents that rapidly reach the path endpoint. $w_u \boldsymbol{u}^2$ is a small effort penalty to discourage extreme motions, labelled the effort term. For both approaches, weighting contributions of $w_d = 1000$, $w_s = 10$, and $w_u = 0.000001$ were selected for each reward.

4.3.1 Reinforcement learning

Based on the available observations $\boldsymbol{x} = (d, s, \Delta\boldsymbol{p}, \boldsymbol{f}_e)$ we aim to learn a policy mapping $\boldsymbol{x}$ to actions in Cartesian velocity space $\boldsymbol{u}$. Each observation $\boldsymbol{x}$ was scaled to the approximate range 0–1. For TD3, we employ a set of deep feed-forward neural networks serving as the actor and dual critics respectively. The critic networks each comprise two input pathways: two hidden layers of 400 and 300 units for observations, and one of 300 units for actions, followed by a common output layer. A learning rate of 5×10^{-4} and L2 regularisation penalty of 2×10^{-4} were selected as critic network hyperparameters.
The actor network comprises 2 hidden layers of 400 and 300 units respectively. An initial learning rate of 5×10^{-4}, L2 regularisation penalty of 1×10^{-5}, batch size of 512, RELU hidden activation and tanh output activation were chosen as the network hyperparameters. The choice of tanh activation bounds the velocities by saturating the policy outputs. The remaining hyperparameters for the learning algorithm were chosen according to Table 4.1. Training was carried out up to a threshold of 3000 episodes. This threshold is established from initial experiments conducted in a simulation environment described in Section 4.3.2. Based on the actions $\boldsymbol{u}$ and sample time T_s, we convert the policy outputs into joint position commands $\boldsymbol{q}$ as:

$$\boldsymbol{q} = T_s \cdot \mathbf{J}^+ \boldsymbol{u} \tag{4.10}$$

where $\mathbf{J}^+$ is the Moore-Penrose pseudo-inverse of the manipulator Jacobian.

Table 4.1
The reinforcement learning hyperparameters and noise options used in training the TD3 policy.

RL parameters		Noise options	
Smooth factor	0.001	Mean	0
Mean attraction	2.5	Variance decay rate	0.00001
Sample time (T_s)	0.02	Variance	0.5
Discount factor	0.99		

Table 4.2
Sample space used for domain randomisation of the simulated workpiece parameters

Property	Range	Distribution
Stiffness k_p (Nm^{-1})	10^4–10^9	Log-uniform
Dyn. coeff. friction μ	0.1–1.2	Uniform

4.3.2 Domain randomisation

We consider the case of a position-controlled manipulator in the KUKA LBR iiwa R820 collaborative robot. The robot is mounted with an external force-torque sensor, an RGBD vision sensor, and a cutting tool. Experiments were carried out in the Gazebo simulator, with the workpiece being represented by a Nissan Leaf 2011 battery module as an application example. Currently, the process of battery disassembly is not fully automated chiefly due to diversity in the design of electric vehicle batteries. The proposed strategy in this study can be generalized to various battery designs (in terms of geometry and material characteristics). For simplicity, the workpiece collision surface is approximated as an elastically compliant cuboid with fixed, but unknown stiffness and isotropic dynamic coefficient of friction. The TCP is positioned at a fixed offset above the workpiece about the path start point $\boldsymbol{p}_{start}$, at which point the agent is given full control over the robot.
To learn a policy representation that is robust to an unknown environment, we establish the environment based on a curriculum-based domain randomisation method. During training, at the beginning of each episode, the properties of the object surface were sampled according to the distributions in Table 4.2. In the traditional DR, the distribution parameters are held constant from the first episode as the range specified in Table 4.2, denoted as l_+, l_- for the maximum and minimum value of a variable l. For example, in the case of the stiffness variable k_p, l_+ corresponds to the upper limit of $10^9 Nm^{-1}$ and l_- to the lower limit of $10^4 Nm^{-1}$. Similarly, for the dynamic coefficient of friction μ, l_+ is 1.2 and l_- is 0.1. These limits are used in the equations to define the range of possible values that the environment variables can take during each episode.
However, the extreme and immediate variation in the environment can greatly increase the difficulty of learning the task, and in some cases reaching the optimal reward is not possible as the learning algorithm converges to a local minimum. To combat this, we introduce the concept of curriculum-based DR. Under this approach, the full random distribution range is not immediately introduced but varied according to each episode as F_N:

$$F_N = F_0 + (1 - F_0)g(N) \tag{4.11}$$

where F_0 is the fraction of the limits at episode zero. The maximum and minimum limits for episode N, l_{N+}, l_{N-}, are computed as

$$l_{N\pm} = l_- + (1 \pm F_N)\,\frac{l_+ - l_-}{2} \tag{4.12}$$

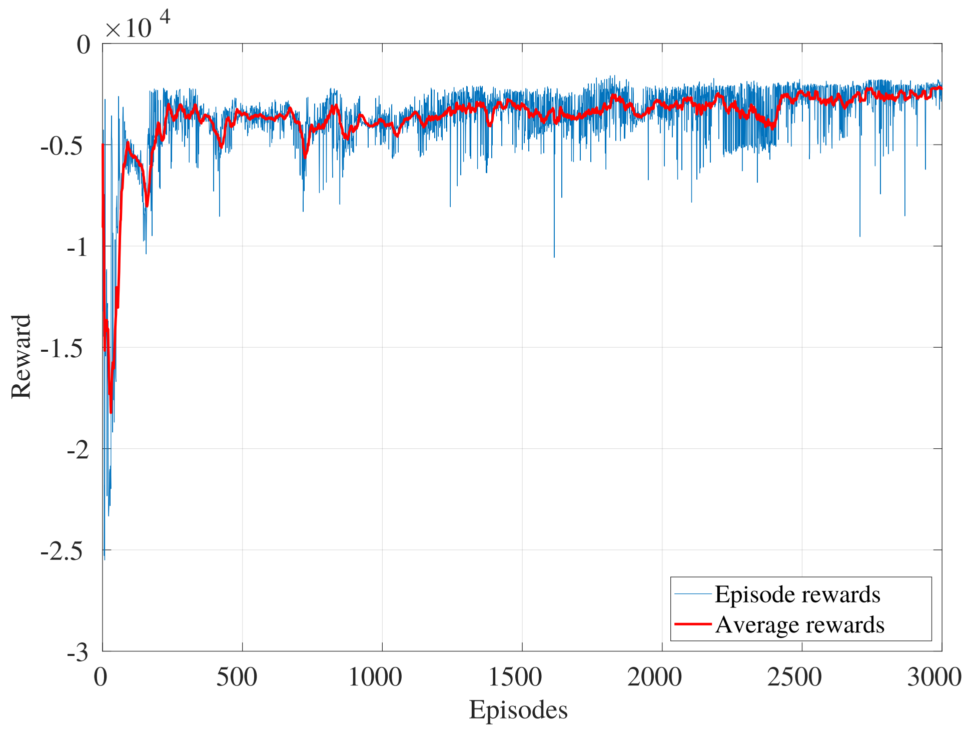


Fig. 4.3. The training graph of the TD3 agent for the compliant path following task on a near-planar surface with unknown surface properties.

Here, l_{N+} and l_{N-} represent the dynamically adjusted upper and lower limits for the parameters in Table 4.2 as the training progresses. For instance, as F_N increases over episodes, the range for k_p gradually expands from an initial smaller range towards the full 10^4 to 10^9 range, thereby easing the learning process.
$g(N)$ is an envelope function that specifies the evolution of the randomisation distribution over the training process. In this study, we select $g(N)$ as a linear function of N.

$$g(N) = \frac{N}{N_{max}} \tag{4.13}$$

This gradual adjustment of the randomisation limits allows the learning algorithm to progressively adapt to the full range of environmental variations, thereby increasing robustness without overwhelming the system with abrupt changes.

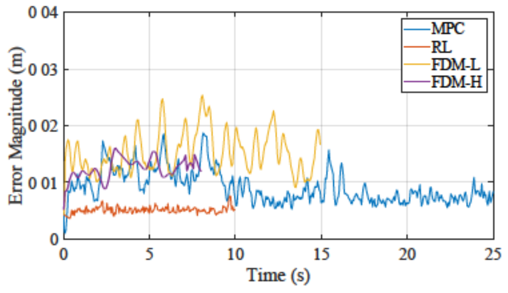
4.3.3 Experiments

We evaluate the trained agents in the simulation environment discussed in Section 4.3.2, for the task of a compliant path following along the surface of a given workpiece. Training for the TD3 agent with the curriculum DR approach was carried out over 4.05×10^5 seconds for 3000 total episodes. The hyperparameters N_{max} and F_0 were chosen as 2500 and 0.05 respectively. The processor of the computer used for simulation and training was an Intel(R) Core(TM) i7-8086K 8-core processor with a 4 GHz base clock and 32 GB RAM. The training graph is shown in Figure 4.3. The agent rapidly converges to an average reward of approximately -2500 and remains close to this value which illustrates that the desired task behaviour was successfully learned.
Based on the learned policy representation from the curriculum DR method, we evaluate the performance of the agent over four path-following case studies with randomly chosen surface properties, shown in Table 4.3. For comparison, we employ a method based on LBMPC

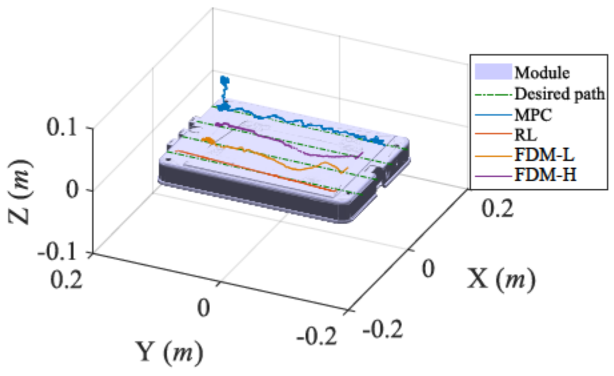
Table 4.3
The stiffness and friction coefficients used in different case studies.

Experiments	Stiffness k_p (Nm^{-1})	Friction μ
Case study 1	9.22116×10^7	0.95413
Case study 2	2.19674×10^6	0.797441
Case study 3	5.06996×10^5	1.32379
Case study 4	9.79874×10^8	1.40105

described in our previous work [87], with data collected using a series of manually designed admittance controllers to train a predictive model of the surface contact dynamics. Due to the difficulty of solving for the optimal trajectory $\boldsymbol{U}$ directly, we employ the forward shooting method using sample-based optimisation to approximate the solution of (4.5). For LBMPC a dataset of 101120 samples was collected, which is comparable to the number of observations exposed to the agent after ~200 episodes of training. The average reward displayed in Figure 4.3 demonstrates that the agent experiences the majority of its performance improvement before completing 200 episodes of training. This is notable because the dataset size used for LBMPC consists of roughly the same number of observations as those encountered by the agent in this early phase of training, indicating that the dataset is a reasonable size for comparison of LBMPC and RL methods. This establishes a benchmark that is less sample-intensive than the exploration required for RL but has greater computational overhead.
The magnitude of tracking error and cutting path for LBMPC, TD3 with curriculum DR, FDM-H, and FDM-L during each example case study are presented in Figure 4.4–4.7. Figures 4.4a–4.7a illustrate the magnitude of trajectory errors, represented as the norm of the error vector in the x, y, and z coordinates, which captures the difference between the current and desired tool TCP positions.
In Figure 4.4a, for case study 1, the task was completed in approximately 25 seconds using MPC, 10 seconds with the trained RL agent, 15 seconds with the FDM-L method, and 6 seconds with the FDM-H method. This comparison highlights that RL achieves faster task execution compared to both MPC and the FDM-L approach. However, the trajectory error of the RL method is significantly lower than that of the other three methods. This suggests that the RL agent is more effective at completing the task compared to the MPC and FDM methods. Figure 4.4b displays the 3D path of the tool-tip for case study 1, comparing the performance of the RL method with others. Although all methods exhibit attempts to correct any deviation from the path, the deviation is less noticeable for the RL method. This observation is supported by the root mean square tracking error (RMSE) between the end-effector position and the desired path. The RMSE was 7.2mm for MPC, 9.6mm for FDM-L, and 9.0mm for FDM-H, greatly exceeding the corresponding RL value of 0.56mm.
The results for case study 2 are depicted in Figure 4.5, where it can be observed that the tool requires approximately 23 seconds to reach the endpoint when using MPC, 15 seconds with FDM-L, approximately 9 seconds with FDM-H and the TD3 agent completes the task in 10 seconds. The performance of the RL agent for case study 2 can similarly be compared by analyzing the 3D TCP path, as shown in Figure 4.5b. Despite all approaches attempting to correct any deviation from the desired path, the deviation is again less prominent in the RL method. This finding is further supported by the RMSE tracking error, where the FDM-L method exhibits the highest RMSE value of 11.1mm, followed by MPC with an RMSE of 10.4mm. In contrast, the FDM-H method achieves a lower RMSE of 4.5mm, while the RL method stands out with a notably lower RMSE value of 0.84mm.

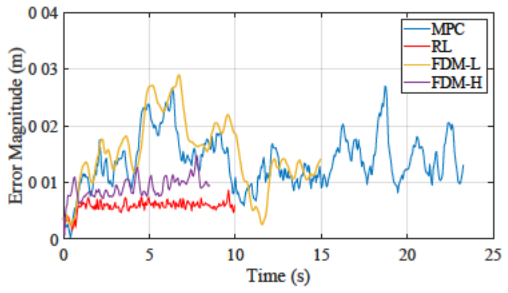


(a) Trajectory error

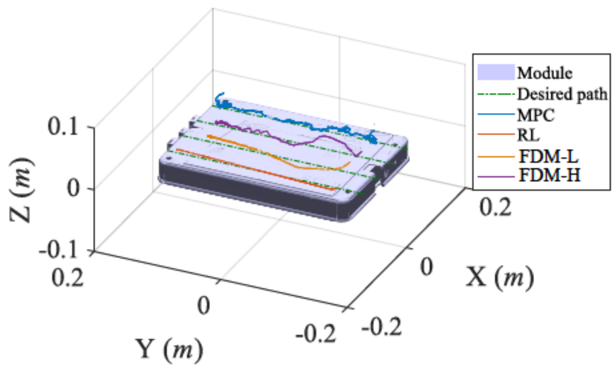


(b) 3D tool path

Fig. 4.4. Case study 1: material stiffness $k_p = 9.22116 \times 10^7 (N/m)$, friction coefficient $\mu = 0.95413$.



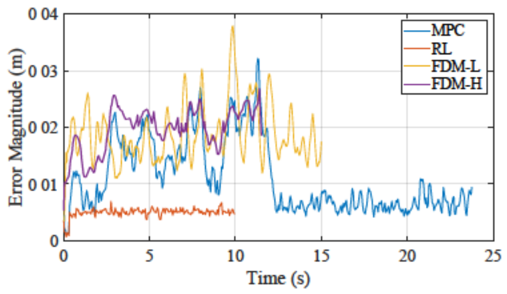
(a) Trajectory error



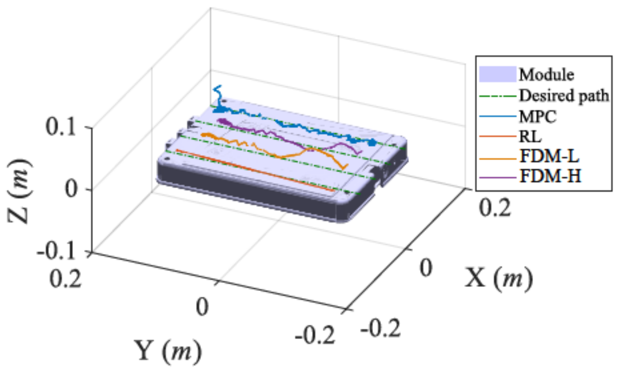
(b) 3D tool path

Fig. 4.5. Case study 2: material stiffness $k_p = 2.19674 \times 10^6 (N/m)$, friction coefficient of $\mu = 0.797441$.

Figure 4.6 presents the outcomes of case study 3, with the endpoint again reached in roughly 23 seconds when using MPC, 15 seconds using the FDM-L method, roughly 12 seconds when using FDM-H and 10 seconds when using the trained RL agent. From Figure 4.6a it is also evident that the magnitude of the trajectory error with the trained RL agent is lower than that of the others during path following. The 3D path of the tool-tip in Figure 4.6b provides a basis for comparing the performance of the RL agent with other methods in case study 3. All methods make an effort to correct any deviations from the intended path, but the deviations are less in the RL method. This observation is consistent with the RMSE values, which were significantly lower for RL (0.34mm) compared to the MPC (9.6mm), FDM-L (10.8mm), and FDM-H (9.7mm). Finally, comparing trajectory errors in Figure 4.7, the RL agent performed 33% faster (10 seconds) than MPC and FDM-L methods (15 seconds). Notably, the RL method was only 1 second slower than the FDM-H method, which completed the task in 9 seconds. It is also illustrated from Figure 4.7 that the error magnitude during path following using the trained TD3 agent is lower than that of the other methods. Figure 4.7b similarly demonstrates the desired path is tracked more closely with RL, with a lower RMSE tracking error (0.11mm) compared to MPC (7.6mm), FDM-L (9.8mm), and FDM-H (7.9mm).

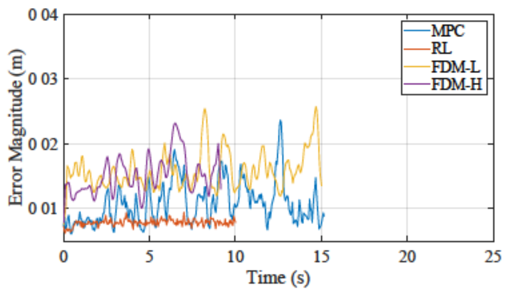


(a) Trajectory error

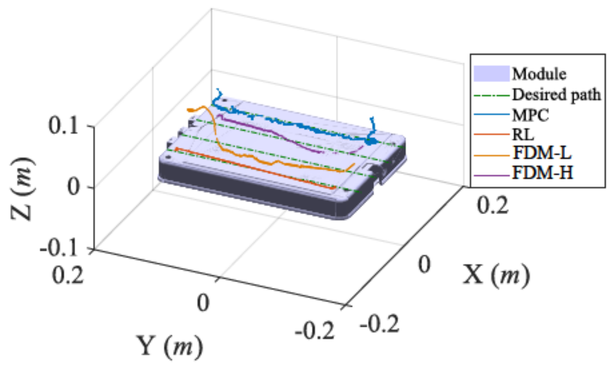


(b) 3D tool path

Fig. 4.6. Case study 3: material stiffness $k_p = 5.06996 \times 10^5 (N/m)$, friction coefficient $\mu = 1.32379$.



(a) Trajectory error



(b) 3D tool path

Fig. 4.7. Case study 4: material stiffness $k_p = 9.79874 \times 10^8 (N/m)$, friction coefficient of $\mu = 1.40105$.

In summary, the results of the four case studies in Figure 4.4–4.7, demonstrate that although all methods are capable of accomplishing the task without prior knowledge of the surface properties, the RL agent outperforms other methods in terms of both speed and effectiveness. This provides evidence for the superiority of the RL agent over MPC and FDM, indicating that the use of RL in completing similar tasks may lead to significant improvements in performance. Moreover, A primary issue with the MPC-based framework is the high computational complexity of the LBMPC method, and hence only approximate solutions for the optimal trajectory may be found at each time step. The degree of variability this introduces leads to the controller exploring areas of the action space for which there is lower confidence in the model predictions, as noted in [83], as they are conditioned on the training data obtained by pre-defined interactions with the system , which are collected offline. In addition, the FDM approach is susceptible to difficulties related to choosing suitable stiffness and damping gains and exhibits a high sensitivity to these parameter values.

4.4 | Case 2: Extension to unknown, non-planar surfaces

We extend the first presented task, considered for the case of unknown material properties with known surface geometry to the more general case where material properties and surface position are both unknown. We procedurally generate height-field surfaces alongside randomizing stiffness and friction, while modulating the contact force to avoid damage to the tool or workpiece. In doing so, we establish the capability of the trained policy to generalize to various types of surfaces besides the presented planar surface case studies. While in the first instance, the TCP orientation $\boldsymbol{R}$ was excluded, for the case of an unknown, non-planar environment, the rotation encodes useful information about the point of contact and external torques acting on the tool, particularly in the case of loss of visual feedback or occlusion of the surface geometry. We therefore extend the observations from Task 1 to include the TCP orientation $\boldsymbol{R}$, as $\boldsymbol{x} = (d, s, \Delta\boldsymbol{p}, \boldsymbol{f}_e, \boldsymbol{R})$ where the sine and cosine of each Euler angle component were taken as the scaled orientation inputs. Training for the TD3 agent with the curriculum DR and randomized heightmaps was carried out over 2.81×10^5 seconds for 2000 total episodes. Similarly to Section 4.3, the hyperparameters for TD3 were established by manual search. The TD3 hyperparameters and noise information are summarized in Table 4.5.
The problem of reward function selection is a further necessary and challenging part of task specification. Based on observations in equations (4.1), (4.2), and (4.3), we extend the definition for the agent reward function r as:

$$r = -w_d d^2 - w_s \frac{|s|}{||\boldsymbol{c}||} - w_u \boldsymbol{u}^2 + w_c C - w_f \left(\max\left(f_{max}, ||\boldsymbol{f}_e||\right) - f_{max} \right) \tag{4.14}$$

where w_d, w_s, w_c and w_f are manually tuned weighting terms. $w_d d^2$ and $w_s \frac{|s|}{||\boldsymbol{c}||}$ are deviation and slicing terms explained in (4.9). While these reward contributions alone may be sufficient for unconstrained path following, in the presence of path planning errors presented by an unknown or approximately known surface geometry, it is necessary to ensure the robot does not apply excessive force to the environment to avoid tool breakage or fail to accomplish the desired tasks by avoiding the surface entirely. This is accomplished by the latter 3 terms. $w_u \boldsymbol{u}^2$ is a small effort penalty to discourage extreme motions. C is a discrete reward contribution encouraging the agent to establish contact with the environment, defined as:

$$C = \begin{cases} 1 & \text{if } f_z > f_{min} \\ 0 & \text{otherwise} \end{cases} \tag{4.15}$$

We introduce a ramping force penalty that penalises forces $\left(w_f \left(\max\left(f_{max}, ||\boldsymbol{f}||\right) - f_{max} \right)\right)$ in excess of a target threshold f_{max}. Finally, for training, an additional terminal penalty is applied, defined as r_{term} in the case of early episode termination, and 0 otherwise. Without this penalty, the cumulative reward may converge to a local minimum corresponding to the agent immediately pursuing the episode termination criteria, versus the case of a prolonged episode, where the cumulative penalties due to path deviation or excessive force may be higher. The terminal penalty was chosen to be sufficient to surmount any negative cumulative reward expected during the prolonged episode. The chosen weighting contributions and termination penalty for each reward are shown in Table 4.4. The movements of the tool are furthermore bounded via workspace limitations about the desired path, which we employ in both task space and joint space.

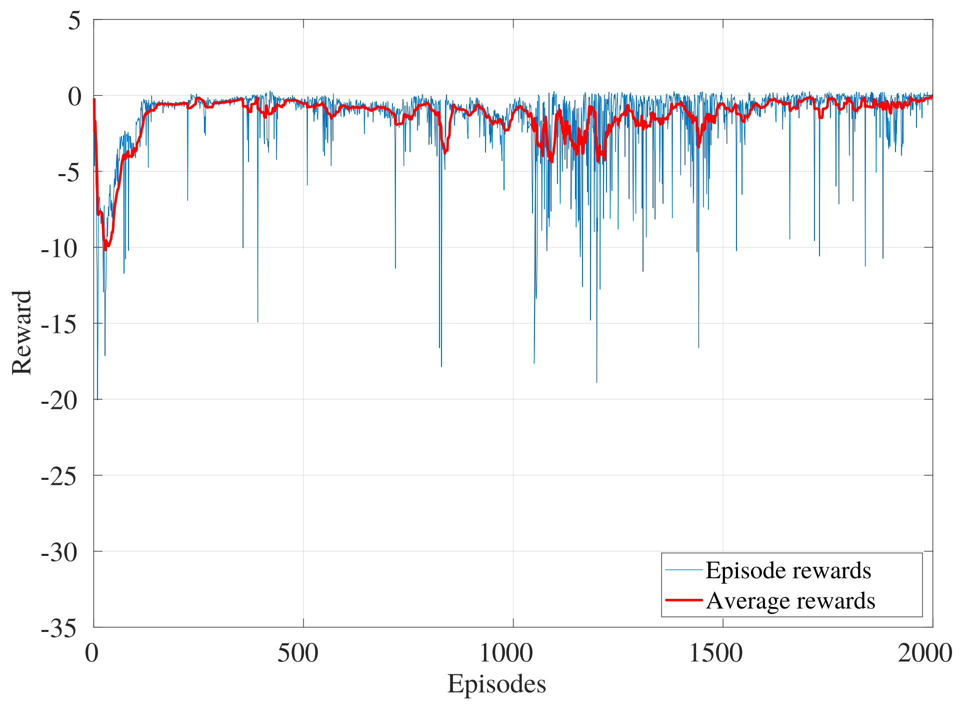
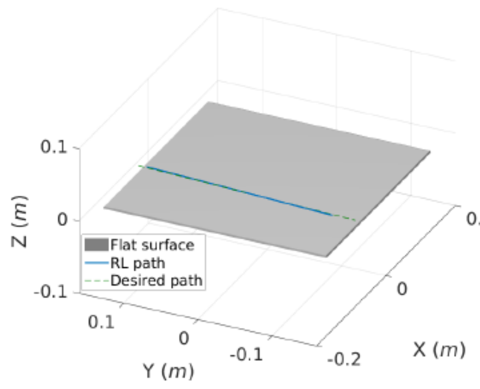


Fig. 4.8. Training graph of TD3 agent for compliant surface path following for the case of unknown surface properties and unknown (heightmap) surface geometry.

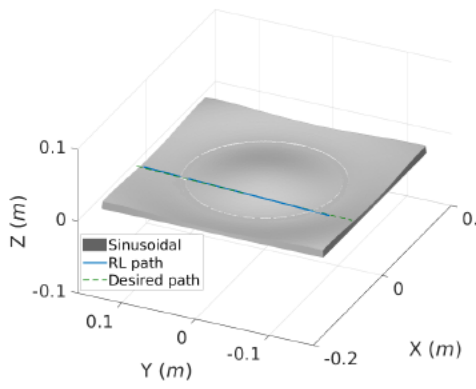
Table 4.4
Reward weighting contributions for the reward function defined in (4.14).

Weighting	Value
Path deviation w_x	1000
Path progress w_s	2
Effort w_u	0.3
Contact w_c	0.75
Force limiting w_f	0.006
Termination penalty r_{term}	400

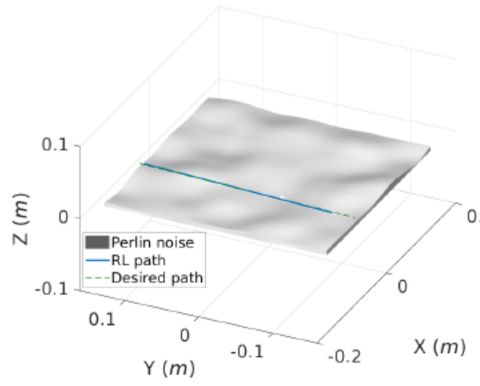
Training was halted when the agent reached the threshold of 2000 episodes and the training progress graph is shown in Figure 4.8. Similarly to the training on the planar surface in Section 4.3, the performance rapidly converges within the first ~ 250 episodes. For the remainder of training, the reward remains approximately constant, with a progressive degradation of performance from 500–1250 episodes as the range of surface properties is introduced by the curriculum schedule. Finally, the performance recovers for the remaining episodes, indicating successful learning of the task. Figure 4.9 illustrates the trajectory of the cutter TCP with the trained RL algorithm on different height maps. In the case of path planning errors or loss of visual feedback, the reference path may be defined slightly below the object surface. Hence, perfect tracking of the path cannot be achieved without violating the force limiting objectives defined in (4.14). However, the trajectories of the tool TCP in Figure 4.9 demonstrate the proposed method results in a learned policy that is robust to an uncertain environment for a variety of surface types.



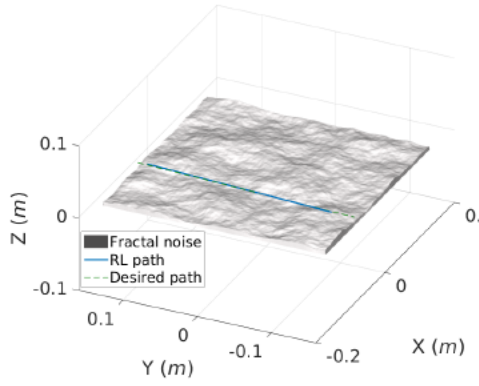
(a) Flat surface



(b) Sinusoidal surface



(c) surface with perlin noise



(d) surface with fractal noise

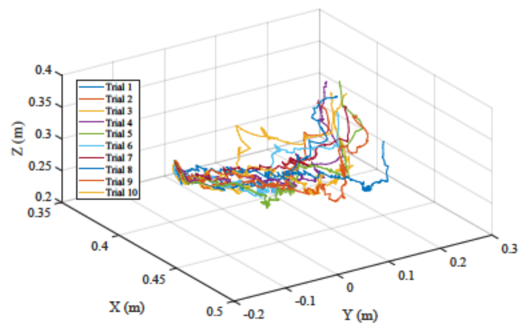
Fig. 4.9. 3D perspective view of the cutter TCP path with the curriculum DR trained TD3 agent for different generated height maps.

Table 4.5
The reinforcement learning hyperparameters and noise options used in training the TD3 policy.

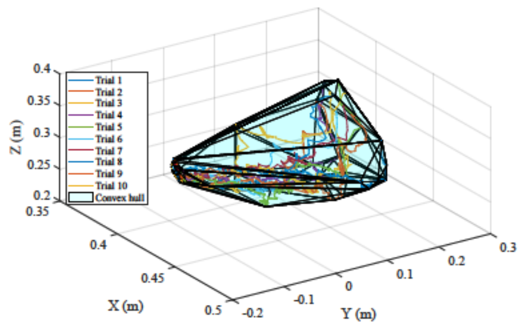
RL parameters		Noise options	
Smooth factor	0.001	Mean	0
Learning rate	5e-04	Mean attraction	2.5
Sample time (T_s)	0.02	Variance decay rate	0.00001
Discount factor	0.99	Variance	0.5
Mini batch size	512		

4.5 | Case 3: Integrating multi-demonstration knowledge to RL

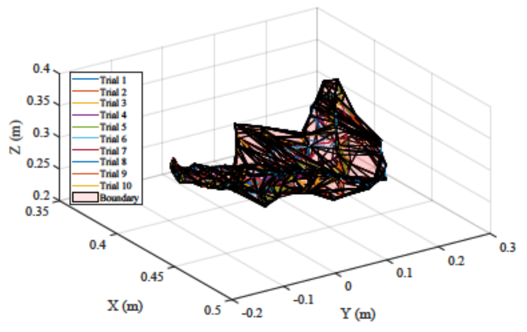
During the evaluation of our proposed approach, we compared four different methods of workspace bounding, along with a baseline approach that solely relied on traditional reinforcement learning (RL) methods. Not to mention that we used domain randomization in all four cases. The methods of workspace bounding included: (1) Convex Hull, (2) Boundary with a shrink factor of 0.4, (3) Shrink factor of 0.1, and (4) Baseline RL. Fig 4.10, visually illustrates the different processes of defining boundaries around the human demonstrator trajectories. The boundaries were computed employing MATLAB's "Convhull" and "Boundary" functions [125, 126].



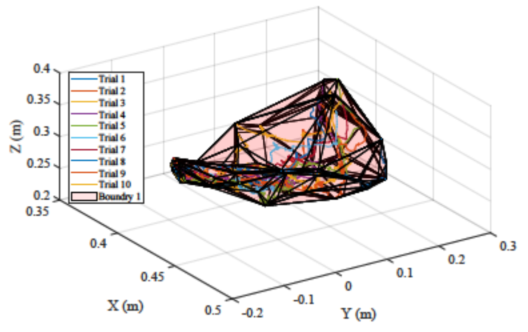
(a) Multiple trajectories were taken by human demonstrators.



(b) The computed convex hull that encompasses all the demonstrators' trajectories, represents the overall workspace (used in case 1).



(c) The boundary around the trajectories with a shrink factor of 0.4, reduced the workspace (used in case 2).



(d) The boundary around the trajectories with a shrink factor of 0.1, restricts the workspace (used in case 3).

Fig. 4.10. Defining boundaries around the human demonstrator trajectories.

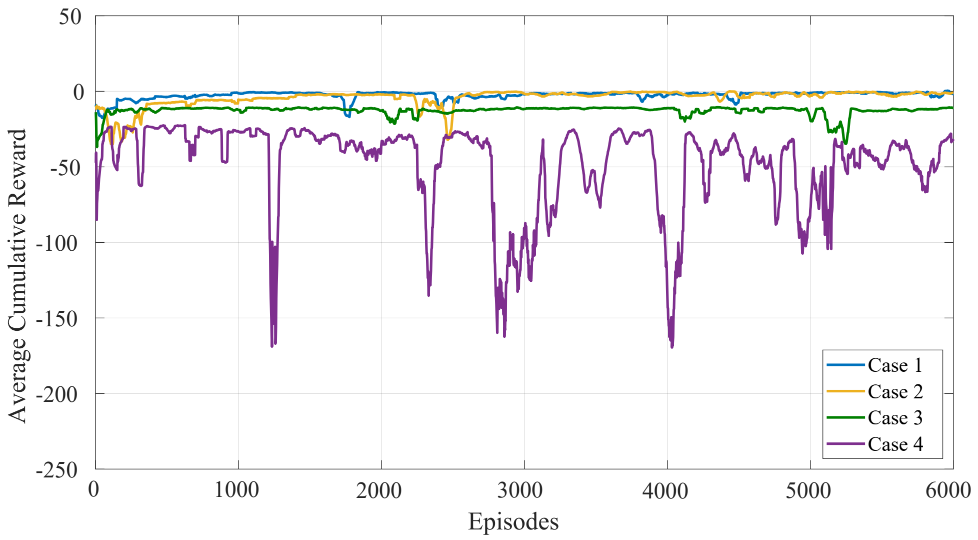


Fig. 4.11. A comparison of training progress: Average cumulative rewards for different workspace bounding methods.

We analyzed the performance of the four methods using a plot depicting the cumulative average rewards over the course of training in Fig 4.11. The x-axis represents the number of episodes, while the y-axis represents the cumulative average reward. The method utilizing Convex Hull workspace bounding (case 1) exhibited the fastest learning progress in Fig. 4.11, reaching the optimal average reward of approximately 0 after only 900 episodes ($\approx 3.6 \times 10^5$ steps). This method provided a workspace area of 0.0016m^3, enabling the agent to explore a range of possibilities in this area. The method employing the boundary with a shrink factor of 0.4 (case 2) achieved the same average reward as case 1 but at a slower learning pace. It reached the optimal reward of 0 after approximately 1500 episodes ($\approx 6 \times 10^5$ steps). The workspace area for this method was 0.0011m^3, slightly smaller than case 1 but still offering a substantial exploration space. On the other hand, the method with a shrink factor of 0.1 (case 3) faced challenges in achieving satisfactory results. It became trapped in a local minima, resulting in an average reward that failed to surpass -10. The significantly reduced workspace area of 0.00062m^3 may have limited the agent's ability to discover optimal solutions. Lastly, the baseline approach (case 4), which relied solely on RL without any workspace bounding, struggled to learn the task effectively. Even after 6000 episodes, it failed to achieve the average reward of -25.

Table 4.6
The average cumulative rewards obtained from 20 randomized trials for each case study.

Method	Slicing	Deviation	Contact/Force	Overal Reward
Case 1	-0.0581	-0.0075	0.0210	-0.0391
Case 2	-0.1993	-0.0091	0.0062	-0.2335
Case 3	-4.7807	-0.7230	-0.0089	-7.6126
Case 4	-8.0741	-3.0414	-0.0782	-22.2341

Table 4.6 displays the average cumulative rewards obtained from 20 randomized trials for each of the four methods employed to train the RL agent and confine its workspace boundaries. Figure 4.12 also presents a bar plot for a visual comparison of the performance of each

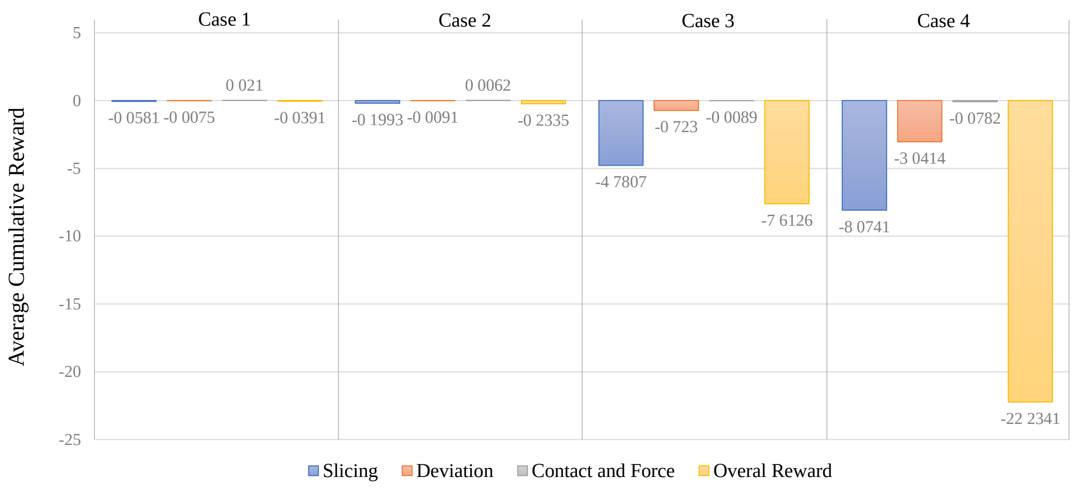


Fig. 4.12. Comparison of the average cumulative rewards obtained from 20 trials across different methods employed in the experiment.

method based on the slicing, deviation, contact/force, and overall rewards. The slicing term is represented by the blue bar in the plot 4.12, reflecting the agent's progress along the desired path and its ability to efficiently reach the endpoint. Therefore, a higher amount (a smaller negative value as observed in Case 1) indicates a better performance in terms of making progress along the path. Case 2, Case 3, and Case 4 follow in decreasing order of performance in this regard. The red bar in Fig. 4.12 represents the deviation reward, which penalizes the robot for deviating excessively from the intended path, with the penalty scaled by the squared distance between the current TCP position and the path. Similarly, a smaller negative value (best seen in Case 1) indicates a better performance in terms of staying close to the desired path. Case 2, Case 3, and Case 4 exhibit progressively lower deviation rewards. The contact/force reward (grey bar in Fig. 4.12) promotes contact between the robot and the environment while ensuring that excessive force is not applied, preventing tool damage or failure to accomplish the task. A higher positive value indicates a better performance in establishing appropriate contact and force interactions. Once again, Case 1 demonstrates the highest contact/force reward, followed by Case 2, Case 3, and Case 4. The overall reward encompasses the cumulative effect of the aforementioned rewards along with the effort term in equation (4.14). It represents the comprehensive performance measure for each method. The results highlight that utilizing a convex hull to limit the agent's workspace (Case 1) yields the highest overall reward, indicating superior performance compared to the other three methods (Case 2, Case 3, and Case 4).

There is no mathematical proof that our method will keep the agent away from local minima outside the created hull. However, by reducing the search space to a bounded region, the agent is forced to explore a smaller and more relevant part of the state space, which may increase the likelihood of finding a better solution and reduce the chance of getting trapped in a local minimum outside the created hull. Additionally, the hull provides a way to constrain the solution space, which can prevent the agent from exploring irrelevant parts of the state space. These results further emphasize the efficacy of our proposed approach and its potential to enhance the development of deep RL.

4.6 | Conclusion

In this work, we proposed a TD3 agent with curriculum-based DR to learn contact-rich path following with parametric uncertainties in the interaction contact dynamics. We specifically considered the case of a robotic path following along a workpiece with unknown stiffness and isotropic friction over a range of values. By validating our approach with four random case studies in simulation, we demonstrated the robustness of the learned policy representation to unknown environments. Comparison with an earlier approach using learning-based model predictive control and a virtual forward dynamic model illustrates RL superior task performance with improvement in tracking error; the LBMPC approach suffers due to computational complexity and the problem of adequate domain coverage in the training dataset when employing established expert policies for data collection. Furthermore, the FDM approach is vulnerable to challenges associated with selecting appropriate stiffness and damping gains and is highly sensitive to these parameter values. Overall, the RL approach shows approximately 15 times improvement over LBMPC method and 18 times improvement over the FDM approach. We extend this concept by procedurally generating height-field surfaces alongside randomizing stiffness and friction, during the online training, which allowed us to generalize the trained policy for various types of surfaces beyond planar surfaces to environments with unknown surface geometries and path-planning errors.
A notable limitation of the current work is sensor limitations, particularly with loss of visual / depth feedback with reflective or occluded surfaces, or close to the target surface. Although the method compensates for vision feedback loss by incorporating both visual and tactile modalities for path following, the so-called "reality gap" between simulations remains a challenge. To directly bridge this gap, RL methods rely on the exploration of the target environment, which is costly in a real setup. Therefore, future work will focus on addressing domain adaptation of the proposed method to a range of target domains, including real-world applications. We have also proposed an offline workspace bounding approach for accelerating the process of deep RL using human demonstrations. Our approach reduces the search space and keeps the agent focused on the most promising areas of the state space, which in turn can lead to faster convergence to the optimal solution while avoiding possible local minimas outside the reduced workspace. We evaluated three different methods for limiting the workspace boundary and training a RL agent. The methods aimed to optimize performance metrics related to slicing, deviation from the desired path, contact, and force interactions. Our results demonstrate that incorporating a convex hull to limit the workspace boundary yields the most favourable outcomes across all performance metrics. The agent trained using this method consistently achieved higher slicing rewards, indicating better progress along the desired path. Additionally, it exhibited reduced deviation from the path, as evidenced by the smaller negative values in the deviation reward metric. Moreover, it successfully established optimal contact and force interactions with the environment. The overall reward, which accounts for all performance metrics and the effort term, further supports the superiority of the convex hull method. Our findings highlight the importance of carefully considering the workspace boundary limitations and their impact on RL agent training.
In future research, it would be beneficial to conduct studies on the generalization capabilities of our method to various configurations and applications.

Data availability

The data and clip supporting the findings presented in this chapter are also available in the following repositories:

https://github.com/aaflakiyan/RL-Contact-Rich-Path-following
https://github.com/aaflakiyan/TouchX-Teleoperation-Matlab-ROS
https://figshare.com/articles/media/RL_MPC_Cutting/25498342

CHAPTER 5

Experiments and Demonstrations: Disassembly of Battery Pack

Some content from this chapter has been published in the following:

Ali Aflakian, Rustam Stolkin, Alireza Rastegarpanah, "Integrating Multi-Demonstration Knowledge and Bounded Workspaces for Efficient Deep Reinforcement Learning", IEEE-RAS International Conference on Humanoid Robots, (Accepted 2023).

Hathaway, J., Shaarawy, A., Akdeniz, C., **Aflakian, A.**, Stolkin, R. and Rastegarpanah, A., 2023. Towards Reuse and Recycling of Lithium-ion Batteries: Tele-robotics for Disassembly of Electric Vehicle Batteries.(Accepted 2023) :2304.01065.

Rastegarpanah, A., Mineo, C. , Contreras, C.A, **Aflakian, A.**, Paragliola, G., Stolkin, R. "Electric Vehicle Battery Disassembly Using Interfacing Toolbox for Robotic Arms", Journal of Batteries (MDPI), (Submitted 2024).

5.1 | Introduction

Chapter 5 of this thesis provides a comprehensive examination of the experimental setups and tasks involved in the disassembly of batteries employing a diverse array of robotic systems. The chapter commences with an overview of the experimental configurations, encompassing robots, grippers, haptic devices, vision systems, and the integration of deep learning techniques. Additionally, it explores the utilization of a model-based tracker to localize the position of each component in 3D space based on 2D images.
The subsequent section delves into specific disassembly tasks, including sorting, unbolting/unscrewing, cutting, and teleoperation. Each task is described, along with its objectives, challenges, and the important roles played by various components in their execution. Each task is further illustrated by a collection of images that offer visual context to both the setups and the disassembly procedures.
Each setup further explores its development into the suggested vision methodology, model-based tracking, and deep learning techniques. In the Model-based tracker, the chapter investigates the employment of component models to determine their 3D positions from 2D images. In the Deep NN section, the focus shifts to the application of deep neural network models trained using transfer learning. These models are designed to autonomously detect objects in the camera scene and localize the borders of rectangles encapsulating the objects.
The case study of battery disassembly is presented, which includes a variety of disassembly tasks like unbolting, sorting, and cutting, following the sequence of operations required for

pack-to-cell disassembly of a Nissan Leaf 2011. A concise overview of the process involved in disassembling an EV battery, from the battery pack down to the individual battery cell is illustrated in Fig 5.1.

In conjunction with the traditional disassembly methods, which rely on predetermined component positions for task execution, our study delves into innovative approaches for the integration of vision feedback. Some of these innovative approaches are explored in prior chapters. In this chapter, we also employ vision in two additional capacities. One of these approaches involves model-based tracking, where we use the built-in knowledge of how things should look to figure out exactly where they are in 3D space (3D model of components) from 2D images. This helps us understand where each part is more precisely and makes the disassembly process based on that. Furthermore, we explore the use of deep learning, to automate the process of object detection. Specifically, we employ transfer learning, a technique enabling the utilization of pre-existing knowledge from trained neural network models. This approach facilitates the development of sophisticated neural network models capable of autonomous object detection within the camera scene and outlining object boundaries.

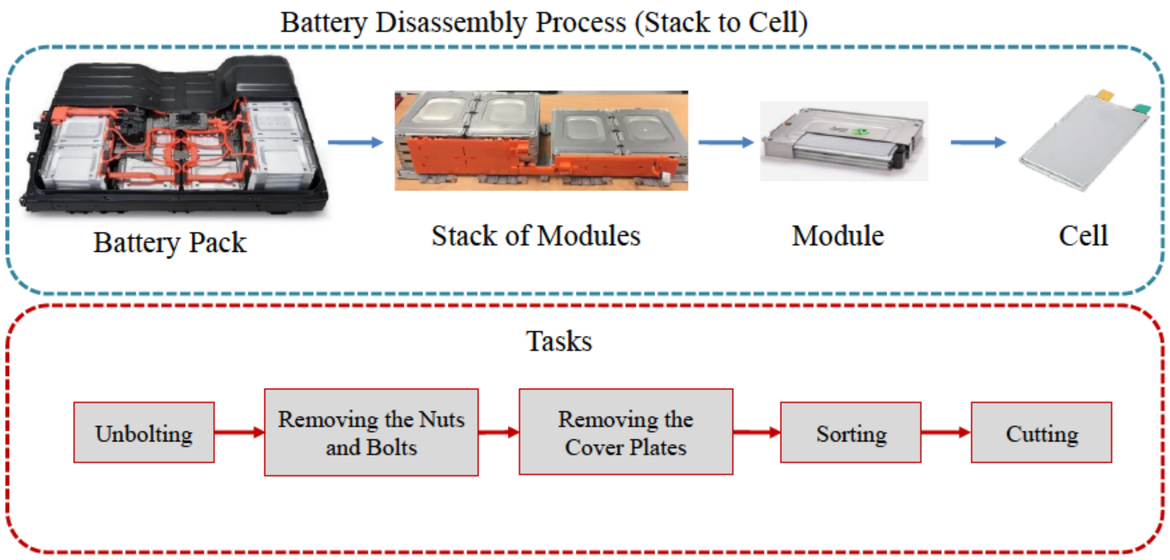

Battery Disassembly Process (Stack to Cell)
Battery Pack
Stack of Modules
Module
Cell
Tasks
Unbolting
Removing the Nuts and Bolts
Removing the Cover Plates
Sorting
Cutting

Fig. 5.1. A brief breakdown of Nissan leaf battery pack disassembly process.

As mentioned, this research centres on the 2011 Nissan Leaf battery pack, serving as a case study to illustrate various disassembly tasks. The 2011 Nissan Leaf pack is composed of 192 cells distributed within 48 modules, organized into two front vertical stacks, each containing 12 modules, and a single rear horizontal stack of 24 modules. In Figure 5.2, a visual depiction of the grippers utilized in our experiments for the disassembly processes is provided. These grippers, carefully selected for their functionality, play a pivotal role in the successful execution of our battery disassembly tasks.

Figure 5.3 offers an overview of the essential robotic systems and devices crucial to our disassembly experimental setups. In the subsequent sections, we will delve into detailed explanations of how these grippers, robotic arms, and devices were employed in various setups for practical demonstrations, providing an understanding of their roles and functionalities.

An overview of the primary disassembly stages for pack-to-cell disassembly is presented in Table 5.1, specifically focusing on the forward module stacks briefly.

The degree of automation detailed in Table 5.1 was derived from an understanding of the current manual disassembly process, considering factors such as the types of fasteners used and component accessibility. While the extent of semi- and fully autonomous tasks may differ

Table 5.1
The order of disassembly steps for the pack-to-cell disassembly of the 2011 Nissan Leaf. To keep it concise, we're only focusing on disassembling the front module stack. The tasks fall into different categories: fully manual (M), where specialized tools or skilled manual dexterity are needed, semi-autonomous (S-At), where a robot can perform the task with human assistance, and fully autonomous (At) tasks that can be completed without human involvement [17].

Step #	Disassembly task		Type
1	Top case	Remove service plug retainer	M
2		Remove upper case bolts & lift top case	At
3	Battery controller	Remove mounting bolts	S-At
4		Disconnect harness connectors & remove battery controller	M
5	Junction box & harnesses	Disconnect interlock circuit harness & heater harness connectors	M
6		Remove mounting nuts & front stack connecting bus-bar	S-At
7		Remove battery member pipe	S-At
8		Remove junction box cover	M
9		Remove central bus bar bolts and remove central bus bar	S-At
10		Remove current sensor bus bar mounting bolt	S-At
11		Remove switch bracket mounting bolts	S-At
12		Invert switch bracket, disconnect harnesses & remove switch bracket	M
13		Remove high voltage (HV) harness bolts & remove HV harnesses	S-At
14		Disconnect voltage & temperature sensor harnesses	M
15		Remove junction box mounting nuts & junction box	M
16	Heaters	Disconnect harness connectors from heater and heater relay unit	M
17		Remove heater & heater relay mounting nuts	S-At
18		Remove heater controller unit & heaters	M
19	*Front module stack(s)*	Remove stack mounting nuts	At
20		Extract module stack	At
21		Remove bus bar cover	M
22		Remove bus bar terminal mounting bolts & mounting screws	S-At
23		*Remove end plate bolts*	At
24		*Remove end plate*	S-At
25		*Electrical test & sort modules*	S-At
26	*Module*	*Separate module cover*	S-At
27		Glue separation	At
28		Separate cell tabs from terminal assembly	S-At

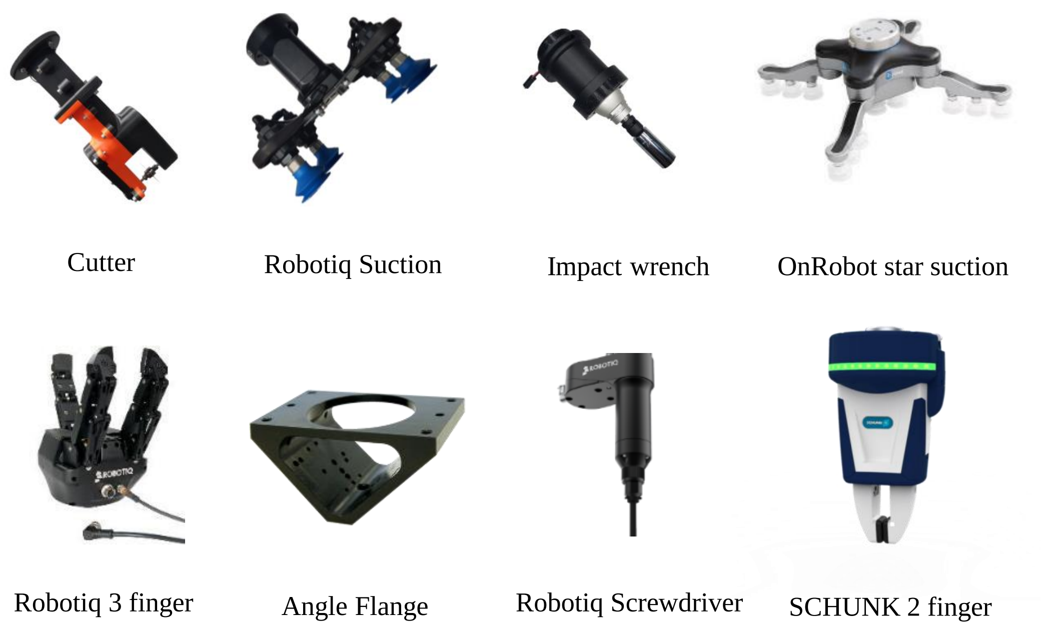


Fig. 5.2. Grippers employed in experimental setups

among various battery designs and manufacturers, there is a shared motivation to enhance automation levels to reduce disassembly costs, as emphasized in the study by Lander et al. [6]. Based on the disassembly sequence outlined in Table 5.1, we explore a range of repetitive tasks commonly identified in related studies [9]. These tasks encompass unbolting, involving the removal of fasteners connecting a stack of modules; the extraction and sorting (pick and place) of disassembled waste components; and cutting, utilized when mechanically separating components becomes necessary due to the inability to remove fasteners non-destructively, and when the goal is to reach inside the components such as reaching to cells inside the battery modules.
To demonstrate these tasks, we employed various methods:

- **Predefined Positions Teaching:** This conventional approach involves instructing the robot with predetermined positions in the task environment. While widely used in industry, it demands strict adherence to initial positions, making it less robust for disassembly tasks where object poses can vary, especially with flexible or tangled items. Changes in the environment or object positions during tasks that need contact can also decrease accuracy.

- **Vision-Based Methods:** Utilizing vision for object identification and localization, we explored different approaches such as model-based trackers and employing trackers based on component models for look-and-move strategies. Deep learning and transfer learning, train neural networks to recognize and identify components, providing flexibility in handling variations. Visual Servoing, Dynamically manipulating the robot based on real-time image features, facilitating adaptive and precise movements.

- **Teleoperation:** We investigated teleoperation, utilizing either a haptic device or an identical robot to remotely control the robot's position, allowing for disassembly tasks to be performed from a distance.

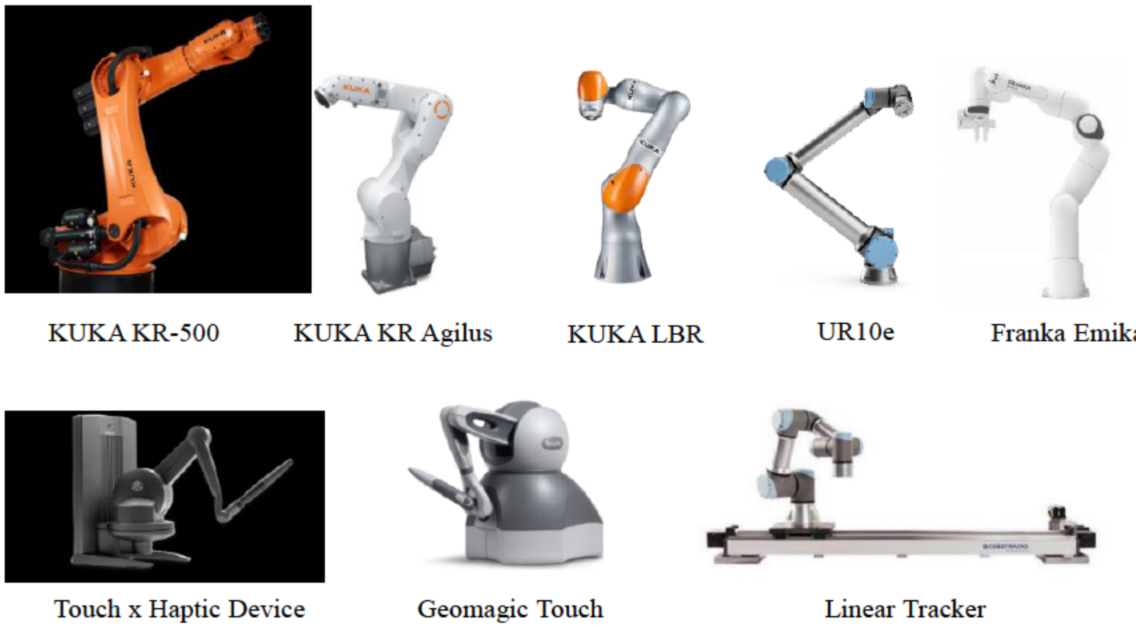


Fig. 5.3. Robotic arms and devices utilized in experimental setups

In the following sections, each method is comprehensively explained, accompanied by detailed insights into various setups designed and utilized for these purposes.

5.2 | Model based tracker

In this section, we introduce a 3D model-based tracking approach, which enables tracking of marker-less objects while providing precise 3D localization of the object's pose in the camera frame. This method is particularly valuable when utilizing a calibrated camera. Our implementation leverages the ViSP library [127] and involves the publication of the object's location on a designated topic, making it compatible with various ROS-friendly platforms. The information obtained through this tracking system holds significant potential for applications such as pick-and-place and sorting tasks, and we have successfully applied this method to track both bolts and modules within the battery pack of the Nissan Leaf 2011. The package is designed to be adaptable, allowing for implementation on a wide range of ROS-compatible robots, regardless of their programming language.
The ViSP library provides three distinct tracking methods to accommodate a variety of objects and scenarios:

- **Moving-Edges Tracker:** This tracker is designed to handle objects lacking textures and relies on the detection of moving edges that may not be visible within the object's model. It proves particularly effective for tracking untextured objects, ensuring robust performance even in the absence of prominent edges.
- **KLT Key Points Tracker:** In contrast, the KLT key points tracker is tailored for objects with textured surfaces, tracking key points detected on each visible face of the model. This method excels at tracking textured objects, even when the edges are not visible.
- **Hybrid Tracker:** The hybrid tracker combines the strengths of both the moving-edges and KLT key points trackers, offering a solution capable of tracking textured objects with visible edges, making it an ideal choice for a wide range of applications.

The adaptability of these trackers is a key feature, allowing users to switch between them by simply modifying the controller in the launch file provided with our implemented model-based tracking package.
To employ the tracking system, an initial setup is required, which involves a one-time manual initialization by a human operator (Figure 5.4). This setup includes the use of helper points to define and initialize the object to be tracked and localized. These points are pivotal for the system to establish an initial understanding of the object's position and characteristics. The Intel RealSense D435i (RGB-D) camera is used in this work. In Figure 5.5, we provide a visual representation of the model-based tracker in action, demonstrating the utilization of the hybrid version of the tracker to detect and localize modules and bolts within the Nissan Leaf 2011 battery pack.

5.3 | Object detection using deep learning

In this section, we explore the process of utilizing deep learning techniques for the detection of battery components within the battery pack. This approach offers a robust and automated method for object detection and localization, enhancing the efficiency and accuracy of battery disassembly procedures and offering a range of benefits to the entire workflow.
The outcome of this project is the development of a Deep Learning package. This package is able to perform the classification of a diverse set of battery components, effectively distinguishing between them. Once a component is detected, its coordinates are pinpointed within the image or video frame. Additionally, the system provides labels to identify these components.

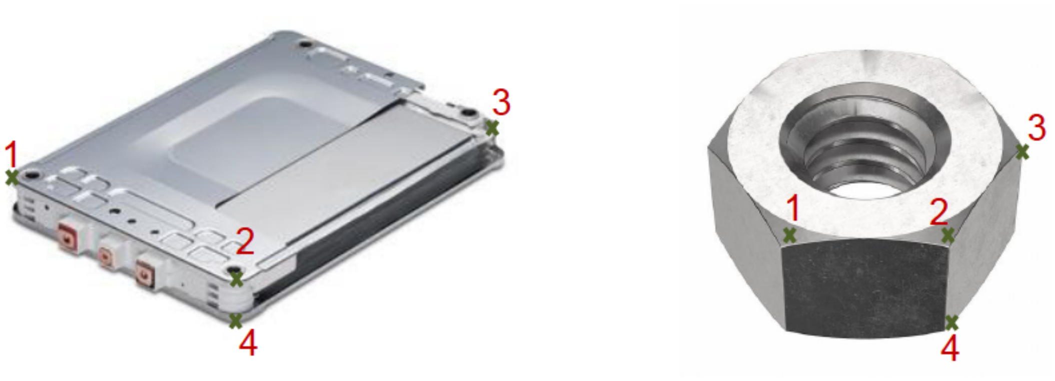


Fig. 5.4. The manual initialization process is depicted, demonstrating the use of helper points to establish the initial object tracking parameters for both the bolt and battery module.

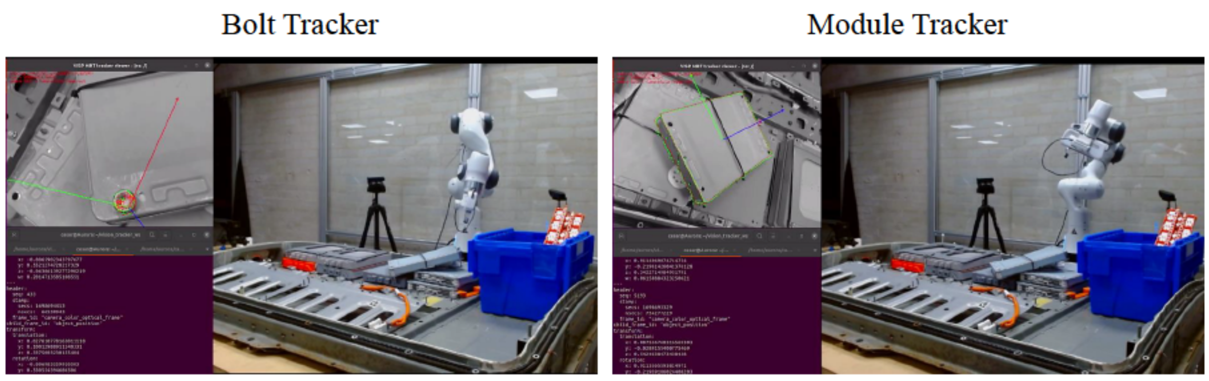


Fig. 5.5. This figure illustrates the implementation of our model-based tracker, showcasing the hybrid tracker in action as it detects and localizes modules and bolts within the Nissan Leaf 2011 battery pack.

Once this package is fully completed it will be used as an asset for a task planner, providing feedback that includes not only the location data but also the labels associated with all detected objects. This information is important in facilitating the planning and execution of subsequent tasks in the battery disassembly process, including, sorting, unbolting, cutting, and so on. In other words, it offers an efficient means of understanding the content and arrangement of battery components, ultimately streamlining the entire disassembly procedure. In what follows, the process of preparing the deep learning package is explained:

5.3.1 Data collection and labeling

The first step in building a deep learning-based object detection system is to collect a substantial amount of data for each component we want to recognize and then label that data. This process begins with the collection of data, typically in the form of images or videos. Since the process of labelling is time-consuming, we make use of the MATLAB Ground Truth Labeler Toolbox. This toolbox assists in the data labelling process by simplifying and automating the annotation of objects within each frame of the video. With this approach, we identify and label components, including battery modules, bolts, bus bards, cables, connectors, battery management systems, and other objects that are of interest to us. Thereafter, the data will be divided into two groups, training and test data (Figure 5.6). The Ground Truth Toolbox

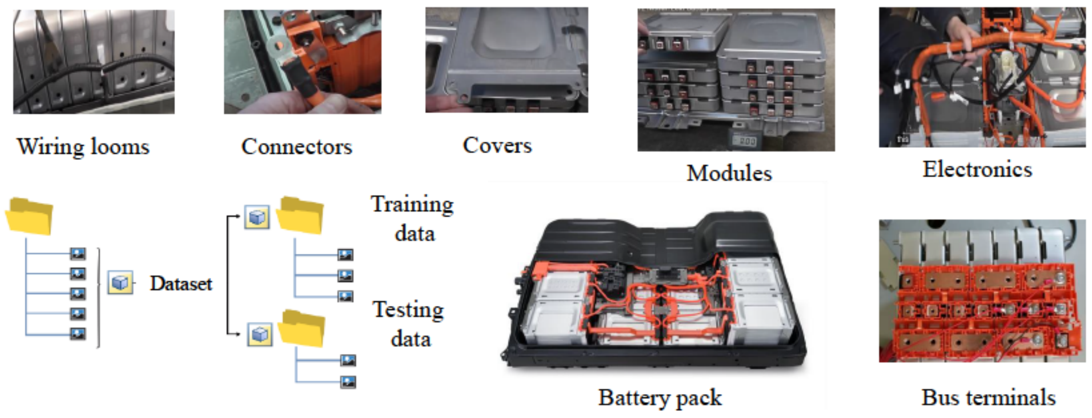


Fig. 5.6. Some of the components in the battery pack data set.

provides powerful built-in algorithms to automate labelling. Among the algorithms provided are:

- **Lane Boundary Detector:** This algorithm helps identify and mark lane boundaries in images, which is particularly useful for tasks involving road scenes or vehicle-related applications.
- **Aggregate Channel Features:** Aggregate Channel Features (ACF) is a technique used for object detection in images, making it easier to highlight and label objects of interest within the data.
- **Point Cloud Temporal Interpolator:** When working with point clouds, this algorithm assists in interpolating and labelling temporal data, which is valuable for applications dealing with time-series information.
- **Point Tracker:** The point tracker algorithm is instrumental in following and labelling the movement of specific points within video frames, a fundamental step in object tracking.
- **Temporal Interpolator:** This algorithm aids in the interpolation of temporal data, which is essential for tasks requiring smooth transitions in time-series data.

These algorithms are a crucial part of the data labelling process, ensuring accuracy and efficiency in preparing the data for deep learning-based object detection. In Figure 5.7, we provide a visual representation of using The Point Tracker algorithm to identify and automatically label bolts, modules, and bus terminals in a video file.

5.3.2 Data pre-processing

After we have labelled the data, the next important step is data pre-processing. This step is all about getting our data ready for our deep learning model. It is a crucial part of the process because it helps our model perform better in a wider range of situations. Techniques like applying filters can help improve image quality and reduce or add noise to the labelled objects. Additionally, data augmentation is employed to expand the dataset. This involves creating variations of the original images through techniques like rotation, scaling, flipping, and colour filters which simulate real-world variations and make the model more robust (using a small and monotonous dataset can cause the model to become too specialized, resulting in

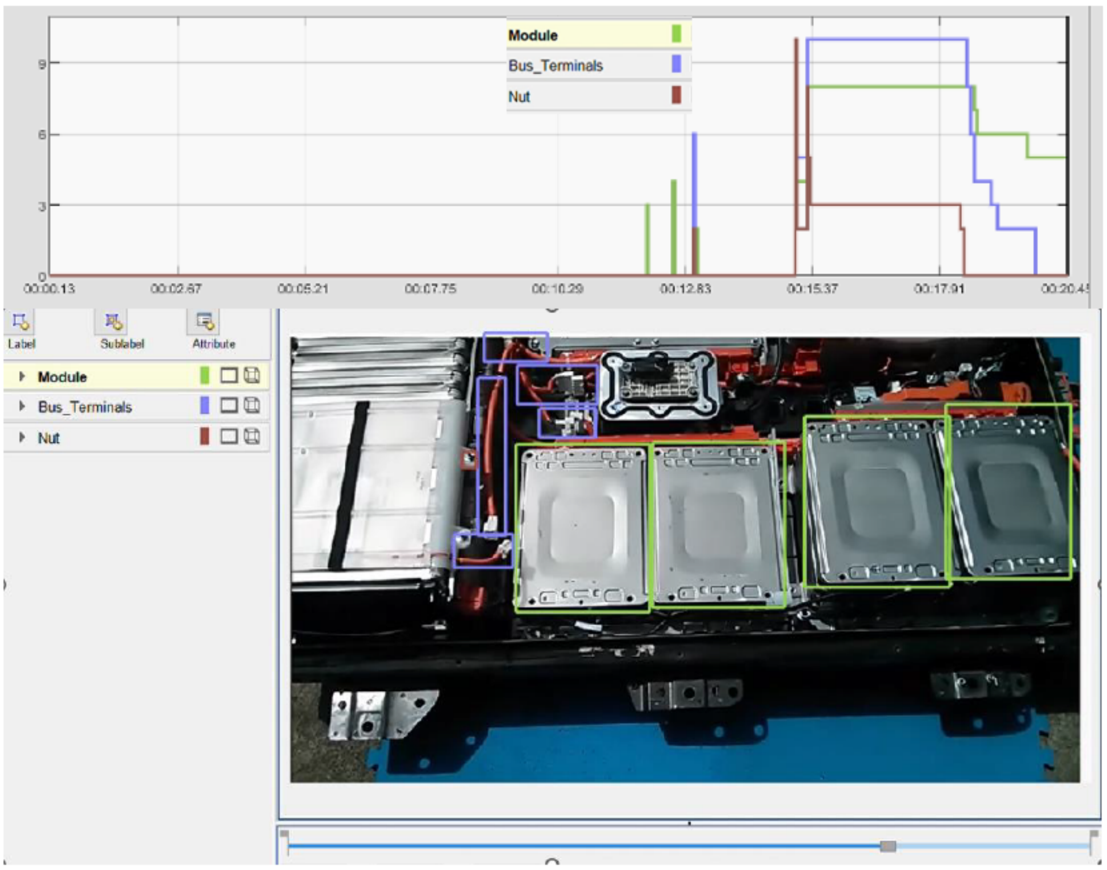


Fig. 5.7. Using the Ground Truth Labeler Toolbox to automatically identify and label nuts, modules, and bus terminals in a MP4 file.

overfitting.). It should be mentioned that all pictures should be resized to the format that is acceptable for the network as inputs (here 224-by-224). Augmentations were applied after the data for training and testing were separated. If we had applied augmentation to the entire dataset before the separation, it could have led to overfitting, as multiple versions of the same data might have ended up in both the training and testing sets.

5.3.3 Transfer learning

The core of our object detection process is deep learning models, particularly pre-trained deep neural networks initially designed for image recognition tasks. These pre-trained models provide a robust starting point due to their ability to extract essential image features, which is incredibly valuable for the object detection objective. We have used Squeezenet for this purpose and trained a YOLOv3 model detector.
To instruct the model in recognizing specific objects, we employ transfer learning. This technique harnesses the knowledge acquired by the pre-trained network in general image recognition tasks and fine-tunes it to custom use cases. Using Transfer learning will adjust the model weights and layers, enabling it to learn the unique features of battery components.
The dataset was partitioned, with 70% reserved for training and the remaining 30% allocated for testing. Before training, we preprocess the training data to conform to the network's input size while preserving the original image aspect ratio.
Furthermore, we calculate anchor boxes using the training data to estimate potential object locations.

5.3.4 Training and validation

The training process involved parameters that influence the accuracy of the model and training speed. To arrive at the final parameter settings, we conducted extensive experimentation by trial and error and compared the results. The key training options employed for the final outcome are shown in Table 5.2. Brief explanations of different parameters are detailed below:

Table 5.2
Training Options and Values.

Epochs	Mini Batch size	Learning Rate	Warm-up Period	L2 Reg	Penalty Threshold
30	8	0.001	7000	0.0005	0.5

- Number of Epochs: An epoch represents the model's complete pass through the training data, and in this case, it encompassed 30 such passes. The choice of the number of epochs is pivotal, as too few may lead to underfitting, where the model does not learn sufficiently, while too many may result in overfitting, limiting the generalization of the model.

- Mini Batch Size: We employed a mini-batch strategy, where data is processed in smaller groups. This not only accelerates training but also helps maintain the quality of the final result. The batch size can be adjusted depending on the available memory resources.

- Learning Rate: The learning rate dictates the step size taken at each iteration towards minimizing the loss function. It is a crucial parameter for achieving effective training.

- Warm-up Period: The warm-up period involves a phase where the model gradually increases its learning rate to stabilize learning in later iterations. Once this period ends, the specified learning rate is maintained for the rest of the training, typically about half of the overall training process.

- L2 Regularization: L2 Regularization introduces an additional penalty to the cost functions used during training. This penalty reduces the weights of coefficients contributing to the cost, thereby mitigating overfitting.

- Penalty Threshold: Predictions that overlap less than the specified threshold with the ground truth are penalized, enhancing the precision of the model.

During each iteration, data is retrieved from the batch queue. If the queue becomes empty, it will reset to proceed to the next epoch. This queue prepares mini-batches of data in the background while the model continues training on previous batches, improving computational efficiency. The queue is also shuffled at the beginning of each epoch to prevent the model from learning the sequence of data. The current state of the model is constantly evaluated against the loss function to assess gradients. Weight decay with L2 regularization is applied to ensure robust training, and the learning rate is adjusted. Detection parameters are updated, and the process repeats until the specified number of epochs is completed. After each iteration, the box loss, object loss, and total loss are recorded. This information allows us to monitor the model performance over time and make informed decisions like stopping training if the loss has saturated for a few epochs. This helps save time and resources while reducing the risk of overfitting. Once the training concludes, the model performance is evaluated by running it on test data and comparing the results with the ground truth. The average precision metric is a critical factor in determining the model's success in this case. The overall results of the deep learning model, particularly the implementation of YOLO, are reported in Table 5.3. The model is more accurate in the detection of larger and isolated components

Table 5.3
Detection accuracy of some components in Nissan leaf battery pack.

Components	Module	Backstack	BMS Board	Terminal Cover	Bus bar
Average Precision	63.97%	88.89%	66.67%	50%	47.28%

such as modules and backstacks. However, challenges are encountered in achieving precise detection for smaller components like bus bars or module connectors. This limitation may be attributed to sub-optimal detection precision, and potential improvements could be explored by utilizing more zoomed-in images instead of directly labelling entire battery packs. Smaller components face difficulties in detection due to factors such as low resolution, making them harder to distinguish from the background. Additionally, their colours often provide limited useful information, as they are typically similar in colour to the modules they are located on. Obstruction further hinders the detection of smaller components, particularly when observing an entire pack. Many battery designs intentionally conceal wiring harnesses for neatness and safety, posing a challenge for the model in accurately detecting and labelling these concealed components. It is important to note that this study is an initial exploration, and further iterations are required. The dataset and YOLO model may need updates and modifications to enhance precision. The study hints at the possibility of utilizing deep learning and transfer learning not only for detecting components within the Nissan Leaf battery pack but also for distinguishing and labelling components from different companies. Future work should consider refining the model architecture, expanding the dataset, and incorporating transfer learning techniques to achieve more accurate and versatile component detection across various battery pack designs and manufacturers.

5.4 | Experimental details and setups

In this section, the experimental setups employed, along with a detailed explanation of the experiments conducted will be presented.

5.4.1 Setup 1

In Figure 5.8, the designed Setup 1 for demonstrating the process of disassembling a stack of modules is illustrated. The image showcases three robotic arms equipped with a two-finger gripper, a suction gripper, and an impact wrench, strategically positioned around the battery pack scene. The arrangement allows for close access to the entire battery pack, enabling efficient execution of disassembly tasks. Two baskets, designated for waste and recycling purposes, are positioned within the scene. Additionally, cameras are placed on the scene to monitor the disassembly process during the execution of the tasks. In this setup, we used preprogrammed robots and predefined positions to perform some parts of the disassembly tasks. In Figure 5.9, a step-by-step visual inspection illustrates the disassembly tasks for a stack of modules. The process begins with unbolting and removing mounting bolts using a two-finger gripper attached to the KUKA LBR. The next steps involve the UR10e with a suction gripper, removing the cover, and placing it in the waste basket. Subsequently, the modules are picked

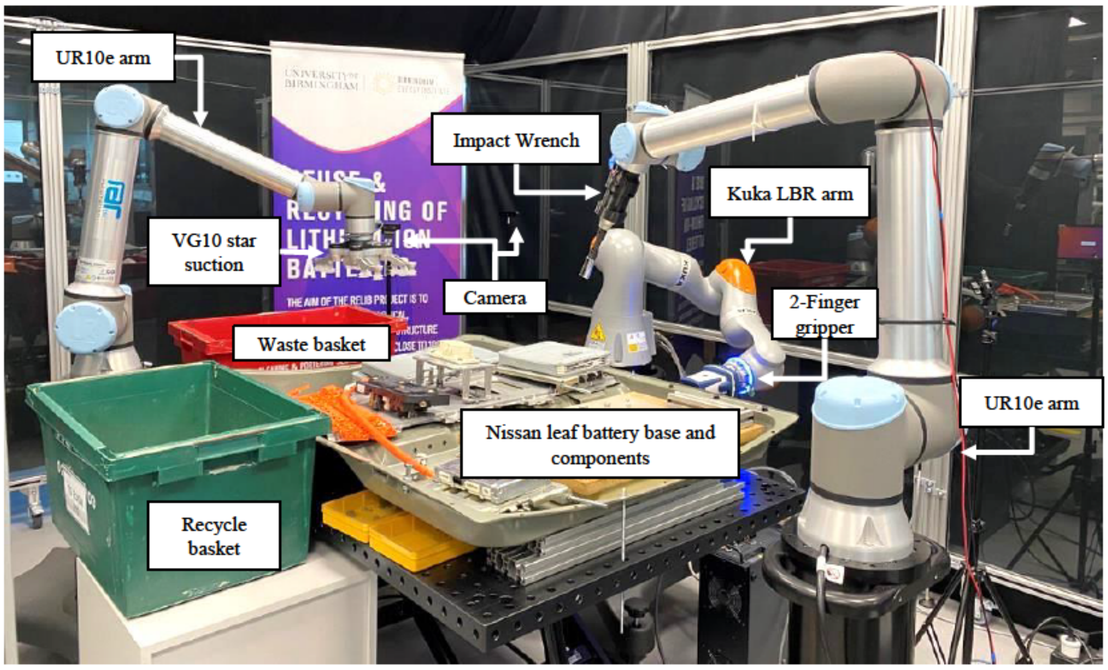
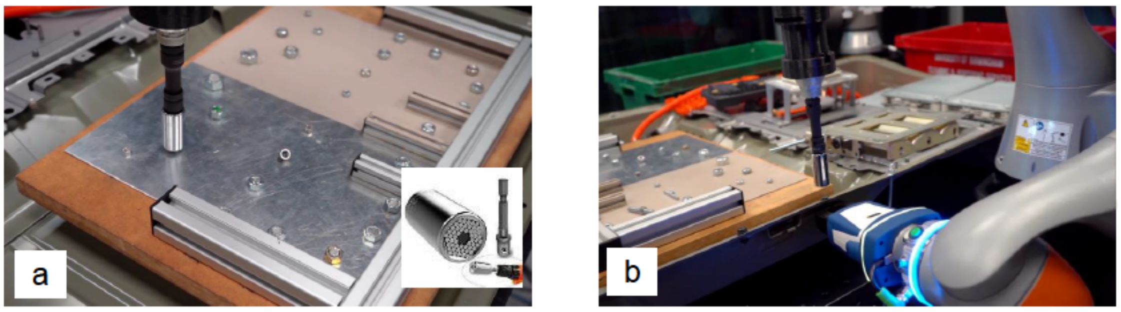
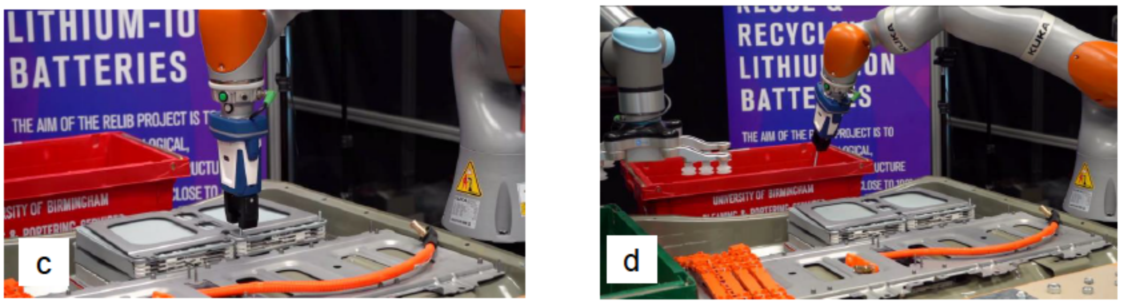


Fig. 5.8. Setup 1 used to demonstrate the disassembly of a module stack in Nissan leaf battery pack.

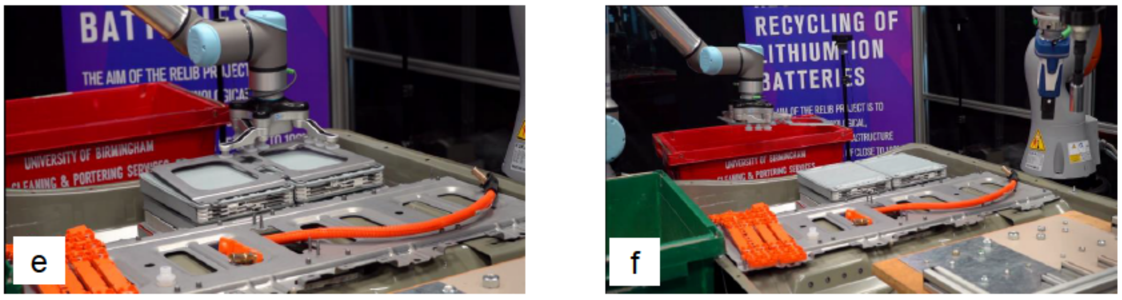
and deposited into the recycle basket, setting the stage for further processes such as cutting and reaching the cell level.
For the unbolting process (Figure 5.9a, and b), we utilized a Universal socket wrench capable of handling bolts of various sizes, ranging from 7 to 19mm. The impact wrench is custom-designed by one of the technicians, in order to be able to mount on the various types of robot wrists and operate it from a distance. This design allows for remote operation, facilitating both bolting and unbolting tasks.
In our experiments, we demonstrate bolting and unbolting of various sizes of bolts, depicted in Figure 5.9a. The impact wrench's adaptability and remote operational contribute to the efficiency and flexibility of the disassembly process. Furthermore, our experimentation extends to illustrating collaborative efforts between two robots engaged in the unbolting of bolts with nuts. As depicted in Figure 5.9b, the KUKA LBR robot, equipped with a two-finger gripper, securely holds the nut at the bottom of a hex bolt. Simultaneously, the UR10e robot, fitted with the impact wrench, moves to the head of the hex bolt, facilitating the unbolting process. This collaborative approach highlights the versatility achieved by combining different robotic arms to accomplish a disassembly task.
The next task, which succeeds the previous unbolting stage, involves the removal of each fastener from the stack, as illustrated in Figure 5.9c, and d. For this specific operation, we utilized the Schunk two-finger gripper, attached to the Kuka LBR arm. The gripper was configured with a grasping force of 40N, ensuring a secure and controlled hold. To execute this task, the robot was programmed to approach each fastener, employ the gripper to securely grasp and remove the bolt from the stack, and then place it into the designated waste basket close to the task space.
Subsequent to the removal of the fasteners, the UR10e robot, equipped with the star suction gripper, undertakes the task of removing the module cover plate (Figure 5.9e, and f) to gain access to the underlying module stack. Given the weight and geometry of the cover plate, achieving a specific configuration of channels and precise positioning of the suction cups



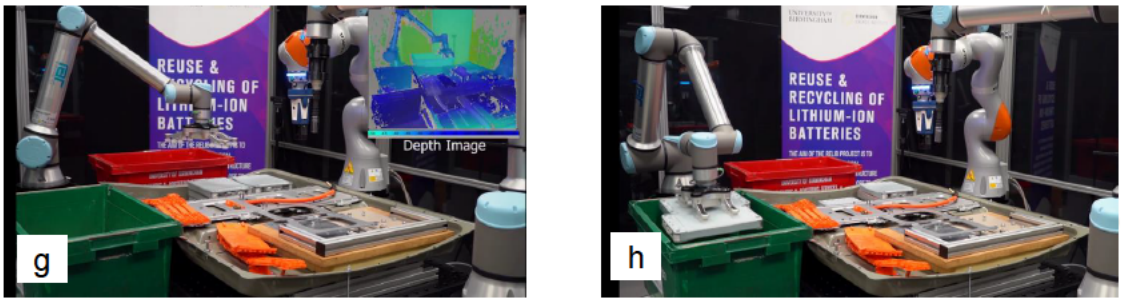
Step 1: Unbolting, a) Unbolting different size of bolts, b) Dual arm cooperation



Step 2: Sorting, grasping bolts with 2 finger gripper and place in the waste basket



Step 3: Sorting, using VG10 suction gripper to pick and place the cover in the waste



Step 4: Sorting, using VG10 gripper to pick and place modules in the recycle bin

Fig. 5.9. Step-by-step disassembly process for a stack of modules

is crucial. This ensures a secure grip on the cover and establishes an optimal grasp point, guaranteeing the safe and stable transportation of the cover during removal. For this particular task, the configured grasping vacuum power was set at 40kPa. This specific parameterization

is designed to provide the necessary suction force, ensuring a reliable grip and facilitating the successful and controlled removal of the cover plate by the UR10e robot.
The next task involves the unstacking and sorting of EV battery modules, employing the star vacuum suction gripper. In this operation, the UR10e robot is tasked with extracting a pair of modules from a stack and depositing them into a designated recycling basket (Figure 5.9g, and h). Precision is crucial in this task, requiring the gripper to be accurately positioned on a suitable surface for effective grasping. It is essential to maintain consistent contact to engage the suction cups with the material, all while ensuring that the force exerted by the robot remains within the permissible force limits of the robot. Similar to the cover removal step, the gripper for this task was configured with a grasping force of 40N. This configuration optimizes the grip on the modules, ensuring a secure hold during the unstacking and sorting process.

5.4.2 Setup 2

In Setup 2, we introduced several enhancements to our disassembly process as illustrated in Figure 5.10. Notably, a camera was integrated into the UR10e along with the implementation of visual servoing, complemented by the utilization of a three-finger gripper (Robotiq). To ensure precise performance, an initial calibration of the camera was done, the details of this calibration process are given in the index 7.1.

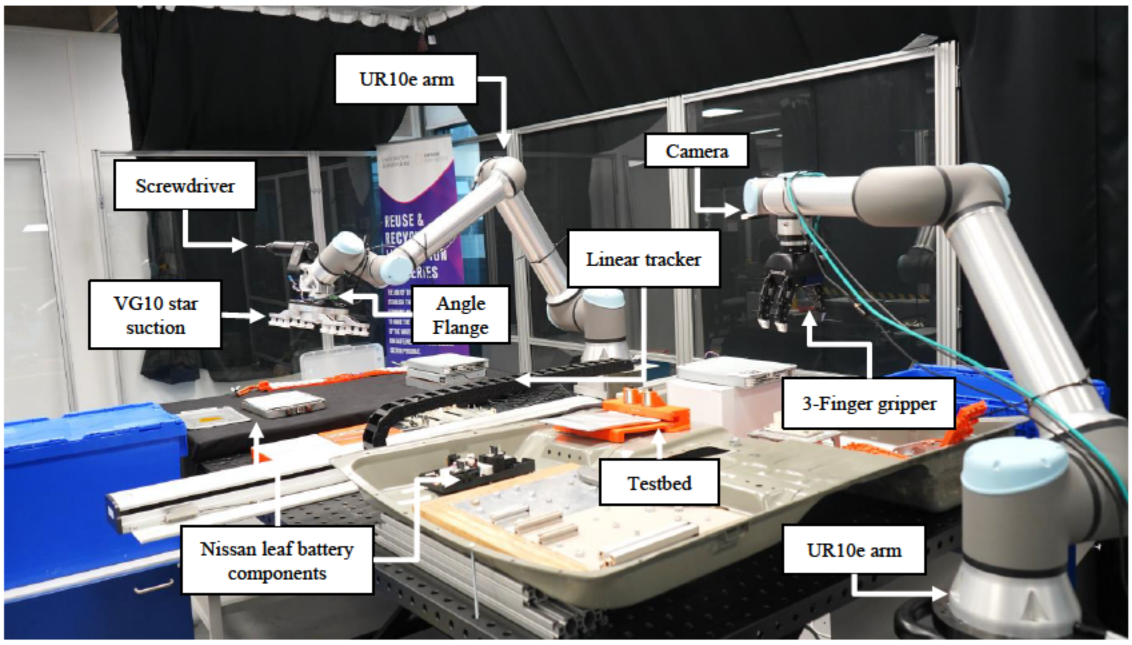


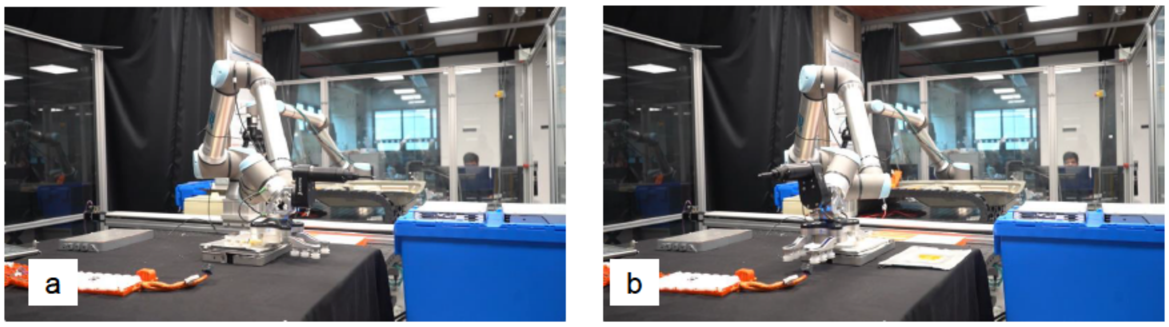
Fig. 5.10. Setup 2 used for the disassembly of a module stack in Nissan leaf battery pack.

One pivotal addition in this setup is the incorporation of an Angle flange, a component that provides the robot with the flexibility of attaching two distinct grippers to its EE. This feature empowers the robot to undertake diverse tasks without necessitating a change in the gripper. A suction gripper and a screwdriver were employed as the two gripper types in this configuration. Within our implemented ROS package for visual servoing, three distinct methods have been incorporated. These include image-based visual servoing (IBVS), position-based visual servoing (PBVS), and our proposed visual servoing method (Chapter 2). It is important to note, however, that due to the absence of redundant joints in the UR robot configuration, we are unable to include a security task during the redundancy resolution process when solving the

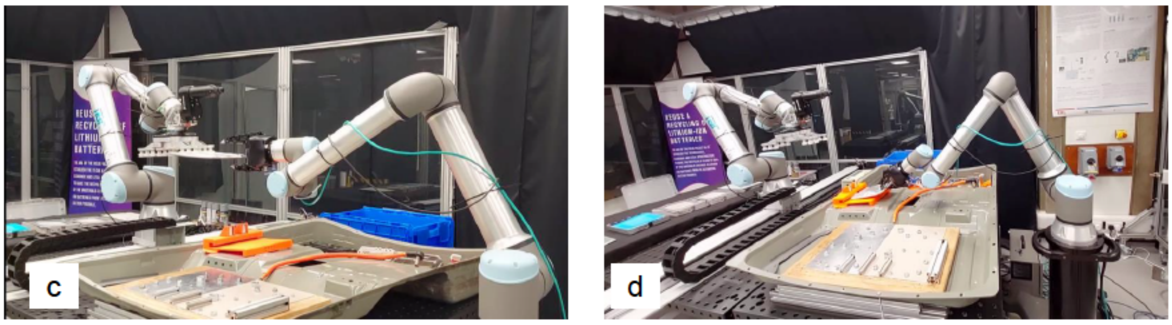
kinematics of the robot. This limitation arises from the non-redundant nature of the robot joints, which restricts the incorporation of additional tasks in redundancy resolution. Despite this constraint, the implemented visual servoing methods remain effective in guiding the robot's movements based on visual feedback for precise and accurate task execution.
Setup 2 also incorporates a linear tracker. By mounting one of the UR10e robots on the linear tracker, the task space of the robot is significantly expanded, enabling it to operate across a broader range. The linear tracker's data cables are connected to the robot's controller, and with the appropriate URCap, it can be controlled directly from within the robot program using the teach pendant. URCaps serves as a platform to add accessories that directly operate within UR robot applications for end-users.
Alongside using VS, another novel task is introduced in this setup, involving the extraction of a cell from the interior of an opened module and placing it on a designated testbed As depicted in Figure 5.11. The testbed is purposefully designed to assess the health of the cell, to determine its reusability. If the cell is healthy, it can be reused; otherwise, it proceeds to the recycling stage, where valuable materials are extracted. After this assessment, the robot is programmed to pick up the cell and, based on the testbed's decision, either place it in an allocated basket for the subsequent task or direct it toward recycling.
The modified HDVS method builds upon our initial introduction of the hybrid VS model. Initially, the HDVS model leveraged the advantages of overconstrained robots, which are robots equipped with redundant joints—having more joints than the number of degrees of freedom they need to control. This redundancy allowed us to incorporate additional tasks, such as obstacle avoidance or singularity avoidance, within the null space of the robot. However, in many industrial robot scenarios, the number of joints is equal to the number of degrees of freedom, eliminating the option of exploiting redundancy for additional tasks. Despite this limitation, we have adapted and modified our proposed hybrid method to remain applicable in these scenarios. Although we may not benefit from the overconstrained nature of the robot, the modifications introduced in our hybrid decoupled method still offer advantages in addressing specific challenges (discussed in 1.2), making the approach relevant and effective even in cases where redundancy is not available. In the proposed visual servoing package, we have implemented not only the HDVS method but also provided the flexibility to utilize two other visual servoing techniques: Image-Based Visual Servoing (IBVS) and Position-Based Visual Servoing (PBVS). Users have the option to simply switch between these three methods based on their preferences and specific application requirements.

5.4.3 Setup 3

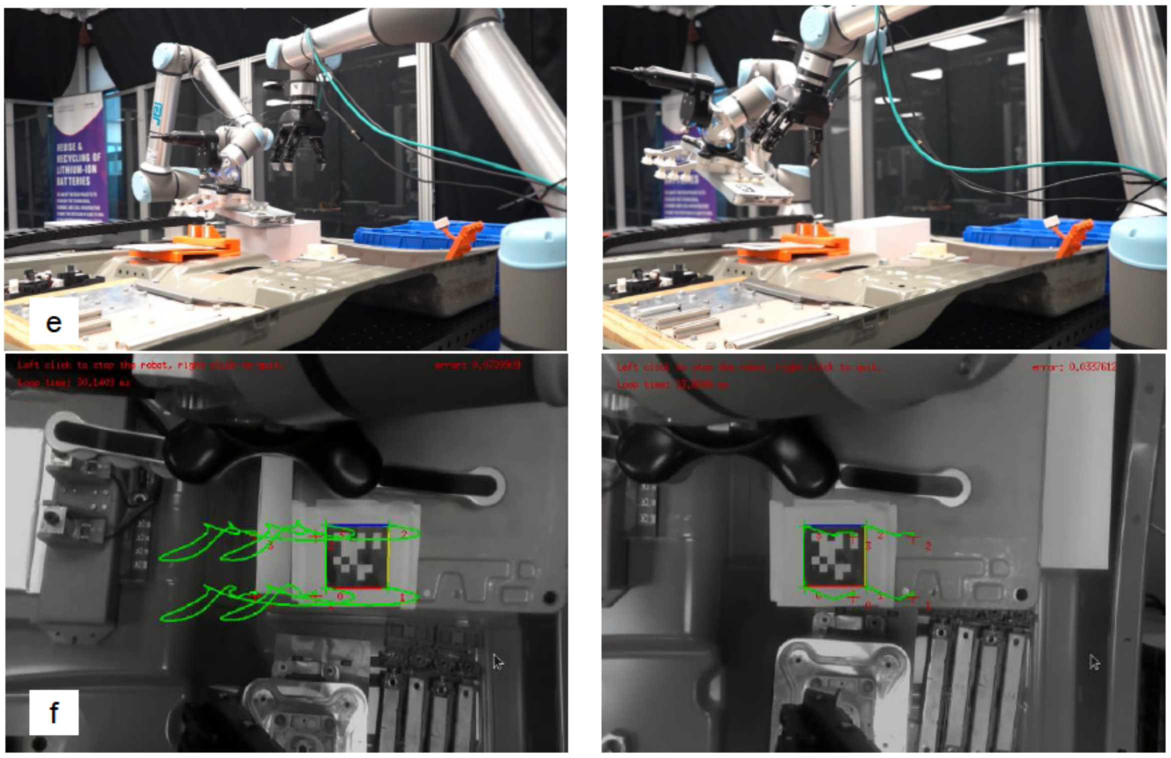
In our subsequent setup, as depicted in Figure 5.12, we employed two Franka arms and one Kuka LBR. This configuration incorporated a motorized cutter and an e-pick suction gripper provided by Robotiq. For the cutting task, we initially applied a hybrid force and position control method, utilizing a marker as a gripper. This demonstration showcased the controller's ability to trace the borders of a dummy module while maintaining constant contact with its surface. The resulting lines produced by the marker on the module can be observed in Figure 5.13a. The cutting algorithm comprises two nested control loops: an outer loop for ensuring the Cartesian x and y directions of the cutter tip, and an inner loop for maintaining contact of the cutter tip with the module surface by monitoring the z component of Cartesian force at each step. The inner loop operates at a higher frequency, and when the detected force is below the desired threshold, the robot moves further in the z-direction until the specified force threshold is reached. Moreover, when the detected force is higher than the threshold the robot will move up until the force is in the boundary. The boundary should change for each material and here we choose the $4N < F < 8N$ to be between 4N and 8N for the dummy module. It is



a) Using suction gripper to pick the opened lid of module b) pick the cell.



c) Dual arm collaboration to grasp a cell and d) place it on the Testbed.



e) Using the modified HDVS to track a dynamic module f) Image screen and trajectory (green)

Fig. 5.11. Visual representation of the 2 new tasks introduced in the setup 2. Sub-figures (a) and (b) demonstrate the task of extracting a cell from an opened module. Sub-figures (c) and (d) illustrate the subsequent placement of the cell on the designated testbed for health assessment. Sub-figure (e) shows the use of visual servoing to track a dynamic module and sub-figure (f) illustrates the camera FOV, where the green trajectories are the path of each feature (red signs).

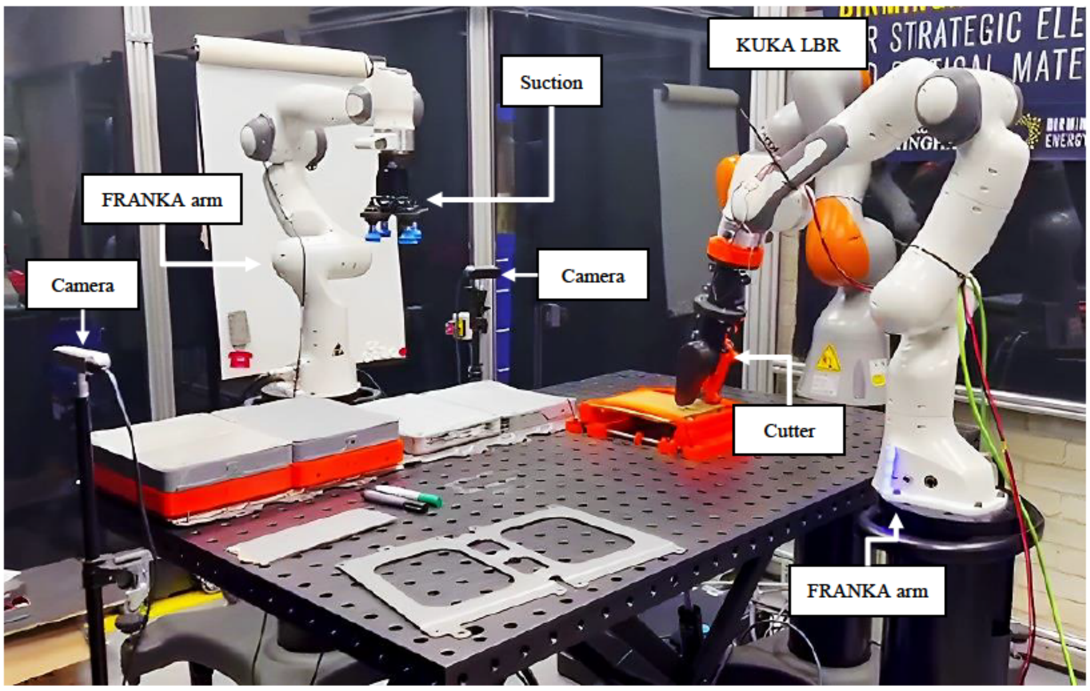
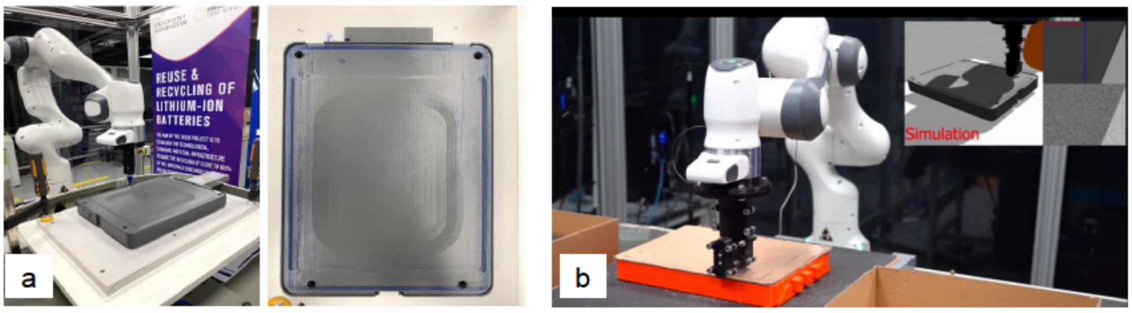


Fig. 5.12. Setup 3 used for the disassembly of a module stack in Nissan leaf battery pack.

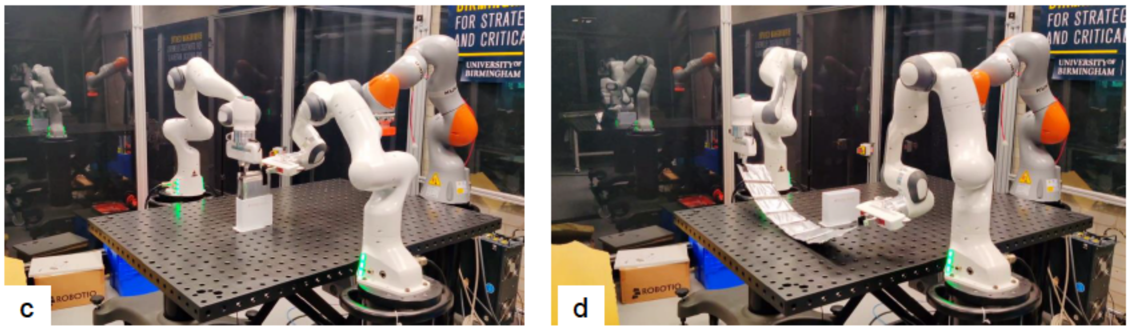
important to note that the z-direction of the robot's motion is constrained as well to prevent damage to both the cutter and the module.
In the subsequent phase of our experimentation, we utilized a custom-designed motorized cutter tool to demonstrate the effectiveness of our proposed control algorithm in cutting through the borders of cardboard, serving as a proof of concept (Figure 5.13b). Furthermore, the execution of a dual-arm cooperative task for cell separation, as outlined in reference [128], was accomplished using two Franka robots (Figure 5.13c, and d).
Furthermore, additional experiments were conducted using a RealSense camera in tandem with the proposed RL-based VS (detailed in Chapter 3) for the task of sorting cells within the battery module. The advantage of VS lies in its capacity to dynamically correct the robot's position based on the error generated between the current and desired features of the object in the image. This alleviates concerns about potential variations in the position of the robot's base or alterations in the environment, as the controller consistently adapts to correct these errors. Figure 5.13e provides a visual representation of the image screen and the robot equipped with the suction and camera. It is noteworthy that the red markers indicate the desired positions of features in the image, while the green trajectories represent the path travelled by the robot to reach the current features. Once the errors converge to zero and the features align, the robot's precise position relative to the object becomes known. Subsequently, the robot executes precise movements, guided by detected contact forces, to vacuum the cells and sort them into designated baskets around the scene (Figure 5.13f).

5.4.4 Setup 4

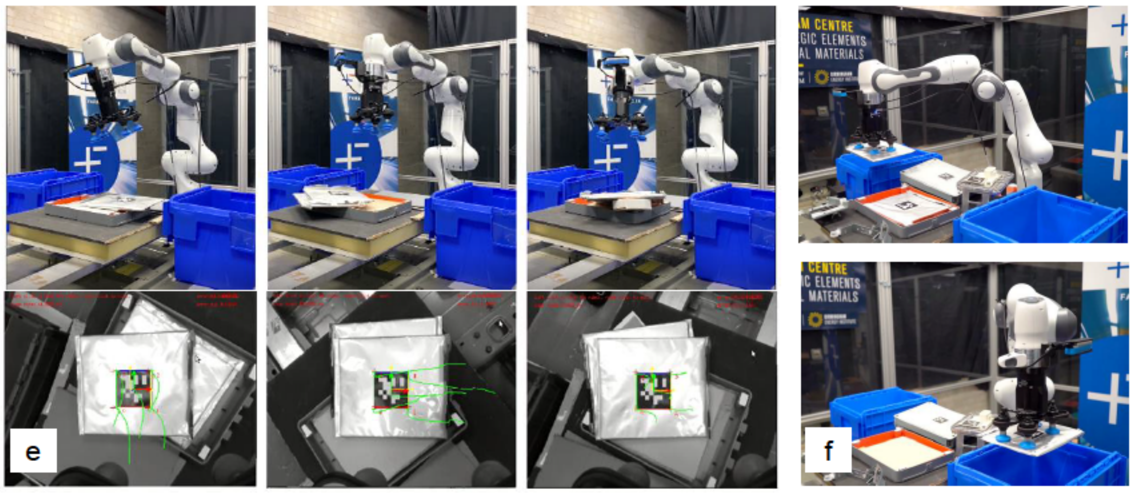
In the final demonstration setup, illustrated in Figure 5.14, heavy-duty industrial robots and grippers have been used. This configuration consists of two Kuka KR500 robots and one Kuka KR16 arm. The Kuka KR500 is categorized in heavy-duty industrial robotics, characterized by its remarkable payload capacity (500kg) and expansive range of motion. KR500 belongs to the KR QUANTEC series and is engineered for applications demanding robustness and strength.



a) Using a marker to demonstrate contact-rich path following b) Cutting borders of cardboard



c) Dual arm manipulation to remove cell d) and separate cells from each other



e) Using VS to follows cells inside a module f) Sorting with the help of VS

Fig. 5.13. Experiments were conducted using Setup 3.

With a high payload capability, the KR500 excels in material handling, welding, machining, and other industrial processes that necessitate the handling of substantial loads. Its versatility and precision are further enhanced by the Kuka KRC4 controller, providing advanced programming and control functionalities. On the other hand, the Kuka KR16 embodies precision and agility. As a compact industrial robot, it is engineered for applications that prioritize accuracy and lightness. Industries such as electronics, assembly, and laboratory settings benefit from the KR16's dexterity and control precision. Coupled with the Kuka KRC4 controller, this robot executes tasks that demand careful handling and intricate manoeuvres, showcasing its adaptability in applications where a lighter touch is key. The integration of Kuka KR500 and Kuka KR16 robots creates an optimal configuration for the disassembly of a battery pack, providing a fusion of precision and heavy-duty capabilities.

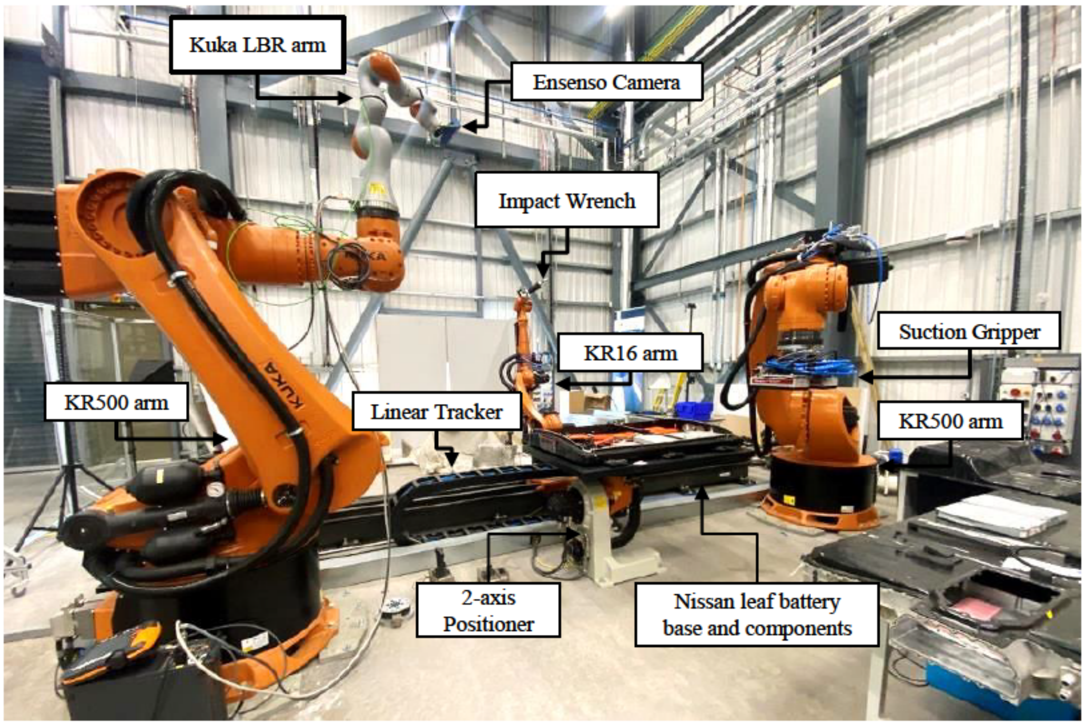


Fig. 5.14. Setup 4, used for some demonstration in disassembly of Nissan leaf battery pack.

The KR16 is mounted on a linear tracker, expanding its workspace capabilities, while a 2-axis positioner allows versatile positioning of the Nissan Leaf battery pack, enabling task execution from different angles. An impact wrench was attached to the EE of the KR16 for unbolting. Within this setup, one KR500 robot is equipped with a vacuum gripper, a RealSense D435 camera, and a force sensor. Another robot, a Kuka LBR, is attached to the wrist of the other KR500, enabling it to perform more delicate tasks. This tandem arrangement significantly expands the workspace of the Kuka LBR, as the KR500 can reposition the base of the LBR robot as needed. The robots could be controlled through their teach pendant or using Kuka Sunrise software and Java language or via MATLAB.
Figure 5.15 provides a visual representation of one KR500 robot picking up the heavy lid of the battery pack and precisely placing it in the designated position. Additionally, the KR500 demonstrates the pick-and-place capability of modules using a vacuum suction gripper. The integration of the VS package and the Model-Based Tracker enhances the robotic capabilities for vision-related tasks. It is crucial to note that the camera should undergo calibration for optimal performance within this robotic system. The conducted VS with the KUKA KR Agilus robot in Figure 5.16 serves as a demonstration illustrating the versatility of the proposed package for application across various types of industrial robots and cobots. Besides, the Model-Based Tracker package is also adaptable to be used in various setups. Through camera calibration on the robot's wrist (index 7.1), these two packages directly would integrate into the system. The Visual Servoing package provides a dynamic solution by generating Cartesian velocity commands through three distinct topics, accommodating various visual servoing methods including, IBVS, PBVS, and DHVS (detailed in Chapter 2). Simultaneously, the Model-Based Tracker package enhances the setup by providing the 3D positions of tracked models, empowering the robot to execute movements with different tools for diverse tasks (detailed in the section 5.2). The package offers multiple tracking options, catering to textureless and textured objects, and a hybrid approach, demonstrating robust tracking capabilities. ROS integration and compatibility

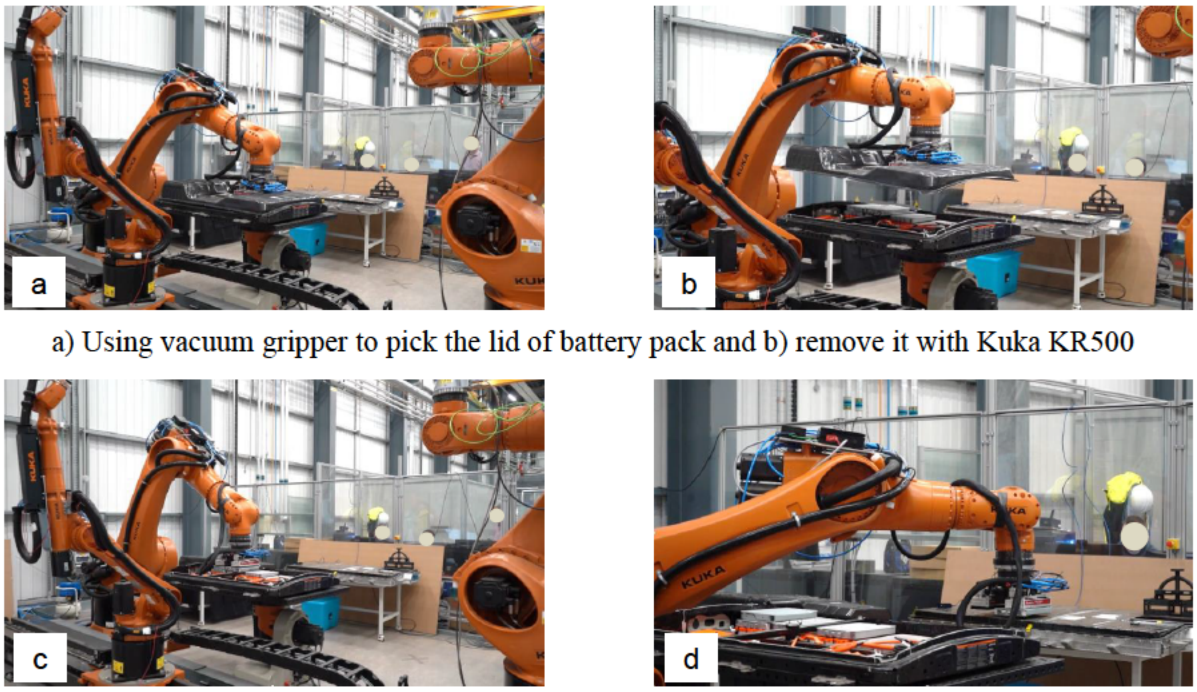

a
b
a) Using vacuum gripper to pick the lid of battery pack and b) remove it with Kuka KR500
c
d
c) Using vacuum gripper to pick modules and d) sort them with Kuka KR500

Fig. 5.15. Some of the performed experiments with Setup 4.

with popular programming languages contribute to the adaptability of both packages in both simulation and real-world scenarios.

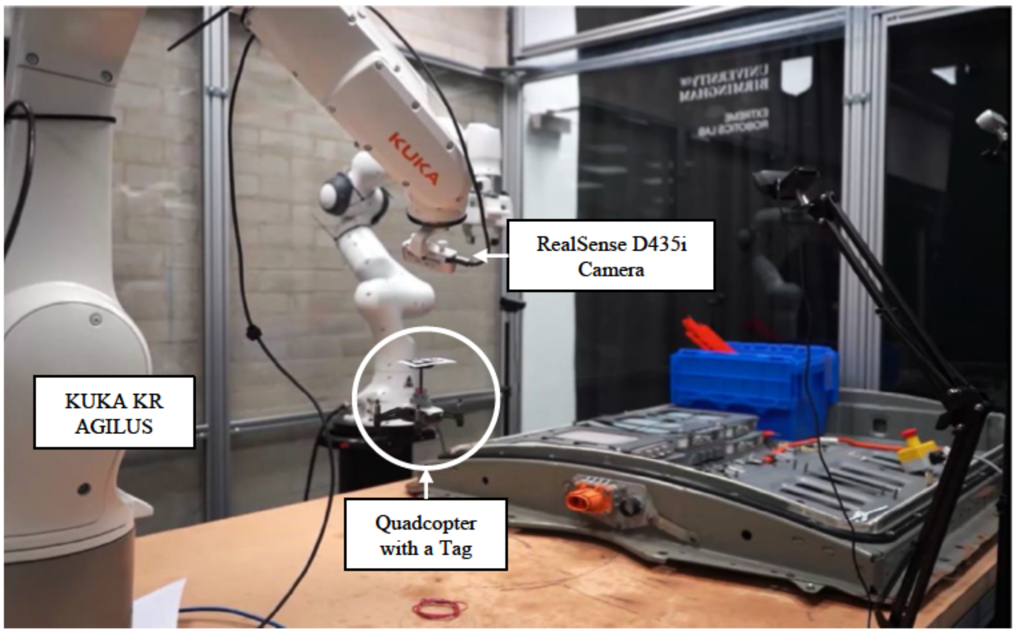


Fig. 5.16. Utilizing the suggested Visual Servoing (VS) package to track a marker attached to a quadcopter.

5.5 | Teleoperation

As previously discussed, teleoperation provides the operator with the capability to remotely control robots, ensuring safety and facilitating tasks that require human flexibility. In this scenario, we employed two distinct master devices for teleoperating the slave robot, and a comparison was conducted between these two methods. The experimental setup is illustrated in Figure 5.12.
In this research, we used a custom-designed impact wrench tool capable of accommodating various fastener sizes. To facilitate low-power cutting tasks, a motorized slitting cutting tool was specifically designed for the low-payload cobots featured in this study. When it comes to pick and place operations, the Franka Hand two-finger gripper was utilized for handling thin and lightweight waste items like bolts and plates, while the Robotiq EPick suction gripper was employed for larger and heavier materials such as battery modules.

5.5.1 Teleoperation using haptic device

A common approach for executing teleoperation tasks involves the use of a cost-effective, handheld platform, such as a haptic device, in conjunction with a robot within a master-slave configuration. In this specific approach, we examine the pairing of a Phantom Omni haptic device with a Panda cobot, as depicted in Figure 5.17. The difference in Degrees of Freedom (DoFs) between the Omni and Panda robot, or any over-actuated robot in general, imposes limitations, preventing independent control of all robot DoFs. This limitation necessitates the establishment of a mapping function, aligning the joint positions of the haptic device with those of the slave arm. The mapping is often achieved by mirroring the Cartesian pose or twist

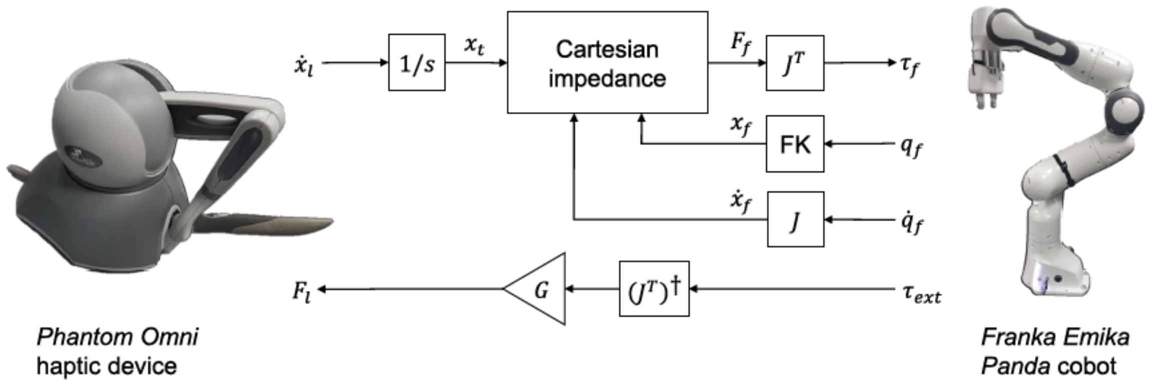


Fig. 5.17. Overview of the telemanipulation framework featuring the Phantom Omni haptic device and Franka Emika Panda cobot. The term FK represents the forward kinematic mapping, linking the follower joint configuration $\boldsymbol{q}_f$ to the end-effector pose $\boldsymbol{x}_f$.

of the haptic end-effector to that of the slave robot, offering an intuitive framework for users to position the robot for task execution. However, this approach introduces susceptibility to singularities and joint limits during inverse kinematics/dynamics calculations to determine the required joint-space motion. Additionally, the smaller range of motion provided by the Omni, both in joint and task space, presents a trade-off between the speed of coarse motion and the precision of fine positional alignment. This trade-off involves either scaling the motion of the haptic device to achieve larger robot motions or directly mapping the motion, potentially even reducing it. Furthermore, the limited force capabilities of the haptic device hinder a 1:1 mapping of force feedback from the robot to the haptic device. Consequently, the feedback may feel unnatural or fail to provide sufficient cues for the operator, especially when safety limits on force are exceeded.

5.5.2 Telemanipulation using two identical cobots

Alternatively, we explore a telemanipulation setup employing two identical Franka Emika Panda cobots configured in a master-slave arrangement (Figure 5.18). In this configuration, the user directly controls the master Franka arm, and its movements are directly replicated by the slave arm. Since both robots share an identical configuration, joint positions, and torques can be seamlessly transferred between them in a 1:1 mapping. While this mapping ensures natural and responsive user feedback, it also introduces potential safety risks as the user may be exposed to the full forces involved in specific tasks.
The joint space control of both arms offers advantages in handling singularities and joint limits, overcoming challenges associated with mapping Cartesian end-effector poses from the master to the task space of the slave robot. However, this control scheme may present challenges in performing tasks along specific directions, such as bolt removal or cutting, making them more complex and less intuitive for the user.

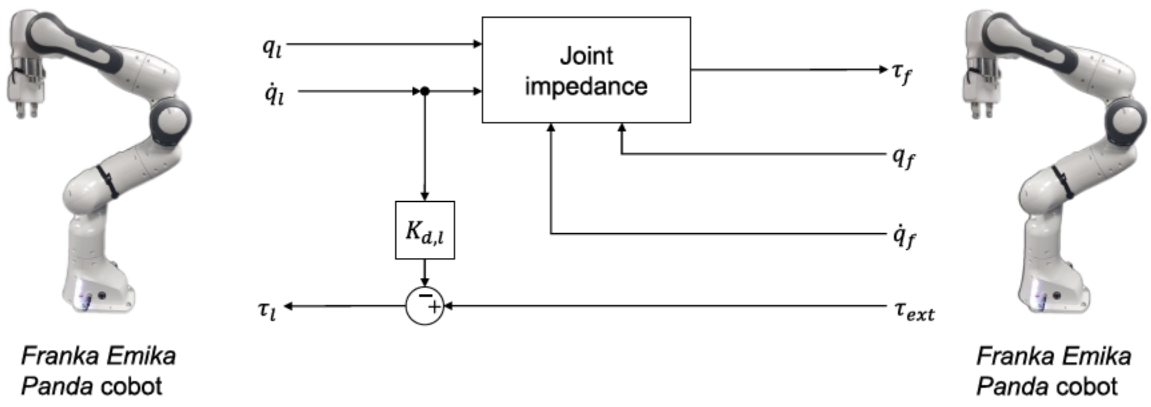


Fig. 5.18. Telemanipulation framework utilizing two identical Franka Emika Panda cobots.

5.5.3 Methodology

To establish a virtual coupling between master and slave devices, we begin by considering the dynamic equation of a rigid N-link manipulator in joint space:

$$\mathbf{M}(\boldsymbol{q})\boldsymbol{q} + \mathbf{C}(\boldsymbol{q}, \boldsymbol{q}) + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau}_{\text{ext}} + \boldsymbol{\tau} \tag{5.1}$$

Here, $\mathbf{M} \in \mathbb{R}^{N \times N}$, $\mathbf{C}(\boldsymbol{q}, \boldsymbol{q}) \in \mathbb{R}^{N \times N}$, $\boldsymbol{g}(\boldsymbol{q}) \in \mathbb{R}^{1 \times N}$ represent the joint-space inertia matrix, Coriolis and centrifugal matrix, and gravitational torques, respectively. Additionally, $\boldsymbol{\tau}_{\text{ext}} \in \mathbb{R}^{1 \times N}$ and $\boldsymbol{\tau} \in \mathbb{R}^{1 \times N}$ are the vectors of external and control torques acting on each link, with subscripts l and f denoting master and slave, respectively.
Given the haptic device's limitations, achieving a 1:1 mapping between $\boldsymbol{q}_l$ and $\boldsymbol{q}_f$ is impractical due to discrepancies in kinematic degrees of freedom and joint ranges compared with the Franka arm. Consequently, either mapping the joint space onto a reduced subset of the robot's full joint space or operating in a mutual task space is required. In this work, we employ a 6DoF Cartesian mapping for motion control of the slave Franka arm with the Phantom Omni master arm. As the operator moves the master arm within its workspace, we compute the delta Cartesian pose ${}^{P}\mathbf{T}\Delta$, which is then mapped to the end-effector delta pose ${}^{F}\mathbf{T}\Delta$ via the workspace transformation:

$${}^{F}\mathbf{T}_{\Delta} = {}^{P}_{F}\mathbf{T} \cdot {}^{P}\mathbf{T}_{\Delta} = \left[\begin{array}{cc} {}^{F}\mathbf{R}_{\Delta} & {}^{F}t_{\Delta} \\ \hline 0 & 1 \end{array}\right] \tag{5.2}$$

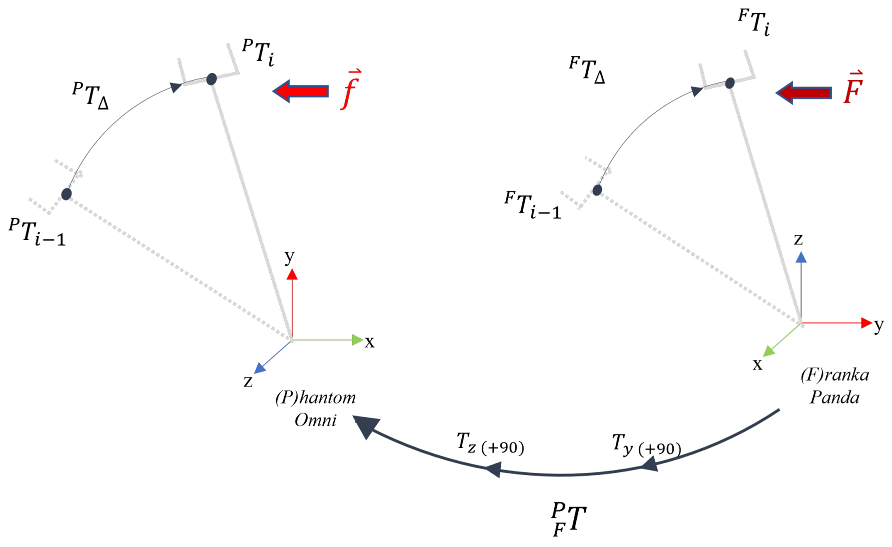


Fig. 5.19. Cartesian mapping for the master-slave teleoperation setup between the haptic device and Franka arm.

Here, $^{P}_{F}\mathbf{T}$ is the homogeneous transformation matrix from the Franka arm's base frame to the Phantom Omni's base frame, and $^{F}\mathbf{T}\Delta$ and $^{P}\mathbf{T}\Delta$ represent the delta transformation matrices for the Franka and Phantom, respectively. The position and orientation components of $^{P}\mathbf{T}\Delta$ are assigned to the desired velocity $\dot{\boldsymbol{x}}_t$, allowing us to compute the task space pose error $\boldsymbol{e}_x = \boldsymbol{x}_f - \boldsymbol{x}_t$ for the slave arm. The control law for the slave arm is then given by:

$$\boldsymbol{\tau}_f = \mathbf{J}^{\mathsf{T}} \left(-\mathbf{K}_p \boldsymbol{e}_x - \mathbf{K}_d \mathbf{J} \boldsymbol{q}_f\right) + \mathbf{C}(\boldsymbol{q}_f) + \boldsymbol{g}(\boldsymbol{q}_f) \tag{5.3}$$

Here, $\mathbf{J} \in \mathbb{R}^{N\times 6}$ is the slave manipulator Jacobian mapping joint to end-effector velocities, and $\mathbf{K}_p$ and $\mathbf{K}_d$ are controller stiffness and damping matrices, respectively. This results in the desired closed-loop dynamic behaviour:

$$\boldsymbol{\Lambda} \boldsymbol{x}_f + \mathbf{K}_d \boldsymbol{x}_f + \mathbf{K}_p \boldsymbol{e}_x = \boldsymbol{F}_{\text{ext}} \tag{5.4}$$

Force feedback is crucial for a bilateral teleoperation system, providing the user with a tactile perception of the slave robot's environment. The external force vector $\boldsymbol{F}_{\text{ext}}$ experienced at the Franka end-effector is transformed with respect to the Phantom, enabling the user to receive the force as $\boldsymbol{F}_l$:

$$\boldsymbol{F}_l = G \cdot {}^{P}_{F}\mathbf{T}^{-1} \cdot \boldsymbol{F}_{\text{ext}} \tag{5.5}$$

Because of the disparity in force capabilities between the master and slave devices, the feedback is adjusted by a factor of $G = 0.1$. This factor was empirically determined by comparing the maximum expected force across all tasks using preliminary data and scaling it to the haptic device's maximum force capabilities (3.3N). To minimize distortion of the force feedback and maintain consistency with the constant 1:1 feedback of the identical cobot setup, this factor was kept constant across all trials.
An essential aspect of the task space control scheme is addressing singularities and joint limits. In this scenario, a null space position regulation term is incorporated into (5.3) to prevent encountering joint limits. In contrast to pseudoinverse control, where large solution values can be obtained, the controller design in (5.3) results in singularities being observed as the torque command $\boldsymbol{\tau}_f$ tends toward zero along the singular directions.

For the Franka arm, we employ a joint impedance control scheme that directly maps the joint configuration of the master arm to the slave arm (Figure 5.18). The control law is defined as follows:

$$\boldsymbol{\tau}_f = -\mathbf{K}_p \boldsymbol{e}_q - \mathbf{K}_d \boldsymbol{e}_q + \mathbf{C}(\boldsymbol{q}_f) + \boldsymbol{g}(\boldsymbol{q}_f) \tag{5.6}$$

where $\boldsymbol{e}_q = \boldsymbol{q}_f - \boldsymbol{q}_l$ is the joint space error. represents the error in joint space. This leads to the closed-loop dynamics in joint space for the slave arm.

$$\mathbf{M}\boldsymbol{q}_f + \mathbf{K}_d \boldsymbol{e}_q + \mathbf{K}_p \boldsymbol{e}_q = \boldsymbol{\tau}_{\text{ext}} \tag{5.7}$$

To deliver force feedback to the user, the master control torques are computed as follows:

$$\boldsymbol{\tau}_l = \boldsymbol{\tau}_{\text{ext}} - \mathbf{K}_{d,l} \boldsymbol{q}_l \tag{5.8}$$

In general, note that the model parameters $\mathbf{C}$ and $\boldsymbol{g}$ are subject to uncertainty, leading to reduced tracking performance and the presence of steady-state error in both controllers. Tuning of the control gains $\mathbf{K}_p$ and $\mathbf{K}_d$ followed a similar process as that for G, through preliminary experiments, in line with other comparative studies like [129]. Specifically, the value of $\mathbf{K}_p$ was incrementally increased for each control strategy on the slave arm in isolation until instability occurred, while $\mathbf{K}_d$ was set to achieve critically damped behavior. This tuning approach enhances tracking performance in free space and minimizes the impact of uncertainties in the model parameters $\mathbf{C}$ and $\boldsymbol{g}$. For the joint control strategy, the gains were scaled down based on the torque capabilities of each joint.
Considering the higher forces applied to the master and the low stiffness of the human operator, an additional damping term $\mathbf{K}_{d,l}$ is introduced to the master's response. The value of $\mathbf{K}_{d,l}$ was determined by identifying the minimum damping value assigned to all joints, aiming to suppress oscillations resulting from feedback effects in the bilateral master-slave coupling.

5.5.4 Teleoperation experiments

For each task, the operator is presented with two fixed camera views of the module stack. In all tasks except cutting, these camera views remain constant throughout. For the cutting task, the operator is provided with two camera views of a material holder containing the cutting workpiece. Before starting each task, both the master and slave robots are initialized to a home position in joint space, maintaining consistency across all tasks. The following provides a more detailed explanation of each task. Figure 5.20, illustrates the experimental case studies conducted in order.

- **Unbolting:** In this task, the goal is to unfasten a set of four bolts securing an individual stack of modules. The operator utilizes a motorized universal socket wrench tool attached to the robot's wrist. Success is achieved when the bolt becomes manually removable without further unscrewing. If the success condition is not met on the first attempt or the robot's configured force thresholds (40N) are exceeded, the task is considered a failure.

- **Removing Fasteners:** Following the unbolting stage, the next step involves extracting each fastener from the stack before removing the cover. For this task, the Franka hand, configured with a grasping force of 50N, is employed. The operator must manoeuvre the hand to each fastener, grasp and remove the bolt from the stack, depositing it into a container. Due to limited camera views, the operator relies on a combination of tactile and visual exploration. Failure occurs if the bolt is not successfully grasped on the first attempt or if the grasp is lost outside the target container.

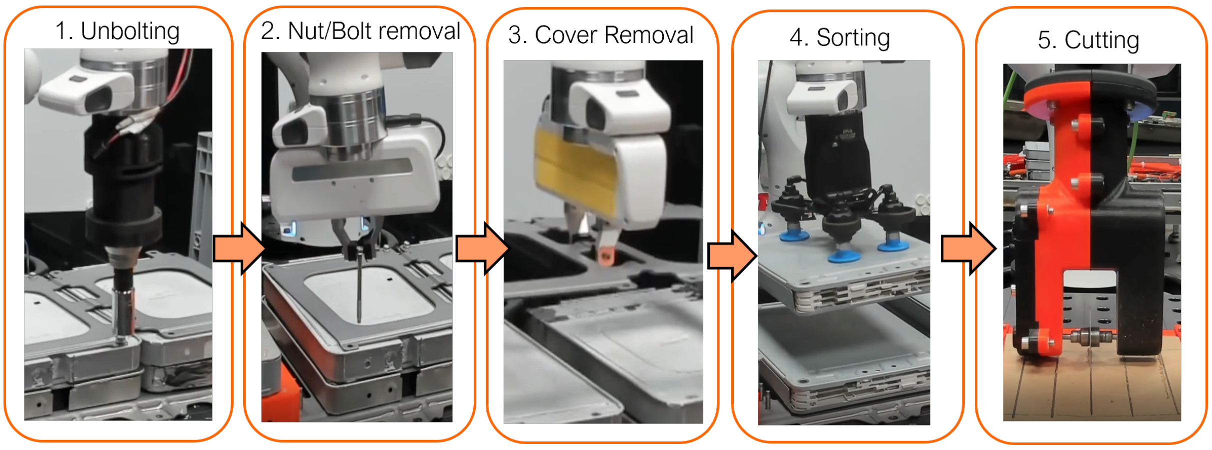


Fig. 5.20. Procedure of an electric vehicle (EV) battery disassembly detailing the case studies explored in sequence.

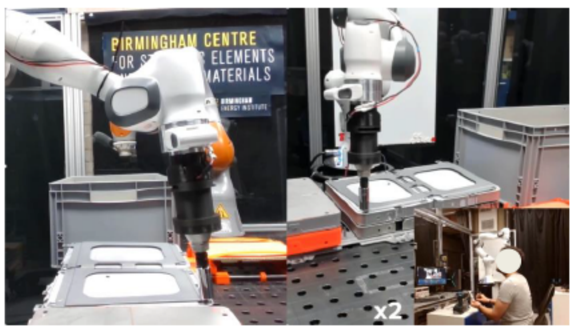
- **Removing Module Cover Plate:** After fastener removal, the operator proceeds to take off the module cover plate to access the underlying module stack. Considering the weight and geometry of the cover plate, finding a suitable grasp point is crucial for safe transportation. The configured grasping force is increased to 60N for all experiments. Failure conditions include an unsuccessful first grasp or losing the grasp during transportation.

- **Sorting Modules:** This case study involves unstacking and sorting EV battery modules using a vacuum suction gripper. The operator must remove a pair of modules from a stack and deposit them into a container. Visual positioning of the gripper and maintaining contact to engage the suction cups without exceeding force limits are essential. Similar to the cover removal, failure conditions include an unsuccessful first grasp or losing the grasp during transportation.

- **Contact Cutting** The operator cuts a planar material along a visually marked path using a slitting saw tool. As a benchmark, cutting a cardboard sheet is considered. Tactile feedback controls the cutting force, while visual feedback ensures precise positioning along the marked path. Failure occurs if the cut deviates more than $\pm$2.5mm from the desired path's centroid or if the cut is incomplete along parts of the path.

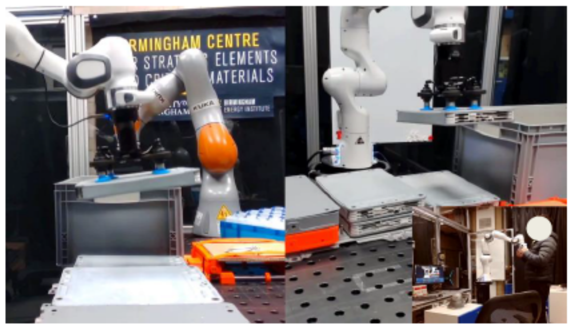
5.5.5 Teleoperation results

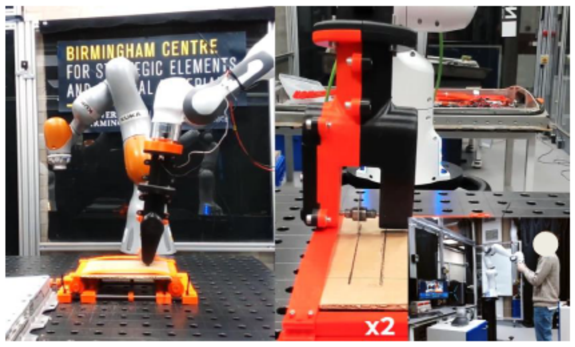
For each case study, Table 5.4 provides a comprehensive summary of the overall success rate and average completion time across all trials. The success rates for all tasks consistently surpass or equal 50%, with the lowest rate of 50% noted for the identical cobot in the task of bolt removal. Most tasks achieve success rates exceeding 75%, showcasing the viability of the module stack disassembly process using both cobot and haptic device telerobotics platforms. Figure 5.21 illustrates selected teleoperation tasks, showcasing the operator's engagement in the disassembly process using the proposed telemanipulation setup.
Comparing the two platforms, success rates are generally comparable for unbolting and bolt removal, with the operator's success rate 10-15% higher than the identical cobot case. Failures in bolt removal trials are attributed to misalignment, constituting 17% of failures, rising to 30% with identical cobots. Other causes include loss of grasp during bolt extraction (15% for both platforms) and releasing the bolt outside of the box (5%, 2.5% for haptic and cobot respectively).

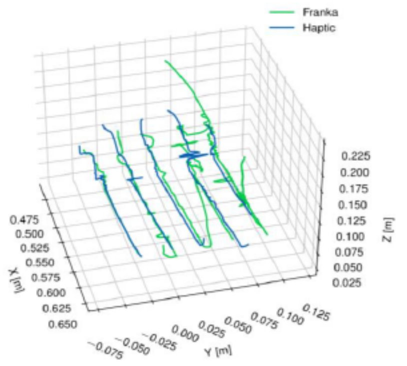

a) Unbolting – Haptic device

b) Nut/Bolt removal – Haptic device

c) Cover removal - Identical robot


d) Sorting - Identical robot


e) Cutting - Identical robot



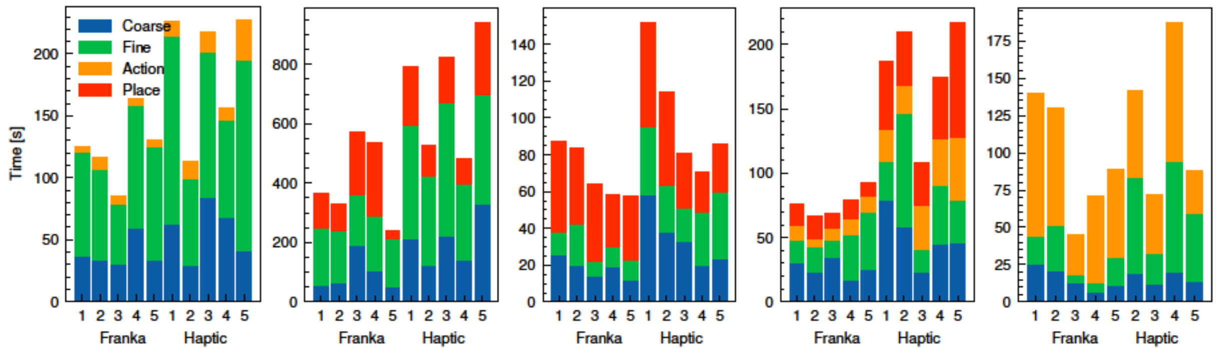
f) Cutting trajectories

Fig. 5.21. Various teleoperation tasks are depicted, offering a visual representation of the operator's involvement in the disassembly procedures through the telemanipulation system.

Cover removal emerges as the simplest task, achieving a 100% success rate across all operator trials with both platforms. In contrast, notable differences arise in unstacking and cutting tasks, where the identical cobot underperforms by 30% in the former and outperforms by 20% in the latter. The challenges in unstacking stem from the design mechanism of the vacuum suction gripper, requiring direct contact and perpendicular orientation for a successful grip. Manipulating the suction gripper's position and orientation in Cartesian space with the haptic platform yields a higher success rate than in joint space with the identical cobot, where failures occur due to orientation misalignment.
The findings from cutting trials underscore the significance of force feedback. The identical cobot platform exhibits a higher success rate because force feedback is maintained at a 1:1 scale, while the haptic platform scales it down due to limited force capabilities (Eq. 5.5). Consequently, the user does not experience the fully scaled forces exerted on the end-effector during cutting. Failure causes in the cobot platform include deviations from the desired path,

Table 5.4
Assessing performance metrics for disassembly tasks during telemanipulation utilizing the Phantom Omni haptic device and an identical cobot (Franka) platform.

	Haptic		Franka	
	Success rate[%]	Avg. time[s]	Success rate[%]	Avg. time[s]
Unbolting (4×)	95	188±23	85	124±13
Nut/bolt removal (8×)	63	713±89	50	410±63
Cover removal	100	101±15	100	70±6
Sorting modules (2×)	90	179±19	60	77±5
Cutting	60	122±26	80	95±18



(a) Unbolting (b) Bolt removal (c) Cover removal (d) Sorting (e) Cutting

Fig. 5.22. A detailed task trial breakdown of completion times between the Haptic and Franka telemanipulation masters. Data are excluded if the failure condition is met before task completion, typically due to a violation of configured force limits. Tasks are categorized into four phases: an initial "Coarse" phase involving approximate visual tool alignment, a "Fine" phase requiring precise visual and tactile alignment, an "Action" phase involving the task-specific action (e.g., unbolting, cutting), and a "Place" phase where the user deposits grasped objects.

while the haptic platform failures result from shallow, incomplete cutting or exceeding the robot's force limits.
Figure 5.22 provides a detailed breakdown of task completion times by trial, illustrating the time spent in each stage of the trials. The stages include a "Coarse" stage for rough visual alignment, a "Fine" stage for precise positioning, an "Action" stage for task interaction, and a "Place" stage for object release after successful grasping. The breakdown aids in interpreting the operators' efforts in completing each stage with respect to the two comparable platforms. Upon reviewing Figure 5.22, it is evident that the identical cobot platform achieved shorter average completion times across all tasks compared to the haptic platform. Notably, the most substantial difference in average time between the two platforms is observed for the bolt removal and sorting tasks, taking approximately 1.7 to 2.3 times longer to complete on average with the haptic device. Task completion times were consistently more stable between trials with the identical cobot platform for all tasks.
For a comprehensive comparison of the effect on completion time across tasks, the Standardized Mean Difference (SMD) effect size metric is employed. Values of $|\mathrm{SMD}| \geq 0.8$ indicate a significant effect, while the converse suggests a small or marginal effect. In the context of the module stack disassembly case study, the highest proportion of time was dedicated to unbolting and removing the retaining fasteners. This is evident in Figure 5.22a and 5.22b,

where the 'fine' stage comprises 60%, 63%, and 47%, 50% of the average task completion time, respectively, for both platforms. Therefore, the fine alignment stage significantly contributed to the overall reduction in task completion speed. This implies that regardless of the platform used, unbolting and bolt removal were the most demanding tasks, requiring a combination of precise visual and tactile alignment of the tool to successfully accomplish the disassembly. However, the SMD values for unbolting and bolt removal are 1.09 and 1.24, respectively, indicating significant improvements afforded by the identical cobot.

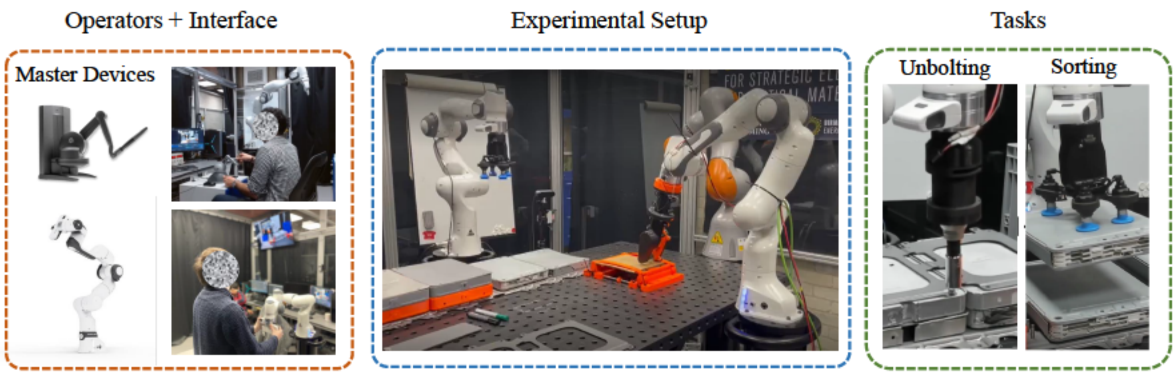


Fig. 5.23. This figure depicts the teleoperation setup where a haptic device and an identical robotic arm are employed as master devices to teleoperate slave robots for the tasks of unbolting and sorting.

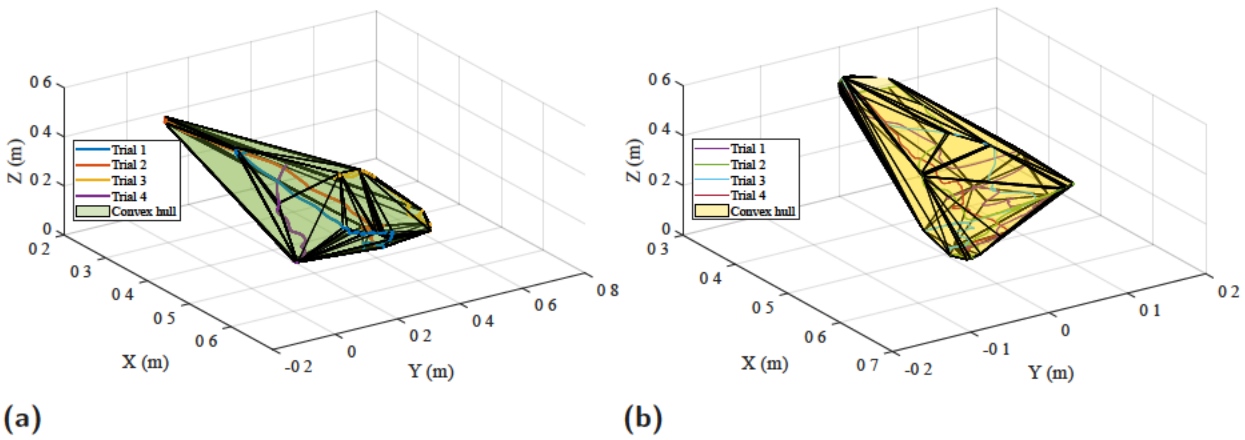


Fig. 5.24. Defining boundaries around the human demonstrator trajectories, (a) for the task of unbolting, (b) for the task of sorting.

Following this study, we have leveraged some of the demonstrations to perform unbolting and sorting tasks (Fig 5.23). Subsequently, we depicted the convex hull created around these demonstrations in Fig 5.24. These convex hulls represent initial demonstrations to highlight the potential of our approach. The bounded workspace would be later used in Reinforcement Learning to automate and optimize these tasks. By confining the workspace, we limit the search space for the RL agent, with a higher probability of finding optimal solutions in these constrained areas. The concept of utilizing these limited bounds extends beyond the proposed applications in this study. Once reward functions are defined for specific tasks, these precomputed bounded workspaces can be readily employed to boost the RL process in a variety of real-world scenarios. This flexibility in applying our method proposed in Chapter 4 underscores its potential to significantly accelerate the development and deployment of RL-based solutions.

5.6 | Conclusion

In conclusion, Chapter 5 provides a comprehensive overview of the experimental setups and disassembly tasks employed in battery disassembly, showcasing the versatility of various robotic systems. The integration of robots, grippers, haptic devices, vision systems, and advanced techniques such as deep learning, model-based tracking, and visual servoing is highlighted. The chapter thoroughly explores specific disassembly tasks, shedding light on their objectives, challenges, and the pivotal role played by different approaches. The visual context provided through images enhances the understanding of both setups and procedures. Moreover, This study investigated the telerobotic disassembly of a module stack from the Nissan Leaf 2011 battery pack. The research conducted a comparative analysis involving two distinct setups: a master-slave configuration employing a haptic device to control a cobot, and a scenario with two identical cobots. The evaluation focused on assessing the success rate and completion time across various tasks, including unbolting, bolt extraction, cover plate grasping and removal, module sorting using a suction gripper, and contact cutting. The findings from this study provide valuable insights into the performance and efficiency of these setups, shedding light on the strengths and limitations of each configuration in the context of battery disassembly.
Moreover, we demonstrated that our proposed Visual Servoing and Model-Based Tracker packages have shown adaptability to various platforms and support different programming languages in both simulation and real-world scenarios. The prerequisite for deploying these packages is the calibration of the attached camera on the wrist of any robot, as explained in section 7.1. For the VS package, the calculated Cartesian velocity commands are transmitted as twist messages through three distinct topics, corresponding to the IBVS, PBVS, and DHVS methods. Additionally, the Model-Based Tracking package provides the 3D position of each model in the camera frame. This enables the robot to move accordingly, allowing for the use of different tools to perform diverse tasks. The package includes models for tracking both modules and bolts. However, by incorporating the CAD model of each component, the package can be extended to detect and localize various components. Users are provided with three tracking options: Moving Edge Tracker for objects without texture, KLT Key points Tracker for objects with textured surfaces, and the Hybrid Tracker, which combines both approaches. Future work will concentrate on assessing the impact of variable autonomy on task completion performance within the framework of the disassembly case studies presented. Additionally, there is the possibility to incorporate our proposed workspace bounding approach from Chapter 4. This involves merging the data obtained from human demonstrations with RL techniques to automate and further enhance tasks that are currently performed manually through telemanipulation. This integration aims to bring about advancements in the efficiency and automation of the disassembly processes, leveraging both human expertise and machine learning algorithms.

Data availability

The data supporting the findings presented in this chapter are also available in the following GitHub repositories:
https://github.com/aaflakiyan/TouchX-Teleoperation-Matlab-ROS
https://github.com/aaflakiyan/DHVS
https://figshare.com/articles/media/Robotic_Disassembly/25498360

CHAPTER 6

Conclusion

In conclusion, this PhD project represents important steps towards improving the capabilities of robotic manipulation systems by integrating artificial intelligence techniques with computer vision methods. The study is divided into multiple chapters, each of which focuses on a different aspect of the main objective. The journey begins with the optimization of hybrid visual servoing approaches, addressing convergence issues and introducing adaptive gains and neuro-fuzzy neural networks for enhanced control precision and robustness. The study explores further the integration of RL with visual servoing, extending AI intervention from image spaces to joint spaces. Incorporating expert demonstrations into the learning process, this extension results in faster and more effective learning. The research then extends into the domain of contact-rich manipulation tasks, introducing a RL framework to navigate complex and uncertain environments. This framework enhances the robustness and adaptability of robotic systems in tasks like compliant path-following. In this study, human demonstrations are also utilised to limit the agent's workspace, preventing it from exploring unnecessary areas and avoiding sub-optimal solutions. Finally, the practical applicability of the developed methods through a series of experiments that explore battery disassembly in the context of various robots, grippers, haptic devices, and deep learning techniques has been investigated.
In the following sections, we will go over a summary of the key findings from the research. This investigation will not only highlight the accomplishments and advancements made in the integration of AI and computer vision with robotic manipulation but will also openly discuss the limitations that have been identified. Following the summary of key findings, the focus will shift to recognising these boundaries and constraints encountered during the research process. These constraints are essential for understanding the scope and applicability of the developed methodologies. Furthermore, an examination of possibilities for future research is explained. Identifying and addressing these future research directions is critical to the field's ongoing development and the refinement of existing approaches. Finally, the story progresses into the realm of applications. The practical implications and potential applications of the techniques that have been developed will be explained, understanding where and how to apply these methodologies.

6.1 | Summary of key findings

In this section, the summary of key findings from my research is explained.
In chapter 2, the proposed Hybrid Decoupled Visual Servoing (HDVS) method emerges as a robust and optimized solution to address limitations inherent in classical 2D, 3D, and Hybrid visual servoing methods. The key findings can be summarized as follows:

- Decoupling and Optimization: HDVS effectively decouples rotations and translation in the Z-axis, regulating these decoupled errors to zero through 3D reconstruction of visual features while regulating X-axis and Y-axis errors in 2D. This decoupling strategy leads to optimized solutions for both the end effector and trajectories in the image space.

- Neuro-Fuzzy Approximation: The use of the Local Linear Model Tree (LoLiMoT) neuro-fuzzy neural network enhances the control system's robustness, approximating the pseudo-inverse of the interaction matrix. This not only reduces computational expenses but also ensures the avoidance of singularities and ill-conditioning, making the method more resilient to image noises and camera parameters.

- Adaptive Gains and Damped Least Square: The incorporation of adaptive gains expedites the visual servoing operation, contributing to reduced convergence times. The application of the Damped Least Squares method mitigates robot singularities and smoothes discontinuities, further enhancing the method's performance.

- Enhanced Performance Metrics: Comparative evaluations against classical image-based, position-based, and hybrid visual servoing methods, conducted both in simulation and with a 7-degree-of-freedom arm robot in the real world, reveal superior performance. HDVS demonstrates more efficient optimized trajectories, shorter camera paths, and heightened manipulability compared to existing methods.

- Robustness and Object Tracking: HDVS exhibits robustness to camera calibration and image noises, outperforming classical position-based visual servoing methods. Furthermore, the method reduces the likelihood of losing the object from the camera's field of view, ensuring a more reliable and controllable robotic manipulation process.

Chapter 3 presents a significant advancement in the integration of RL techniques with the visual servoing method. Key findings include:

- Expansion to Joint Space: The incorporation of RL techniques extends the VS method beyond image spaces to the joint space of the robot, establishing a direct mapping from image features to desired joint velocities.

- Addressing Data Insufficiency: A novel approach is introduced to mitigate the data insufficiency challenge inherent in RL. By leveraging demonstrations from multiple controllers, the method combines RL, Learning from Demonstration (LfD), and Ensemble Learning, resulting in a more efficient learning paradigm.

- Case Studies and Performance Improvement: Two case studies demonstrate the efficacy of the proposed method. Notably, the integration of a hypercube to constrain the action space significantly reduces training time, resulting in a 51% reduction in achieving the desired performance level compared to RL alone. The method also exhibits improved performance compared to classical VS methodologies.

- Bounding Actions: Four methods for bounding actions of the RL agent during training are explored, with the convex hull and modified loss function (AORLD-HL) proving the most effective. This method enhances the exploration-exploitation trade-off, leading to improved average reward progress during training.

- Versatility: The AORLD method shows versatility in adapting to various expert supervisors and provides a promising approach for addressing challenges in high-dimensional action spaces.

Chapter 4 introduces a framework for learning contact-rich tasks using Reinforcement Learning (RL), with a focus on robustness to parametric uncertainties. The key findings include:

- Curriculum-Based Domain Randomization (DR): The proposed RL method employs curriculum-based domain randomization with a time-varying sampling distribution. This approach enhances the robustness of the policy to parametric uncertainties in the robot-environment system, demonstrating superior performance in compliant path-following tasks.

- Improved Trajectory Accuracy: Evaluation in simulation for compliant path-following case studies reveals a substantial improvement in trajectory accuracy compared to traditional methods. The RL approach shows approximately 15 times improvement over a previous Learning-Based Model Predictive Control (LBMPC) method and 18 times improvement over a Virtual Forward Dynamics Model (FDM) approach.

- Boosting RL with Human Demonstrations: A novel approach is introduced to boost deep RL using human demonstrations and offline workspace bounding. The method reduces the search space, expedites learning, and improves the policy's performance and resilience to local minima.

Chapter 5 of the thesis focuses on the experimental setups and disassembly tasks involved in battery disassembly using various robotic systems. The key findings include:

- Versatile Robotic Systems: The chapter highlights the versatility of robotic systems, including robots, grippers, haptic devices, vision systems, and advanced techniques such as deep learning, model-based tracking, and visual servoing.

- Showcasing Disassembly Tasks: Specific disassembly tasks, such as sorting, unbolting/unscrewing, cutting, and teleoperation, are thoroughly described. The chapter provides insights into the objectives, challenges, and the crucial role of different components in executing these tasks.

- Vision Methodologies: The exploration extends into vision methodologies, with a focus on model-based tracking and deep learning approaches. Model-based tracking involves using component models to determine 3D positions from 2D images, while deep neural network models, trained using transfer learning, autonomously detect objects in the camera scene.

- Telerobotic: A comparative analysis of two distinct setups for telerobotic disassembly of a module stack from the Nissan Leaf 2011 battery pack is presented. The evaluation assesses the success rate and completion time across various tasks, providing valuable insights into the performance and efficiency of each configuration.

- Adaptability of Visual Servoing and Model-Based Tracker Packages: The study demonstrates the adaptability of the Visual Servoing and Model-Based Tracker packages to various platforms and programming languages in both simulation and real-world scenarios. These packages enhance robot movements based on calculated Cartesian velocity commands and provide 3D positions of tracked components.

- The Visual Servoing and Module Tracker packages are provided in two versions for different needs. The standalone C++ version ensures flexibility and independence, enabling users who may not be utilizing the Robot Operating System (ROS) to incorporate the functionalities into their applications. On the other hand, the ROS-compatible version is designed to integrate into ROS-based robotic systems, taking advantage of ROS features like communication between nodes, topics, services, and visualization tools. This version facilitates integration with other ROS packages and tools.

6.2 | Limitations

In this section, we will elaborate on the limitations associated with each proposed method and algorithm introduced in this thesis.

6.2.1 Algorithmic Limitations

HDVS Approach (Chapter 2): While the HDVS approach, introduced in Chapter 2, showcases impressive capabilities, it is important to acknowledge certain limitations, including adaptability to high dimensions, sensitivity to camera model and impact of structural changes in camera mount. The HDVS method is currently optimized for scenarios involving four features in the image, with predefined desired convergence points. This design choice caters to specific applications where the same number of features are considered. However, there are applications in which the user needs to define more features, and the input space of the NN will increase. Investigating the adaptability of the learning method to higher dimensions needs further investigation. In this study, we exclusively applied the HDVS with the LoLiMoT neural network to the Franka robotic arm and its specific camera structure. For other camera platforms (with new extrinsic parameters), a modified version of our proposed method (we call it DHVS) was utilized. The modification involves excluding the learning component responsible for approximating the proposed image Jacobian matrix. The core methodology remains consistent with the approach employed for the Franka arm, with the necessary adjustments made to accommodate the new parameters. Therefore, to use HDVS for alternative robots and camera structures, one needs to collect relevant data with the specific setup of interest and subsequently retrain the model. This retraining procedure ensures the adaptability of the HDVS method to provide the camera's desired velocities based on the unique structure of the new setup.

Reinforcement Learning for Visual Servoing (Chapter 3): In chapter 3, we have investigated the Use of Reinforcement learning for the task of VS with stationary robots. Utilizing RL for VS introduces notable advantages, particularly in addressing issues related to robot control and image Jacobian calculations. However, it comes with limitations to the chosen robot and camera structure in training. Despite the comprehensive integration of both image space and robot control in RL for VS, the trained RL policy is highly specific to the model of the chosen robot. The mapping of feature errors to joint velocities is linked to the geometry and kinematics of the robot involved in the training process. Consequently, deploying the RL-based VS method on a different robot type requires the repetition of the training process. However, it should be mentioned that the proposed AORLD approach can be applied to a broad spectrum of applications where systematic controllers exist, and the objective is to enhance their behaviour. This broader applicability underscores the versatility of AORLD across diverse scenarios, making it a valuable tool for improving the performance of systematic controllers in various contexts.

Simulation-based Policy Development (Chapter 4): A notable limitation from Chapter 4 lies in the fact that all experiments conducted, involving the application of RL and curriculum-based domain randomization to develop a robust policy for environments with uncertainties, were carried out in simulation. The transition from simulation to the real-world scenario introduces a set of challenges, particularly due to the inherently contact-rich nature of the task which could be studied in future works. The study aimed to showcase the possibilities and demonstrate the effectiveness of our method primarily in simulation, serving as a foundation for understanding its potential real-world applications. However, the simulation environment may not perfectly replicate the complexities and dynamics of the actual contact-rich interactions that occur in the real-world setting. Therefore, the inherent challenges of translating simulation results to real-world performance, especially in tasks involving significant physical contact, highlight a limitation that should be acknowledged.

Deep Learning for Object Detection (Chapter 5): Another limitation of this thesis pertains to the deep learning method, introduced in chapter 5, which could be enhanced through the expansion of the dataset and fine-tuning of training parameters. The inclusion of this section aimed to showcase the potential of employing deep learning for object detection, an alternative to initializing the image in the model-based tracker. Further improvements in the deep learning model's performance may be achieved through a more extensive dataset and optimization of training parameters to develop a more robust object detector. Moreover, in the model-based tracker package, while the tracker offers adaptability, the scope is limited to the CAD model of each component. Therefore, extending the package to detect and localize various components requires incorporating CAD models.

6.2.2 Hardware and Structural Limitations

Camera Setup and Mounting (Chapter 2): It is noted that changes in the camera setup, such as replacement or adjustment, necessitate retraining of the HDVS model. The model's robustness to camera calibration is evident, but significant structural changes in the camera mount, such as tilting or rotating, present challenges that require pre-training the model to handle various angles. This limitation is particularly relevant in scenarios where frequent changes in camera setup are necessary.

Real-world Applications (Chapter 4): The reliance on simulation for developing and testing the RL and curriculum-based domain randomization methods limits the direct applicability of the results to real-world scenarios. The inherent differences between simulated environments and real-world contact-rich interactions could affect the performance of the developed policies when applied to actual tasks.

6.3 | Implications for future research

Task Planner Development: The outcome of the battery disassembly project would be to create a general task planner. This would be a big step for future research. This task planner would serve as a higher-level decision-making component, determining optimal actions for various robotic systems and grippers during the disassembly process. By leveraging the information about the environment obtained through various sensors, the task planner can intelligently allocate specific tasks to different robots and grippers. It would strategically plan the sequence of actions to streamline and optimize the overall disassembly process. Using a well-designed task planner aims to enhance efficiency, reduce processing time, and ensure the systematic execution of disassembly tasks, contributing to the advancement of automation in disassembly projects. In this thesis, we have focused on various tasks individually. Our proposed approaches involve the integration of artificial intelligence with vision and robotic control, resulting in the development of robust algorithms designed to address various sorts of challenges. Moving forward, an important direction for future work involves incorporating the developed algorithms into a unified task planner. This task planner not only does routine disassembly tasks but also learns and gets better over time (utilizing the power of RL).

Deep Learning Expansion: Another possible future work for the disassembly project includes exploring the integration of deep learning not only for specific component detection but also for identifying the car company that utilizes each component. This extension could broaden the scope of the disassembly project. Additionally, using more robust and advanced YOLO methods in future setups could enhance the accuracy and efficiency of component detection, localization, and segmentation. Furthermore, future works could explore the application of deep learning techniques to detect defective and distorted objects within the disassembly process. Implementing a system that considers the percentage of defectiveness could enhance the precision and adaptability of the disassembly process, contributing to a more efficient and reliable overall system. This ability would enable the system to identify issues in components and take appropriate actions, such as sorting, handling the task by an operator, or telemanipulation from a distance, based on the degree of defectiveness. Another aspect with room for improvement is the model-based tracker, which could be further modified to extend its capability to track various types of components within the battery pack. In this study, the model-based tracker was employed for tracking bolts and modules, and there is potential for future development to enable multi-object tracking. Enhancements to the model-based tracker could involve adaptations to accommodate diverse components, expanding its application to a broader range of object-tracking scenarios within the battery pack. The integration of object detection, segmentation, and localization techniques could play a crucial role in the development of the general task planner.

Vision Technique Enhancement: Another future exploration would be enhancing the versatility of the proposed VS technique. The current focus on tracking features using tag markers can be expanded to accommodate more sophisticated scenarios. The aim would be to adapt the VS method for tracking identified components through deep learning or the model-based tracker package. This expansion would expand to situations where a greater set of features needs to be tracked.

Teleoperation and RL Integration: Furthermore, the integration of Reinforcement Learning and the proposed workspace limitation approach, in Chapter 5, provides a pathway to automate challenging tasks done with teleoperation. To this end, the data collected from human demonstrations would be used to limit the search space of the RL agent. Thereafter, the agent will learn to perform the same tasks done in teleoperation, automatically by learning from its behavior and improving itself.

An Example of Systematic Task Planner Development: An example of a systematic task planner could be as follows: The initial phase involves the detection and localization of objects within the workspace. Subsequently, an in-depth analysis of objects takes place, coupled with task determination. This process begins with objects requiring minimal robot motion and effort, gradually extending to more complex tasks. Additionally, the task planner considers the robot's effort and power, potentially informed by a learning process. Task priorities are established, taking into account different criteria such as the nature of the task and the robot's capabilities. The subsequent stages encompass robot control implementation, ensuring precise and effective robotic movements, followed by the execution of specific tasks. These tasks encompass a range of actions, including cutting, grasping, pick-and-place manoeuvres, suction operations, and measurements, all orchestrated to enhance the overall efficiency and automation of the disassembly workflow. In case of unsuccessful task execution, the planner will flag the issue, prompting intervention. An operator can then take over and perform the task manually using teleoperation.

6.3.1 Possible Applications

Based on the contribution of this study, there are some likely end applications for my research: **Industrial Robotic Automation:**This research findings could be applied to enhance the capabilities of industrial robots in manufacturing and assembly lines. The optimized visual servoing and reinforcement learning techniques can improve the efficiency and precision of tasks such as pick-and-place operations, assembly of complex components, and handling objects in dynamic environments.

Medical and Healthcare Robotics: The integration of AI-driven control strategies can find applications in medical robotics, enabling more accurate and delicate procedures such as surgical tasks or handling medical instruments. The enhanced manipulation and adaptability of robotic systems can lead to safer and more efficient medical interventions.

Logistics and Warehousing: This research could be extended to logistics and warehousing automation, where robots are increasingly utilized for tasks like sorting, packing, and moving items. The developed control methods could optimize the manipulation of objects with varying shapes, sizes, and weights, improving the overall efficiency of distribution centers and warehouses.

Agricultural Robotics: The AI-enhanced manipulation techniques could be employed in agricultural settings to handle delicate tasks such as fruit harvesting or plant maintenance. Robots equipped with optimized control strategies could navigate complex environments, ensuring minimal damage to crops while improving harvesting efficiency.

Space Exploration and Manufacturing: The concepts developed in this research could be adapted for space missions, where robots play a crucial role in tasks like assembling structures or conducting experiments in microgravity environments. The robust control methods could aid in precise and reliable manipulation of objects in space.

Assistive and Rehabilitation Robotics: The research outcomes might have implications for assistive devices and rehabilitation robotics. Optimized manipulation strategies could be integrated into robotic prosthetics or exoskeletons, enabling more natural and effective movement for individuals with mobility impairments.

Construction Robotics: The advanced control techniques we have developed could be applied to construction robots, improving their ability to handle tasks like bricklaying, concrete pouring, and material transportation on construction sites. This could lead to faster and more efficient construction processes.

Environmental Monitoring and Maintenance: This research findings could be utilized in the development of robots designed for environmental monitoring and maintenance tasks. These robots could navigate challenging terrains to perform tasks like sensor placement, data collection, and maintenance of infrastructure in remote or hazardous locations.

Human-Robot Collaboration: The adaptive control strategies and AI-driven manipulation techniques could facilitate safer and more productive collaboration between humans and robots. In settings like warehouses, factories, or healthcare facilities, robots could work alongside humans, assisting in tasks that require precision, strength, or repetitive actions.

In the end, the findings of this study contribute to a new paradigm in robotic manipulation in which the interaction of AI and traditional control strategies redefines the capabilities of robotic systems. The proposed methodologies open the path for higher levels of autonomy, precision, and adaptability, making major improvements to the fields of robotics and artificial intelligence.

Bibliography

1. Lander, L., Tagnon, C., Nguyen-Tien, V., Kendrick, E., Elliott, R. J., Abbott, A. P., Edge, J. S. & Offer, G. J. Breaking it down: A techno-economic assessment of the impact of battery pack design on disassembly costs. *Applied Energy* **331,** 120437 2023.
2. Ai, N., J. Zheng & W.-Q. Chen, "U.S. end-of-life electric vehicle batteries: Dynamic inventory modeling and spatial analysis for regional solutions". *Resources, Conservation and Recycling* 145, 208–219 2019.
3. Thies, C., K. Kieckhäfer, C. Hoyer & T. S. Spengler. Recycling of Lithium-Ion Batteries: The LithoRec Way. eds Kwade, A. & Diekmann, J. , 253–266. Cham, Springer International Publishing, 2018.
4. Berckmans, G., Messagie, M., Smekens, J., Omar, N., Vanhaverbeke, L. & Van Mierlo, J. Cost projection of state of the art lithium-ion batteries for electric vehicles up to 2030. *Energies* **10,** 1314 2017.
5. Harper, G., Sommerville, R., Kendrick, E., Driscoll, L., Slater, P., Stolkin, R., Walton, A., Christensen, P., Heidrich, O., Lambert, S., *et al.* Recycling lithium-ion batteries from electric vehicles. *Nature* **575,** 75–86 2019.
6. Lander, L., C. Tagnon, V. Nguyen-Tien, E. Kendrick, R. J. Elliott, A. P. Abbott, J. S. Edge & G. J. Offer, "Breaking it down: A techno-economic assessment of the impact of battery pack design on disassembly costs". *Applied Energy* 331, 120437 2023.
7. Pehlken, A., S. Albach & T. Vogt, "Is there a resource constraint related to lithium ion batteries in cars?". *The International Journal of Life Cycle Assessment* 22, 40–53 Jan. 2017.
8. Tang, Y., & M. Zhou, *Fuzzy-Petri-net based disassembly planning considering human factors*. 2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583). 5. 2004, 4195–4200 vol.5.
9. Tan, W. J., C. M. M. Chin, A. Garg & L. Gao, "A hybrid disassembly framework for disassembly of electric vehicle batteries". *International Journal of Energy Research* 45, 8073–8082. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/er.6364` 2021.
10. Zorn, M., C. Ionescu, D. Klohs, K. Zähl, N. Kisseler, A. Daldrup, S. Hams, Y. Zheng, C. Offermanns, S. Flamme, C. Henke, A. Kampker & B. Friedrich, "An Approach for Automated Disassembly of Lithium-Ion Battery Packs and High-Quality Recycling Using Computer Vision, Labeling, and Material Characterization". *Recycling* 7 2022.
11. Zhang, H., Yang, H., Wang, H., Wang, Z., Zhang, S. & Chen, M. *Autonomous Electric Vehicle Battery Disassembly Based on NeuroSymbolic Computing* in *Intelligent Systems and Applications* ed Arai, K. Springer International Publishing, Cham, 2023, 443–457.
12. Choux, M., E. Marti Bigorra & I. Tyapin, "Task Planner for Robotic Disassembly of Electric Vehicle Battery Pack". *Metals* 11 2021.

13. Farhan, A. S., A. M. Bassiouny, Y. T. Afif, A. A. Gamil, M. A. Alsheikh, A. M. Kamal, K. S. Elenany, M. A. Bahour, M. I. Awad & S. A. Maged, *Autonomous Non-Destructive Assembly/Disassembly of Electronic Components using A Robotic Arm*. 2021 16th International Conference on Computer Engineering and Systems (ICCES). 2021, 1–7.

14. Rastegarpanah, A., Aflakian, A. & Stolkin, R. Improving the Manipulability of a Redundant Arm Using Decoupled Hybrid Visual Servoing. *Applied Sciences* **11,** 11566 2021.

15. Rastegarpanah, A., Aflakian, A. & Stolkin, R. Optimized hybrid decoupled visual servoing with supervised learning. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering,* 09596518211028379 2021.

16. Aflakian, A., Rastegarpanah, A. & Stolkin, R. Boosting Performance of Visual Servoing using Deep Reinforcement Learning from Multiple Demonstrations. *IEEE Access* 2023.

17. Hathaway, J., Shaarawy, A., Akdeniz, C., Aflakian, A., Stolkin, R. & Rastegarpanah, A. Towards Reuse and Recycling of Lithium-ion Batteries: Tele-robotics for Disassembly of Electric Vehicle Batteries. *arXiv preprint arXiv:2304.01065* 2023.

18. Servoing, V. *Real Time Control of Robot Manipulators Based on Visual Sensory Feedback* 1993.

19. Chaumette, F., Hutchinson, S. & Corke, P. in *Springer Handbook of Robotics* 841–866 Springer, 2016.

20. Li, C., Li, B., Wang, R. & Zhang, X. A survey on visual servoing for wheeled mobile robots. *International Journal of Intelligent Robotics and Applications* **5,** 203–218 2021.

21. Dewi, T., Risma, P., Oktarina, Y. & Muslimin, S. *Visual servoing design and control for agriculture robot; a review* in *2018 International Conference on Electrical Engineering and Computer Science (ICECOS)* 2018, 57–62.

22. Peller-Konrad, F. H^{2}T Research-Grasping and Manipulation-Visual Servoing. *IEEE/ASME Transactions on Mechatronics* 2023.

23. Sun, C., Liu, C., Feng, X. & Jiao, X. Visual servoing of flying robot based on fuzzy adaptive linear active disturbance rejection control. *IEEE Transactions on Circuits and Systems II: Express Briefs* **68,** 2558–2562 2021.

24. Shkurti, F., Chang, W.-D., Henderson, P., Islam, M. J., Higuera, J. C. G., Li, J., Manderson, T., Xu, A., Dudek, G. & Sattar, J. *Underwater multi-robot convoying using visual tracking by detection* in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* 2017, 4189–4196.

25. Saeidi, H., Opfermann, J. D., Kam, M., Wei, S., Léonard, S., Hsieh, M. H., Kang, J. U. & Krieger, A. Autonomous robotic laparoscopic surgery for intestinal anastomosis. *Science robotics* **7,** eabj2908 2022.

26. Abdul Hafez, A. H., V. V. Anurag, S. V. Shah, K. M. Krishna & C. V. Jawahar, *Reactionless visual servoing of a dual-arm space robot*. 2014 IEEE International Conference on Robotics and Automation (ICRA). 2014, 4475–4480.

27. Rastegarpanah, A., Ahmeid, M., Marturi, N., Attidekou, P. S., Musbahu, M., Ner, R., Lambert, S. & Stolkin, R. Towards robotizing the processes of testing lithium-ion batteries. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering,* 0959651821998599 2021.

28. Vicentini, F., Pedrocchi, N., Beschi, M., Giussani, M., Iannacci, N., Magnoni, P., Pellegrinelli, S., Roveda, L., Villagrossi, E., Askarpour, M., *et al.* in *Bringing Innovative Robotic Technologies from Research Labs to Industrial End-users* 57–96 Springer, 2020.

29. Roveda, L., Castaman, N., Ghidoni, S., Franceschi, P., Boscolo, N., Pagello, E. & Pedrocchi, N. *Human-robot cooperative interaction control for the installation of heavy and bulky components* in *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* 2018, 339–344.

30. BOGE, C. Visual servoing for single and dual arm manipulation tasks in humanoid robots. *at-Automatisierungstechnik* **60,** 309–317 2012.

31. Francois, C. & Hutchinson, S. Visual servo control Part I: Basic approaches. *IEEE Robot. Autom. Mag* **13,** 82–90 2006.

32. Deng, L., Janabi-Sharifi, F. & Wilson, W. J. *Stability and robustness of visual servoing methods* in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)* **2** 2002, 1604–1609.

33. Han, X.-F., Laga, H. & Bennamoun, M. Image-based 3D Object Reconstruction: State-of-the-Art and Trends in the Deep Learning Era. *arXiv preprint arXiv:1906.06543* 2019.

34. Chaumette, F. in *The confluence of vision and control* 66–78 Springer, 1998.

35. Corke, P. I. & Hutchinson, S. A. A new partitioned approach to image-based visual servo control. *IEEE Transactions on Robotics and Automation* **17,** 507–515 2001.

36. Palmieri, G., Palpacelli, M., Battistelli, M. & Callegari, M. A comparison between position-based and image-based dynamic visual servoings in the control of a translating parallel manipulator. *Journal of Robotics* **2012** 2012.

37. Hafez, A. A., Cervera, E. & Jawahar, C. *Hybrid visual servoing by boosting IBVS and PBVS* in *2008 3rd International Conference on Information and Communication Technologies: From Theory to Applications* 2008, 1–6.

38. Hafez, A. A. & Jawahar, C. *Probabilistic integration of 2D and 3D cues for visual servoing* in *2006 9th International Conference on Control, Automation, Robotics and Vision* 2006, 1–6.

39. Gans, N. R. & Hutchinson, S. A. Stable visual servoing through hybrid switched-system control. *IEEE Transactions on Robotics* **23,** 530–540 2007.

40. Kumar, D. S. & Jawahar, C. in *Computer vision, graphics and image processing* 906–918 Springer, 2006.

41. Chesi, G., Hashimoto, K., Prattichizzo, D. & Vicino, A. Keeping features in the field of view in eye-in-hand visual servoing: A switching approach. *IEEE Transactions on Robotics* **20,** 908–914 2004.

42. Cervera, E., Del Pobil, A. P., Berry, F. & Martinet, P. Improving image-based visual servoing with three-dimensional features. *The International Journal of Robotics Research* **22,** 821–839 2003.

43. Malis, E., Chaumette, F. & Boudet, S. 2 1/2 D visual servoing. *IEEE Transactions on Robotics and Automation* **15,** 238–250 1999.

44. Hu, G., Gans, N. & Dixon, W. Quaternion-based visual servo control in the presence of camera calibration error. *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal* **20,** 489–503 2010.

45. Marey, M. & Chaumette, F. *A new large projection operator for the redundancy framework* in *2010 IEEE international conference on robotics and automation* 2010, 3727–3732.

46. Mansard, N. & Chaumette, F. Directional redundancy for robot control. *IEEE Transactions on Automatic Control* **54,** 1179–1192 2009.

47. Yoshikawa, T. Basic optimization methods of redundant manipulators. *Laboratory Robotics and Automation* **8,** 49–60 1996.

48. Chaumette, F. & Marchand, E. A redundancy-based iterative approach for avoiding joint limits: Application to visual servoing. *IEEE Transactions on Robotics and Automation* **17,** 719–730 2001.

49. Nelson, B. J. & Khosla, P. K. Strategies for increasing the tracking region of an eye-in-hand system by singularity and joint limit avoidance. *The International journal of robotics research* **14,** 255–269 1995.

50. Liegeois, A. *et al.* Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE transactions on systems, man, and cybernetics* **7,** 868–871 1977.

51. Singh, A., Kalaichelvi, V. & Karthikeyan, R. A survey on vision guided robotic systems with intelligent control strategies for autonomous tasks. *Cogent Engineering* **9,** 2050020 2022.

52. Ramachandram, D. & Rajeswari, M. *A short review of neural network techniques in visual servoing of robotic manipulators* in *Malaysia-Japan Seminar On Artificial Intelligence Applications In Industry* 2003, 2425.

53. Massoud Farahmand, A., Shademan, A., Jagersand, M. & Szepesvari, C. *Model-based and model-free reinforcement learning for visual servoing* in *2009 IEEE International Conference on Robotics and Automation* 2009, 2917–2924.

54. Zhou, Z., Zhang, R. & Zhu, Z. *RETRACTED: uncalibrated dynamic visual servoing via multivariate adaptive regression splines and improved incremental extreme learning machine* 2019.

55. Raj, P., Namboodiri, V. P. & Behera, L. Learning to Switch CNNs with Model Agnostic Meta Learning for Fine Precision Visual Servoing. *arXiv preprint arXiv:2007.04645* 2020.

56. Jokić, A., Petrović, M., Kulesza, Z. & Miljković, Z. *Visual Deep Learning-Based Mobile Robot Control: A Novel Weighted Fitness Function-Based Image Registration Model* in *International Conference "New Technologies, Development and Applications"* 2021, 744–752.

57. Hua, J., Zeng, L., Li, G. & Ju, Z. Learning for a Robot: Deep Reinforcement Learning, Imitation Learning, Transfer Learning. *Sensors* **21,** 1278 2021.

58. Castelli, F., Michieletto, S., Ghidoni, S. & Pagello, E. A machine learning-based visual servoing approach for fast robot control in industrial setting. *International Journal of Advanced Robotic Systems* **14,** 1729881417738884 2017.

59. Jin, Z., Wu, J., Liu, A., Zhang, W.-A. & Yu, L. Policy-based deep reinforcement learning for visual servoing control of mobile robots with visibility constraints. *IEEE Transactions on Industrial Electronics* **69,** 1898–1908 2021.

60. Ramirez, J., Yu, W. & Perrusquia, A. Model-free reinforcement learning from expert demonstrations: a survey. *Artificial Intelligence Review* **55,** 3213–3241 2022.

61. Ross, S., Gordon, G. & Bagnell, D. *A reduction of imitation learning and structured prediction to no-regret online learning* in *Proceedings of the fourteenth international conference on artificial intelligence and statistics* 2011, 627–635.

62. Takeda, T., Hirata, Y. & Kosuge, K. Dance step estimation method based on HMM for dance partner robot. *IEEE Transactions on Industrial Electronics* **54,** 699–706 2007.

63. Krishnan, S., Garg, A., Liaw, R., Thananjeyan, B., Miller, L., Pokorny, F. T. & Goldberg, K. SWIRL: A sequential windowed inverse reinforcement learning algorithm for robot tasks with delayed rewards. *The International Journal of Robotics Research* **38,** 126–145 2019.

64. Ho, J. & Ermon, S. Generative adversarial imitation learning. *Advances in neural information processing systems* **29** 2016.

65. Ding, Z. & Dong, H. Challenges of reinforcement learning. *Deep Reinforcement Learning: Fundamentals, Research and Applications,* 249–272 2020.

66. Cruz Jr, G. V., Du, Y. & Taylor, M. E. Pre-training neural networks with human demonstrations for deep reinforcement learning. *arXiv preprint arXiv:1709.04083* 2017.

67. Ball, P. J., Smith, L., Kostrikov, I. & Levine, S. Efficient online reinforcement learning with offline data. *arXiv preprint arXiv:2302.02948* 2023.

68. Liu, Y., Datta, G., Novoseller, E. & Brown, D. S. Efficient preference-based reinforcement learning using learned dynamics models. *arXiv preprint arXiv:2301.04741* 2023.

69. Wu, J., Huang, Z., Hu, Z. & Lv, C. Toward human-in-the-loop AI: Enhancing deep reinforcement learning via real-time human guidance for autonomous driving. *Engineering* **21,** 75–91 2023.

70. Kumar, V., Gupta, A., Todorov, E. & Levine, S. Learning dexterous manipulation policies from experience and imitation. *arXiv preprint arXiv:1611.05095* 2016.

71. Ross, S., Gordon, G. J. & Bagnell, J. A. *No-regret reductions for imitation learning and structured prediction* in *In AISTATS* 2011.

72. Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Osband, I., *et al. Deep q-learning from demonstrations* in *Proceedings of the AAAI Conference on Artificial Intelligence* **32** 2018.

73. Kelly, M., Sidrane, C., Driggs-Campbell, K. & Kochenderfer, M. J. *HG-DAgger: Interactive imitation learning with human experts* in *2019 International Conference on Robotics and Automation (ICRA)* 2019, 8077–8083.

74. Hoque, R., Balakrishna, A., Novoseller, E., Wilcox, A., Brown, D. S. & Goldberg, K. ThriftyDAgger: Budget-aware novelty and risk gating for interactive imitation learning. *arXiv preprint arXiv:2109.08273* 2021.

75. Sun, W., Bagnell, J. A. & Boots, B. Truncated horizon policy search: Combining reinforcement learning & imitation learning. *arXiv preprint arXiv:1805.11240* 2018.

76. Kang, B., Jie, Z. & Feng, J. *Policy optimization with demonstrations* in *International conference on machine learning* 2018, 2469–2478.

77. Yang, C., Ma, X., Huang, W., Sun, F., Liu, H., Huang, J. & Gan, C. Imitation learning from observations by minimizing inverse dynamics disagreement. *Advances in neural information processing systems* **32** 2019.

78. Mayne, D. Q., Rawlings, J. B., Rao, C. V. & Scokaert, P. O. Constrained model predictive control: Stability and optimality. *Automatica* **36,** 789–814 2000.

79. Forbes, M. G., Patwardhan, R. S., Hamadah, H. & Gopaluni, R. B. Model predictive control in industry: Challenges and opportunities. *IFAC-PapersOnLine* **48,** 531–538 2015.

80. Zanon, M. & Gros, S. Safe reinforcement learning using robust MPC. *IEEE Transactions on Automatic Control* 2020.

81. Kober, J., Bagnell, J. A. & Peters, J. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research* **32,** 1238–1274 2013.

82. Omer, M., Ahmed, R., Rosman, B. & Babikir, S. F. *Model predictive-actor critic reinforcement learning for dexterous manipulation* in *2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)* 2021, 1–6.

83. Bellegarda, G. & Byl, K. *An online training method for augmenting MPC with deep reinforcement learning* in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* 2020, 5453–5459.

84. Hewing, L., Wabersich, K. P., Menner, M. & Zeilinger, M. N. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems* **3,** 269–296 2020.

85. Zhang, K., Wang, J., Xin, X., Li, X., Sun, C., Huang, J. & Kong, W. A survey on learning-based model predictive control: Toward path tracking control of mobile platforms. *Applied Sciences* **12,** 1995 2022.

86. Shin, C., Ferguson, P. W., Pedram, S. A., Ma, J., Dutson, E. P. & Rosen, J. *Autonomous tissue manipulation via surgical robot using learning based model predictive control* in *2019 International conference on robotics and automation (ICRA)* 2019, 3875–3881.

87. Rastegarpanah, A., J. Hathaway & R. Stolkin, "Vision-Guided MPC for Robotic Path Following Using Learned Memory-Augmented Model". *Frontiers in Robotics and AI* 8 2021.

88. Padalkar, A., Nieuwenhuisen, M., Schneider, S. & Schulz, D. *Learning to Close the Gap: Combining Task Frame Formalism and Reinforcement Learning for Compliant Vegetable Cutting.* in *ICINCO* 2020, 221–231.

89. Martin-Martin, R., Lee, M. A., Gardner, R., Savarese, S., Bohg, J. & Garg, A. *Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks* in *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)* 2019, 1010–1017.

90. Zhang, X., L. Sun, Z. Kuang & M. Tomizuka, "Learning Variable Impedance Control via Inverse Reinforcement Learning for Force-Related Tasks". *IEEE Robotics and Automation Letters* 6, 2225–2232 2021.

91. Faulwasser, T., T. Weber, P. Zometa & R. Findeisen, "Implementation of Nonlinear Model Predictive Path-Following Control for an Industrial Robot". *IEEE Transactions on Control Systems Technology* 25, 1505–1511 2017.

92. Matschek, J., J. Bethge, P. Zometa & R. Findeisen, "Force Feedback and Path Following using Predictive Control: Concept and Application to a Lightweight Robot". *IFAC-PapersOnLine* 50, 9827–9832 2017.

93. Meng, L., S. Yu, H. Chang, R. Findeisen & H. Chen, *Path following and terminal force control of robotic manipulators*. IEEE International Conference on Control and Automation, ICCA. 2020-October. 2020, 1482–1487.

94. Maldonado-Ramirez, A., R. Rios-Cabrera & I. Lopez-Juarez, "A visual path-following learning approach for industrial robots using DRL". *Robotics and Computer-Integrated Manufacturing* 71 2021.

95. Peng, X. B., M. Andrychowicz, W. Zaremba & P. Abbeel, "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization". *CoRR* abs/1710.06537. arXiv: 1710.06537 2017.

96. OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba & L. Zhang, "Solving Rubik's Cube with a Robot Hand". *CoRR* abs/1910.07113. arXiv: 1910.07113 2019.

97. Nelles, O., Fink, A. & Isermann, R. Local linear model trees (LOLIMOT) toolbox for nonlinear system identification. *IFAC Proceedings Volumes* **33,** 845–850 2000.

98. Kalhor, A., Araabi, B. N. & Lucas, C. Evolving Takagi–Sugeno fuzzy model based on switching to neighboring models. *Applied Soft Computing* **13,** 939–946 2013.

99. Nelles, O. *Local linear model trees for on-line identification of time-variant nonlinear dynamic systems* in *International Conference on Artificial Neural Networks* 1996, 115–120.

100. Hu, G., Gans, N. R. & Dixon, W. E. *Adaptive Visual Servo Control.* 2009.

101. Hutchinson, S. & Chaumette, F. Visual servo control, part i: Basic approaches. *IEEE Robotics and Automation Magazine* **13,** 82–90 2006.

102. Kermorgant, O. & Chaumette, F. Dealing with constraints in sensor-based robot control. *IEEE Transactions on Robotics* **30,** 244–257 2013.

103. Spong, M. *Hutchinso, n. Seth, and MV Vidyasagar,"Robot Modeling and Control," John Wiley& Sons* 2006.

104. Baerlocher, P. & Boulic, R. *Task-priority formulations for the kinematic control of highly redundant articulated structures* in *Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No. 98CH36190)* **1** 1998, 323–329.

105. Maciejewski, A. A. & Klein, C. A. Numerical filtering for the operation of robotic manipulators through kinematically singular configurations. *Journal of Robotic Systems* **5,** 527–552 1988.

106. Fruchard, M., Morin, P. & Samson, C. A framework for the control of nonholonomic mobile manipulators. *The International Journal of Robotics Research* **25,** 745–780 2006.

107. Vahrenkamp, N., Asfour, T., Metta, G., Sandini, G. & Dillmann, R. *Manipulability analysis* in *2012 12th ieee-ras international conference on humanoid robots (humanoids 2012)* 2012, 568–573.

108. Rezaie, J., Moshiri, B., Rafati, A. & Araabi, B. N. *A Modified LOLIMOT Algorithm for Nonlinear Estimation Fusion* in *2007 IEEE International Conference on Information Reuse and Integration* 2007, 520–525.

109. Kalhor, A., Araabi, B. N. & Lucas, C. Reducing the number of local linear models in neuro-fuzzy modeling: A split-and-merge clustering approach. *Applied Soft Computing* **11,** 5582–5589 2011.

110. Aflakian, A., Safaryazdi, A., Masouleh, M. T. & Kalhor, A. Experimental study on the kinematic control of a cable suspended parallel robot for object tracking purpose. *Mechatronics* **50,** 160–176 2018.

111. Corke, P. *Robotics, vision and control: fundamental algorithms in MATLAB® second, completely revised* Springer, 2017.

112. Wegener, K., Andrew, S., Raatz, A., Dröder, K. & Herrmann, C. Disassembly of electric vehicle batteries using the example of the Audi Q5 hybrid system. *Procedia CIRP* **23,** 155–160 2014.

113. Alfaro-Algaba, M. & Ramirez, F. J. Techno-economic and environmental disassembly planning of lithium-ion electric vehicle battery packs for remanufacturing. *Resources, Conservation and Recycling* **154,** 104461 2020.

114. Pistoia, G. & Liaw, B. *Behaviour of lithium-ion batteries in electric vehicles: battery health, performance, safety, and cost* Springer, 2018.

115. Wells, L. & Bednarz, T. Explainable AI and Reinforcement Learning—A Systematic Review of Current Approaches and Trends. *Frontiers in artificial intelligence* **4,** 48 2021.

116. Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P. & Zaremba, W. Hindsight experience replay. *arXiv preprint arXiv:1707.01495* 2017.

117. Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. & Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* 2015.

118. Franks, J., Huo, L. & Baron, L. The joint-limits and singularity avoidance in robotic welding. *Industrial Robot: An International Journal* 2008.

119. Bellman, R. Dynamic programming, princeton univ. *Press Princeton, New Jersey* 1957.

120. Fujimoto, S., Hoof, H. & Meger, D. *Addressing function approximation error in actor-critic methods* in *International conference on machine learning* 2018, 1587–1596.

121. Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W. & Abbeel, P. *Domain randomization for transferring deep neural networks from simulation to the real world* in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)* 2017, 23–30.

122. MATLAB. *version 9.9.0 (R2020b)* The MathWorks Inc., Natick, Massachusetts, 2021.

123. Scherzinger, S., A. Roennau & R. Dillmann, *Inverse Kinematics with Forward Dynamics Solvers for Sampled Motion Tracking*. 19th International Conference on Advanced Robotics (ICAR). Dec. 2019, 681–687.

124. Fujimoto, S., van Hoof, H. & Meger, D. *Addressing Function Approximation Error in Actor-Critic Methods* 2018. arXiv: `1802.09477 [cs.AI]`.

125. Barber, C. B., Dobkin, D. P. & Huhdanpaa, H. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)* **22,** 469–483 1996.

126. Inc., T. M., *MATLAB version: 9.13.0 (R2020b)*. Natick, Massachusetts, United States, 2020.

127. Trinh, S., Spindler, F., Marchand, E. & Chaumette, F. *A modular framework for model-based visual tracking using edge, texture and depth features* in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'18* Madrid, Spain, Aug. 2018.

128. Scott, S., Z. Islam, J. Allen, T. Yingnakorn, A. Alflakian, J. Hathaway, A. Rastegarpanah, G. D. Harper, E. Kendrick, P. A. Anderson, J. Edge, L. Lander & A. P. Abbott, "Designing lithium-ion batteries for recycle: The role of adhesives". *Next Energy* 1, 100023 2023.

129. Nakanishi, J., R. Cory, M. Mistry, J. Peters & S. Schaal, "Operational Space Control: A Theoretical and Empirical Comparison". *The International Journal of Robotics Research* 27, 737–757. eprint: `https://doi.org/10.1177/0278364908091463` 2008.

130. Marchand, E., Spindler, F. & Chaumette, F. ViSP for visual servoing: a generic software platform with a wide class of robot control skills. *IEEE Robotics & Automation Magazine* **12,** 40–52 2005.

CHAPTER 7

Index 1

7.1 | Camera calibration

Camera calibration is a fundamental process in computer vision, involving the measurement and adjustment of camera parameters to ensure that it accurately captures and represents the real-world environment. The main goal of camera calibration is to establish a mathematical relationship between the 3D world coordinates and the 2D image coordinates produced by the camera. This relationship is vital for a wide range of applications, including object measurement, 3D reconstruction, augmented reality, and robotics.
Camera calibration is necessary for several reasons:

- Accuracy: Cameras, even high-quality ones, have inherent imperfections, such as lens distortion and variations in manufacturing. Calibration helps correct these imperfections, ensuring the accuracy of measurements and reconstructions.
- Consistency: Calibration ensures that different cameras produce consistent and comparable results, making it possible to integrate data from multiple cameras or sensors into a single coordinate system.
- Spatial Mapping: For computer vision tasks, it's essential to map 2D image points to corresponding 3D world points. Calibration provides the transformation necessary for this mapping.
- Virtual and Real World Alignment: In augmented reality, virtual objects must align correctly with the real world. Calibration enables this alignment by establishing the camera's position and orientation in space.
- Robotics: In robotics, precise camera calibration is crucial for robot navigation, object manipulation, object detection and localization, and obstacle avoidance.

Camera calibration is divided into two main groups, intrinsic camera calibration and extrinsic camera calibration. In the following subsections, these two categories are explained.

7.1.1 Intrinsic Camera Calibration

Intrinsic camera calibration deals with the internal characteristics of the camera, which primarily affect the image formation process. These characteristics include parameters like the focal length, principal point, and lens distortion. Intrinsic calibration is necessary to correct distortions caused by the camera's lens and ensure that measurements in the image are accurate and undistorted.

The goal of calibration is to estimate specific camera parameters that find the relation between the pixel positions in the image and normalized positions in meters on the image plane (real-world units). The mathematical relationship between 3D world coordinates and 2D image coordinates are as follows:

$$u = f \cdot \frac{X}{Z} + C_x \tag{7.1}$$

$$v = f \cdot \frac{Y}{Z} + C_y \tag{7.2}$$

Here (u, v) are Pixel coordinates in the image, (X, Y, Z) are 3D world coordinates, f is Focal length, and (C_x, C_y) are Principal point coordinates.
Intrinsic camera calibration also addresses lens distortion, which can affect the accuracy of measurements in images. A common model used for this purpose is the radial distortion model. Two distortion coefficients, k_{ud} and k_{du} are Distortion parameters that correct for lens distortion effects. From meters to pixels, the following formulas are considered:

$$u = u_0 + xp_x(1 + k_{ud}r^2) \tag{7.3}$$

$$v = v_0 + yp_y(1 + k_{ud}r^2) \tag{7.4}$$

where $r^2 = x^2 + y^2$. From pixels to meters, the following formulas are applied:

$$x = (u - u_0)\left(1 + k_{du}r^2\right)/p_x \tag{7.5}$$

$$y = (v - v_0)\left(1 + k_{du}r^2\right)/p_y \tag{7.6}$$

with $r^2 = ((u - u_0)/p_x)^2 + ((v - v_0)/p_y)^2$.
Here, (u_0, v_0) are Principal point coordinates in pixels (p_x, p_y) are the Ratio between focal length and pixel size. The calibration process involves several steps:

1. **Image Acquisition:** Capture images of a calibration pattern (e.g., checkerboard) from various angles and orientations. Ensure the pattern is visible in each image.

2. **Corner Detection:** Detect the corners of the calibration pattern in the images using computer vision libraries.

3. **World Points and Image Points:** Create a list of 3D world points corresponding to the corners of the calibration pattern and a list of corresponding 2D image points.

4. **Camera Calibration:** Use a calibration algorithm to estimate the intrinsic parameters, including $(f, C_x, C_y, p_x, p_y, k_{ud}, k_{du})$.

5. **Refinement:** Refine the calibration results by minimizing the reprojection error, ensuring that the 3D world points projected onto the images match the detected corner positions.

6. **Extraction of Parameters:** The calibrated camera parameters are then extracted from the calibration results and can be used in various computer vision applications.

In summary, intrinsic camera calibration is a crucial process that estimates the internal camera parameters. These parameters enable accurate mapping between the 3D world and 2D image coordinates. The calibration procedure involves capturing images of a calibration pattern, detecting corners, and using calibration algorithms to estimate camera intrinsic parameters, as well as correcting lens distortion. The resulting parameters are essential for ensuring accurate measurements in computer vision tasks. In our work, we utilized the standard predefined intrinsic parameters provided for the intrinsic calibration of each camera. It is essential to clarify that our focus was on applying these pre-established parameters rather than conducting the intrinsic calibration process from scratch. These parameters could be found in the camera firmware or SDK and in VISP library we get them from $vpRealSense2 :: getCameraParameters()$ function.

7.1.2 Extrinsic camera calibration

Extrinsic camera calibration deals with determining the pose and position of the camera in the 3D world. It includes camera orientation and its translation (in general homogeneous transformation) in the world coordinate system. These parameters are essential for mapping the 2D image to the 3D world. This calibration is crucial for various computer vision applications, including robotics, object tracking, and augmented reality, where knowing the camera's viewpoint in the world is essential.
Extrinsic Calibration Procedure: Les us define ${}^{f}\mathbf{M}_{e}$, as the homogeneous transformation between the robot base frame (often referred to as the fixed frame) and the robot end-effector. ${}^{c}\mathbf{M}_{o}$, as the homogeneous transformation between the camera frame and a calibration grid frame (also known as the object frame). Typically, this calibration grid is the OpenCV chessboard. ${}^{e}\mathbf{M}_{c}$, as the homogeneous transformation between the end-effector and the camera frame. This transformation corresponds to the extrinsic eye-in-hand transformation that needs to be estimated, then:
Extrinsic camera calibration when the camera is attached to the robot's EE, can be performed using the following steps:

- Image Acquisition: Capture images of a known calibration pattern from various angles and orientations and their correspondence homogeneous transformation between the robot base frame and EE (${}^{f}\mathbf{M}_{e}$). The pattern should be visible in each image. Also, acquire the camera's intrinsic parameters. In Figure 7.1, the calibration process involves placing a chessboard pattern underneath some of the robots is illustrated. This setup allows for the precise estimation of the camera's position with respect to each robot's EE position. The camera on the robot captures images of the chessboard from different angles, providing essential information to determine the camera's pose. The resulting extrinsic parameters are crucial for various applications, such as accurate robot navigation, visual servoing, and object localization.

- World Points and Image Points: Establish correspondences between 3D world points and their 2D image coordinates. The calibration pattern provides known 3D points in the world coordinate system. In other words, computing the corresponding homogeneous transformation between the camera frame and a calibration grid frame (${}^{c}\mathbf{M}_{o}$). Figure 7.2 displays a subset of images employed during the extrinsic calibration procedure. Each image represents the robot in a distinct position, ensuring coverage of a significant portion of the half-sphere above the chessboard. Capturing images from various angles and orientations is essential for robust calibration, as it enables accurate determination of the camera's rotation and translation with respect to the robot's EE position.

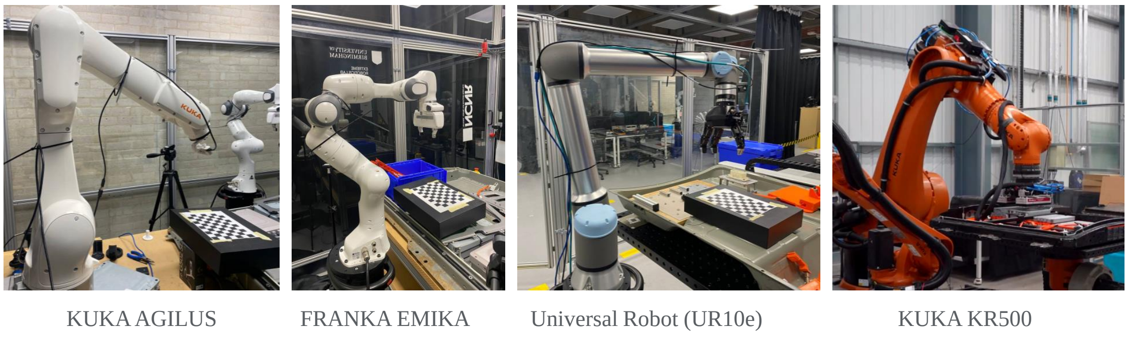


Fig. 7.1. The extrinsic calibration setups, where a chessboard pattern is positioned beneath the robots for the calibration of the camera's extrinsic parameters.

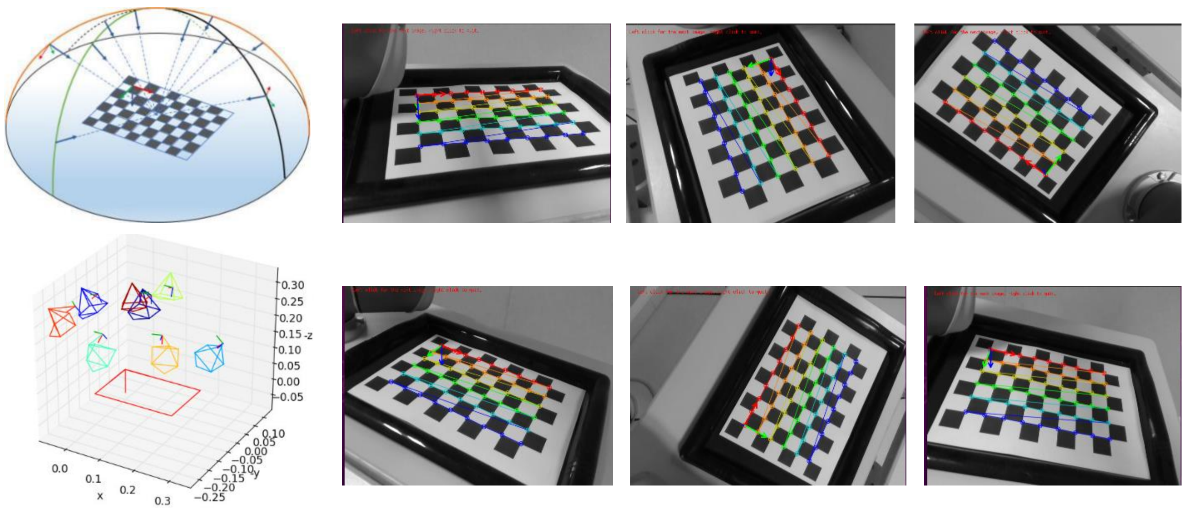


Fig. 7.2. A selection of images used for the calibration process. These images depict the robot in different positions, covering a substantial portion of the half-sphere above the chessboard.

- Camera Calibration: Use an extrinsic calibration algorithm and the collected data in previous steps to estimate the homogeneous transformation between the EE and the camera frame (${}^{e}\mathbf{M}_{c}$). We have used the VISP camera calibration tutorial [130] for calibrating our cameras.

CHAPTER 8

Index 2

8.1 | Designed SimMechanics model

One setup involves creating a SimMechanics model for the LBR robot (Figure 8.1). In this simulation, the robot model is constructed from scratch using SimMechanics, with revolute joints, transformation matrices, and CAD parts defining the robot's structure. Friction and damping coefficients for each joint are specified based on the robot's documentation.

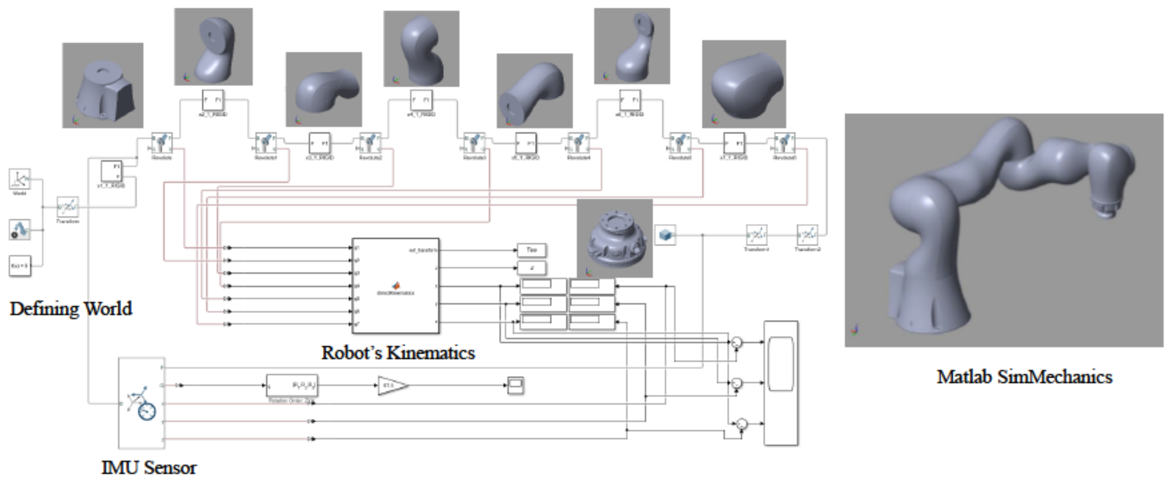


Fig. 8.1. Design SimMechanics environment to simulate and visualise Kuka LBR behavior.

8.2 | RL in MATLAB

Reinforcement Learning in MATLAB/Simulink provides a powerful framework for training intelligent agents to perform tasks in various simulated environments. In this overview, I will discuss the application of RL in two distinct scenarios: VS and Contact-Rich Manipulation.

8.2.1 Visual servoning application

Figure 8.2 illustrates the Simulink environment tailored for VS. Leveraging the MATLAB Reinforcement Learning Toolbox, we establish an RL framework where observations, termination rules, and rewards are defined. The action vector for RL is configured to be joint velocities, allowing the agent to control the robotic arm's movements. Three different approaches to defining a camera are explored. The first one is defining a dummy camera within MATLAB's Robotics Vision and Control Toolbox, providing a straightforward way to simulate visual observations. The second approach is Utilizing camera information from the Gazebo simulation

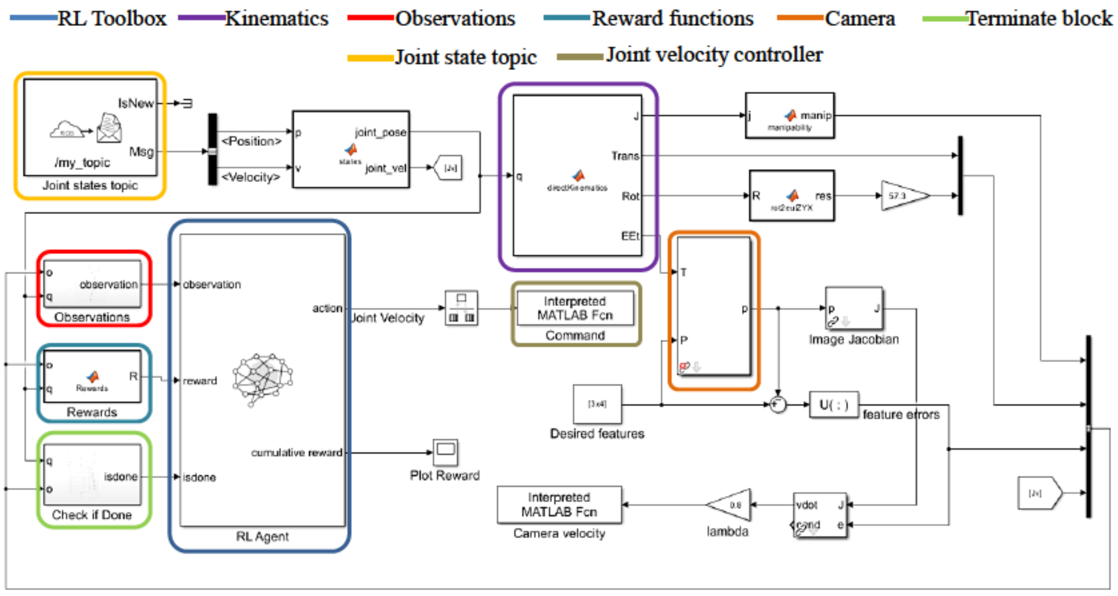


Fig. 8.2. Simulink environment for Visual Servoing using RL

environment, offering a more realistic representation. The last option is incorporating a real camera to get the visual input during training. Figure 8.3 illustrates "is Done" and "Observations" blocks in Figure 8.2, used in training the agent for the task of VS.

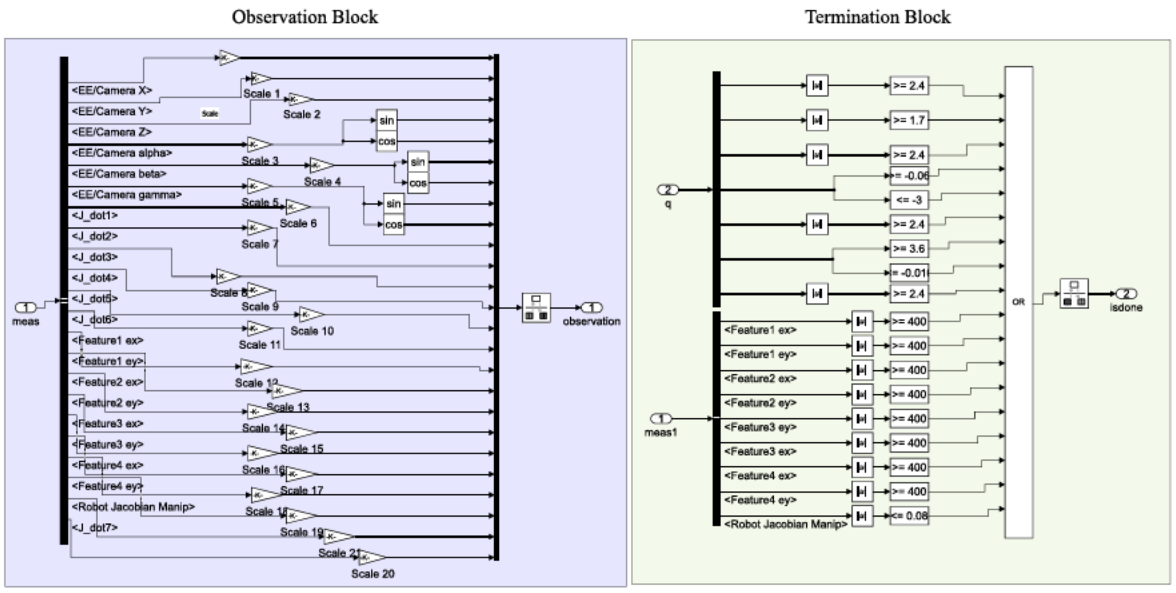


Fig. 8.3. Is Done and Observations used in training the agent for the task of VS.

In Section 3.2.1, we provide a complete explanation of the proposed action-constrained strategies, along with detailed algorithms and the theoretical formulation of RL applied to VS. The strategies, such as AORLD-HL, AORLD-CL, AORLD-HF, and AORLD-HP, are thoroughly explained, outlining their respective algorithms and the underlying theory.

8.2.2 Cutting application

Figure 8.4 showcases a Simulink environment designed for the contact-rich manipulation task. The action space is defined as Cartesian velocities, and as with the Visual Servoing task, the simulation has been done in Gazebo. Communication between MATLAB and Ros packages in Gazebo has happened through ROS topics and services, enabling direct integration of the RL agent with the simulated environment. Figure 8.5 illustrates is Done and Observations blocks in Figure 8.4, used in training the agent for the task of contact-rich manipulation. Figure 8.6 shows the design of reward functions in the Simulink used in training the agent for the task of contact-rich manipulation.

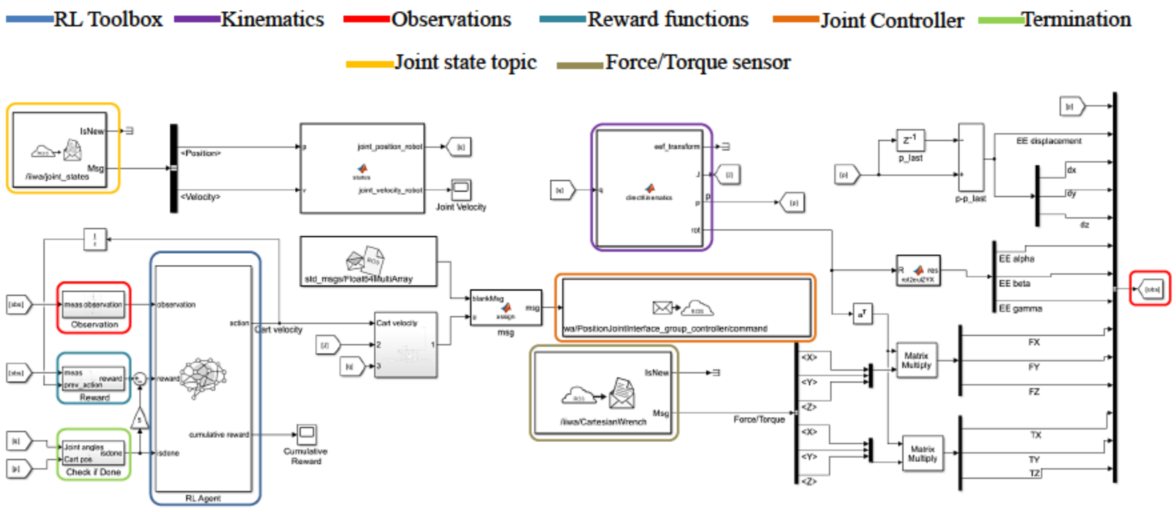


Fig. 8.4. Simulink environment for contact-rich manipulation task using RL

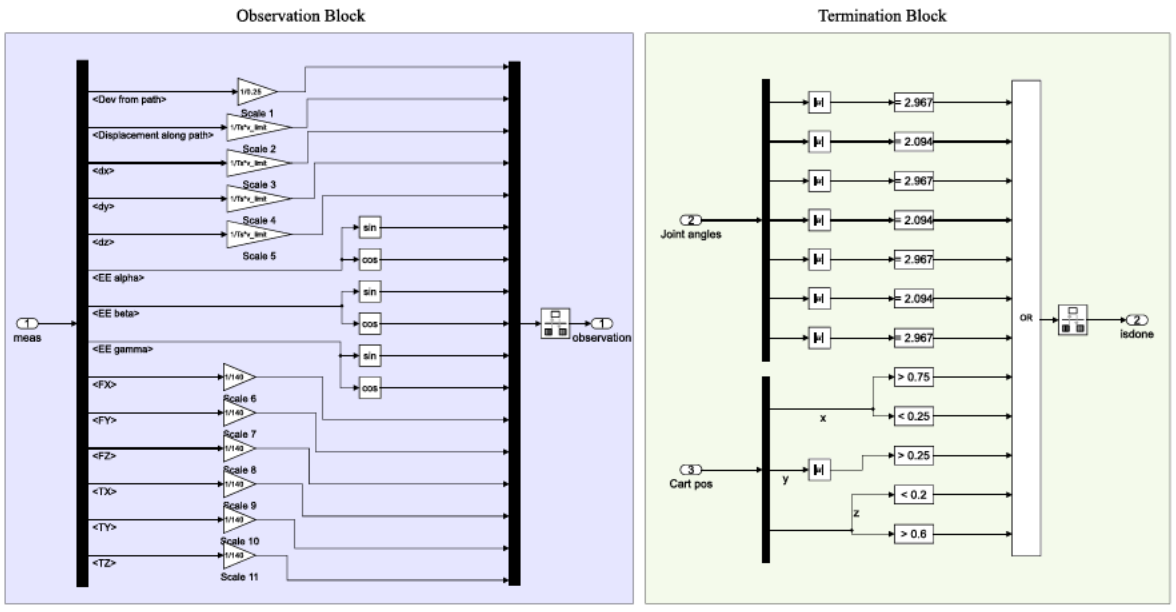


Fig. 8.5. Is Done and Observations used in training the agent for the task of contact-rich manipulation.

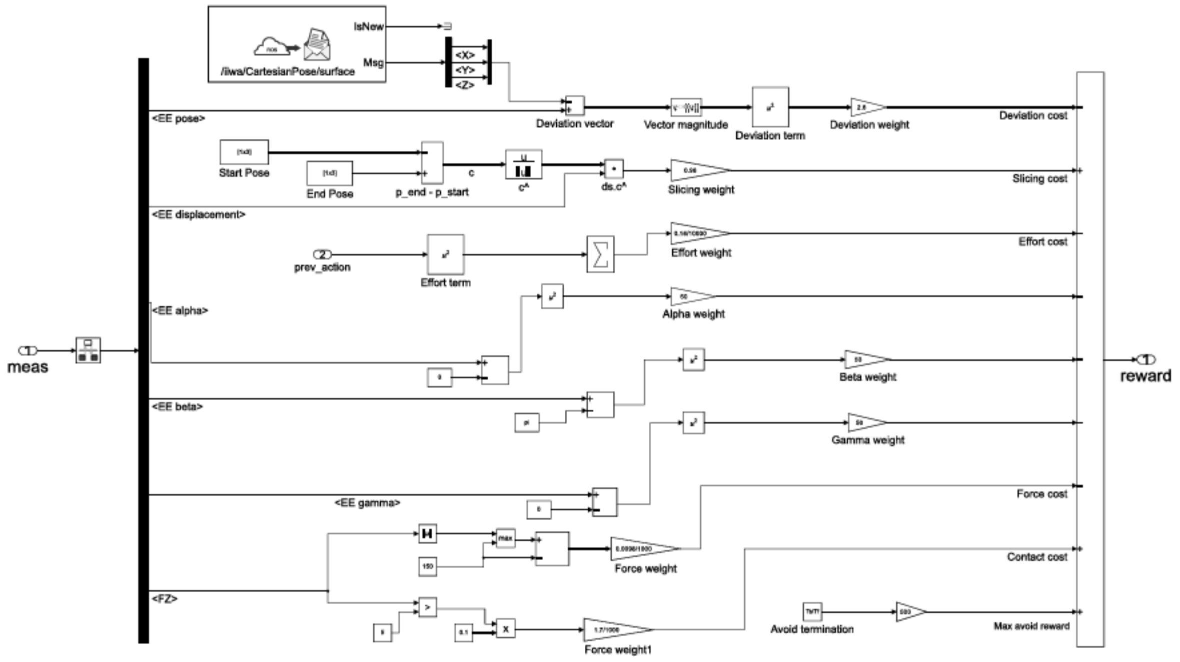


Fig. 8.6. Defined rewards in training the agent for the task of contact-rich manipulation.

8.3 | Teleoperation Simulink Model

To collect human demonstrations for our investigation in Section 4.5, a Simulink model has been designed. The model utilizes ROS-MATLAB integration to enable teleoperation, employing a Phantom Haptic device to control a simulated KUKA robot. Additionally, the model incorporates functionality to map contact forces experienced by the robot to the operator's hand during telemanipulation. This setup facilitates the gathering of human demonstrations, contributing to our research in the specified section. Figure 8.7 depicts the Simulink model designed for telemanipulating the simulated robot. It should be mentioned that using the features provided by ROS, we have the flexibility to transition from controlling the simulated robot to operating a real robot by adjusting the assigned topics. This capability enhances the flexibility of the telemanipulation setup, allowing for easy integration with both simulated and real-world robotic systems.

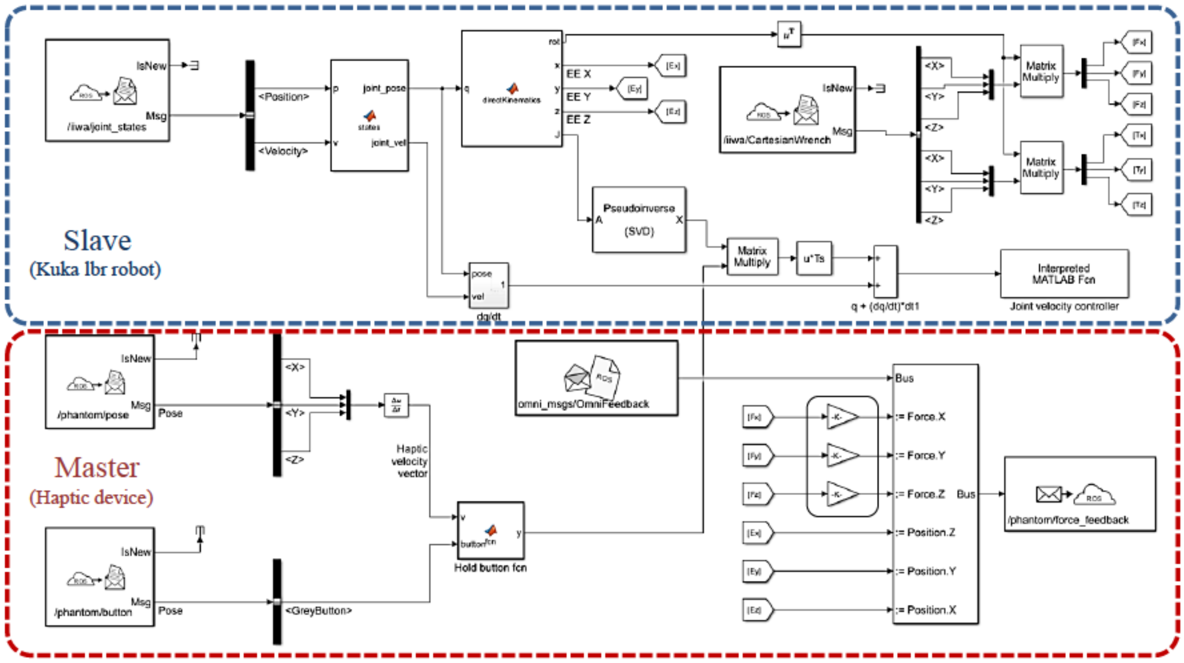


Fig. 8.7. Simulink environment for Teleoperating the simulated KUKA robot in Simulink using ROS