IMPERIAL

ROBOT PERCEPTION AND MANIPULATION OF DEFORMABLE LINEAR OBJECTS

HAINING LUO

Thesis submitted for the degree of Doctor of Philosophy

Supervised by PROFESSOR YIANNIS DEMIRIS

Personal Robotics Laboratory
Department of Electrical and Electronic Engineering
Imperial College London

December 2024

COPYRIGHT DECLARATION

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution-Non Commercial-No Derivatives 4.0 International Licence (CC BY-NC-ND).

Under this licence, you may copy and redistribute the material in any medium or format on the condition that; you credit the author, do not use it for commercial purposes and do not distribute modified versions of the work.

When reusing or sharing this work, ensure you make the licence terms clear to others by naming the licence and linking to the licence text.

Please seek permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.

ORIGINALITY DECLARATION

I hereby declare that this thesis and the work herein detailed, was composed and originated by myself, except where appropriately referenced and credited.

London, December 2024

Haining Luo

ABSTRACT

Robot manipulation of Deformable Linear Objects (DLOs) spans applications in industry, healthcare, and daily life. Long-term DLO manipulation is challenging due to complex object dynamics and logical reasoning requirements. A task common in both everyday life and manufacturing is Shoe Lacing (SL). This thesis focuses on the perception and manipulation of DLOs under the task of SL.

First, this thesis approaches SL is by intuitively decomposing it into primitives and planning with symbolic planners. The task is structured across three levels: high-level optimisation of SL patterns, mid-level task planning, and low-level execution. Experiments are conducted using different shoes and conditions to evaluate system performance.

This thesis proposes a simulation environment and benchmarking framework for bimanual SL systems. The environment supports controlled training and testing, while the framework offers evaluation metrics and task protocols to ensure performance comparison and repeatability. Baseline results demonstrate the effectiveness of the framework and the environment.

Existing tracking algorithms overlook interactions with scene objects. This thesis introduces a method that tracks shoelaces while preserving topological relations with the shoe. Two evaluation metrics are proposed: sequential *H-Signature*, representing topological state, and Dynamic Time Warping (DTW) distance, quantifying tracking error. The algorithm consistently yields correct signatures and achieves errors comparable to state-of-the-art methods.

Learning from Demonstration (LfD) offers an efficient way to learn robot skills. While existing work often focuses on learning policies, few consider the operator's role. This thesis presents a bimanual teleoperation system and analyses how first- and third-person perspectives affect completion time, trajectory diversity, and policy quality.

Overall, this thesis advances Deformable Object (DO) manipulation by addressing the full pipeline of robot SL, including task planning, perception, simulation, and primitive learning. These developments offer foundational tools and insights for robust, generalisable DLO manipulation in real-world settings.

ACKNOWLEDGEMENTS

I have been incredibly fortunate to receive the unwavering support of my colleagues and family over the past four transformative years. The journey towards completing my PhD has been long and often challenging, but also filled with moments of joy and excitement. Among those who have supported me, I would like to express my deepest gratitude to my supervisor, Professor Yiannis Demiris, for his invaluable guidance in my research, career development, and personal growth. Perhaps the most pivotal moment in my life was the day I selected my Master's project with Prof. The time working with him changed the trajectory of my life. It was his encouragement and trust that kept me going through the years, teaching me how to think and work as a researcher and how to face challenges and setbacks with resilience.

I am deeply grateful to my friends and colleagues at the Personal Robotics Lab, who shared in both my happiest and most challenging moments. I thank Rodrigo for his all-around support and mentoring since my first day in the lab. I am thankful to Dimitra for being an incredible listener and for her emotional support. I am grateful to Fernando for his research guidance and his rather unique sense of humour. I thank Cedric for always providing technical assistance when I needed it. I feel fortunate to have started my PhD alongside Haziq and Nico, sharing the worries and difficulties of the journey together. I also thank all the other lab members for their engaging lunch break discussions, which brought both inspiration and laughter. Out of the 4-year journey, I had the privilege of overlapping with some of the previous lab members. I am thankful to Xingchen for his insightful advice on research and career development, and to Fan, Young, and Mohammed for their guidance during the early stage of my PhD.

I would like to give special thanks to my best friends from middle school, Runsheng and Yuanzheng. Their friendship over all these years has played a significant role in shaping who I am today.

Finally, I would like to express my deepest gratitude to my parents for their unconditional support and love.

CONTENTS

1 Introduction 19
- 1.1 Research Goals . . . 20
  - 1.1.1 Task Planning with Primitives . . . 20
  - 1.1.2 Simulation and Benchmarking . . . 21
  - 1.1.3 Topology-preserving Tracking . . . 22
  - 1.1.4 Learning from Human Demonstrations . . . 23
  - 1.1.5 Summary of Research Questions . . . 23
- 1.2 Thesis Roadmap . . . 24
- 1.3 Contributions . . . 25
- 1.4 Publications . . . 27

2 Background 29
- 2.1 Robot Shoe Lacing . . . 30
  - 2.1.1 Industrial Endeavours . . . 30
  - 2.1.2 Shoe Lacing with Different Patterns . . . 31
- 2.2 Planning for Deformable Linear Object Manipulation . . . 31
  - 2.2.1 Planning with Topology . . . 32
  - 2.2.2 Planning with General Purpose Planners . . . 32
- 2.3 Simulating DLOs . . . 33
  - 2.3.1 Physics Engines Supporting DLO Simulation . . . 34
  - 2.3.2 Environments for DLO Simulation . . . 36
- 2.4 Benchmarking DLO Manipulation . . . 36
- 2.5 Perception of DLOs . . . 37
  - 2.5.1 Segmenting DLOs . . . 37
  - 2.5.2 State Estimation of DLOs . . . 38
  - 2.5.3 Topology in DLO Tracking . . . 39
- 2.6 Learning to Manipulate DLOs from Demonstrations . . . 40
  - 2.6.1 Imitation Learning in DLO Manipulation . . . 40
  - 2.6.2 Teleoperation Systems and Behaviour Cloning . . . 41
  - 2.6.3 Comparing Teleoperation Interfaces . . . 43
- 2.7 Summary . . . 43

3 DLO Manipulation with Task Planning and Hierarchical Primitives 45
- 3.1 Definition of the Shoe Lacing Task . . . 46
  - 3.1.1 Definition of Terms and Concepts . . . 46

3.1.2 Task Description . . . 47
3.2 System Overview . . . 48
3.3 Design of the Shoe Lacing Patterns . . . 48
3.3.1 Definition of the Pattern Designing Problem . . . 49
3.3.2 Designing Shoe Lacing Patterns with Genetic Algorithms 51
3.4 Planning Shoe Lacing Actions . . . 51
3.4.1 Shoe Lacing Action Primitives . . . 51
3.4.2 Planning Primitive Sequences . . . 53
3.5 Localisation with Active Perception . . . 54
3.6 Experiments and Results . . . 55
3.6.1 Implementation . . . 55
3.6.2 Pattern Designer Experiments . . . 56
3.6.3 Active perception strategy experiments . . . 58
3.6.4 Optimal Position Experiments . . . 59
3.6.5 Complete System Experiments . . . 59
3.7 Conclusion . . . 61

4 Simulating and Benchmarking Long-term DLO Manipulation 63
4.1 The Simulation Environment . . . 63
4.1.1 System Architecture . . . 64
4.1.2 Handling Deformation and Collision for Shoe Lacing . . 66
4.2 Protocol and Benchmark Description . . . 67
4.2.1 Setup Description . . . 67
4.2.2 Experimental Procedure . . . 68
4.2.3 Evaluation Metrics . . . 69
4.3 The Baseline System . . . 71
4.3.1 The Baseline Setup . . . 71
4.3.2 The Baseline Structure . . . 72
4.3.3 The Baseline Results . . . 73
4.3.4 Baseline Experiments in Simulation . . . 74
4.3.5 Difference with the Reality . . . 75
4.4 Conclusion and Discussion . . . 76

5 Tracking DLOs with Physical Simulation 79
5.1 Structure of TSL . . . 80
5.2 Statistical Registration . . . 82
5.3 Physical Simulation . . . 83
5.3.1 Simulation Environment . . . 83
5.3.2 Anomaly Detection . . . 84
5.4 Evaluation Metrics . . . 85

5.4.1 Sequential H-Signature . . . 85
5.4.2 Dynamic Time Warping Distance . . . 87
5.5 Experiments . . . 88
5.5.1 Implementation . . . 88
5.5.2 Qualitative Analysis with *Sequential H-Signature* . . . 89
5.5.3 Quantitative Analysis with the DTW Distance . . . 93
5.6 Conclusions and Future Work . . . 95

6 Influence of Teleoperation Interface in BC for DLO manipulation 97
6.1 Problem Formulation . . . 98
6.2 Teleoperation Interfaces . . . 98
6.2.1 GuMi - Third-Person Perspective Interface . . . 99
6.2.2 Ego-GuMi - First-Person Perspective Interface . . . 100
6.3 Learning Shoelace Insertion from Demonstration . . . 101
6.4 Experiments . . . 102
6.4.1 Hardware Setup . . . 102
6.4.2 Data Collection . . . 103
6.4.3 Result Analysis Metrics . . . 104
6.5 Results and Discussion . . . 105
6.6 Conclusions . . . 110

7 Conclusions and Future Work 111
7.1 Summary of Thesis Contributions . . . 111
7.2 Limitations . . . 112
7.2.1 Failure Recovery . . . 112
7.2.2 Physical Parameters . . . 113
7.2.3 Perception of the Key Components . . . 113
7.3 Future Work . . . 115
7.3.1 Planning Actions with Foundation Models . . . 115
7.3.2 Learning Physical Parameters . . . 115
7.3.3 Learning to Segment DLOs . . . 116
7.3.4 Aligning Aglets with Tactile Sensing . . . 117
7.4 Epilogue . . . 117

A Robot Platform 121
A.1 Robot Platform . . . 122
A.2 Sensors . . . 122
A.3 Hand-Eye Calibration . . . 123
A.4 Mobile Platform . . . 124
A.4.1 Base-robot Interface . . . 124

A.4.2 Camera Module . . . 125
A.4.3 The Navigation Stack . . . 126
A.5 Hardware Components . . . 127

B GuMi: A Bi-manual Teleoperation Device 129
B.1 Device Design . . . 129
B.1.1 Calculating the Joint Configuration . . . 129
B.1.2 Mechanical Design . . . 130
B.1.3 Gravity Compensation . . . 133
B.2 Hardware Components . . . 133
B.3 Calculating Motor Offsets . . . 134
B.4 GuMi-ROS Interface . . . 134

C Simulation with Unity3D 137
C.1 Unity Interface for Robot Simulation . . . 137
C.2 Fast Communication Between Unity and Python with gRPC . . . 139
C.3 Simulating Cameras in Unity . . . 139
C.3.1 Shader for Depth Cameras . . . 139
C.3.2 Camera Parameters . . . 140
C.3.3 Transferring Images between Unity and ROS . . . 141

Bibliography 143

LIST OF FIGURES

Figure 1.1 Cable Harnessing . . . 22
Figure 1.2 Thesis roadmap . . . 26
Figure 2.1 Existing SL machine . . . 30
Figure 3.1 ABB YuMi IRB14000 conducting Shoe Lacing (SL) . . . 46
Figure 3.2 Components of a shoe and a shoelace . . . 47
Figure 3.3 Requirements on completing SL . . . 48
Figure 3.4 Proposed robot SL system . . . 49
Figure 3.5 Three placing sites for proposed system . . . 52
Figure 3.6 Active perception for eyelet observation . . . 54
Figure 3.7 Shoes used in the experiments . . . 56
Figure 3.8 Sample Planning Domain Definition Language (PDDL) action definitions . . . 56
Figure 3.9 Generated patterns . . . 57
Figure 3.10 Results of single eyelet insertion experiments . . . 58
Figure 3.11 Average completion rate of the system experiments . . . 60
Figure 4.1 Real and Simulated environments . . . 64
Figure 4.2 Structure of the simulation environment . . . 65
Figure 4.3 Baseline algorithm operating in the simulation environment . . . 66
Figure 4.4 Obi Attachment . . . 67
Figure 4.5 Measuring shoelace branches . . . 69
Figure 4.6 Initial configuration for the baseline system . . . 72
Figure 4.7 Structure of the baseline algorithm . . . 73
Figure 4.8 Failure cases in simulation . . . 76
Figure 5.1 Sample image of Tracking for Shoe Lacing (TSL) with robot data. . . . 80
Figure 5.2 physical simulation environment of TSL . . . 83
Figure 5.3 Anomaly monitoring during an experiment . . . 85
Figure 5.4 Intersections and *Sequential H-Signature* of the shoelace 86
Figure 5.5 DTW between two sets of estimated states . . . 87
Figure 5.6 Simulation environment for data collection . . . 89
Figure 5.7 Tracking results plotted in 2D and 3D . . . 90

Figure 5.8 *H-Signature* of the shoelace with simulated data. Positive values indicate the shoelace is passing through the eyelet same as the eyelet's normal direction. The shoelace is considered to be passing through an eyelet if the absolute value is over 0.5. TSL maintains a stable shoelace-eyelet relation for all eyelets, whereas the baselines fail to preserve the shoelace paths through the eyelets. 91
Figure 5.9 Occlusions during the tracking process. In SL, the manipulators inevitably occlude the shoelace from different perspectives. TSL keeps the tracking process to continue with its simulated model and the End-Effector (EE) movement. 92
Figure 5.10 DTW distance for all tested methods. TSL and Constrained Deformable CPD 2 (CDCPD2) achieve similar levels of tracking error. Although CDCPD2 is not able to estimate the shoelace-eyelet relation correctly, the difference makes a small impact on the overall tracking error. 93
Figure 6.1 Comparison between GuMi and Ego-GuMi 98
Figure 6.2 GuMi Interface 99
Figure 6.3 The Ego-GuMi framework 100
Figure 6.4 Learning From Demonstration 101
Figure 6.5 Data collection process 104
Figure 6.6 Dynamic Time Warping Analysis of Expert Demonstrations 106
Figure 6.7 Effect of the interfaces on the demonstration time T . . 107
Figure 6.8 Distributions of the variability in demonstrations . . . 107
Figure 6.9 Comparison between Ego-GuMi and GuMi on task performance metrics 109
Figure 7.1 Eyelet segmentation 114
Figure A.1 Hardware Platform 121
Figure A.2 Realsense L515 Camera Mount 122
Figure A.3 Realsense D435 Camera Mount 123
Figure A.4 Hand-Eye Calibration 124
Figure A.5 Base-Robot Interface 125
Figure A.6 Camera module 126
Figure A.7 RTAB-Map with Move Base 126
Figure A.8 Navigation System 127
Figure B.1 YuMi Joint Configuration 129

Figure B.2 Motor positioning and connection in GuMi 130
Figure B.3 Motor stopper and alignment in GuMi 131
Figure B.4 GuMi Power Circuit Protection 132
Figure B.5 GuMi model 132
Figure B.6 Assembled GuMi 133
Figure C.1 Unity ROS Interface Structure 137
Figure C.2 Unity ROS Interface Demo 138
Figure C.3 RGB and Depth Camera in Unity 140

LIST OF TABLES

Table 3.1 Computational cost of different algorithms 57
Table 3.2 Occurrence rate for the sources of Failure for each shoe 59
Table 4.1 Shoes used in the benchmark 71
Table 4.2 Performance of the baseline system 74
Table 4.3 Failure modes occurrence ratio 74
Table 5.1 Tracking Error (mm) with different simulation parameters: Particle Mass (kg), Particle Radius (m) and Max Bending (m). The values used to collect the simulation data are bolded. 94
Table 5.2 Tracking Error with different number of eyelets 94
Table 6.1 Comparison between Ego-GuMi and GuMi on task performance metrics 108
Table A.1 Hardware Components 128
Table B.1 Hardware components of GuMi 134

ACRONYMS

SL	Shoe Lacing
DO	Deformable Object
DLO	Deformable Linear Object
DOO	Deformable One-dimensional Object
STRIPS	Stanford Research Institute Problem Solver
SAS	Simple Action Specification
PDDL	Planning Domain Definition Language
ROS	Robot Operation System
CPD	Coherent Point Drift
RL	Reinforcement Learning
IL	Imitation Learning
AMBF	Asynchronous Multi-Body Framework
EE	End-Effector
LiDAR	Light Detection and Ranging
RTAB-Map	Real-Time Appearance-Based Mapping
SLAM	Simultaneous Localization and Mapping
GA	Genetic Algorithm
BF	Brute-Force
EM	Expectation-Maximization
GMM	Gaussian Mixture Model
GLTP	Global-Local Topology Preservation
CDCPD	Constrained Deformable CPD
CDCPD2	Constrained Deformable CPD 2
MCT	Motion Coherence Theory
TSL	Tracking for Shoe Lacing
ACT	Action Chunking with Transformers
BC	Behavioural Cloning

CVAE	Conditional Variational Autoencoder
LfD	Learning from Demonstration
ROPE	Region of Practical Equivalence
HDI	Highest Density Interval
VR	Virtual Reality
FoV	Field of View
DTW	Dynamic Time Warping
CAD	Computer-Aided Design
EGM	Externally Guided Motion
gRPC	Google Remote Procedure Call
UDP	User Datagram Protocol
GMM-EM	Gaussian Mixture Model with Expectation-Maximization
FEM	Finite Element Method
PBD	Position-Based Dynamics
MuJoCo	Multi-Joint dynamics with Contact
UI	User Interface
SOFA	Simulation Open Framework Architecture
PID	Proportional-Integral-Derivative
MSD	Mass-Spring-Damper
LSTM	Long Short-Term Memory
VLM	Visual Language Model
LLM	Large Language Model

1

INTRODUCTION

Deformable Linear Objects (DLOs) [1] are common in everyday tasks such as knot tying, cable routing and shoelace tying. Recent studies on DLO manipulation have shown promising results in controlling the state of DLOs (Yin et al., 2021). Short-term tasks [2] such as rope shaping (Wang et al., 2022; Yang et al., 2022*a*), cable routing (Sánchez et al., 2020; Zhu et al., 2018), wire insertion (De Gregorio et al., 2018; Wang et al., 2015) have been investigated. However, manipulation of DLOs in long-term tasks, such as knot tying, surgical suturing and Shoe Lacing (SL), remains a challenge.

The manipulation of DLOs is challenging due to their infinite-dimensional state space and the complex non-linear dynamics. The tasks typically exhibit two characteristics: The requirement for high precision and the need for cognitive reasoning. The high precision requirement derives from the size and the deformability of the DLOs. The cognitive reasoning requirement arises from the diversity of the interaction between the DLOs and the surrounding environment.

SL is a representative task in DLO manipulation. The task is challenging from both cognitive and sensorimotor perspectives. From the cognitive perspective, logical planning is required to determine the order of the eyelets being laced and to avoid the shoelace getting entangled. Regarding the sensorimotor aspect, a collaboration between both arms and the visual system is involved. An example is getting the shoelace through the narrow eyelet, which requires precise eyelet localisation and dexterous insertion skills.

In real life, SL is difficult for the disabled and elderly due to its requirements for accurate sensorimotor coordination and cognitive reasoning. Research conducted in the United States shows that about one-quarter of the participants, with an average age of 71, reported foot pain aching, or stiffness in either of

1 Also known as Deformable One-dimensional Objects (DOOs) (Sanchez et al., 2018)

2 This thesis defines long-term manipulation tasks as extended sequences of robot actions involving different action types to achieve intermediate sub-goals. These tasks require high-level planning to coordinate actions across different stages toward the overall objective.

their feet (Menz et al., 2013). A proper re-lacing of shoes by trained nurses helps the elder people better secure the feet to the back of the shoes and results in a more comfortable walking posture, better balance and steadier gait, reducing the risk of falls (Clayton-Jones, 2021).

Meanwhile, as a challenging engineering problem, SL is also manually completed in manufacturing processes. The manual lacing process is time-consuming and requires skilled workers. The inconsistency in the tension of the laces can affect the aesthetics of the shoes and potentially make them less comfortable.

1.1 RESEARCH GOALS

This thesis addresses the manipulation of DLOs under the task of robot SL with four contributions: a SL system using task planning with primitives, a tracking algorithm of shoelaces, a simulation framework and a benchmark to evaluate the performance of the robot manipulation system, as well as a study of the human factor on learning manipulation skills from demonstrations.

1.1.1 Task Planning With Primitives

Long-term DLO manipulation tasks often have various goal states due to complex system dynamics. For example, in knot tying, the goal is to tie a knot, but the knot can be tied in different ways; in surgical suturing, the goal is to stitch a wound, but the suture can be done in different patterns. More importantly, different goals, such as the types of knots and suture patterns, might be more desired in different scenarios. The variability of the goal states makes it difficult to craft a single policy that can achieve all the goal states. Manipulation policies need to decide on the target goal state and plan the actions accordingly.

In SL, a shoe can be laced in various ways. Hagen and Hennig (2009) shows that different SL patterns can affect foot biomechanics. Suitable patterns lead to better comfort and more effective use of the shoe in different scenarios. Polster (2002) discussed the shortest and the strongest SL patterns mathematically. However, there was no information on how to find such patterns with real shoes. This brings the problem of finding the optimal pattern for a given shoe configuration and planning the actions to lace the shoe accordingly. The solution should reflect aesthetics and the preferences of different people. Meanwhile, the constraints of the robot, such as the workspace and

the kinematic limits, as well as the entanglements of the shoelace, also need to be taken into consideration.

1.1.2 *Simulation And Benchmarking*

Simulated benchmarks played a significant part in the development of Reinforcement Learning (RL) and Imitation Learning (IL) for robotic manipulation (Lee et al., 2021). Although several previous works have explored the simulation of deformable objects, simulating objects with complex geometries and interactions remains an open problem (Erickson et al., 2020; Lin et al., 2021; Tagliabue et al., 2020). In this case of SL, unlike single-object manipulation tasks, the interaction between the shoes and shoelaces amplifies its complexity. The slim shoelaces and tiny eyelet openings require the simulation to have millimetre-level accuracy and thus pose a significant challenge to collision detection. The penetration of shoelaces through the eyelets is particularly challenging due to the tunnelling effect [3] (Redon et al., 2002), which highlights the trade-off between real-time performance and simulation accuracy. Meanwhile, the shoes and shoelaces have both rigid and soft components. The force propagation between the rigid and soft parts further complicates the simulation.

Meanwhile, benchmarking is crucial in the development of robotics (Calli et al., 2015). Several benchmarks have been proposed (Chatzilygeroudis et al., 2020; Garcia-Camacho et al., 2020; Huang et al., 2021; Lin et al., 2021) in recent years. However, many of the benchmarked tasks consider only one skill with relatively short horizons, such as rope winding, water cup transporting and pizza dough flattening (Lin et al., 2023). This lack of consideration of long-term tasks is one of the bottlenecks in advancing research towards complex real industrial and domestic settings. In terms of SL, due to the complexity of the setup, detailed experiment protocols and rigorous evaluation metrics become necessary. To validate the usefulness of the benchmark, baseline results should also be provided for comparison.

3 In this thesis, the tunnelling effect refers to the unintended phenomenon where a simulated object passes through another object or a collision boundary without proper physical interaction, typically due to limitations in the simulation's temporal resolution, the size of the object or collision detection algorithms.

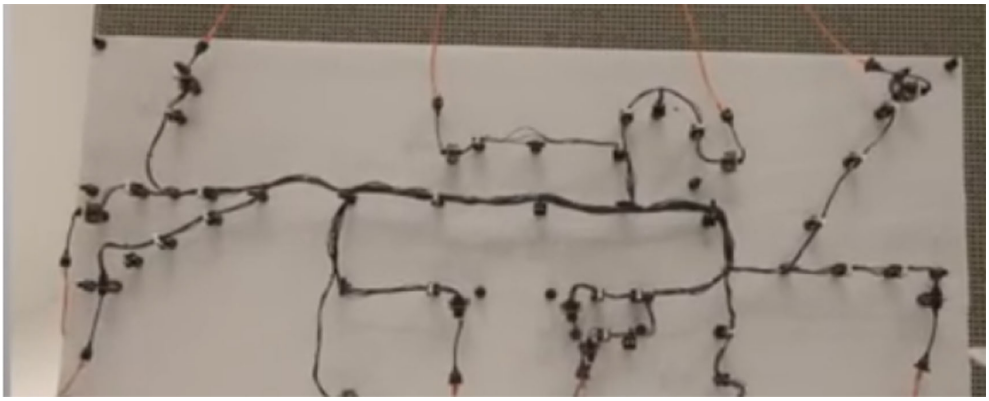


Figure 1.1: Cable Harnessing (reproduced from Zhu et al. (2019))

1.1.3 *Topology-Preserving Tracking*

In DLO manipulation tasks such as cable harnessing and surgical suturing, achieving an accurate 3D understanding of the DLO state is essential (Zhaole et al., 2023; Zhu et al., 2022). Compared with rigid objects, DLOs exhibit more variations in their topology, which is an important factor to consider during tracking (Tang and Tomizuka, 2022). Previous works have tackled the preservation of the topology of the object itself from both the perspective of the local segment scale and the global structure (Chi and Berenson, 2019; Tang et al., 2017; Wang et al., 2021; Xiang et al., 2023). However, in real-world applications, DLOs commonly interact with the surrounding environment and it is also important to consider its relation with the scene objects. For example, in surgical suturing, the success of an insertion is determined by whether the thread passes through appropriate areas of the wound. Similarly, in cable harnessing, the stability and durability of the structure depend on whether the cable properly loops around the nails or merely passes by them (Figure 1.1). The relation with the scene objects provides spatial understanding for task planning to produce realistic and executable plans.

In the context of SL, the task requires precise control of the shoelace states, as well as the understanding of the overall shoelace topology that dynamically evolves during the interaction with the shoe. Correctly estimating the shoelace-shoe relation is crucial for the planning of the task. The estimated shoelace should pass through eyelets after successful insertions; otherwise, corresponding recovery actions must be taken. Meanwhile, tracking the shoelace states is challenging due to the constant occlusion caused by the shoe and the manipulator. The tracked shoelace states should provide accurate information for grasping and the length estimation for stretching the shoelace.

Another question that arises is the performance evaluation of the tracking algorithm. Since the previous works do not consider the DLO-scene objects relations, there have been no standardised evaluation metrics. The metrics should consider the qualitative topological correctness, such as whether the DLO visits the scene objects and the sequence of the scene objects visited; as well as the quantitative accuracy of the tracking algorithm, to provide numerical distances between the estimated and the groundtruth states.

1.1.4 Learning From Human Demonstrations

Manually constructing and tuning such primitives, although robust, is not scalable. Learning from Demonstration (LfD) approaches provide an efficient solution to obtain the desired robot skills without accurate modelling of the dynamics of the DLOs and reduce the injection of domain-specific knowledge from hand-crafted primitives. Among the LfD techniques, BC, leverages demonstrations collected through teleoperated robots, which makes it suitable for tasks with high precision requirements (Liang et al., 2024).

While many approaches have been proposed for LfD (Yin et al., 2021), few consider the human factor in the learning process. The comfort and ease of use of the teleoperation device can affect the performance of the operator, which may lead to a variance in the quality of the demonstrations. Among the different comfort factors, the perspective of the operator is one of the most noticeable aspects. Placing the operator in a third-person perspective can provide a wide view of the workspace and the robot, however, the operator can suffer from heavy occlusion caused by the robot and the target object. It is also difficult to estimate the action of the robot since all movements are mirrored. In the case of SL, this is equivalent to lacing a shoe on someone else's foot. Placing the operator in a first-person perspective, on the other hand, can provide a more immersive experience. This is similar to lacing a shoe from one's own point of view. The operator can see the workspace from the robot's perspective and can estimate the action of the robot more easily. However, this is not always achievable since the operator typically cannot stand in the robot's position. One potential solution is to use Virtual Reality (VR), which may provide the operator with an immersive experience.

1.1.5 Summary Of Research Questions

To summarize, the thesis aims to address the following research questions:

- How can a robot perform long-term DLO manipulation tasks to achieve various goals across different scenarios?
- How can a DLO manipulation task be tested in simulation and evaluated with fair metrics?
- How can a robot perceive the states of a DLO and correctly understand its relation with the scene objects?
- How does the teleoperation interface influence the effectiveness of expert demonstrations in Behavioural Cloning (BC) for DLO manipulation?

1.2 THESIS ROADMAP

As shown in Figure 1.2, the thesis is structured as follows:
The thesis is structured as follows:

- Chapter 1 introduces the manipulation of DLOs and SL as a representative task. It outlines the motivation and research objectives of the thesis.
- Chapter 2 provides an overview of the background relevant to the goals of this thesis. It reviews previous approaches to task planning for robotic manipulation and examines tracking methods for DLOs. Additionally, it discusses learning from human demonstrations, benchmarks proposed for robotic manipulation tasks, and the simulation environments developed for DLOs.
- Chapter 3 presents a complete system designed to address the robot SL task. It begins by providing a detailed definition of the task and explores the optimization of goal states. The chapter then describes how the system utilizes task planning algorithms to determine the sequence of actions necessary to achieve a given goal state.
- Chapter 4 introduces a simulation environment specifically designed for training and evaluating algorithms for the SL task. Additionally, it presents a comprehensive benchmark for assessing the performance of SL systems. The benchmark incorporates a series of metrics to evaluate system performance in a controlled setting. A baseline system is tested using the benchmark in both real-world scenarios with actual robots and the simulated environment.

- Chapter 5 introduces an algorithm designed to track the states of DLOs while maintaining their topology. This algorithm is applied within the context of the SL task to estimate the states of the shoelace. Two metrics are proposed for the evaluation of this algorithm. Its performance is evaluated through experiments conducted using human demonstrations, simulated data, and real robot data.

- Chapter 6 proposes a bimanual teleoperation system and applies it to the SL task. It presents a comparative study evaluating the effectiveness of first and third-person perspective user interfaces for BC in shoelace insertion. This chapter includes both qualitative and quantitative evaluations of the results and discusses how the different interfaces influence learning outcomes.

- Finally, Chapter 7 provides a comprehensive summary of this thesis, synthesising the findings and outcomes of the research presented. It emphasises the significance of the contributions while acknowledging the limitations of the studies. The chapter also explores potential avenues for future work aimed at enhancing the performance of the methods and approaches discussed. Additionally, it highlights the broader potential of the interdisciplinary efforts in this thesis to inspire research in various other domains.

- Appendix A details the robot platform utilised for the experiments in this thesis. It provides an overview of the robot, the sensors, the calibration process between them, and the mobile base developed for the robot.

- Appendix B outlines the design and manufacturing process of the bimanual teleoperation device used in the experiments. It provides details on the selection of hardware components, the key design considerations, and the assembly process.

- Appendix C discusses the development of the Unity simulation environments and the interfaces employed in this thesis. It offers an overview of the robot simulation within Unity, the communication protocols utilised, and the creation of simulated sensors in Unity.

1.3 CONTRIBUTIONS

The work in this thesis makes the following contributions:

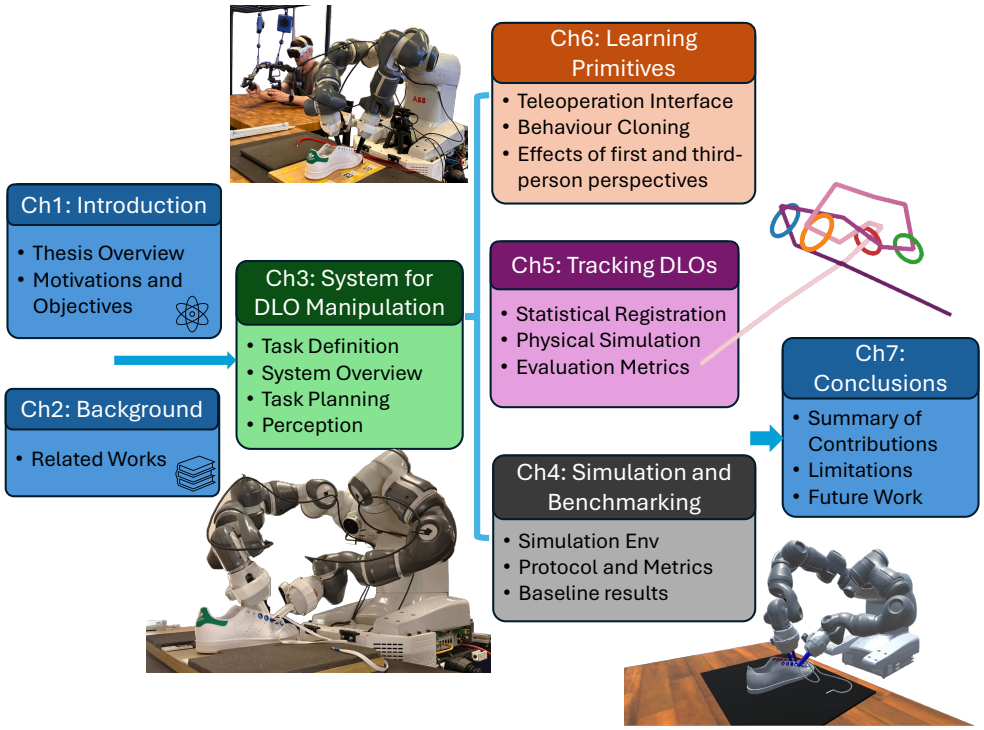


Figure 1.2: Thesis roadmap. Each box represents a chapter in the thesis.

- This thesis presents a complete system for addressing the novel task of robotic SL. It divides the SL into the 3 levels. The high-level SL pattern design task is formulated as an optimisation problem, and a genetic algorithm is applied to solve it. The middle-level SL planning problem is described, and task planning algorithms are employed to determine the sequence of actions required for a given pattern. A set of bi-manual action primitives is defined at the low level to support robotic SL, complemented by an active perception strategy. A series of experiments evaluate the proposed system under five sets of preference criteria, using three shoes varying in size, material, and number of eyelets.

- This thesis proposes an open-source simulation environment for robot SL, aimed at enabling the research community to develop novel methods for solving the task and advancing the field of long-horizon robot task planning. A comprehensive set of performance evaluation metrics is introduced for the bi-manual robot SL task. The validity of the benchmark is demonstrated through the results of a baseline approach, evaluated in both real-world scenarios with actual robots and within the simulated environment.

- This thesis addresses the novel task of shoelace tracking in SL with the proposed algorithm, Tracking for Shoe Lacing (TSL). The algorithm integrates probabilistic registration with physical simulation [4] to maintain the shoelace topology during tracking. A sequential *H-Signature*is introduced for representing the DLO topological state, along with the Dynamic Time Warping (DTW) distances to quantify the algorithm performance. Experiments compare TSL to multiple algorithms using the proposed signature, with evaluations performed on real, simulated, and human demonstration data.

- This thesis presents a bespoke 3D kinematic model designed to replicate the bi-manual robot SL task, along with two interfaces enabling teleoperation from first and third-person perspectives. Additionally, it introduces a dataset comprising over 100 demonstrations for each interface modality, collected using a real bi-manual manipulation platform. The thesis provides both qualitative and quantitative evaluations of the results and examines the impact of the different interfaces on the learning outcomes of BC.

1.4 PUBLICATIONS

This section presents the peer-reviewed publications that are derived from this thesis. Each publication is presented with the full reference, the supplementary materials and the contribution to the thesis.

H. Luo and Y. Demiris (2023), Bi-Manual Robot Shoe Lacing, *in* 'IEEE/RSJ International Conference on Intelligent Robots and Systems', pp. 8772–8775. doi: 10.1109/IROS55552.2023.10341934.

- Supplementary video available at: https://www.youtube.com/watch?v=vazYrF-KfXA&t=54s

- Chapter 3 is based on this article.

H. Luo and Y. Demiris (2024), Benchmarking and Simulating Bimanual Robot Shoe Lacing, *IEEE Robotics and Automation Letters* vol. 9, no. 10, pp. 8202-8209, Oct. 2024, doi: 10.1109/LRA.2024.3440101.

- Supplementary video available at: https://www.youtube.com/watch?v=3UWQLZWmGyM

4 In this thesis, physical simulation refers to the computational modelling of physical systems and their behaviours over time with physical laws.

- Part of Section 3.1 and Chapter 4 are based on this article.

H. Luo and Y. Demiris (2024), TSL: Tracking Deformable Linear Objects for Bimanual Shoe Lacing, *IEEE Robotics and Automation Letters*, Accepted.

- Chapter 5 is based on this article.

H. Luo, R. Chaćon Quesada, F. Estévez Casado, N. Lingg, and Y. Demiris (2025), Interface Matters: Comparing First and Third-Person Perspective Interfaces for Bi-Manual Robot Shoe Lacing Behavioural Cloning, *in* 'IEEE International Conference on Robotics and Automation', Accepted.

- Chapter 6 is based on this article.

2

BACKGROUND

Robot manipulation of Deformable Object (DO) spans a wide range of applications and has gained significant attention in recent years (Yin et al., 2021; Zhu et al., 2022). Tasks involving DLOs form an active area of research within the broader field of DO manipulation. Sanchez et al. (2018) categorised DLOs research into three primary areas: deformation sensing, which includes state estimation (Chi and Berenson, 2019; Tang et al., 2017; Wang and Yamakawa, 2022; Wang et al., 2021; Xiang et al., 2023); grasping and manipulation, encompassing tasks such as cable insertion (Baksic et al., 2020; De Gregorio et al., 2018; She et al., 2021; Wang et al., 2015), rope untangling (Grannen et al., 2020; Huang et al., 2023; Viswanath et al., 2022); and knot tying, a particularly challenging task due to the complex topology involved in the process (Sundaresan et al., 2020; Suzuki et al., 2021; Wakamatsu et al., 2006; Yan, Li, Zhu and Bohg, 2020).

As discussed in Chapter 1, this thesis focuses on the task of SL, which serves as a representative task in the domain of DLO manipulation. SL involves the sensing of shoelace deformation, the grasping and manipulation of the shoelace, and the complex topology created during the lacing process. This chapter provides the theoretical background for DLO manipulation and perception, with an emphasis on SL. Section 2.1 begins by exploring the history of SL research and the challenges associated with automating this task. Section 2.2 then provides an overview of planning methods in DLO manipulation, focusing on the use of action primitives and heuristic search algorithms. Section 2.5 examines tracking algorithms for DLO state estimation. Section 2.4 discusses simulation and benchmarking techniques for DLO manipulation tasks. Finally, Section 2.6 reviews recent Imitation Learning (IL) works on DLO manipulation and the development of teleoperation interfaces.

2.1 ROBOT SHOE LACING

The history of footwear can date back to 40,000 years ago when the structure of human foot bones changed (Trinkaus and Shang, 2008). For any type of shoe that ever existed, there are always some fastening mechanisms to secure the shoe onto the human feet. Lacing shoes with fully autonomous robots has been a part of the endeavours of shoe manufacturers for the past two decades.

2.1.1 *Industrial Endeavours*

Despite years of effort in developing SL machines, shoe manufacturers continue to rely heavily on manual labour for lacing shoe uppers. Manual lacing often results in inconsistent tension, which can affect the aesthetics of the shoes and potentially reduce their comfort. Existing machine designs are typically limited to a single type of shoe, requiring modifications to accommodate different shoe models (Figure 2.1). Moreover, these machines depend on specialised tools, making them expensive to produce and maintain. This thesis aims to address these limitations by employing general-purpose robots and sensors without the need for specialised tools so that the solutions are adaptable to various types of shoes and shoe models. The robot platform used in this thesis is discussed in Appendix A.

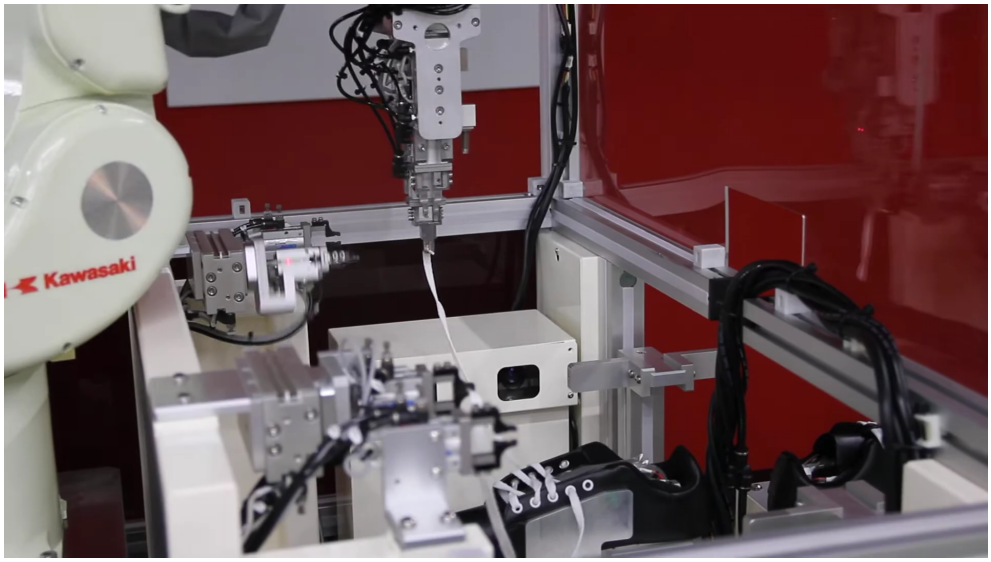


Figure 2.1: Existing SL machine (figure reproduced from Solomon AI and 3D Vision (2017))

2.1.2 *Shoe Lacing With Different Patterns*

A SL problem, like many other DLO tasks, can be solved in many ways due to the infinite number of dimensions of the state space and the complex non-linear dynamics. In the case of SL, lacing a shoe can be done following different patterns, such as the commonly used crisscross or bowtie. As a reference, a preliminary study with four participants was conducted as part of this thesis. All participants selected the crisscross lacing pattern, with an average completion time of 85 seconds. However, their lacing strategies varied: one participant laced all eyelets with the aglet pointing inward, while the other three began with an inward insertion for the first eyelet and outward insertions for the remaining ones. Human participants were typically able to insert the aglet accurately regardless of the insertion direction, without needing to adjust its orientation or perform reinsertion.

While people naturally choose the crisscross pattern, it is not evident that this is the optimal choice. To find the shortest and strongest patterns, Polster (2002, 2006) formulated the SL task as a mathematical problem. The author simplifies the problem by representing a theoretical shoe with eyelets on a 2D plane. It is shown that depending on the number of eyelets and the gap between the eyelets, the shoelace patterns vary remarkably. It is also proven that for real shoes, whose eyelets are not on the same 2D plane and do not fall into two parallel lines, some patterns may not apply at all. This thesis formulates pattern designing as a constrained optimisation problem following the settings from Polster (2002, 2006), and aims to numerically compute the optimal lacing patterns in different scenarios.

2.2 PLANNING FOR DEFORMABLE LINEAR OBJECT MANIPULATION

A planning problem is defined by its initial states, the actions available to alter those states, and the desired goal state (Russell and Norvig, 2016). Since the earliest days of robotics, efforts have been made to automate planning (Haslum et al., 2019). Long-horizon robotic manipulation tasks require reasoning about both high-level logical task actions and precise geometric motions (Lagriffoul et al., 2018). While many studies focus on controlling the states of ropes (Nair et al., 2017) or cables (Zhu et al., 2018), less attention has been given to planning the sequence of actions needed to achieve a goal state. Earlier works, such as Moll and Kavraki (2006); Tandon et al. (n.d.), introduced local planners for DLOs based on minimum energy curves. However, these approaches are limited to short-horizon simple tasks and are prone to

local minima (Yan, Li, Zhu and Bohg, 2020). High-level logical planning is therefore needed to guide the motion planning process.

2.2.1 *Planning With Topology*

For long-horizon tasks, Wakamatsu et al. (2004) introduced the concept of topological states to represent the DLO. They developed strategies for selecting grasping points and determining robot motions for each topological action step. Similarly, Saha and Isto (2007) proposed a method that constructs search trees by sampling possible robot motions. By leveraging knot theory to prune unpromising branches, they enhanced the efficiency and feasibility of the planning process.

In recent studies, Grannen et al. (2020); Huang et al. (2023); Lui and Saxena (2013); Sundaresan et al. (2021); Viswanath et al. (2022) addressed knot untangling tasks by leveraging manipulation primitives, while Sudry et al. (2023); Sundaresan et al. (2020); Yan, Li, Zhu and Bohg (2020) focused on knot tying using primitives learned from human demonstrations. In both cases, the selection and application of primitives were informed by domain-specific insights derived from knot theory. Knot theory as a branch of algebraic topology (Akbulut and King, 1981), studies the properties of mathematical knots, which are closed loops in three-dimensional space. In the case of SL, the inclusion of shoes introduces additional topological complexity in the relation between the shoelace and the shoe, complicating the problem compared to knot tying.

2.2.2 *Planning With General Purpose Planners*

A more general approach involves manually defining the action-state space and using a general-purpose planner to identify the optimal sequence of actions. Over the years, Various methods for general-purpose planning have been developed (Guo et al., 2023), including hierarchical planning, plan graphs, and heuristic-based approaches. One notable heuristic-based planner is LAMA (Richter and Westphal, 2010), which effectively balances action costs with time constraints, producing fast yet high-quality plans. LAMA incorporates landmarks, the critical states that must be achieved en route to the goal, as heuristic guides. This enables LAMA to generate high-quality plans quickly and makes it a widely adopted tool for applications requiring effective and scalable planning solutions.

Problems in task planning are typically described using formal languages such as Stanford Research Institute Problem Solver (STRIPS) (Fikes and Nilsson, 1971), Simple Action Specification (SAS) (Bäckström and Nebel, 1995), and Planning Domain Definition Language (PDDL) (David et al., 1998). Over the years, PDDL has emerged as one of the most widely adopted representations for planning problems (Haslum et al., 2019). PDDL is an expressive modelling formalism to set up general planning problem (David et al., 1998). It provides a standardised framework for defining actions, states, and goals, enabling planners to generate sequences of actions to achieve desired outcomes. This thesis leverages PDDL to describe the SL problem and uses LAMA to generate the sequence of actions.

2.3 SIMULATING DLOS

As DO manipulation gains increasing attention, there is a growing demand for models that approximate real-world physical behaviour based on physical laws, in order to control object states and support reinforcement learning. Such models are referred to as physically-based models (Sanchez et al., 2018). Deformable objects possess infinite degrees of freedom, adding significant complexity to their simulation. The accuracy of these models directly impacts the performance of planning algorithms and the effectiveness of learnt policies. Common physically-based models for DOs include particle-based methods, discrete-based approaches such as Mass-Spring-Damper (MSD) models, and continuum-based methods like Finite Element Method (FEM) (Sanchez et al., 2018; Sin et al., 2013).

FEM is a numerical technique used to solve complex problems in engineering and physics by dividing a structure or domain into smaller, simpler elements, such as tetrahedra, referred to as finite elements. It is widely applied to simulate and analyse material behaviour under various conditions, such as stress and deformation. FEM-based simulations are highly accurate and well-suited for applications requiring high precision. However, due to their computational complexity, FEM methods are not widely adopted for simulating DOs (Baksic et al., 2020; Ficuciello et al., 2018; Seita et al., 2020).

A more commonly used approach is the MSD model, which represents objects as a system of particles connected by springs and dampers. This method is intuitive and computationally efficient, making it a popular choice for robotic manipulation and particularly well-suited for simulating linear deformable objects (Almaghout et al., 2024; Grannen et al., 2020; Schulman, Lee, Ho and Abbeel, 2013; Tang and Tomizuka, 2022; Yan, Zhu, Jin and Bohg,

2020). Another widely adopted method is particle-based systems, which represent objects as a collection of discrete particles, each carrying properties such as mass, velocity, and position (Tonnesen, 1998). When subjected to external forces, these particles move to new positions according to physical laws.

To improve the real-time performance, position-based models have been proposed as a computationally efficient alternative to FEM and MSD models. It iteratively updates object positions to satisfy geometric constraints (Bender et al., 2014). Position-based models are widely used in computer graphics due to their simplicity and speed. This makes them favourable for real-time applications such as game development and physics engines.

2.3.1 *Physics Engines Supporting DLO Simulation*

Physics engines are virtual environments designed to simulate the laws of physics for general-purpose applications. Compared with manually coding the physics of objects for each application, physics engines provide a more efficient and flexible solution. They are widely used in fields such as motion planning to predict the future states of objects (Sanchez et al., 2018). Numerous physics engines have been developed to simulate DOs, including Bullet, Unity, Multi-Joint dynamics with Contact (MuJoCo), NVIDIA PhysX, and Simulation Open Framework Architecture (SOFA).

Bullet is an open-source physics library primarily developed by Erwin Coumans (Coumans et al., 2013), featuring Position-Based Dynamics (PBD) and FEM solvers. Its Python wrapper, PyBullet (Coumans and Bai, 2017), has been widely adopted in the research domain due to its accessibility and open-source nature. Applications include sim-to-real transfer (Matas et al., 2018), object tracking (McConachie et al., 2020), and assistive dressing (Erickson et al., 2020). Additionally, Bullet Physics has been integrated into other engines such as Blender (Community, 2018).

MuJoCo is a physics engine developed for high-fidelity simulation of robotic systems, biomechanics, and other dynamic environments (Todorov et al., 2012). It is renowned for its ability to simulate complex, continuous dynamics with precise handling of contact, soft constraints, and multi-body systems, making it a widely used tool in robotics research (Erez et al., 2015). MuJoCo has been extensively employed in robot simulation and reinforcement learning, particularly for tasks requiring high accuracy and control. Its support for soft constraints enables the modelling of deformable objects

such as cloth and rope (Yan, Li, Zhu and Bohg, 2020), further broadening its applicability.

NVIDIA PhysX is a real-time physics engine commonly used for simulating rigid and soft body dynamics, collision detection, and fluid dynamics in applications such as gaming, robotics, and virtual environments. Famous for its high performance and GPU acceleration, PhysX offers robust simulation capabilities suitable for interactive and computationally demanding tasks. It supports cloth simulation using PBD, making it well-suited for tasks involving deformable objects. PhysX has been employed in reinforcement learning, as demonstrated by Clegg et al. (2017).

SOFA is an open-source framework designed for multi-physics simulation, with a focus on real-time interaction and soft-body dynamics. It is widely used in domains such as medical simulation, robotics, and haptics, owing to its modular architecture and support for highly customised simulations (Westwood et al., 2007). SOFA includes built-in mass-spring models and FEM support, making it well-suited for applications involving deformable objects. Ficuciello et al. (2018) utilised SOFA to control deformations into desired shapes using Proportional-Integral-Derivative (PID) controllers, while Petit et al. (2017) employed its FEM solver for simulating and tracking pizza deformations.

Unity is a versatile game engine that integrates NVIDIA PhysX as its physics solver, featuring an intuitive User Interface (UI) design alongside well-documented plug-ins and assets (Technologies, 2005). Among its physics plug-ins specialised for soft objects is the real-time particle-based solver *Obi*, which supports simulations involving a wide range of objects, including ropes, cloth, liquids, and general volumetric bodies (Studio, 2016). Weng et al. (2021); Yu, Lv, Zhong, Song and Li (2022) demonstrated the capability of *Obi* in simulating ropes and cloth, showcasing its utility in complex soft-body dynamics. In SL, an essential aspect is the interaction between the rigid aglets and the soft shoelace. Compared to other physics engines, *Obi* offers a more intuitive approach to simulating interactions between soft and rigid bodies. The *Obi attachment* functionality facilitates the exchange of impulses between soft and rigid bodies, making it particularly suitable for simulating shoes and shoelaces.

Overall, due to the ability to simulate the interaction between both soft and rigid objects, as well as the efficiency of particle-based dynamics, this thesis explores the use of Unity and Obi for simulating SL.

2.3.2 *Environments For DLO Simulation*

Several simulation environments have been developed to simulate DOs, aimed at accelerating learning and advancing research in this domain. SoftGym, for instance, uses NVIDIA PhysX to simulate water, rope, and cloth for tasks such as water pouring and cloth folding. NVIDIA Isaac Gym (Makoviychuk et al., n.d.) has been utilised to generate grasping data for DOs (Huang et al., 2022). PyBullet is employed in Assistive Gym (Erickson et al., 2020) for tasks like bed bathing and assistive dressing, as well as in OmniHang (You et al., 2021) for collision estimation. Additionally, Seita et al. (2021) used PyBullet to generate simulated data for cables, fabrics, and bags. Reform (Laezza et al., 2021) provides environments for tasks such as wire bending and cable closing using AGX Dynamics. MuJoCo has also been utilised for simulating cloth and rope (Yan et al., 2021).

From the medical robotics domain, several simulation environments have been proposed for surgical applications. Munawar et al. (2022) introduced an environment using Asynchronous Multi-Body Framework (AMBF) for learning and practising surgical sutures, while Tagliabue et al. (2020) tackled soft tissue simulation using NVIDIA PhysX. Similar to the physics engines discussed in Section 2.3.1, these environments primarily focus on simulating a single type of DO. This thesis aims to develop a simulation environment that supports the simulation of both the shoe and the shoelace, as well as their interactions.

2.4 BENCHMARKING DLO MANIPULATION

In recent years, although several benchmarks have been proposed for DO manipulation, researchers have highlighted the lack of comprehensive and well-established benchmarks in this domain (Yin et al., 2021).

As early pioneers, Chatzilygeroudis et al. (2020) introduced benchmarks for semi-deformable object manipulation tasks, including watchmaking and rubber-band handling. Garcia-Camacho et al. (2022, 2020) proposed benchmarks for bimanual cloth manipulation, focusing on tasks such as cloth folding and spreading. SoftGym (Lin et al., 2021) expanded the scope by supporting tasks with linear, planar, and liquid objects[1]. However, existing benchmarks often emphasise relatively short-horizon tasks and lack a focus

1 Some tasks exhibit different characteristics depending on their definitions. For example, cloth folding in Lin et al. (2021) involves only one type of action, while Zhou et al. (2024) defines it as requiring multiple actions, making planning essential.

on the complexity of planning. DaxBench (Chen et al., 2022) supports tasks involving ropes and liquids, such as liquid pouring and rope shaping. While these tasks span more steps, they typically involve repeating a singular type of action rather than requiring a sequence of coordinated actions and logical planning.

In the medical domain, long-horizon tasks such as surgical suturing are common. However, existing benchmarks in this area primarily focus on evaluating surgeon skills (McKendy et al., 2017) or assessing robot-assisted surgeries (Raison et al., 2017), rather than on fully autonomous robotic systems. This thesis focuses on providing a benchmark on the bimanual manipulation task of SL, which demands both long-horizon task planning and dexterous manipulation skills with DOs.

2.5 PERCEPTION OF DLOS

Perception of DOs has been extensively studied across various domains (Wang et al., 2021). In DLO manipulation tasks, achieving an accurate 3D understanding of the state of DLOs is crucial (Zhaole et al., 2023; Zhu et al., 2022). Research on the perception of DLOs can broadly be divided into two main categories: segmentation and state estimation (Zhaole et al., 2023).

2.5.1 Segmenting DLOs

Segmentation involves identifying the boundaries of objects within an image or point cloud (Minaee et al., 2021). In many studies on DLO manipulation, segmentation is simplified using colour thresholding under conditions without occlusion (Zhaole et al., 2023), or bypassed entirely by attaching markers to the DLO (Yang et al., 2022*a*; Zhou et al., 2021).

Segmentation of thin structures has been extensively studied within the computer vision domain, including applications such as blood vessel segmentation in medical imaging (Wang et al., 2020), drive lane detection in autonomous driving (Zhang et al., 2021), and road network extraction in satellite image analysis (Yuan and Cheriyadat, 2016). These studies often employ data-driven methods to enhance segmentation performance. To address the specific challenges of occlusion and complex backgrounds in DLO segmentation, Caporali et al. (2022); Thananjeyan et al. (2022); Zanella et al. (2021) proposed deep learning-based approaches that harness the power of neural networks to improve accuracy. However, collecting large-scale annotated datasets for training these models remains a significant challenge,

particularly for DLO tasks that require precise annotations (Zhaole et al., 2023). While segmentation is a challenging aspect of DLO perception, this thesis focuses on the state estimation of DLOs and assumes a clear colour contrast between the shoelace and the background.

2.5.2 *State Estimation Of DLOs*

Common state estimation approaches discretise DLOs as a set of ordered key nodes (Tang and Tomizuka, 2022), where the DLO is represented by these nodes and the edges connecting them. DLO tracking is typically formulated as a point set registration problem, with one set representing the maintained key nodes (also referred to as state points) and the other consisting of point clouds from new observations. While Iterative Closest Point is one of the most well-known registration methods, the soft assignment of correspondences is generally preferred in DLO tracking due to its suitability for handling the flexibility and deformation inherent in DLOs.

During tracking, common methods align observed point clouds with the maintained state representation using point registration techniques based on Gaussian Mixture Model with Expectation-Maximization (GMM-EM) and Coherent Point Drift (CPD). The key nodes are treated as Gaussian centroids, with Gaussian Mixture Model (GMM) used to compute correspondences through probability distributions (Chui and Rangarajan, 2000). Node positions are then updated based on new point cloud observations using Expectation-Maximization (EM), while CPD is designed to preserve the global topology (Myronenko and Song, 2010). The global topology refers to the overall structure of the DLO, which should remain invariant under continuous deformations. To further maintain the local topology, Global-Local Topology Preservation (GLTP) was introduced to regularise the local structure using Locally Linear Embedding (Ge et al., 2014; Roweis and Saul, 2000). Tang et al. (2017) applied GLTP to DLO tracking, demonstrating the robustness of the estimator to sensor noise, outliers, and occlusions.

However, the estimator does not account for physical constraints, and probabilistic registration alone is insufficient to produce physically plausible estimations (Tang and Tomizuka, 2022). From this limitation, two distinct research directions have emerged. The first approach incorporates dynamic simulation to enforce physical constraints (Tang et al., 2017; Wang and Yamakawa, 2022). The second approach applies selected posterior constraints, either after the EM step or directly to the cost terms in the M-step, to restrict the DLO states, as demonstrated in Constrained Deformable CPD (CDCPD) (Chi and

Berenson, 2019) and Constrained Deformable CPD 2 (CDCPD2) (Wang et al., 2021). Building on these efforts, TrackDLO (Xiang et al., 2023) introduced the use of Motion Coherence Theory (MCT) to estimate the spatial velocity of occluded tip nodes based on the spatial velocity of visible nodes, eliminating the need for tip position priors.

2.5.3 *Topology In DLO Tracking*

The aforementioned methods tackle the preservation of the local and global structure but lack the consideration of interactions with the environment. This limitation arises primarily from the tasks they address. To this end, this thesis refers to the concept of homotopy. Homotopy refers to a property of the DLO that remains invariant under continuous deformation (Bhattacharya et al., 2011). Two paths are *homotopic* or belong to the same *homotopy class* if one can be continuously deformed into the other. In 3D space, homotopy classes can only be introduced by obstacles with one or more holes, or by obstacles stretching to infinity in two directions (Bhattacharya et al., 2011). For tasks such as shaping a rope or cable into desired forms, there is typically only one homotopy class. However, homotopy becomes a critical factor in the SL task, as the shoelace interacts with the shoe, which includes the hole structure eyelets. The eyelets create new homotopy classes, which increases the complexity of the task. This thesis considers the eyelets during the tracking process and constrains the tracked states to preserve the shoelace-eyelet relations.

The concept of homotopy has long been studied in motion planning of mobile robots (Hernandez et al., 2015). In the path planning of tethered robots, the homotopy classes of robot paths are identified to prevent entanglement (Cao et al., 2023; Yang et al., 2023). To avoid obstacles, *H-Signature* is proposed to characterise the topological class of paths in a given environment (Bhattacharya et al., 2011; Hernández et al., 2011). It assigns a value to each obstacle, enabling the identification of topologically distinct trajectories (the formulation is introduced in Section 5.4.1). This is particularly useful in motion planning, where distinguishing between homotopy classes enables more efficient exploration of the solution space. Mitrano et al further enhance this representation by incorporating the robot state into *H-Signature* (Mitrano and Berenson, 2024). A similar representation is also used in guiding DLOs for insertion tasks (Wang et al., 2015). While being extensively studied in motion planning, it cannot be directly applied to evaluate the tracking performance. The continuous signature values it provides for each of the obstacles

do not reflect the sequence of the obstacles being passed through. In tasks like SL, the sequence of the eyelets being passed through is crucial for the success of the task. This thesis aims to incorporate the concept of *H-Signature* to evaluate the tracking performance in the SL task.

2.6 LEARNING TO MANIPULATE DLOS FROM DEMONSTRATIONS

Yin et al. (2021) identified two main approaches for DLO manipulation: model-based and data-driven methods. Model-based approaches rely on physical models to predict the behaviour of DLOs, but they often face challenges due to the infinite degrees of freedom and non-linear dynamics, making them computationally demanding.

Data-driven methods, on the other hand, leverage learning-based techniques such as IL and Reinforcement Learning (RL) to develop manipulation policies Yan, Li, Zhu and Bohg (2020); Yu, Zhong and Li (2022). IL uses expert demonstrations to teach agents human-like behaviours, while RL allows agents to learn optimal actions through interaction with the environment. These methods address some of the limitations of model-based approaches by using machine learning to handle the complexities of DLO manipulation (Yin et al., 2021).

2.6.1 *Imitation Learning In DLO Manipulation*

IL is a learning technique designed to replicate human behaviour in specific tasks (Hussein et al., 2017). It has gained popularity for teaching manipulation skills as it removes the need for explicitly programming primitives or defining task-specific rewards (Zare et al., 2024). By leveraging expert demonstrations, IL enables robots to acquire complex skills that are otherwise challenging to encode manually.

Over the years, IL has been successfully applied to a wide range of DO manipulation tasks (Yin et al., 2021). These include cloth manipulation (Jia et al., 2019), where robots learn to smooth or fold fabrics, sand shaping (Cherubini et al., 2018), assistive dressing (Pignat and Calinon, 2017), and garment folding (Yang et al., 2016). The versatility of IL in handling diverse tasks highlights its potential to address the challenges associated with DO manipulation in unstructured and dynamic environments. Among the IL techniques, Behavioural Cloning (BC), leveraging demonstrations collected through teleoperated robots and training complex skills by direct imitation from the demonstrations, enables the policies to complete tasks with high

precision (Liang et al., 2024). This makes BC a suitable choice for learning DLO manipulation skills.

As discussed in the introduction of this chapter, one of the most commonly studied DLO manipulation tasks is knot tying. Kudoh et al. (2015) proposed an approach for in-air knot tying by segmenting demonstrations into motion primitives, which are then recombined to create different types of knots. Similarly, Rambow et al. (2012) applied this concept to tube routing, introducing admittance controllers to handle positioning uncertainties during the task. These methods generally require a single demonstration to replicate the task and are effective for scenarios that closely resemble the demonstrated configuration (Yin et al., 2021). However, environments with greater variability pose additional challenges. To address such cases, Schulman, Gupta, Venkatesan, Tayson-Frederick and Abbeel (2013) introduced a method for surgical suturing that uses non-rigid registration techniques to adapt the demonstration to new scenarios. Similarly, Lee et al. (2015) proposed a method for learning force-based manipulation skills, such as knot tying, towel flattening, and whiteboard cleaning, to enhance adaptability in dynamic environments.

Moving to more recent works, Yan, Li, Zhu and Bohg (2020) proposed a method for learning knot tying from demonstrations using Long Short-Term Memory (LSTM) with attention mechanisms (Vaswani, 2017). Luo et al. (2024) proposed a hierarchical scheme to learn manipulation policies for long-term tasks such as cable routing. Similarly, Zhao et al. (2024) achieved impressive results on complex and dexterous manipulation tasks, such as tying shoelaces, hanging shirts, and inserting gears, by employing a Transformer-based neural network trained with Diffusion Policy. The authors showcased the high level of performance that can be achieved in enabling robots to perform intricate bimanual tasks, leveraging large-scale datasets autonomously.

2.6.2 *Teleoperation Systems And Behaviour Cloning*

One of the main challenges in Learning from Demonstration (LfD) is the collection of high-quality demonstrations. The learned policies can only be as effective as the demonstrations provided Schettino and Demiris (2019); Wang et al. (2023). Meanwhile, the quality of the teleoperation interface can have a large influence on the quality of the collected data for BC.

Robot teleoperation systems enable the integration of human expertise and cognitive skills into physical robots (Darvish et al., 2023). These systems are also critical in hazardous environments where humans cannot safely enter and in unstructured settings where autonomous robotic operation is

not feasible. The design of teleoperation systems is highly influenced by the design of the physical robot and the specific task requirements, leading to a wide range of system designs and variations in affordability.

One of the most renowned high-end teleoperation systems is the da Vinci Surgical Robot, widely used in minimally invasive surgery (Kim et al., 2024). This system enables surgeons to perform precise and complex procedures with enhanced dexterity and control, revolutionising surgical techniques. However, the cost of such systems can reach up to 2.5 million dollars (Turchetti et al., 2012), making them prohibitively expensive for many research institutions and smaller healthcare facilities. High-end teleoperation systems are characterised by their precise control and haptic feedback capabilities. For example, the DLR bimanual haptic device (Hulin et al., 2011) and NimbRo Avatar (Schwarz et al., 2023) are specifically designed for fine-grained manipulation tasks. These systems provide operators with accurate force feedback, allowing them to perceive the robot's interactions with its environment in real time. This level of precision makes them ideal for applications requiring delicate and highly controlled manipulations, such as intricate assembly tasks.

In recent years, there has been a surge in the development of low-cost teleoperation systems that trade the fidelity of haptic feedback for affordability and accessibility. ALOHA is a notable example of a low-cost, open-source hardware system designed for bimanual teleoperation (Zhao et al., 2023). It uses budget servo arms as controllers, enabling precise and dynamic manipulation tasks such as threading zip ties and tying shoelaces. Building on this concept, Wu et al. (2023) introduced GELLO, a framework for constructing economical teleoperation systems using not just low-cost servo motors but also 3D-printed components. The framework creates kinematically equivalent, scaled-down replicas of existing robot arms, allowing users to adapt from existing platforms while remarkably reducing system costs. Other teleoperation devices used for robot control include 3D mice (Zhu et al., 2023), Virtual Reality (VR) devices (Iyer et al., 2024; Seo et al., 2023), and vision-based hand tracking (Qin et al., 2023; Sivakumar et al., 2022). These devices provide intuitive and cost-effective control solutions, but they often fail to account for the physical constraints of the robot, which can impact the accuracy and repeatability of the teleoperation system.

One of the main limitations of low-cost teleoperation systems, such as ALOHA and GELLO, is the perception of the workspace. These systems rely on the visual feedback from the operators, which can be hindered by occlusions from the robot or target objects, depending on the task. To

address these challenges, recent works have explored more immersive teleoperation experiences. Bunny-VisionPro integrates VR-based control with haptic feedback using low-cost vibration actuators (Ding et al., 2024). Similarly, Open-TeleVision enhances operator spatial awareness by combining stereoscopic video feedback with active camera control (Cheng et al., 2024).

This thesis aims to develop an affordable and accessible bi-manual teleoperation system for collecting demonstrations for the SL task, addressing the aforementioned limitations while enabling precise and reliable data collection.

2.6.3 *Comparing Teleoperation Interfaces*

One of the research questions arising from the wide variety of teleoperation interfaces is how the choice of interface affects the quality and effectiveness of demonstration data collected for BC. Despite its importance, few studies have investigated the impact of teleoperation interface quality on the effectiveness of collected demonstration data for BC (Si et al., 2021). This thesis aims to address this gap by comparing the performance of different teleoperation interfaces, each offering different perspectives of view, under consistent conditions for the SL task.

2.7 SUMMARY

This chapter provided a background on the challenges and existing solutions for robot DLO manipulation from the planning, simulation, perception, and learning perspectives, with a particular focus on robot SL. It first reviewed industrial efforts and mathematical formulations for SL patterns, highlighting the task's long-horizon and topologically complex nature. Various planning approaches were then discussed, from topology-aware methods to general-purpose symbolic planners using PDDL.

The chapter also surveyed physically-based models and physics engines used for simulating DLOs, motivating the use of Unity and Obi for simulating the interactions between shoes and shoelaces. The benchmarking landscape was examined, revealing a lack of long-horizon benchmarks.

In the context of perception, the chapter reviewed segmentation and tracking methods, identifying a gap in preserving the DLO-scene object during tracking. Finally, learning from demonstration (LfD) techniques were discussed, with a focus on teleoperation systems for collecting high-quality data. The chapter concluded with a discussion on how different teleoperation interfaces may affect the quality of behavioural cloning in SL.

Together, these topics provide the foundational context and reveal critical gaps in existing research. The subsequent chapters build on this foundation, presenting contributions that directly address these gaps across the planning, simulation, perception, and learning aspects.

3

DLO MANIPULATION WITH TASK PLANNING AND HIERARCHICAL PRIMITIVES

As discussed in Section 1.1.1, long-term DLO manipulation tasks often have various goal states. Shoe Lacing (SL), for example, can be completed with different patterns, which require different action plans to achieve. This chapter addresses the first research question:

> *How can a robot perform long-term DLO manipulation tasks to achieve various goals across different scenarios?*

This chapter takes the example of the SL task and presents a system achieving different SL patterns using a bi-manual robot. Section 3.1 first presents a comprehensive definition of the SL task to explain the related concepts and requirements, followed by a system overview in Section 3.2. Using the mathematical formulation, Section 3.3 proposes a pattern optimiser to search for the best lacing pattern corresponding to the shoe configuration and the user preferences. Section 3.4 proposes a set of action primitives and a primitive sequence planner to generate action sequences according to the designed pattern. The system plans the trajectories based on the perceived position of the eyelets and aglets with an active perception strategy (Section 3.5), and the trajectories are deployed on a bi-manual robot shown in Figure 3.1. Section 3.6 presents the experimental results and analyses the system performance. Section 3.7 concludes this chapter. Experiments demonstrate that the proposed system can successfully lace 3 different shoes in different configurations, with a completion rate of 92.0%, 91.6% and 77.5% for 6, 8 and 10-eyelet patterns respectively.

The work from this chapter has been published in Luo and Demiris (2023). A supplementary video is available under this link.

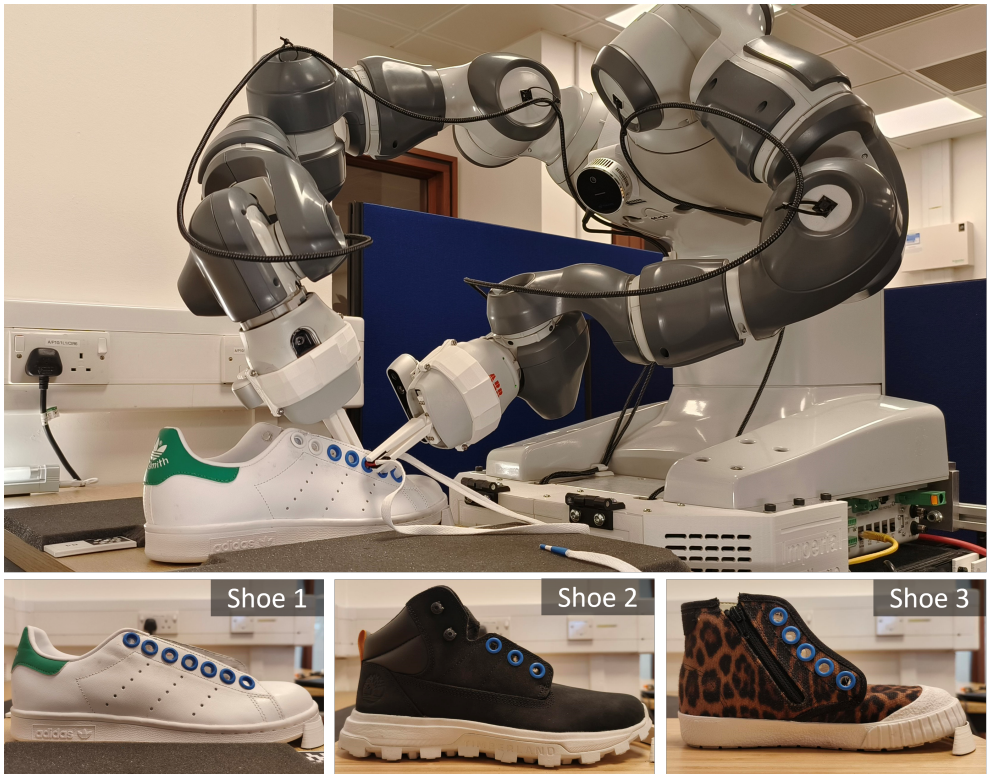


Figure 3.1: Autonomous bimanual SL using an ABB YuMi robot and the proposed system. The system is designed to achieve different SL patterns, leveraging genetic algorithms and task planning. The proposed system is validated with real-world experiments on 3 shoes varying in size, material and number of eyelets.

3.1 DEFINITION OF THE SHOE LACING TASK

This section introduces the relevant terms and concepts, followed by the proposed task description.

3.1.1 *Definition Of Terms And Concepts*

The SL task involves a shoe and a shoelace. As shown in Figure 3.2, an eyelet $e_i \in E$ ($i \in [1, N]$) is the small opening on eyestays accepting the shoelace. A shoe has 2 eyestays with N eyelets. The eyelets are numbered from left to right, bottom to top.

The shoelace is composed of the rigid tips, named the aglets $\{a_1, a_2\}$, and the flexible part. The flexible part connecting an eyelet and an aglet is a shoelace branch b and the part connecting two eyelets is a shoelace segment $s(e_i, e_j)$, where e_j is closer to the aglet. A segment is leftward if e_i and e_j are

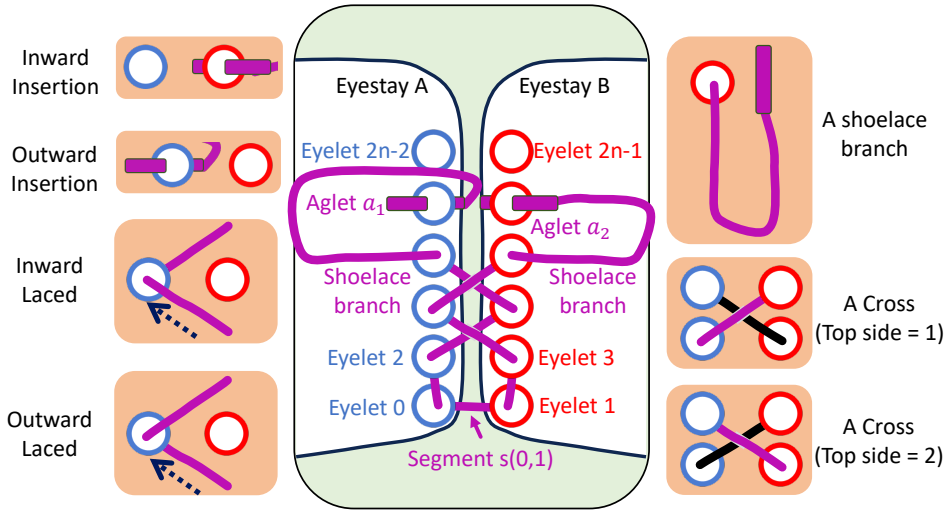


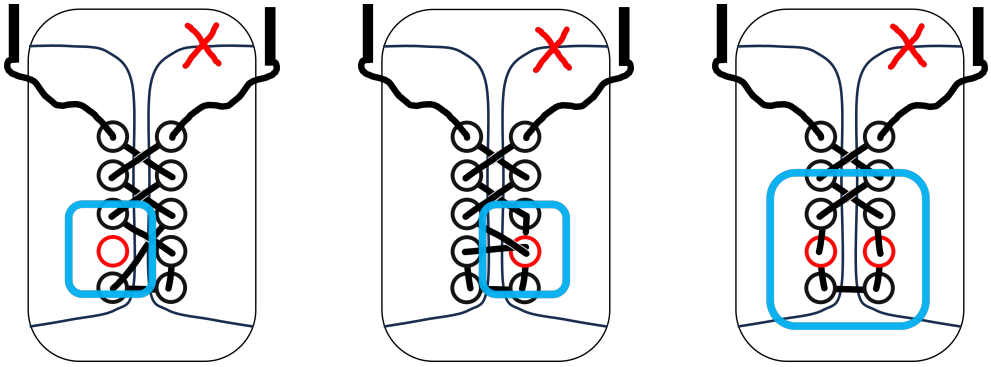
Figure 3.2: Components of a shoe and a shoelace. Laced in mixed ways for illustration of different lacing techniques. **Aglets** are the tips of shoelaces. **Eyelets** are circular openings on eyestays to accept shoelaces. A **shoelace branch** connects an aglet to the closest eyelet. **Segments** are parts of the shoelace between two eyelets. A segment can be leftward, rightward, vertical and horizontal. Two segments form a **cross** when they intersect. A cross has **top side modes** 1 and 2 when the leftward and rightward segments are on top respectively.

in the left and right column respectively and rightward for the opposite; it is vertical if e_i and e_j are in the same column.

3.1.2 *Task Description*

A lace action $\texttt{lace}(a, e)$ is the process where aglet a passes through eyelet e and is pulled outside the shoe cavity. The laced eyelets form a new group E'. The action is inward if a points towards the shoe centre, or outward if the opposite. The SL task is a series of lace operations where the aglets a_1 and a_2 of the shoelace are inserted through every eyelet e once, leaving each eyelet, apart from the last row, connected to two other eyelets. There are 3 requirements for the SL patterns, as illustrated in Figure 3.3.

- The shoelace must connect the eyestays ($E = E'$) and miss no eyelets.
- The shoelace must not travel through the same eyelet more than once.
- Two consecutive vertical segments must not connect. In this case, the middle pair of eyelets is not constrained in the horizontal direction (e.g. the red eyelets in Figure 3.3c).



(a) Missing (b) Repeating (c) Caving

Figure 3.3: 3 Requirements on completing SL. 1) The shoelace should not miss any eyelet; 2) The shoelace should not travel through an eyelet more than once; 3) Two vertical segments should not connect.

A SL pattern is a permutation of the eyelets $[e_{\sigma(1)}, ...e_{\sigma(N)}]$, where σ gives the permuted eyelet indices. A pattern is considered *simple* if the indices of the eyelets that both aglets travel through are non-decreasing. In this thesis, only the *simple* patterns are considered.

3.2 SYSTEM OVERVIEW

The system solves the SL problem in 3 stages: 1) The pattern designer computes the optimal SL pattern based on the eyelet positions and user-defined preference criteria (Section 3.3.2). 2) The task planner then details this pattern into a sequence of actions (Section 3.4.2). 3) Finally, these primitives are consecutively executed by the execution manager on a bi-manual robot (Section 3.4.1). A flowchart of the proposed system is shown in Figure 3.4.

3.3 DESIGN OF THE SHOE LACING PATTERNS

The pattern affects the aesthetics, comfort and functionality of the shoe. However, given a shoe configuration, there are numerous patterns to select from. According to Polster (2002, 2006), the total number of solutions to a lacing problem is

$$\frac{(n!)^2}{2}\sum_{k=0}^{m}\frac{1}{n-k}\binom{n-k}{k}^2,\ m=\begin{cases}\frac{n}{2}, & n \text{ even},\\ \frac{n-1}{2}, & n \text{ odd},\end{cases} \tag{3.1}$$

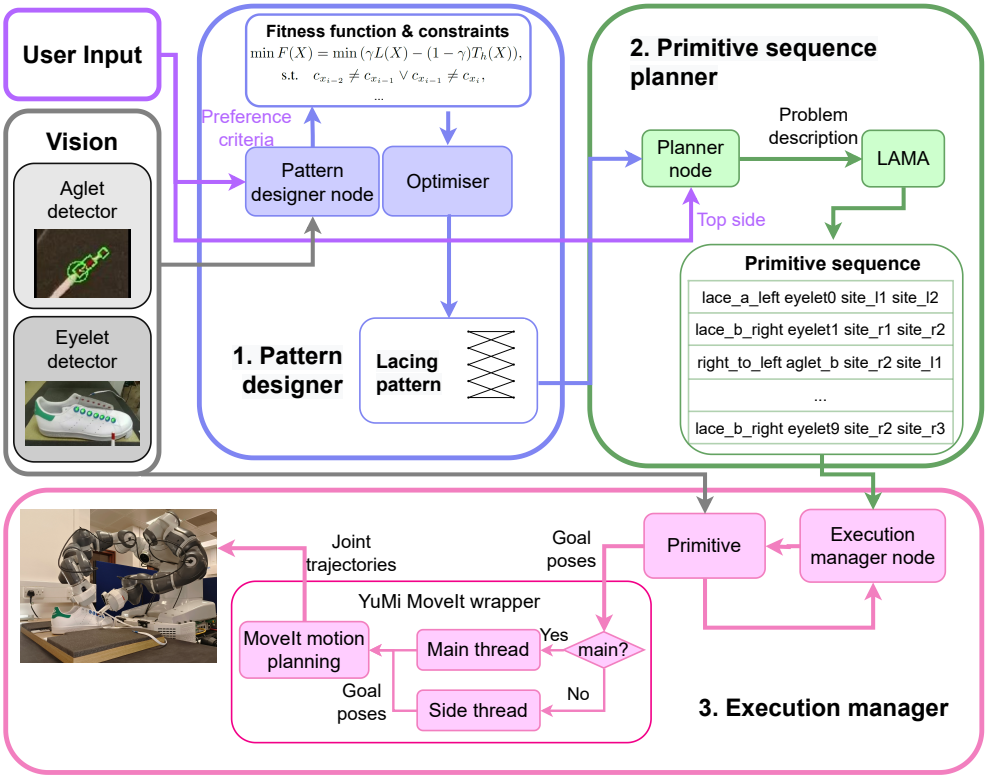


Figure 3.4: **The proposed system for robot SL.** It is composed of 3 stages: 1) The pattern designer generates the optimal SL pattern based on the preference criteria and the eyelet positions (Sec Section 3.3.2); 2) the planner finds the primitive sequence achieving this pattern while satisfying the top side requirement (Sec Section 3.4.2); 3) the list of primitives is consecutively sent to the execution manager and deployed onto the real robot (Sec Section 3.4.1).

where n is the number of rows of eyelets. For a standard shoe with 6 rows of eyelets, there exists a total of 3,758,400 patterns. Brutally traversing and comparing each one of them is time-consuming and computationally expensive. Instead of using exact methods, this section defines the pattern designing task as an optimisation problem and solves it with the real shoe eyelet configurations.

3.3.1 *Definition Of The Pattern Designing Problem*

A SL pattern is essentially a permutation of the eyelet indices

$$X = [x_0\ x_1\ x_2\ x_3\ ...\ x_{2n-1}],$$

where $x_i \in \mathbb{N}, x_i < 2n-1, x_i \neq x_j, \forall i,j \in [0, 2n-1]$.

Let the length l of a segment $s(p, q)$ be the Euclidean distance between eyelets p and q. The horizontal distance between the two eyelets in the same row is

$$h(k-1) = h(k) = l(k-1, k), \tag{3.2}$$

where k is odd. The length of the shoelace used during SL

$$L(X) = \sum_{i=1}^{2n-1} l(x_{i-1}, x_i). \tag{3.3}$$

Assume the tension along the shoelace is a constant T. Its horizontal component T_h holds the eyestays together. A pattern is the strongest when $\frac{T_h}{T}$ is maximised (Polster, 2006).

$$T_h(X) = T \sum_{i=1}^{2n-1} \frac{h(x_i)}{l(x_{i-1}, x_i)}. \tag{3.4}$$

This thesis considers two aesthetic properties of a pattern:

- *Dense*: Patterns with no vertical segment. This leads to a denser look of the pattern.
- *Straight*: Patterns include all possible horizontal segments. This leads to a more organised look.

Pattern designing is the process of numerically finding an X that minimises $L(X)$ or maximises $\frac{T_h}{T}$ considering *dense* and *straight* constraints.

Designing a SL pattern is essentially a permutation-based combinatorial optimisation problem with user-defined preference criteria as constraints. The preference criteria include:

1. the objective factor λ, which selects between the objectives for the shortest and the strongest pattern;
2. whether the *dense* mode is required;

Let the fitness function be $F(X)$. The optimisation problem is as follows:

$$\begin{aligned} \min F(X) &= \min\left(\gamma L(X) - (1-\gamma)T_h(X)\right), \\ \text{s.t.} \quad & c_{x_{i-2}} \neq c_{x_{i-1}} \vee c_{x_{i-1}} \neq c_{x_i}, \\ & c_{x_{i-1}} \neq c_{x_i}, \\ & r_{x_{i-2}} \neq r_{x_{i-1}} \wedge r_{x_{i-1}} \neq r_{x_i}. \end{aligned} \tag{3.5}$$

where $\gamma \in \{0, 1\}$ selects the objective; $i \in [2, 2n-1]$ is the index of x. The first constraint ensures no more than 2 consecutive eyelets in the same column.

The second and third constraints are used when *dense* and/or *straight* are selected.

3.3.2 *Designing Shoe Lacing Patterns With Genetic Algorithms*

To solve the optimisation problem, a Genetic Algorithm (GA) is adopted (Dwivedi et al., 2012). GAs are derivative-free and highly flexible in defining objective functions, making them well-suited to the discrete and combinatorial nature of the problem. As a global optimisation technique, a GA is also less susceptible to local minima, increasing the likelihood of providing global optimal solutions (Dwivedi et al., 2012).

The proposed algorithm employs ordered crossover and insertion mutation (Larranaga et al., 1996). The selection process is a tournament selection. During the tournament selection, chromosomes are selected according to the principles proposed in Deb (2000). Same as in Deb et al. (2002), no penalty values are added to the fitness as the fitness values are never compared with constraints.

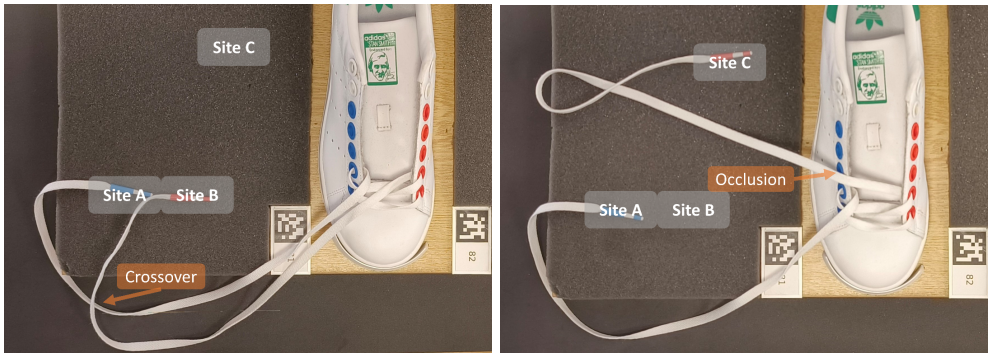
3.4 PLANNING SHOE LACING ACTIONS

Planning refers to a group of algorithms that take the goal state and the available actions as input and outputs a solution in the form of an action sequence (Russell and Norvig, 2016). The pattern designer produces the goal condition for the SL task, which specifies the connection between eyelets. One still needs to decide the order to lace the eyelets and the movement of the aglet while avoiding entanglements. In this section, this decision-making process is formulated as a planning problem. The proposed solution includes a series of parameterised action primitives and details the planning of the action sequences to achieve the target pattern. The execution of the primitives is aided by the vision module, which is detailed in Section 3.5.

3.4.1 *Shoe Lacing Action Primitives*

The following 8 sets of atomic bi-manual primitives are proposed for a SL system. Each set of primitive has a left and a right version in practice.

1. **Pick**(g, a): Vision module finds the pose of target aglet a. Gripper g reaches the coordinate and picks up the aglet.



(a) Site A and B (b) Site A and C

Figure 3.5: Three placing sites are proposed for each side of the shoe (grey boxes). Shoelace branches at sites A and B may overlap with each other, while those at site C may occlude the eyelet (orange boxes). The placing sites are selected by the task planner (Section 3.4.2).

2. **Place**(g, a, s): Gripper g releases aglet a to placing site s. To avoid entanglement, 3 placing sites are proposed on each side of the shoe (Figure 3.5).

3. **Re-orient**(g): The second gripper regulates the held aglet to be perpendicular to the gripper g.

4. **Stretch**(g): Gripper g moves an aglet towards the target direction by the estimated remaining shoelace length. This stretches the connected shoelace branch taut to avoid tangling.

5. **Observe**(g): This action finds the pose of the target eyelets or aglets. The vision module first finds the target from default positions. This reading is then corrected with the active perception strategy (Section 3.5).

6. **Reset**(g): Arm with gripper g moves to $\mathtt{reset}$ configuration in case the robot goes into difficult positions.

7. **Insert**(g, e): This action inserts an aglet with g through the eyelet e and pulls it out with the second gripper. The remaining shoelace length is updated according to the distance from the previous eyelet afterwards.

8. **Handover**(g): Gripper g moves the aglet to the $\mathtt{transfer}$ point. The second gripper comes to grasp this aglet and takes it to the other side.

Empirically, many atomic primitives are always executed together. 3 sets of composed high-level action primitives are proposed for the left and right sides:

1. **Lace**(g, a, e, s): **Observe**(g)-**Pick**(g, a)-**Re-orient**(g)-**Insert**(g, e)-**Handover**(g)-**Stretch**(g)-**Place**(g, a, s).
2. **Transfer**(g_1, g_2, a, s): **Pick**(g_1, a)-**Handover**(g_1)-**Stretch**(g_2)-**Place**(g, s).
3. **Replace**(g, a, s): **Pick**(g, a)-**Stretch**(g_2)-**Place**(g, s).

3.4.2 *Planning Primitive Sequences*

To find the sequence of actions required for a given pattern while considering user preferences, a primitive sequence planner is constructed. First, a symbolic description of the SL planning problem is provided.

The SL planning problem involves 2 aglets, 6 placing cites, 2 grippers and $2n$ eyelets. The states are defined as follows:

1. *aglet-at*(a, s): Aglet a is at site s.
2. *eyelet-laced*(e): Eyelet e has been laced.
3. *covered*(s): Placing site s is covered by the shoelace. Placing at sites A and B creates an overlap (Figure 3.5).
4. *occupied*(s): Placing site s is occupied by an aglet.
5. *preferred*(e_1, e_2): Eyelet e_1 prioritised over eyelet e_2. The user-specified top side of the shoelace segments is controlled with this state. For example, in Figure 3.2, there is a cross with segments $s(2, 1)$ and $s(3, 0)$. With top side mode 2, eyelet 3 is considered preferred.
6. *lace-with*(e, a): According to designed pattern, eyelet e is supposed to be laced with aglet a.
7. *on-correct-side*(a): Aglet a is at the same side as the eyelet it is directly connected. *Correct sides* is introduced to avoid the shoelace branches twisted together. The free aglet is required to be on the correct side.
8. *left-correct-side*(a): The eyelet which aglet a is directly connected is on the left side.
9. *right-correct-side*(a): The eyelet which aglet a is directly connected is on the right side.

Using the proposed states, 8 actions α corresponding to the 3 sets of composed primitives are constructed:

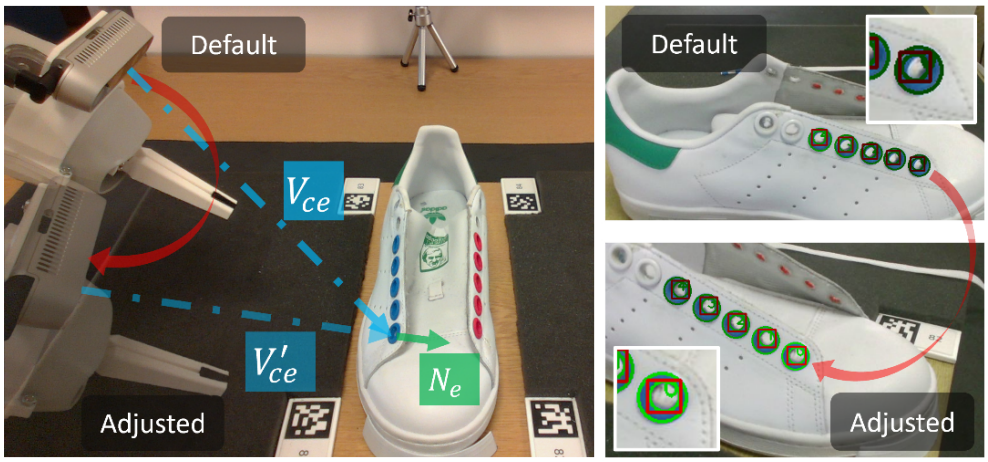


Figure 3.6: Eyelet observation before and after active perception strategy engages. Brighter contours indicate higher confidence. The camera to the target eyelet vector V_{ce} is adjusted to V'_{ce} which better aligns with the eyelet normal N_e. As a result, the detected target eyelet (rightmost) is corrected to the eyelet centre and also the image centre.

1. *lace-a/b-left/right*(e, s_1, s_2): 4 actions which lace eyelet e from the left-/right side with aglet a/b. This action changes states *aglet-at*, *eyelet-laced*, *occupied* and *at-correct-side*.

2. *left/right-to-right/left*(a, s_1, s_2): 2 actions which moves an aglet a from s_1 to opposite side site s_2. This action changes states *aglet-at*, *occupied* and *at-correct-side*.

3. *left/right-re-place*(a, s_1, s_2): 2 actions which moves an aglet a from s_1 to same side site s_2. This action changes states *aglet-at* and *occupied*.

The primitive sequence planner incorporates a task planning algorithm to solve this problem. It considers each action to be of the same cost and minimises this cost by outputting the plan with the least total number of actions.

3.5 LOCALISATION WITH ACTIVE PERCEPTION

In this chapter, the research focuses on the completion of the SL task. The perception is relaxed with coloured eyelets and aglets (Figure 3.4). The vision module localises the eyelets and aglets with colour thresholding.

Although the view of the cameras at the *observe* position covers the whole workspace, in practice, an eyelet can be at the corner of the image, affecting the accuracy of the inner opening detection (Figure 3.6).

To improve the accuracy of the detection, an active perception strategy is deployed to tackle this problem. The strategy evaluates the quality of detection with a confidence score. This score is defined by the angle between the camera to eyelet vector v_{ce} and the normal of the eyelet n_e (Figure 3.6).

$$\text{Confidence}(e) = \frac{1}{\pi} \arccos \frac{v_{ce} \cdot n_e}{|v_{ce}||n_e|}. \tag{3.6}$$

After observing from the default position, the strategy checks the confidence score of the target eyelet. If the confidence is lower than a threshold, the gripper adjusts the observing pose so that the optical axis is perpendicular to the eyelet. Figure 3.6 shows a detection result, the bounding box of the target bottom right eyelet is off the centre of the eyelet. After the active perception strategy is engaged, the eyelet is moved to the centre of the image facing the camera.

3.6 EXPERIMENTS AND RESULTS

The proposed system is validated from four aspects. First, the effectiveness of the pattern designer is verified. Two experiments are conducted to demonstrate the variety of the patterns and evaluate the computational cost of the optimiser (Section 3.6.2). Then, the performance of the active perception strategy is tested by lacing single eyelets (Section 3.6.3). The insertions results provide the optimal shoe position in the workspace (Section 3.6.4). Finally, complete system experiments are conducted and the failure cases are analysed (Section 3.6.5).

3.6.1 Implementation

The experiments are conducted with an ABB YuMi robot (Figure 3.1). Two Intel Realsense D435s are mounted on the grippers and an Intel Realsense L515 is mounted on the front side of the robot. More details of the hardware setup can be found in Appendix A. Four ArUco markers are used to log the shoe positions. Figure 3.7 show the three shoes used in experiments: (Shoe 1) The right shoe of UK size 6.5 Adidas Stan Smith (14 eyelets with leather upper); (Shoe 2) the right shoe of 6.5 Timberland Treeline Mid Hiker (6 eyelets with leather upper); (Shoe 3) the left shoe of 1.5 Clarks Cyrus Geo (10 eyelets with textile upper). A 137 cm flat fabric shoelace is used in all experiments. Limited by the shoelace length, only the first 10 eyelets of shoe 1 are used. Both the eyelet grommets and the aglets are coloured for detection.



(a) (b) (c) (d) (e) (f)

Figure 3.7: Shoes used in the experiments and some of the lacing results. (a) Shoe 1 (14 eyelets with leather upper); (b) Shoe 2 (6 eyelets with leather upper); (c) Shoe 3 (10 eyelets with textile upper); (d) Shoe successfully 1 laced with C1; (e) Shoe 2 successfully laced with C1; (f) Shoe 3 failed with C4.

```
(:action left_insert_a
   :parameters (?el - eyelet_left
      ?el_n - eyelet ?c_b - eyelet)
   :precondition (and
      (cursor_a ?el)
      (cursor_b ?c_b)
      (next ?el ?el_n)
      (> (id ?c_b) (id ?el))
      (aglet_at aglet_a site_left)
   )
   :effect (and
      (eyelet_laced ?el)
      (cursor_a ?el_n)
      (not (cursor_a ?el))
   )
)
```

```
(:action left_to_right_transfer
   :parameters (?al - aglet)
   :precondition (and
      (aglet_at ?al site_left)
   )
   :effect (and
      (aglet_at ?al site_right)
      (not (aglet_at ?al site_left))

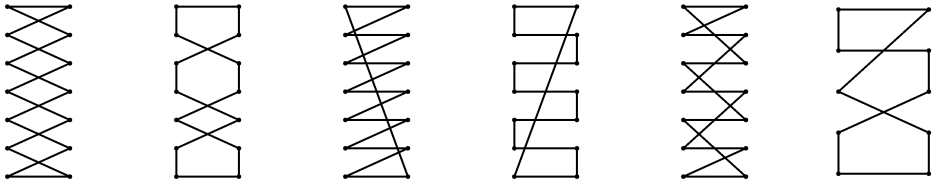
   )
)
```

Figure 3.8: Sample PDDL action definitions

The primitive sequence planner incorporates a description of the SL problem with Planning Domain Definition Language (PDDL) (Figure 3.8) and the LAMA planner, as discussed in Section 2.2.2. MoveIt is used for robot trajectory planning and visualisation. A 2-thread execution system is created to reduce the execution time (Figure 3.4). The main thread executes actions that are pivotal, such as **insert** and **picking**. The side thread is used for actions such as **reset**.

3.6.2 *Pattern Designer Experiments*

Figure 3.9 shows some patterns generated under 5 sets of preference criteria with shoes 1 and 3. The sets of criteria include (C1) max strength; (C2) min



(a) C1 (b) C2 (c) C3 (d) C4 (e) C5 (f) C2

Figure 3.9: Some patterns generated by the proposed optimiser with shoe 1: (a) max strength, (b) min shoelace, (c) max strength (straight), (d) min shoelace (straight), (e) min shoelace (straight, dense); and shoe 3: (f) min shoelace. (a) is known as the *crisscross* pattern.

shoelace; (C3) max strength (straight); (C4) min shoelace (straight); (C5) min shoelace (straight, dense).

Patterns generated with shoe 1 (Figure 3.9a to Figure 3.9e) are more 'canonical', being either symmetric or rotationally symmetric. However, for shoes like shoe 3, the patterns are not. The two eyelet columns of shoe 1 are roughly parallel to each other, whereas for shoe 3 they do not converge (Figure 3.7). Depending on the actual shoe configuration, optimal patterns for the second type of shoe can have a mixture of crosses and long diagonals (Figure 3.9f).

Next, the computational cost of the proposed GA is evaluated by comparing against the Brute-Force (BF) method. 35 sets of eyelet positions of shoe 1 in random poses are collected. The GA and the BF methods are used separately to compute the optimal pattern supposing the shoe had 6, 8, 10, 12 and 14 eyelets with the first 2 sets of criteria.

Table 3.1: Computational cost by times of evaluations with the BF method and the GA in terms of average times of fitness function evaluations. Experiments used C1 and C2 with shoe 1 at 35 different positions. Values in parenthesis are estimated results due to computation limits. The GA computational cost increases slowly with the number of eyelets but its computation takes less than 1s for all experiments.

		6	8	10	12	14
BF	C1	42	1080	51840	(3.76×10^6)	(3.83×10^8)
	C2	42	1080	51840	(3.76×10^6)	(3.83×10^8)
GA	C1	4051	4245	4707	5443	6346
	C2	4059	4305	5503	6062	7221

All patterns generated by BF and GA are the same for shoes with less than 10 eyelets, proving the global optimality of the proposed designed patterns. For other scenarios, BF becomes intractable (Table 3.1). The BF results in parenthesis are estimated with Equation (3.1). The computational costs of GA are noticeably lower than BF for normal 10, 12 and 14-eyelet shoes. Although

the GA computational cost also increases slowly with the number of eyelets, all computations are completed within 1s.

3.6.3 *Active Perception Strategy Experiments*

The effectiveness of the proposed active perception strategy is validated with an ablation study. Shoe 1 is moved randomly within the workspace and the **lace** action is applied with and without the active perception strategy at each location. The success rates for insertion with active and static perception are 29/30 and 20/30 respectively. With active perception, the insertion success rate is overall 30% higher. Most of the failure cases with passive perception happen during the pulling rather than insertion stage. The size of the eyelet provides enough accuracy margin to tolerate the localisation errors, however, the exit angle of the aglet varies with the insertion pose. Since the pulling does not perceive the aglet again, the aglet is missed during pulling in these cases.

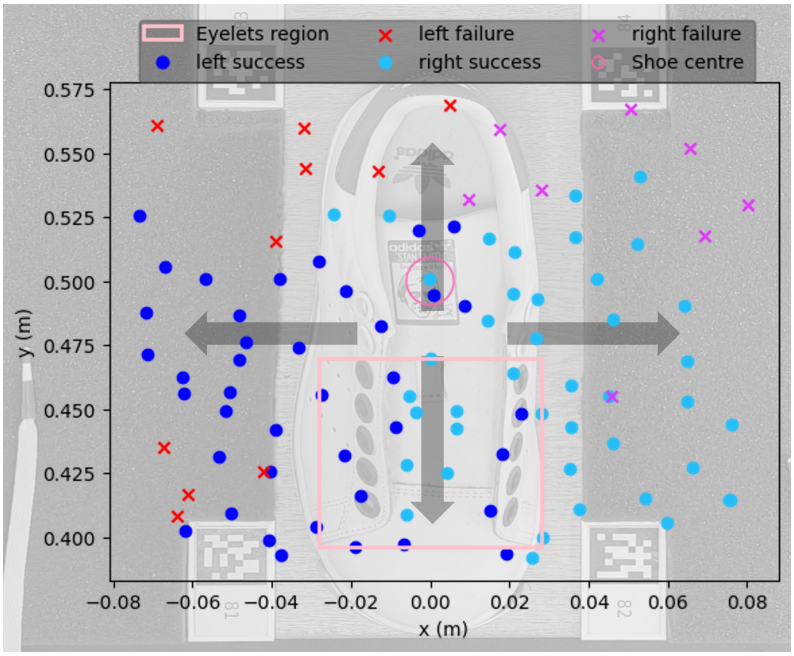


Figure 3.10: Results of single eyelet insertion experiments. The shoe is mounted on a movable plate and moved to different locations. At each location, the robot executes the **lace** action on a selected eyelet. Blue and light blue dots signify the successful insertions from the left and right sides respectively, while the red and pink crosses indicate the failures. The desired shoe centre for the complete system experiment is picked according to the distribution of the dots, indicated by the black circle. The black box shows the consequential eyelet positions.

3.6.4 *Optimal Position Experiments*

To select the optimal shoe location within the workspace, 100 insertions (50 on each side) are conducted and Figure 3.10 shows the results. The blue and light blue dots show successful insertions from the left and right sides respectively, while red and pink crosses are the failure cases.

The success cases are observed within a rectangular shape. The eyelets are not visible when YuMi is at *observe* configurations, which resulted in the left, right, and bottom borders. The reachability of YuMi also limits the shoe from getting too far away which leads to the formation of the top border. As a result, the desired shoe centre for system experiments is selected at $x = 0.50\text{m}, y = 0.0\text{m}$.

3.6.5 *Complete System Experiments*

The complete system is tested with 3 shoes under all 5 sets of criteria introduced in Section 3.6.2. The shoes are placed in the position acquired from the previous experiment. Experiments are repeated 3 times for each shoe per set of criteria. For asymmetric patterns, the required length for two shoelace branches is imbalanced. Only the first 6 eyelets for patterns in Figure 3.9c, Figure 3.9d, and the first 8 for Figure 3.9e are laced.

As shown in Figure 3.11, the proposed system achieves a completion rate of 92.0%, 91.6% and 77.5% for 6, 8 and 10-eyelet patterns respectively. In term of time usage, the system takes 837s, 679s, and 501s to complete 6, 8 and 10-eyelet patterns respectively. The system robustly laces shoe 1 and shoe 2 in all patterns, while the results for shoe 3 are worse. Overall, the success rates are lower for longer patterns. 3 sources of failures are observed (Table 3.2):

Table 3.2: Occurrence rate for the sources of Failure for each shoe

	Source 1	Source 2	Source 3
Shoe 1	25.0%	50.0%	25.0%
Shoe 2	0.0%	33.3%	66.7%
Shoe 3	73.3%	13.3%	13.3%

EXCESSIVE TENSION APPLIED TO GRIPPERS This is caused by the error in the shoelace length estimation. Due to the elasticity of the shoelace, the **stretch** primitive cannot always fully stretch the shoelace. Since there is no actual feedback on the shoelace length, the estimation accuracy deteriorates, causing more tension to be applied to the shoelace. Eventually, either the aglet is

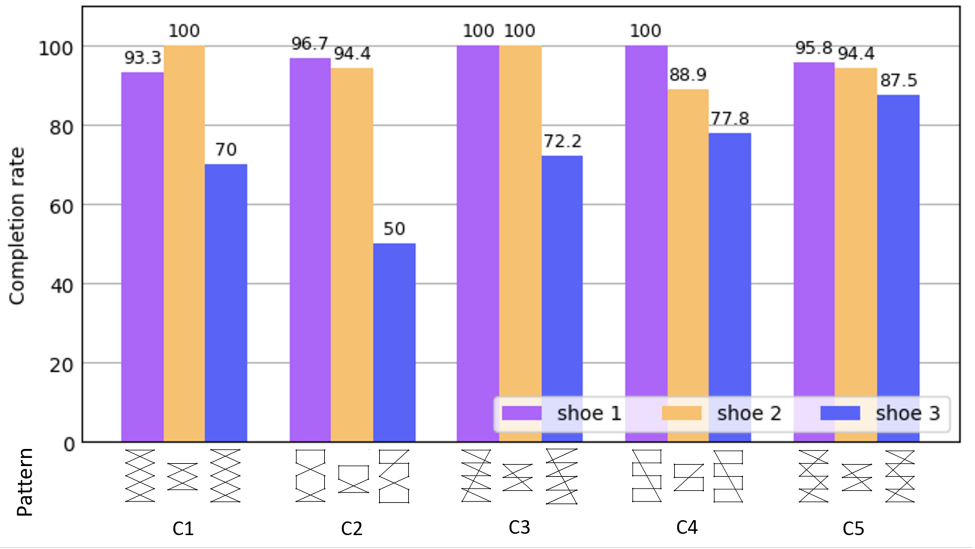


Figure 3.11: Average completion rate of the system experiments. The pattern completed under each set of criteria is shown at the bottom. For Shoe 3 with C2, the resulting pattern is asymmetric, likely due to the non-parallel eyestays of the shoe. Robot SL is performed with 3 shoes under the 5 sets of criteria. The completion rate is the number of laced eyelets over all eyelets. The system is more robust with shoe 1,2 compared to shoe 3.

pulled off from the fingers or the gripper gets stuck. This explains the overall lower completion rate with longer patterns, and the significant performance drop between the 8 and 10-eyelet patterns. The estimation error is amplified for shoe 3 whose textile upper is highly deformable (Figure 3.7f). This requires a more accurate estimation of the remaining shoelace length. The shoelace tracking problem is two-folded: estimating the shoelace configuration and the relation with the eyelets. Chapter 5 proposes a method to tackle this problem.

INSERTION FAILURES This is due to the non-canonical aglet orientation prior to insertion. Although the orientation of the aglet is regulated with the **Re-orient** primitive, occasionally they are not perfectly perpendicular to the fingers. As the insertion is open-loop, this error can not be corrected. Chapter 6 proposes a method to learn an insertion policy from demonstrations that achieves insertion without **Re-orient**.

OTHER FAILURES These include aglets stuck between the eyestays, eyelet localisation failure and picking failure. With shoe 3, the eyestays are highly deformable, and the gap between them shrinks when tightened with the shoelace, trapping the aglet after insertion. The estimated depth can be wrong

in rare cases, which causes a grasping failure when the gripper travels too deep.

3.7 CONCLUSION

In this chapter, a complete system for bimanual robot SL is presented to achieve different SL patterns. A GA is used to design the SL pattern according to the shoe configuration and the preference criteria; a LAMA planner is used to generate action primitive sequences from the pattern; the execution manager then deploys primitives in sequence to move the robot and lace the shoe. The pattern designer gives consistent results with different criteria and finds solutions within 1 second. The lacing system can achieve a completion rate of 92.0%, 91.6% and 77.5% for 6, 8 and 10-eyelet patterns respectively. The system demonstrates the possibility of using action primitives with symbolic planners to achieve various goal states in DLO manipulation. At the current stages, the main limitations of the proposed system is the lack of recovery mechanisms and the tracking of the shoelace states. More future work is discussed in Chapter 7. This includes replacing the task planner with a Foundation Model to reduce the requirement of human-introduced domain knowledge, as well as recovery strategies for the system to handle the failure cases.

4

SIMULATING AND BENCHMARKING LONG-TERM DLO MANIPULATION

As discussed in Section 1.1.2, simulation and benchmarking play an important role in the development of robot manipulation systems. Simulated environments provide a safe and cost-effective platform for testing and validating algorithms. In the context of SL, comparison between different algorithms is challenging due to the limited availability of the physical setups and the lack of standardised benchmarks. This chapter tackles the following research goal:

> *How can a DLO manipulation task be tested in simulation and evaluated with fair metrics?*

This chapter presents a simulation environment and a benchmark for bimanual SL. Section 4.1 proposes an open-source high-fidelity simulation environment (Figure 4.1) for SL to enable the community to develop new methods for solving the task and advance the field of long-horizon robot task planning. Section 4.2 introduces a benchmark for evaluating the performance of SL systems. The benchmark includes an experimental protocol and six performance metrics. Section 4.3 discusses a baseline approach evaluated with the benchmark, showcasing its validity in both real-world scenarios with actual robots and the simulated environment. Section 4.4 concludes the chapter with a discussion on the results and the limitations of the proposed simulation environment and the benchmark.

The work from this chapter has been published in Luo and Demiris (2024). A supplementary video is available under this link. The open-source simulation environment can be found via this link.

4.1 THE SIMULATION ENVIRONMENT

To make the SL task accessible to a wider range of researchers, this section presents a SL simulation environment that replicates the real setup

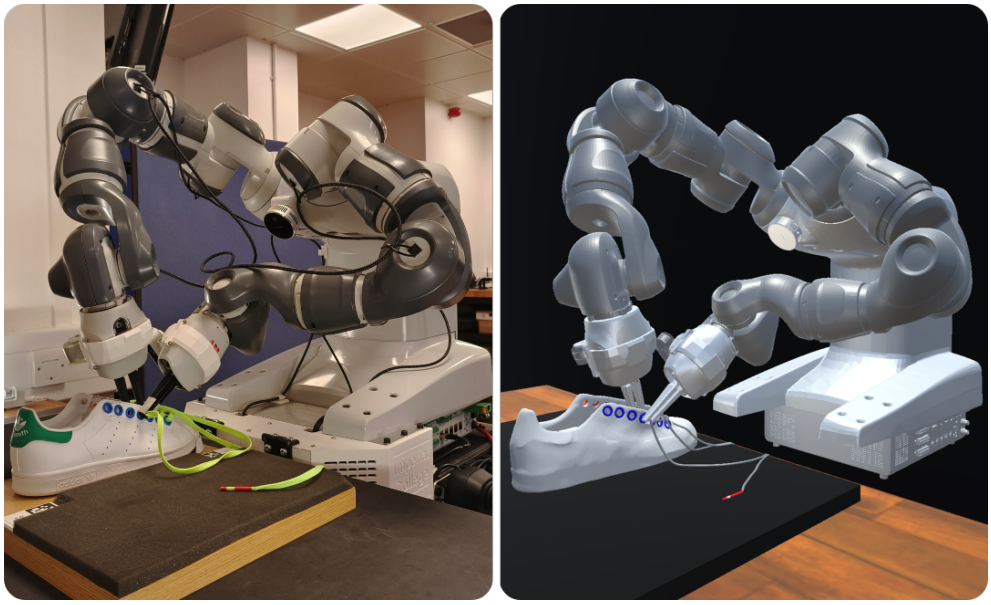


Figure 4.1: **Left**: Using the system proposed in Chapter 3 as the baseline, results and analysis are provided applying the proposed benchmark. The outcomes reveal substantial challenges that remain in effectively solving this task. **Right**: An exact replica of the real system in high-fidelity simulation is provided to bypass the hardware requirements and aid the development of data-driven methods. The interaction between the deformable and rigid components is considered while balancing the trade-off between real-time performance and simulation accuracy.

in Figure 4.1. Section 4.1.1 presents the environment and its components. Section 4.1.2 describes the simulation of deformable objects and collision handling. Lastly, Section 4.3.4 presents the experiment results with the baseline system.

4.1.1 *System Architecture*

The simulation environment consists of overall 3 parts (Figure 4.2). First, there are 4 building components of the *Unity* scene: the robot controllers, the shoe manager, the shoelace manager and the camera wrappers. The second part is the main process. It constructs the scene using the building components, monitors the execution and decides on the simulation outcomes. The third is the supporting packages outside *Unity*. They manage tasks such as reading the client parameters and exposing the simulated robot to MoveIt.

The shoelace manager generates the shoelace and configures the physical parameters such as length, friction, and stretching compliance. It monitors the collision and tension of the shoelace during execution. The shoe manager oversees the creation of the shoe and sets the parameters such as deformation

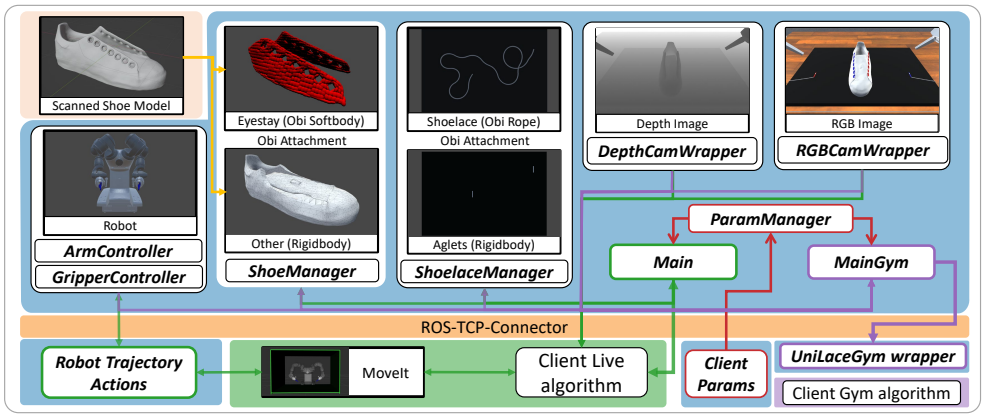
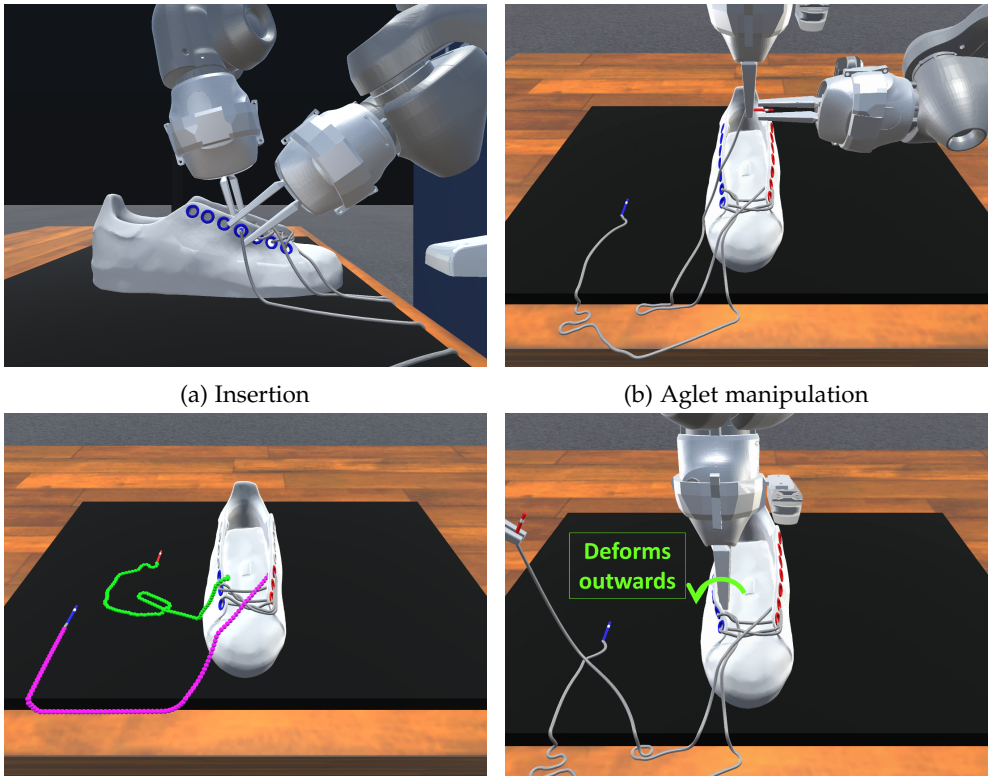


Figure 4.2: Structure of the simulation environment. *Unity* communicates with other modules through ROS-TCP-Connector. The simulation environment is controlled by a main process initialised with the parameter server. The robot is controlled through robot trajectory actions and the trajectories are planned with *MoveIt*. The simulation environment provides RGB and Depth images, as well as groundtruth information such as the aglet and eyelet poses.

resistance and friction. It also tracks the eyelet poses in runtime. The robot model includes joints and rigid bodies connected to two sets of arm and gripper controllers. The controllers execute position commands and publish the joint states. Controller parameters such as stiffness, damping and friction are configurable. Camera wrappers customise cameras using client-defined parameters such as poses, focus length and resolution.

The main process is where the rules of SL are defined. It automatically counts the eyelets that have been laced and declares success when finished. A service is provided to estimate the shoelace branch lengths afterwards, as shown in Figure 4.3c. The main process monitors 2 types of failures: The idle time defined in Section 4.2.2 being reached; the maximum shoelace tension has been exceeded for some continuous frames. The simulation is paused afterwards and outcome messages are logged on the screen. The proposed environment provides two options for the main process: as a live system with ROS topics and services, or as a sequential decision-making problem with the Gym wrapper.

Domain randomisation is also considered in this environment. Gaussian noises are added to the depth image to achieve more realistic depth readings. The environment also supports randomisation of the positional configuration of the shoe and shoelace, as well as the lighting intensities.



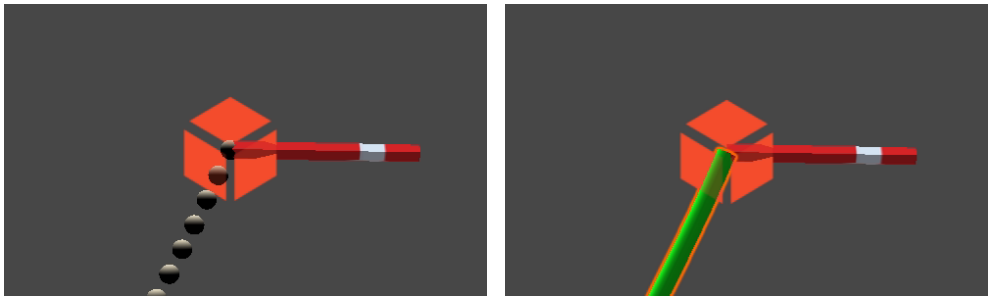
(a) Insertion (b) Aglet manipulation

(c) Branch length estimation (d) Eyestay deformation

Figure 4.3: Baseline algorithm operating in the simulation environment.

4.1.2 *Handling Deformation And Collision For Shoe Lacing*

One of the challenges in SL simulation is that the shoe and the shoelace are composed of parts with different materials. Simulating them with full deformability, while being more realistic, is computationally prohibitive. The shoe is separated into a deformable *Obi Softbody* eyestay and the rest *Unity Rigidbody*s. The eyestay deforms when manipulated as shown in Figure 4.3d. The two parts are connected with an *Obi Attachment* (Figure 4.4) where impulses are propagated through. Within the deformable part, a dense particle representation is generated to increase the robustness of the collision detection. The particles around eyelets are selected and binded to different groups, to avoid the deformation of the eyelet regions. Similarly, the shoelace is composed of the deformable part *Obi Rope* and the rigid aglets, connected with *Obi Attachment*. A dense particle resolution is also assigned to the deformable part to reduce the chances of missing particle contacts and use the surface collision detection pipeline from *Obi* to further improve the robustness of the simulation.



(a) Particle Renderer (b) Mesh Renderer

Figure 4.4: Obi Attachment between the soft rope and the rigid aglet. The end particle of the rope is rigidly attached to the aglet's collider. Hexagon symbol shows the attachment. Left: Rope with rendered particles; Right: Rope rendered as a continuous mesh along the particle chain.

The high particle density brings high computational demand when processing the particle contacts. *Burst* from *Unity* is used to multi-thread computation and allocate objects into different layers to reduce the collision detection complexity. For rigid objects, Simplified mesh models are employed for collision checking and high-resolution representations for rendering.

4.2 PROTOCOL AND BENCHMARK DESCRIPTION

To standardise the SL experiments and the comparison between different systems, this section proposes a protocol and a benchmark. The proposed protocol and benchmark follow the guidelines of the YCB protocols and benchmarks (Calli et al., 2015). The protocol describes the requirements and the procedure to reproduce the experiments, including descriptions of the standardised setup for the experiments (Section 4.2.1), and the experimental procedure (Section 4.2.2). The benchmark includes a set of evaluation metrics for the experiment results (Section 4.2.3).

4.2.1 Setup Description

The experimental setup consists of the shoe, the shoelace and the robot. The first property that characterises a shoe is its material. In SL, the shoe sole is not manipulated and therefore only the eyestay material is considered. This work considers 2 common types of shoe eyestays: hard leather-based materials and soft fabric-based materials. The second property of a shoe is the eyelets. From the material perspective, the most common type of eyelet is considered: the punched eyelets. They are made by punching holes directly

through the eyestay. The punched eyelets have two common types: reinforced with metal grommets and directly using the eyestay fabric. Regarding the number of eyelets, lacing a 2-eyelet shoe does not require a pattern, therefore a shoe must have at least 4 eyelets for the SL task to be valid. The number of eyelets varies with the size of the shoe. An average-size shoe can have 6-14 eyelets. Given the vast number of shoe brands, it is infeasible to list all shoes with all materials invented by different manufacturers. Specifying certain shoe models might also discourage researchers due to the unavailability of different shoe models in different regions. Instead, the experimenters are encouraged to freely choose shoe models and specify the model name, shoe size, eyestay and eyelet material, as well as the number of eyelets in their reports.

The most important characteristic of a shoelace is its length. Common lengths include 70cm for 6 eyelets, 90cm for 10 eyelets and 137cm for 16 eyelets. To guarantee enough workspace for the robot arms, 142cm flat shoelaces (Figure 4.5) are proposed to be used for all testing cases.

There are no restrictions on the initial configuration of the shoe and the shoelace and instead the experimenters are encouraged to explore innovative solutions. The initial configurations should be explicitly stated in the reports, alongside other system limitations and modification requirements. For example, whether the system requires specific modification and positioning of the shoe and the shoelace before lacing. Experimenters are encouraged to use different hardware platforms, and the hardware requirement should be clearly stated in the reports. This includes the robot, the operation speed and the control method, the gripping tools and the sensors. An example with the baseline system is provided in Section 4.3.1.

4.2.2 *Experimental Procedure*

During the experiment, the robot must operate autonomously without interference. A `lace` action is successful if the aglet has gone through the eyelet and pulled outside the shoe cavity. A trial is deemed successful if all eyelets have been laced and both manipulators are free. It is considered failed if the manipulator stops movement after 30 seconds with unlaced eyelets. After each experiment, the shoe can be manually unlaced and restored to the system-required initial configuration. The experiments on each shoe should be repeated M times ($M \geqslant 3$).

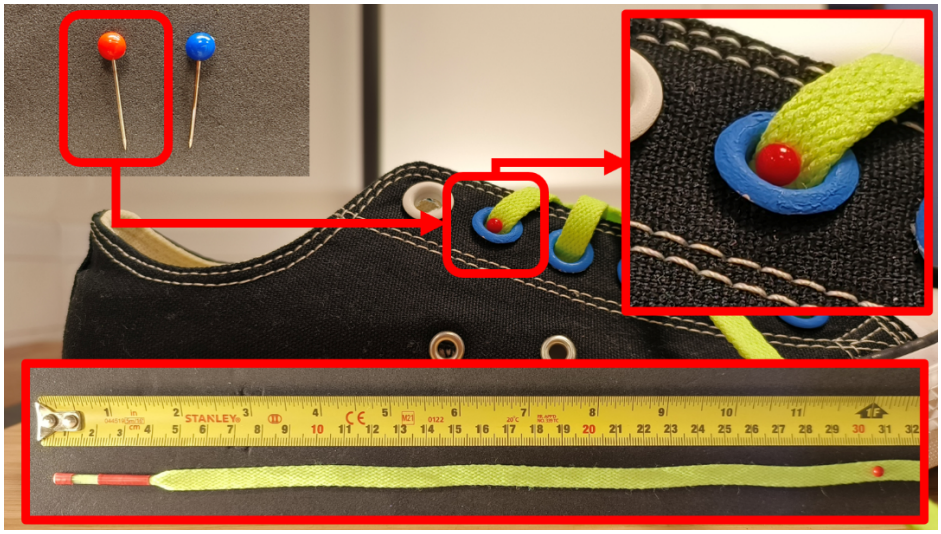


Figure 4.5: Demonstration on measuring the length of the shoelace branches. The top images show how the roots of the shoelace branches are marked. The bottom image gives a sample reading of the branch length.

4.2.3 *Evaluation Metrics*

To promote reproducible approaches and allow easy comparison between algorithms, the following metrics are proposed. All metric scores are averaged over M trials.

1. **Success Rate** (**SR**): The success rate is the number of successful trials M' divided by the total number of trials M:

$$SR = \frac{M'}{M}.$$

2. **Completion Rate** (**CR**): A more detailed criterion of the success of long-term manipulation systems is the completion rate, calculated as the number of laced eyelets $|E'_m|, m \in [1, M]$ divided by the total number of eyelets N:

$$CR = \frac{1}{M} \sum_{m=1}^{M} \frac{|E'_m|}{N}.$$

3. **Time Usage per Eyelet** (**TU**): The time usage $t_m, m \in [1, M]$ is measured from the start of the algorithm till the stop of the robot:

$$TU = \frac{1}{M} \sum_{m=1}^{M} \frac{t_m}{N}.$$

4. **Number of Grasps per Eyelet** (**NG**): The efficiency of the algorithms is also evaluated from the perspective of path planning by counting the number of grasps $G_m, m \in [1, M]$ that the end-effectors make during the experiment. The nature of the parallel grippers limits the dexterity of the movements and the robot may need to pick and place the shoelace constantly:
$$NG = \frac{1}{M} \sum_{m=1}^{M} \frac{G_m}{N}.$$

5. **Branch Discrepancy per Eyelet** (**BD**): Tying the shoelace after SL requires the two branches to have similar lengths. The length of the two branches $b_m, m \in [1, M]$ should be measured in a rest-stable state after the experiment and the discrepancy is the absolute difference between the two readings (Figure 4.5). This difference can be accumulated over the SL process and hence affected by the size of the shoes. Instead of reporting the raw difference, the protocol includes the discrepancy $|L_{b_{1m}} - L_{b_{2m}}|$ per eyelet,
$$BD = \frac{1}{M} \sum_{m=1}^{M} \frac{|L_{b_{1m}} - L_{b_{2m}}|}{N},$$
where L_b is the length of the branch b.

6. **Shoelace Overrun** (**SO**): With the initial position of eyelets, the shoelace length required for a shoelace pattern can be calculated. This is achieved by aggregating the length of the individual segments. The ratio of the actual shoelace cost to the shoelace budget reflects how tight the shoe is laced. While a tightly laced shoe often presents a tidy appearance and may benefit marketability, it is also more difficult to put on. This shoelace overrun is calculated by
$$SO = \frac{L_l - L_{b_{1m}} - L_{b_{2m}}}{\sum_{i=1}^{N-1} L_{s(e_{\sigma(i)}, e_{\sigma(i+1)})}} - 1,$$
where L_l is the length of the shoelace.

On top of the metrics, the following failure modes are observed. They can be optionally reported to identify the algorithm shortcomings.

1. **Motion planning failure**: Motion planner fails to generate a trajectory. This may occur when the gripper is holding an aglet between the eyestays.

2. **Collision**: Unrecoverable robot collision against the shoe. This can also be caused by overstretching the shoelace.
3. **Grasping failure**: Grippers fail to grasp the aglet. For instance, the aglet slips from the fingers due to insufficient grasping force, or the grasped region is not enough to generate sufficient friction.
4. **Perception failure**: The perceived object state is missing or incorrect. For example, when shoelace branches occlude the aglets and eyelets.
5. **Insertion failure**: The aglet fails to enter eyelets. This is an accumulative error from perception and actuation.

4.3 THE BASELINE SYSTEM

To demonstrate the usage of the benchmark and provide results for comparison, the performance of the baseline system in this section is reported. First, the setup used in the system is introduced (Section 4.3.1). Then the structure of the baseline is provided (Section 4.3.2). The performance of the system is analysed in the end (Section 4.3.3).

4.3.1 The Baseline Setup

The experiments are performed using an ABB YuMi with positional control and a target End-Effector (EE) velocity of 400 mm/s. Two Realsense D435s are attached to the grippers; a Realsense L515 is mounted on the front of the robot. The shoe is rigidly attached to the table 45cm in front of the robot and the aglets are placed on both sides of the shoe while the fabric part of the shoelace is between the shoe and the robot (Figure 4.6). The shoes used have punched eyelets in Table 4.1. Due to the limitation of the vision module, the baseline system is not able to perform experiments on shoes without reinforced eyelets. The eyelets and aglets are distinctively coloured, with the eyelet diameter enlarged from 5 mm to 8 mm and the aglets elongated from 15 mm to 30 mm.

Table 4.1: List of shoes used in the benchmark

Type	Stiffness	Material	Rep. Model	Size	# Eyelets
T1	Hard	Leather	Adidas Stan Smith	6.5	14
T2	Soft	Fabric	Converse All Star	8.0	12

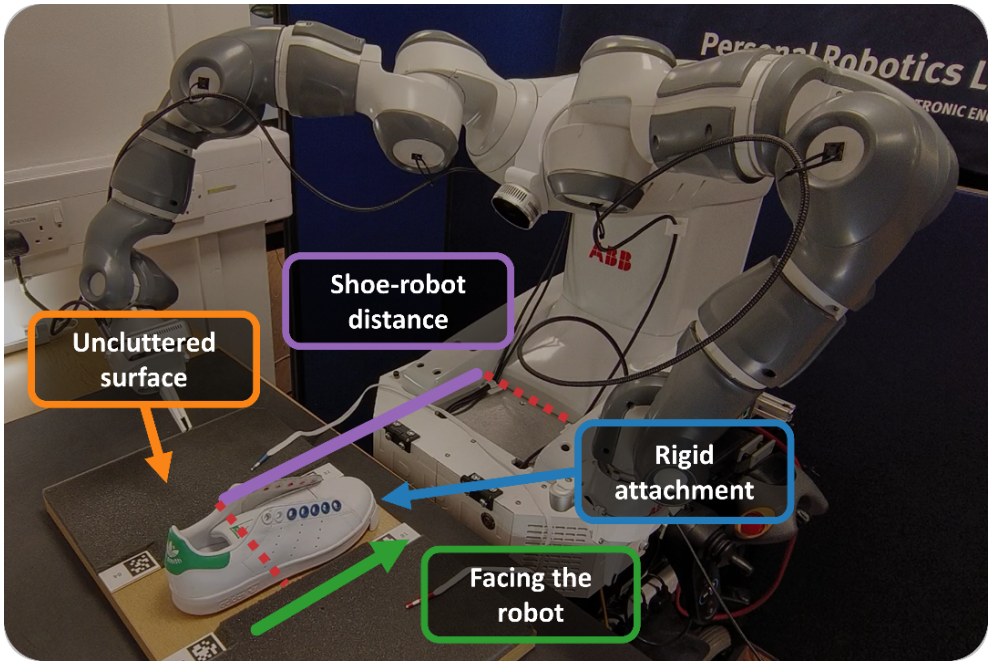


Figure 4.6: Initial configuration for the baseline system. The shoe is fixed on a flat surface with the toe box placed towards the robot. The aglet and eyelets are available in the workspace and visible from the camera default positions. In this trial, the top 2 rows of eyelets are covered to simulate a 10-eyelet shoe.

4.3.2 *The Baseline Structure*

The baseline system is described in Chapter 3. It comprises a vision module, a pattern designer, a primitive planner and a set of action primitives (see Figure 4.7). In the initial phase, the vision module scans for the eyelet poses with colour thresholding. 4 box corners are selected between the inner and outer contours of an eyelet. The pose of the eyelet is then represented by the corresponding pose of the box. The pattern designer then computes the optimal pattern with a genetic optimiser based on user requirements. The optimiser is required to generate a dense pattern that maximises tension over the horizontal direction. The crosses generated are set to follow top side mode 1. This pattern is passed onto the sequence planner which contains a set of symbolic representations of the SL scene. A *LAMA* (Richter and Westphal, 2010) solver leverages Planning Domain Definition Language (PDDL) representation of the problem and determines the order of the primitives to be deployed. There are 6 action primitives for the SL task: **lace**, **transfer** and **re-place** from both left and right sides. The **lace** only supports lacing inwards. Finally, the execution manager executes the complete process.

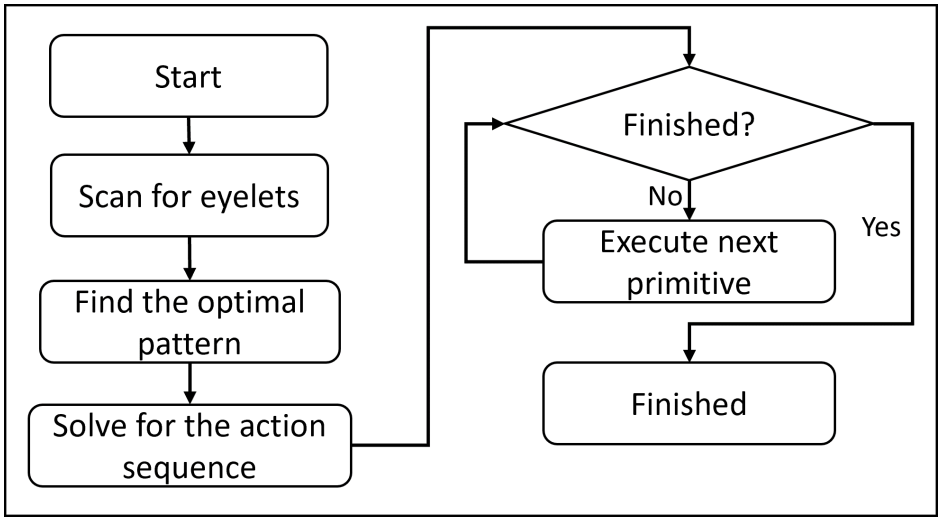


Figure 4.7: Structure of the baseline algorithm (based on the system proposed in Chapter 3).

4.3.3 *The Baseline Results*

To test the performance under different numbers of eyelets, parts of the eyelets are covered to simulate shoes with 6, 8, 10, 12 and 14 eyelets respectively using shoe 1 and shoe 2. 3 trials are conducted for each scenario.

As shown in Table 4.2, the completion rate reduces as the number of eyelets increases. Overall, the baseline system achieved better results with shoe 1. The completion rates are similar when the shoes are simulated to have 6 and 8 eyelets at around 90%, and become around 25% lower for shoe 2 when the shoes have 10 and 12 eyelets. This is due to the deformation of the shoes in repeated lacing actions. When the robot stretches the shoelace, the eyestays move to each other. The baseline system does not consider the shoe deformation, meaning that it cannot capture and compensate for this change. Meanwhile, the deformation is also affected by the eyestay material. The fabric eyestay of shoe 2 allows more plastic deformation, exacerbating this issue. As a result, the shoelace branches become shorter than expected. The shoelace branch tension is eventually higher than the friction between the aglet and gripper, and the shoelace slips from the fingers.

The time usage and number of grasps did not show clear patterns across the testing cases. It takes the baseline 72.83s and 5.56 grasps to lace an eyelet. The average branch discrepancy and the shoelace overrun are smaller with more eyelets, showing that the difference and the overrun generated by the baseline system are not dependent on the number of eyelets.

Table 4.2: Performance of the baseline system.

Shoe	# eyelets	SR	CR	TU (s)	NG	BD	SO
S1	6	2/3	0.94	75.59	5.51	3.87	0.4935
	8	2/3	0.88	72.39	5.57	0.20	0.4924
	10	2/3	0.97	72.44	5.72	0.34	0.2796
	12	0/3	0.81	69.47	5.58	n/a	n/a
	14	0/3	0.62	72.64	5.55	n/a	n/a
S2	6	2/3	0.94	73.18	5.51	1.08	0.3616
	8	1/3	0.92	73.65	5.54	0.28	0.1994
	10	0/3	0.73	76.29	5.54	n/a	n/a
	12	0/3	0.56	70.15	5.46	n/a	n/a
Sim	6	2/3	0.83	122.57	5.33	12.29	0.5171
	8	2/3	0.71	122.92	5.17	9.66	0.3005
	10	2/3	0.90	132.56	5.68	13.90	0.3091
	12	0/3	0.44	132.24	5.38	n/a	n/a
	14	0/3	0.43	119.34	5.06	n/a	n/a

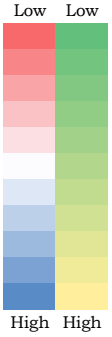


Table 4.3: Failure modes occurrence ratio

Shoe	Mot. Plan.	Collision	Grasping	Perception	Insertion
S1	0.11	0.11	0.56	0.11	0.11
S2	0	0	0.78	0.11	0.11
Sim	0.11	0.56	0	0.22	0.11

Overall the baseline system can complete SL in certain tested scenarios with modified shoes and shoelaces. There is significant room to improve the speed and success rate and remove the modification requirements. Although the baseline is restricted to a specific initial configuration, experimenters are encouraged to test different setups and hardware platforms.

4.3.4 *Baseline Experiments In Simulation*

The baseline experiments are conducted with simulation on a platform with an AMD Ryzen Threadripper Pro 5965WX and an Nvidia RTX 4090[1]. The measurable parameters such as shoelace length and aglet length are configured the same as in the baseline experiments. Other parameters such as damping and stretching compliance are set empirically to maximise the visual resemblance with the real system. The camera depth error rates are modelled after the real cameras described in Section 4.3.1. The shoe model is constructed from a 3D scan of shoe 1, and the robot model is constructed from YuMi's URDF[2]. Figure 4.3a and Figure 4.3b show sample images of the process.

1 The simulation environment is also tested functional on an Intel 8700K with a GTX 1080, and an Intel 10875H with an RTX 2070.

2 YuMi's URDF from https://github.com/kth-ros-pkg/yumi

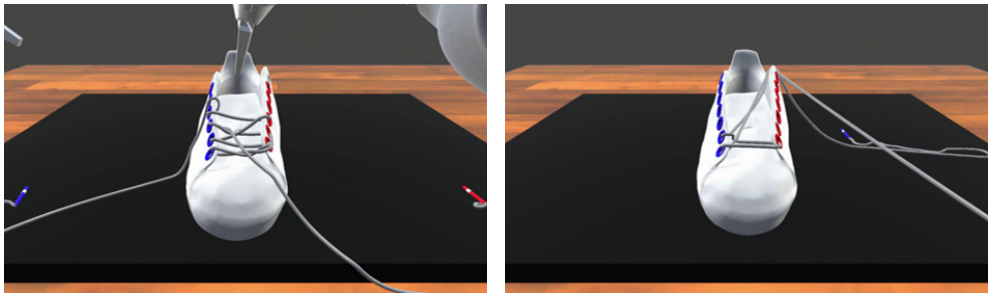
The experiment results are shown in Table 4.2. While generally consistent with real-world experiments, due to the disparity between simulation and reality, the results exhibit some different properties. The baseline system demonstrated an overall similar completion rate in simulation. However, the failure reasons show a different distribution (Table 4.3). Notably, 56% of simulated failures happened because of collision and shoelace overstretching. As mentioned in Section 4.1.1, overstretching leads to the shoelace tension exceeding the maximum threshold, which is recognised as a collision failure. This shifts the grasping failures to the collision failures. Regarding the time usage, the baseline shows a longer completion time in the simulation. Physics engines typically separate tasks such as collision detection, state updates, and scene rendering into sequential phases, which limits the achievable frame rate. Consequently, slower camera updates result in delayed observations, which lead to a longer completion time.

4.3.5 *Difference With The Reality*

3 simulation and reality differences are observed:

- **Collision response:** Unlike in reality, where friction secures the aglets between gripper fingers, in simulation, they are connected using fixed joints. During insertions, with the small contact region, aglets may penetrate through the eyestay. To prevent the simulation from failing, an aglet-eyestay collision response mechanism is implemented. The main process starts monitoring the collision distance once it turns negative. Penetration is confirmed if the contact persists over a series of consecutive frames and the aglet is subsequently detached from the gripper. While this mirrors a typical outcome in real-world scenarios, it does not cover all possible outcomes.

- **Overstretching failure:** Apart from normal collision failures (Figure 4.8a), some other failures are also categorised as collision in simulation. When the shoelace undergoes excessive stretching beyond the tolerance of finger friction, aglets cannot be pulled out, which can lead to the shoelace breaking out from the eyelets and breaking the simulation. This scenario may result in either the gripper being trapped by the shoelace, or the aglet being disengaged. Due to the inability to distinguish the two outcomes, shoelace overstretching is categorised as a collision failure (Figure 4.8b).



(a) Collision (b) Overstretching

Figure 4.8: Failure cases in simulation. Due to discrepancies between simulation and reality, the simulated results exhibit a higher number of collision failures.

- **Behaviour of the eyestays:** The eyestays do not demonstrate the same level of deformability when interacting with the shoelace as they do with the grippers. Given the particle nature of the eyestays, the shoelace particles require sufficient mass to exhibit influence on them. However, when this happens, the eyestays are not the first to deform. Instead, the particles near the eyelets deform first, resulting in the shoelace escaping from the eyelets.

4.4 CONCLUSION AND DISCUSSION

In this chapter, a benchmark and a simulation environment for bi-manual robot SL is proposed to facilitate further research into related task planning algorithms. A comprehensive protocol for the reproduction of the experiments is provided, alongside 6 metrics for algorithm evaluation. The performance of a baseline system is analysed with the proposed criteria. The results show that the baseline achieved over 70% completion rate across all tested scenarios with less than 10 eyelets. The proposed simulation environment provides reliable collision checking for the task and easy-to-use interfaces.

From the experiments, several potential research directions in the domain of SL are identified. First, improving the perception of shoes, eyelets, and shoelaces is essential, as the baseline system relies on specially coloured, reinforced components and does not track the shoelace or shoe deformation, limiting its success rate. Second, achieving accurate shoelace insertion without enlarged or modified eyelets and aglets remains a challenge, as bimanual SL with standard components has not yet been demonstrated. Lastly, physical parameter estimation for realistic simulations is another open area, as the provided environment lacks physics parameters calibrated to specific shoes or shoelaces, a topic explored more generally in the literature (Caporali et al.,

2024; Lim et al., 2022). This coincides with the limitation of Tracking for Shoe Lacing (TSL) proposed in Chapter 5. Furthermore, recent advances in differentiable simulators, such as ChainQueen (Hu, Liu, Spielberg, Tenenbaum, Freeman, Wu, Rus and Matusik, 2019) and DiffTaichi (Hu, Anderson, Li, Sun, Carr, Ragan-Kelley and Durand, 2019), offer a new direction for Deformable Object (DO) simulation. These tools could be applied to the SL task in future work to accelerate the development of learning-based manipulation policies. More discussion on the physics parameter estimation and key component perception is provided in Chapter 7.

5

TRACKING DLOS WITH PHYSICAL SIMULATION

Results from Section 3.6.5 indicate the importance of tracking the states of the DLO during manipulation. As discussed in Section 1.1.3, DLO commonly interact with the surrounding environment in real-world applications and the understanding of its relation with the scene objects affects the task planning and execution. The research question this chapter addresses is:

> *How can a robot perceives the states of a DLO and correctly understand its relation with the scene objects?*

This chapter proposes an algorithm, Tracking for Shoe Lacing (TSL), to track the states of the shoelace while considering its topological relation with the scene objects. Section 5.1 provides an overview of the algorithm. The algorithm combines probabilistic registration with physical simulation to enforce the physical and topological constraints. Section 5.2 presents the statistical registration module that estimates the shoelace states with new observations. Section 5.3 introduces the simulation environment and the physical constraints used to regulate the states. The physical simulation also monitors the system for anomalies such as excessive shoelace tension or insertion failures. To evaluate the performance of TSL, Section 5.4 proposes a representation, *Sequential H-Signature*, to qualitatively showcase the correctness of the estimated shoelace-eyelet relation; as well as using the Dynamic Time Warping (DTW) distance to numerically compare the tracking accuracy. Section 5.5 compares the performance of TSL with existing methods (Tang and Tomizuka, 2022; Wang et al., 2021) using human-demonstrated data, simulation data, and real robot data with a system shown in Figure 5.1. Section 5.6 discusses the results and the limitations of the proposed algorithm. The results show that TSL achieves tracking distance error on a par with the state-of-the-art algorithm, and correctly estimates the shoelace-eyelet relation in all tested cases while the other algorithms do not produce the correct *Sequential H-Signatures*.

The work presented in this chapter has been accepted for publication in IEEE Robotics and Automation Letters (RA-L).

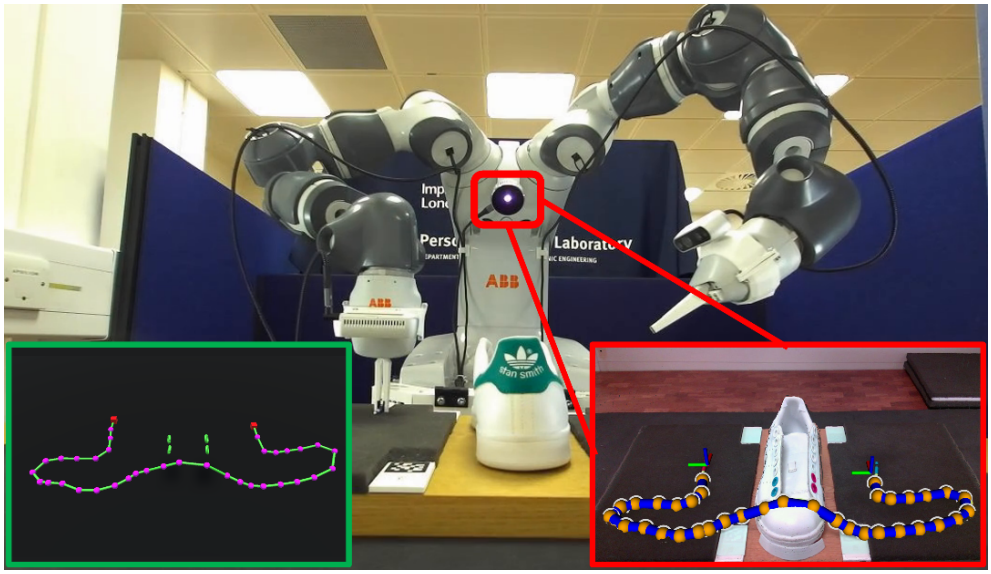


Figure 5.1: TSL for shoelace tracking in robot SL. Leveraging statistical registration and physical simulation, TSL tracks shoelaces while preserving the topological relations between the shoelace and the shoe. The DLO shoelace is represented by a set of ordered key nodes and the edges connecting adjacent points. The bottom left figure shows the physical simulation constructed for TSL, including the tracked shoelace and the eyelets. The estimated shoelace states are projected to the image space in the bottom right for illustration purposes. The spheres indicate the tracked key nodes.

5.1 STRUCTURE OF TSL

As discussed in Section 1.1.3, there are two types of topology that TSL considers: the topological property with respect to the DLO itself; as well as the properties with respect to the scene objects. In the context of SL, the DLO is represented by the shoelace, and the scene object corresponds to the shoe. TSL achieves this with 2 components:

- The registration module which estimates the Gaussian centroids and variance given a new frame (Section 5.1). This module preserves the topology of the DLO with respect to itself.
- The physical simulation that maintains a SL environment and constantly updates its internal shoelace, End-Effector (EE) and eyelet states based on the registration results (Section 5.3). This module preserves the topology of the DLO with respect to the scene objects and detects system anomalies (Section 5.3.2).

TSL maintains the states of three components: the EEs, the shoelace, and the eyelets. The shoelace is modelled as a series of key nodes, among which

the manipulated ones follow the movement of the EE. As discussed in Section 2.5.3, the shoes are represented with their eyelets in TSL. The eyelets are modelled as closed-loop points in 3D space with the pose derived from eyelet detection.

The state updates of the three components are performed in parallel to improve efficiency (Algorithm 1).

- Prediction: the update of the EE states q.
- Adjustment: the update of the shoelace state represented by the key nodes $Y = [y_1, \ldots, y_M] \in \mathbb{R}^{M\times 3}$ based on new observed point cloud $X = [x_1, \ldots, x_N] \in \mathbb{R}^{N\times 3}$, where M is the number of key nodes and N is the size of the point cloud.
- Correction: the update of the eyelet states Z.

Algorithm 1: The Tracking for Shoe Lacing algorithm

Input: New Observation
Output: Shoelace States
do prediction in parallel
  Update EE target q from the observation
  Calculate forces to update simulated EEs
  Apply forces to EEs in simulation
  Estimate shoelace tension
do correction in parallel
  Detect eyelet poses Z
  Update Z in simulation
  Check shoelace-eyelet relations
do adjustment in parallel
  if *Selected as key frame* **then**
    Get simulated states $\tilde{Y}$
    Register states Y to new observation X
    Calculate particle regularisation forces
    Apply forces to shoelace particles in simulation
return *states*

The registration function receives new observations and decides if this frame should trigger adjustment. It then applies for registration and estimates the new shoelace states, which are consecutively sent to the physical simulation for adjustment. Meanwhile, the EE states are updated in the simulation based on the robot commands; and the scene object states are updated based on the detected poses. The simulation environment constantly monitors

the collision between the shoelace and eyelets and determines whether an insertion is successful.

5.2 STATISTICAL REGISTRATION

Following the previous works discussed in Section 2.5.2, this work considers the problem of estimating shoelace states with new observation as non-rigid registration and uses an Expectation-Maximization (EM) algorithm to estimate the new states. The Gaussian centroids are represented by the key nodes $y_m (m \in 1, \ldots, M)$ and the observed point cloud is considered as the sampled points from the Gaussian Mixture Model (GMM). The probability distribution of $x_n (n \in 1, \ldots, N)$ is:

$$p(x_n) = p(m)p(x_n|m). \tag{5.1}$$

where $p(m)$ is the membership probability of each Gaussian and $p(x_n|m)$ represents the distribution of the mth Gaussian. Consider that the registration is in a D dimensional space. Based on Bayes' theorem, the posterior probability distribution with the old (x_n, σ^2) is updated by the E-step:

$$p(m|x_n) = \frac{\exp\left(-\frac{\|x_n - y_m\|^2}{2\sigma^2}\right)}{\sum_{m=1}^{M} \exp\left(-\frac{\|x_n - y_m\|^2}{2\sigma^2}\right) + \frac{(2\pi\sigma^2)^{\frac{D}{2}}}{(1-\mu)N}}. \tag{5.2}$$

To maximise the log-likelihood of the observation, the cost function in M-step is defined as:

$$Q_{GMM} = \sum_{m=1}^{M} \sum_{n=1}^{N} p(m|x_n) \frac{\|x_n - y_m\|^2}{2\sigma^2} + \frac{N_p D}{2} \log(\sigma^2), \tag{5.3}$$

where $N_p = \sum_{m=1}^{M} \sum_{n=1}^{N} p(m|x_n)$.

To regularise the DLO structure, a cost term is used for cross-frame consistency with Coherent Point Drift (CPD), and a term for centroid movement consistency with Locally Linear Embedding (Ge et al., 2014). The final cost function to minimise is given by

$$Q = Q_{GMM} + Q_{CPD} + Q_{LLE}. \tag{5.4}$$

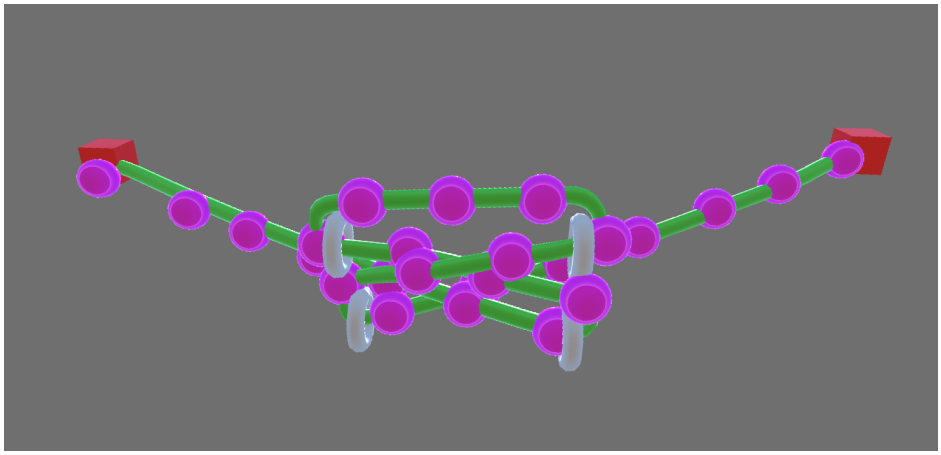


Figure 5.2: Physical simulation environment of the proposed TSL. The simulated shoelace correctly passes through every eyelet. The shoelace and the shoe structure are simplified to speed up the simulation. The eyelets are represented by rings of particles. The spheres show the target key node positions and regulation forces are applied to adjust the shoelace particles towards them. The boxes indicate the EEs.

5.3 PHYSICAL SIMULATION

5.3.1 *Simulation Environment*

To maintain the topology of the DLO with respect to the scene objects, a physical simulation environment is constructed. The environment maintains the states of the EE, the shoelace, and the eyelets. The environment detects collisions between the shoelace and the eyelets and prevents the shoelace from penetrating the eyelets. The physical model of the shoelace also includes physics constraints such as stretching, friction and damping. Meanwhile, the simulation environment keeps the complete states of the shoelace and monitors system anomalies.

The states estimated by the registration module are statistically reasonable, but not restricted by physical laws. TSL uses a physics engine to regulate the states. Equivalent to the posterior constraints in Constrained Deformable CPD 2 (CDCPD2), the simulation environment of TSL includes physical constraints. Known correspondence points are restricted and treated as rigid kinematic bodies. Stretching constraints for the shoelace are added by minimising the displacement between particles along the inter-particle direction. Collision constraints are added by adjusting predicted contact points between a shoelace edge and other edges or objects. Apart from the constraints used in CDCPD2, bending constraints, friction and damping are also added. Using

physics simulation expands the physics regulations that can be applied to the shoelace and allows physically interpretable parameters.

Due to the nature of physical simulation, the states of simulated objects cannot be changed instantly. Abrupt robot movements or shoelace state changes cause significant interruptions of the simulated physics. Therefore, updating signals are converted to regulation forces and pull the components into desired states. To avoid sudden drastic motion, a PD force controller is used to regulate the movement. This reduces the *tunnelling effect* where the simulated shoelace penetrates through other objects. The simulated rope states are also regulated by PD force controllers.

During robot SL, the shoelace often remains stationary. The robot may place and re-grasp the shoelace to adjust its arm configuration. Tasks such as eyelet perception and motion planning contribute to the stationary periods. Meanwhile, if the registered shoelace configuration did not change noticeably due to slow robot motion or occlusion, it is not necessary to adjust the simulated states for every frame. To this end, this work introduces the notion of keyframes. A keyframe is when the simulation updates the shoelace configuration. As the shoelace is represented with a GMM, the cost function Q_{GMM} reflects the quality of the estimated states to the current frame. A keyframe is selected using its matrix form

$$\begin{aligned}\overline{Q}_{GMM} = &\frac{1}{\sigma^2 N}(\mathrm{Tr}(X^T d(P^T 1)X - 2\,\mathrm{Tr}((PX)^T Y) \\ &+ \mathrm{Tr}(Y^T d(P1)Y))),\end{aligned} \tag{5.5}$$

where $\mathrm{Tr}(A)$ is the trace of a matrix A, and $d(a)$ is a diagonal matrix constructed from a vector a. A frame is selected when the cost is over an empirical threshold, depending on the fidelity of the simulated model. This reduces unnecessary updates and computation.

5.3.2 *Anomaly Detection*

The simulation environment provides complete states of simulated objects. In the case of SL, the physics engine is able to continuously monitor the relative positions of the shoelace and the eyelets; as well as the tension of the shoelace.

The obstacle coordinate systems are defined so that the eyelet skeleton lies in XZ planes and Y is the vertical axis. The shoelace particles are represented in the eyelet spaces. A shift in the sign of the Y coordinate of the particles indicates crossing over the XZ plane. An eyelet is passed through if this

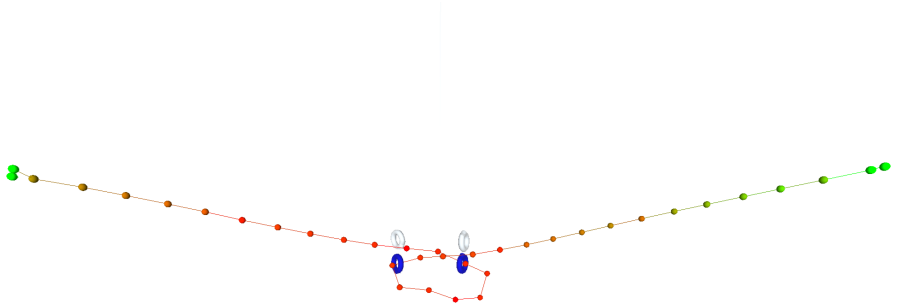


Figure 5.3: Anomaly monitoring during an experiment. The blue obstacles have been passed through. The red region indicates higher strain.

crossing lies within the skeleton on the XZ plane. The shoelace configuration Y that passes through an eyelet Z is represented as

$$Y_{passed} = \{\exists y_m \in Y \mid y_{m,y} \cdot y_{m-1,y} < 0, y_m \in \text{int}(Z)\}, \tag{5.6}$$

where $\text{int}(Z)$ is the interior of the eyelet Z.

Meanwhile, the physical simulation monitors the tension of the shoelace, represented by the strain along the tensile axis. This is calculated by

$$\text{Strain}(y) = \frac{\|y_{m+1} - y_m\|}{\|y^0_{m+1} - y^0_m\|}, \tag{5.7}$$

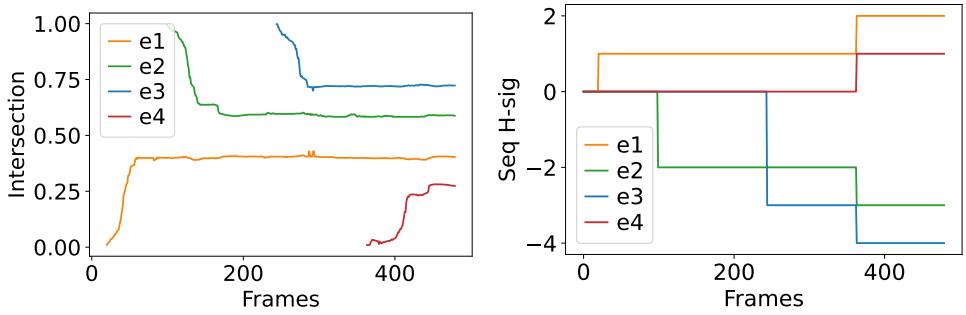
$$\text{Strain} = \sum_{m=1}^{M} \text{Strain}(y_m). \tag{5.8}$$

The particle information within the physics engine can be easily retrieved and used for monitoring. Figure 5.3 shows the strain of the shoelace during an experiment. The red region indicates higher strain.

5.4 EVALUATION METRICS

5.4.1 *Sequential H-Signature*

As discussed in Section 2.5.3, *H-Signature* has been widely used to categorise trajectories in motion planning in the presence of scene objects. Although it is designed to capture the topological relation to the scene objects, it does not provide the sequential information of the relations. For tasks such as



(a) Intersections on the shoelace with eyelets (b) *Sequential H-Signature* of the shoelace

Figure 5.4: Example of intersections on the shoelace with the eyelets and *Sequential H-Signature* of the shoelace. The intersections range from 0-1 with 0 and 1 indicating the two ends of the shoelace. At frame 500 of this example, the shoelace in sequence passes through *e4, e1, e2, e3*. The signature gives the index of an eyelet within this sequence and its sign indicates the direction of the passing. At frame 500, the *Sequential H-Signature* is $[2, -3, -4, 1]$.

SL, the order of obstacles that the DLO passes through is crucial for the success of the task. To evaluate the correctness of the estimated shoelace-eyelet relations, this work proposes a new form of representation namely *Sequential H-Signature* (Figure 5.4).

Suppose a scene object skeleton S is a current conductive wire with unit current and $B(y)$ is its Virtual Magnetic Field over a point y on the DLO. Then the Magnetic Field

$$B(y) = \int_S \frac{(\vec{s} - \vec{y}) \times d\vec{s}}{\|\vec{s} - \vec{y}\|^3}. \tag{5.9}$$

H-Signature $\mathcal{H}$ is defined as the line integral of the Virtual Magnetic Field along the DLO:

$$\mathcal{H}(S) = \int_Y B(y) \cdot dy. \tag{5.10}$$

If the absolute *H-Signature* value is over 0.5, the DLO is considered to pass through the scene object. The key node y that produces the maximum Virtual Magnetic Field is the closest point y_{m^*} to the skeleton:

$$y_{m^*} = \begin{cases} \arg\max_y B(y), & |\mathcal{H}(S)| > 0.5 \\ 0. & \text{otherwise} \end{cases} \tag{5.11}$$

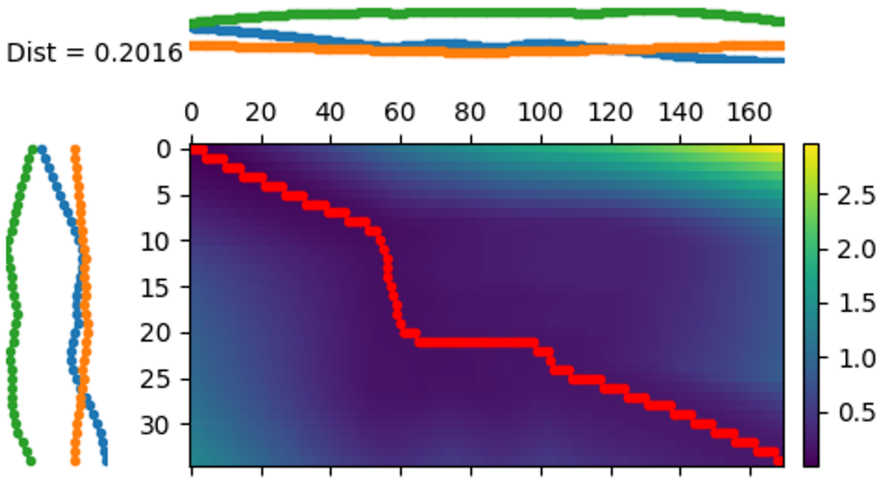


Figure 5.5: DTW between estimated states and the groundtruth at a frame. The plot shows the pointwise distances between nodes from the two sources. The DTW distance represents the minimum alignment cost between the two state sequences. The red diagonal line illustrates the node correspondences between the two sequences that result in the computed DTW distance.

Suppose $M^*_{sorted} \in \mathbb{N}^R$ is the list obtained by sorting $M^* \in \mathbb{N}^R$ in descending order. Define $\mathcal{H}_{seq} \in [1, R] \cap \mathbb{N}$ as the rank of each m^* in M^*_{sorted},

$$\mathcal{H}_{seq} = \{\text{rank}(m^*) \mid m^* \in M^*\}. \tag{5.12}$$

Sequential H-Signature $\mathcal{H}_{seq}$ provides information on both the state of the DLO with respect to the scene object and the order in which the scene objects are encountered. An example is provided in Figure 5.4.

5.4.2 *Dynamic Time Warping Distance*

To evaluate the similarity of the estimated states and the groundtruth states, this work proposes to use DTW distance to evaluate the similarity of estimated states and groundtruth states. The DTW distance is a measure of similarity between two sequences that may vary in time (Figure 5.5). In the case of DLO tracking, the groundtruth sequence Y and the estimated sequence $\hat{Y}$ often differ in the number of nodes and may not have one-to-one correspondence. The DTW distance is defined as the minimum cost of aligning the two sequences. The cost of aligning two points is the Euclidean distance between them. The DTW distance is defined as

$$\text{DTW}(Y, \hat{Y}) = \min_{\pi} \sum_{(i,j)\in\pi} \|y_i - \hat{y}_j\|, \tag{5.13}$$

where π is the warping path that minimises the cost.

5.5 EXPERIMENTS

In this section, the performance of TSL is evaluated and compared against CDCPD2, TrackDLO, and the classic Global-Local Topology Preservation (GLTP) algorithms. The testing data is acquired from a bimanual robot, a simulated SL scene (Figure 5.6), and human demonstrations. The simulated data is used to verify the tracking accuracy using DTW. An analysis sweeping through ranges of physical parameters is conducted to examine their effects on the accuracy. In addition, a separate simulation dataset is used to study how tracking accuracy changes with an increasing number of eyelets.

5.5.1 Implementation

Accurately detecting and managing collisions between the shoelace and eyelets, as well as self-collisions of the shoelace, is challenging due to the *tunnelling effect* Redon et al. (2002). At sufficiently high speeds, shoelace particles can penetrate through eyelets between consecutive frames. This work constructs the simulation environment with a multi-purpose game engine Unity[1] and a particle-based physics solver Obi[2] specialised for soft objects. Obi surface-based collision effectively reduces the *tunnelling effect* and provide reliable collision detection. To efficiently communicate between the simulation and the registration module with low latency, we use the cross-language communication Google Remote Procedure Call (gRPC) framework. The server is implemented in C++ with the registration module and the client in C# with the simulation environment.

TSL assumes that the shoelace is distinctively coloured and free of tangles or occlusions at the beginning. This simplification could be relaxed in future work using data-driven methods Lv et al. (2023). Point clouds corresponding to the shoelace areas are segmented using colour thresholding. The initial states are estimated by extracting the skeletons of the mask and bridging the segments Keipour et al. (2022); Xiang et al. (2023). All tested algorithms use the same initial states. In TSL, the estimated initial states are used to define the control particles and construct the Obi Rope. The EEs are simulated with points rigidly connected to the first and last particles of the shoelace.

1 Unity Engine: https://unity.com/
2 Obi: https://obi.virtualmethodstudio.com/

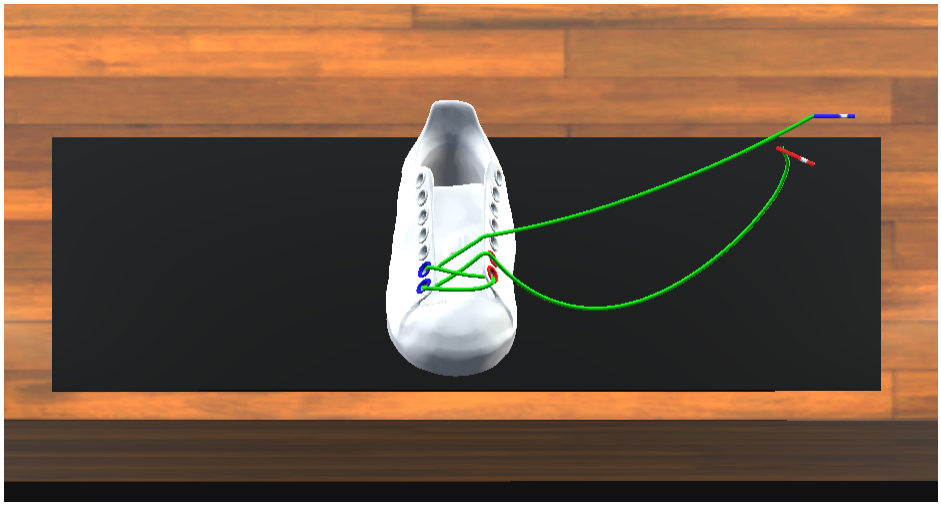


Figure 5.6: Simulation environment used to collect simulated data. More details are discussed in Chapter 4.

The eyelets are represented by a ring of interconnected particles with their normals pointing towards the inside of the shoe.

The simulated data is collected from a SL simulation environment (Figure 5.6) (Luo and Demiris, 2024). The eyelet, aglet and shoelace particle positions are recorded as groundtruth. The shoe model is a 3D scan of the real shoe used in human demonstration. The other shoe from the same pair is used in robot data collection. The robot data is collected with the system proposed in Luo and Demiris (2023), where a dual-arm ABB YuMi IRB 14000 is used to complete the task. The system combines PDDL-based high level reasoning for overall strategy planning and multimodal low-level sensorimotor primitives to execute the low-level actions. The eyelet poses detected by the system and EE poses computed from the joint states are recorded. For the human demonstration data, a single participant was involved in the data collection. The eyelet poses and EE positions were manually annotated. Both the physical robot data and the human demonstration are recorded with Intel Realsense L515s. 4 eyelets are laced and RGBD images are recorded in all cases. Data is recorded as *ROS Bags* and the algorithms are tested on a platform with an Intel 8700K and an Nvidia GTX 1080.

5.5.2 *Qualitative Analysis With Sequential H-Signature*

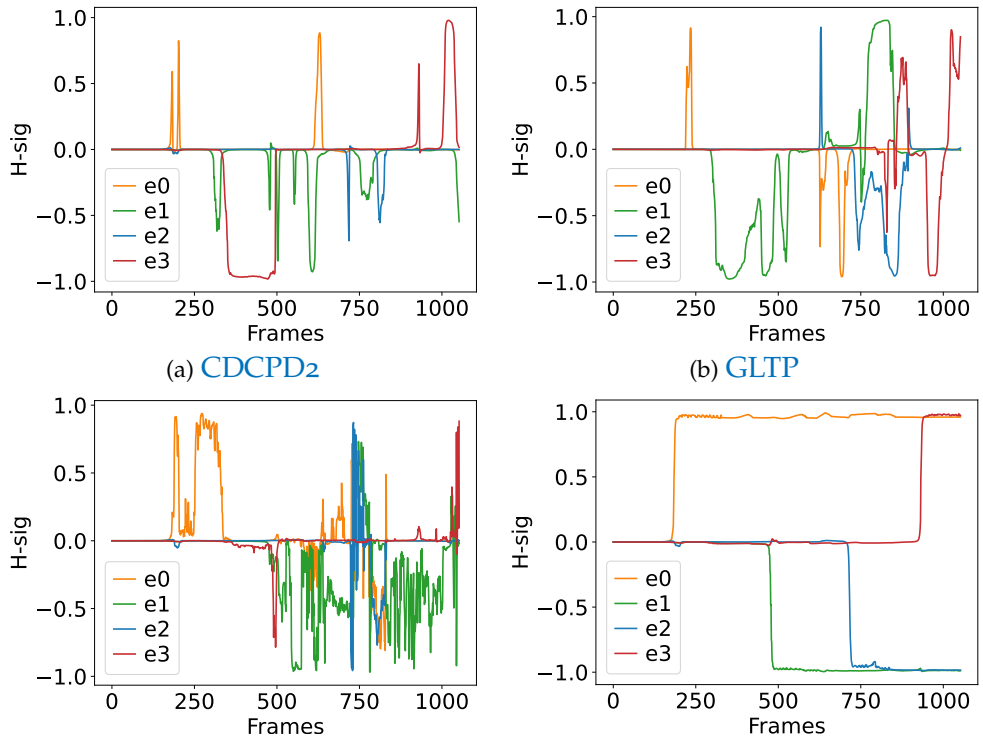
Figure 5.7 shows the results after lacing the second and fourth eyelets with all three data sources. Each section includes: (1) the frame image overlaid by estimated states; (2) a close-up view of the eyelet region (top right); (3) a 3D



Figure 5.7: Tracking results after lacing the second and fourth eyelets. Each section provides the frame image overlaid by the estimated states. Close-up views around eyelets are displayed at the top right of a section. The top left of each section presents a 3D plot of the estimated states around the eyelets, where coloured eyelets indicate that the shoelace passes through, and white indicate otherwise. The *Sequential H-Signatures* follow the same colour scheme as their corresponding eyelets and are shown at the bottom of each section. TSL consistently produces the correct signatures across all cases, whereas the baselines estimate the shoelace to be outside the eyelets (zero in the signatures).

plot of the states around the eyelets (top left), where coloured eyelets indicate that the shoelace passes through, and white eyelets indicate otherwise; and (4) the *Sequential H-Signatures* that follow the same colour scheme as their corresponding eyelets (bottom).

The left two columns of Figure 5.7 shows the results with **simulated data**. Overall, TSL, TrackDLO and CDCPD2 achieved similar levels of shoelace coverage. The close-up view for CDCPD2 shows the shoelace path does not cover all the eyelets. As shown in the top right 3D plots, the shoelace



(a) CDCPD2 (b) GLTP

(c) TrackDLO (d) TSL (Proposed)

Figure 5.8: *H-Signature* of the shoelace with simulated data. Positive values indicate the shoelace is passing through the eyelet same as the eyelet's normal direction. The shoelace is considered to be passing through an eyelet if the absolute value is over 0.5. TSL maintains a stable shoelace-eyelet relation for all eyelets, whereas the baselines fail to preserve the shoelace paths through the eyelets.

estimated by CDCPD2 does not pass through these eyelets. This is further verified by the *Sequential H-Signature* where CDCPD2 produces zeros for these eyelets. TrackDLO maintains better alignment between the estimated states and the actual shoelace compared to CDCPD2, but it also generates incorrect *H-Signatures*. The results from GLTP only partially align with the shoelace structure, and its *Sequential H-Signature* indicates that the shoelace does not pass through any eyelets.

To further examine the evolution of the shoelace–eyelet relationship, we plot the *H-Signature* in Figure 5.8. The signature is 1 or -1 when the shoelace passes through the corresponding eyelet in the same or opposite direction of the eyelet normal. Although the *Sequential H-Signatures* produced by CDCPD2, TrackDLO, and GLTP are incorrect, their tracked shoelace paths do momentarily pass through the eyelets immediately after insertion. However, due to the absence of constraints to preserve the topology, the tracked shoelaces frequently penetrate the eyelets throughout the task. As a result, the *H-Signatures*



(a) Human Demo (b) Real Robot

Figure 5.9: Occlusions during the tracking process. In SL, the manipulators inevitably occlude the shoelace from different perspectives. TSL keeps the tracking process to continue with its simulated model and the EE movement.

for CDCPD2, TrackDLO, and GLTP are unstable, whereas TSL consistently preserves the correct shoelace–eyelet relationship across all eyelets.

With respect to the **human demonstration data** (the middle two columns of Figure 5.7), the results are similar to the simulated condition. TSL correctly estimates the shoelace-eyelet relation and produces the correct *Sequential H-Signatures*, while the results of other algorithms fails to fail to preserve the shoelace paths through the eyelets. CDCPD2 have key nodes overall aligned with the eyelet, but nodes near the eyelets are squeezed outside and edges near the shoelace tips are bent. This shows how CPD tries to maximise the coverage of the tracked nodes against the segmented shoelace masks. GLTP pushes the key nodes close to the shoelace and keeps a good coverage, however, the structure of the shoelace is lost. TrackDLO lost track of the tips affected by occlusion, where the motion of the visible part of the shoelace does not reflect the behaviour of the hidden region. Figure 5.9a highlights critical frames in which the tip, along with a significant portion of the shoelace, is occluded, making it difficult to estimate the tip motion based on the visible shoelace components.

The bottom section of Figure 5.7 shows the results from the **real robot data**. Compared with the previous two experiments, all tested algorithms show worse performance in overlapping the tracked nodes with the shoelace. Although TSL kept the shoelace through the eyelets, the estimated structure does not completely align with the shoelace. The baseline algorithms also exhibited varying degrees of mismatch with the shoelace, which can be attributed to severe occlusions and the robot's fast motion. Due to the complexity of the SL task, the robot introduces significant occlusions regardless of camera placement. For instance, in Figure 5.9b, the shoelace is almost completely obscured from view. Meanwhile, YuMi EEs move at a

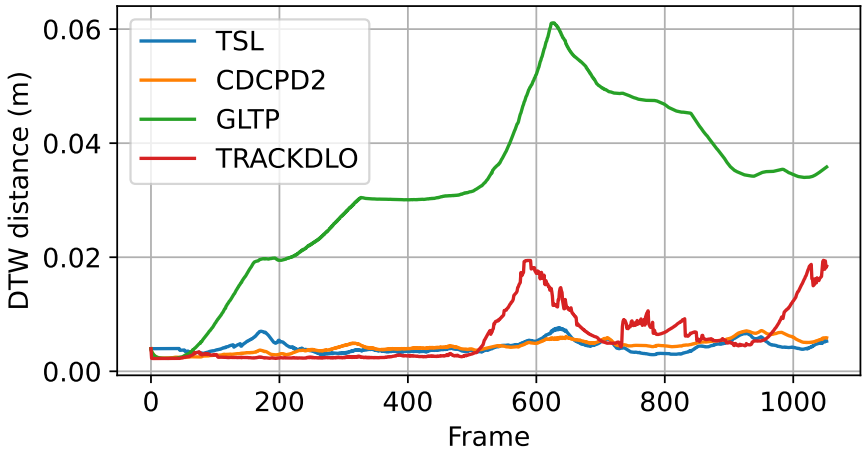


Figure 5.10: DTW distance for all tested methods. TSL and CDCPD2 achieve similar levels of tracking error. Although CDCPD2 is not able to estimate the shoelace-eyelet relation correctly, the difference makes a small impact on the overall tracking error.

speed of 300 mm/s which is significantly faster than the EEs in the other two experiments. This also caused severe motion blur in the RGB images, which decreased the quality of the segmented shoelace mask. The quality of depth information is also affected by the fast robot's movement. The tracking accuracy can be potentially improved with slower motion, however, this will hinder manipulation performance.

5.5.3 *Quantitative Analysis With The DTW Distance*

To understand whether correctly estimating the shoelace-eyelet relation impacts the overall tracking error, Figure 5.10 shows the DTW distance of the tested algorithms. While TSL achieves correct *Sequential H-Signatures*, overall, its tracking error is similar to that of CDCPD2, with mean values at 4.32 and 4.36 mm/node respectively. Without the EE information, TrackDLO achieves a 5.98 mm/node mean error. GLTP instead reached a mean error at 32.48 mm/node. Since the regions near the eyelets form a small portion of the shoelace, the performance difference in the shoelace-eyelet relation only makes a small difference to the tracking error.

The choice of physical parameters is an important factor in DLO simulation. In the previous experiments, the same parameter values used for generating the simulated data are adopted, and applied empirically tuned values for other datasets to maximise visual similarity with the real shoelace. To investigate whether inaccuracies in physical parameters affect tracking error, this work conducts a sensitivity analysis by sweeping through ranges of three key

Table 5.1: Tracking Error (mm) with different simulation parameters: Particle Mass (kg), Particle Radius (m) and Max Bending (m). The values used to collect the simulation data are bolded.

Mass	**0.001**	0.002	0.003	0.004	0.005
Error	4.32 ± 0.10	4.48 ± 0.11	4.80 ± 0.12	5.05 ± 0.13	5.38 ± 0.15
Radius	0.001	0.00125	**0.0015**	0.00175	0.002
Error	4.48 ± 0.11	4.32 ± 0.10	4.32 ± 0.10	4.34 ± 0.10	4.36 ± 0.11
Bend	0.001	0.005	**0.01**	0.015	0.02
Error	4.51 ± 0.10	4.36 ± 0.10	4.32 ± 0.10	4.34 ± 0.10	4.33 ± 0.10

Table 5.2: Tracking Error with different number of eyelets

# Eyelets	2	4	6	8	10
Error	4.75±1.26	4.99±1.31	4.98±1.28	4.84±1.35	4.69±1.35

physical parameters: particle mass, particle radius and max bending. Particle mass influences how particles respond to constraints such as collisions and bending. Particle radius affects the number of particles generated. Max bending defines the threshold at which the bending constraint becomes active, effectively controlling the shoelace's stiffness.

Table 5.1 presents the results of this experiment, with the values used during data collection highlighted in bold. For the mass parameter, tracking error increases with higher mass values, as the regulating forces become less effective when particles are heavier. The mass could not be reduced further, as the shoelace becomes too light and oscillates under the regulating forces. The radius sweep was limited to a relatively narrow range due to constraints imposed by the eyelet size. Nonetheless, it is clear that tracking error increases as the radius deviates from the original value. For the bending parameter, which serves as the threshold for activating bending constraints, tracking becomes less accurate as the value decreases. Meanwhile, once the value exceeds that used in data collection, the error differences become negligible. This suggests that making the simulated shoelace softer than in the dataset does not lead to improved tracking accuracy.

In previous experiments, TSL was evaluated using 4 eyelets. To examine the impact of increasing homotopy classes on tracking accuracy, this work tests TSL on a simulated dataset with 10 eyelets. As shown in Table 5.2, the tracking error initially increases and then decreases as the number of eyelets grows. This trend is attributed to the latency between the tracked and actual states during fast movements. Initially, the tracked states lag behind the actual states, resulting in larger errors. However, as the SL progresses, more

sections of the shoelace become constrained, or effectively "frozen", between the eyelets. The influence of latency is reduced as the remaining free shoelace branches are shorter. As a result, fewer key nodes fall behind, leading to a reduction in overall tracking error.

From the time aspect, CDCPD2 runs at on average 83.76 ms/frame, GLTP at 71.20 ms/frame, TrackDLO at 52.83 ms/frame, and TSL at 114.18 ms/frame. Benefitting from the lightweight physics solver and the simplified simulation setup (Figure 5.2), an adjustment cycle in simulation uses only 25.00 ms. However, comparing with the baselines, the time cost for TSL remains one of its limitations. In future work, the time performance of the current implementation could potentially be improved with more software engineering optimising communication between the registration module and the simulation environment, e.g., more efficient serialisation to avoid conversion between Unity and Eigen data formats.

Overall, TSL correctly estimates the shoelace-eyelet relation in all experimental conditions while the other three algorithms are not able to produce the correct *Sequential H-Signatures*. In terms of the tracking error, TSL achieves a similar level of accuracy as the CDCPD2. With the support of TSL, future SL systems may improve their performances from two perspectives: 1) Task planning: Leveraging the estimation of shoelaces states and the relation with eyelets, systems can detect tangles or failures and act accordingly; 2) Motion planning: The estimated states and shoelace dynamic models can inform the planning of new actions.

5.6 CONCLUSIONS AND FUTURE WORK

This chapter presents TSL for tracking shoelaces and correctly estimating their topology in the task of bimanual SL. The algorithm leverages statistical registration to estimate the states of the shoelaces and physical simulation to maintain the topology. The performance of proposed TSL is evaluated with simulated data, human demonstration data and real robot data. The experiments demonstrate that TSL successfully preserves the shoelace topology in all experimental conditions, with tracking error on par with the state-of-the-art CDCPD2.

The proposed TSL algorithm at present has several limitations. For example, the physical parameters of the shoelace are set empirically and the segmentation of the shoelace is not robust. The future directions to improve the performance of the algorithm are discussed in Section 7.3 with more details.

6

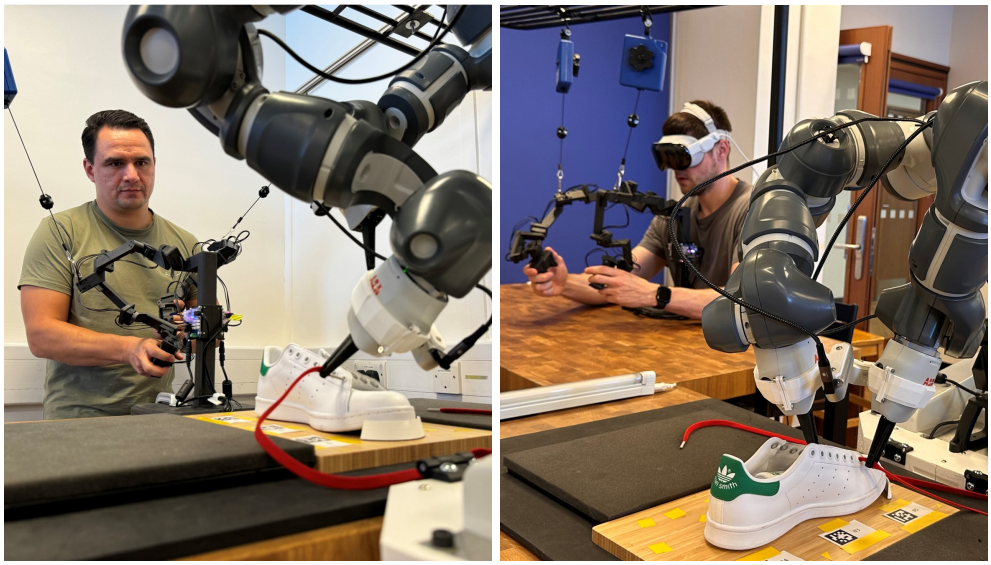
INFLUENCE OF TELEOPERATION INTERFACE IN BEHAVIOURAL CLONING FOR DLO MANIPULATION

The system presented in Chapter 3 automates the manipulation of DLOs with a set of hand-crafted primitives. Manually constructing and tuning such primitives is time-consuming and not scalable. Learning from Demonstration (LfD) approaches provide an efficient solution to obtain the robot primitives without accurate modelling of the DLO dynamics. Among the LfD techniques, Behavioural Cloning (BC), which leverages demonstrations collected through teleoperated robots, enables policies to complete tasks with higher precision (Liang et al., 2024). Chapter 2 discussed the use of teleoperation to learn DLO manipulation and its relation to policy performance. Although many works focused on proposing new teleoperation interfaces, few studies have investigated the impact of teleoperation interface on the effectiveness of collected demonstration data (Si et al., 2021). This chapter aims to address this limitation:

> *How does the teleoperation interface influence the effectiveness of expert demonstrations in BC for DLO manipulation?*

This chapter focuses on a comparative study on the effectiveness of first and third-person perspective user interfaces, applied in shoelace insertion, a subtask of SL. Section 6.1 provides a formal definition of the insertion task. Section 6.2 introduces two teleoperation interfaces: a third-person perspective interface (GuMi) and a first-person perspective interface (Ego-GuMi) (Figure 6.1). Section 6.3 presents the learning framework used to train the robot to perform the task. Section 6.4 describes the hardware setup, data collection, and analysis metrics. Section 6.5 presents an analysis of the results and discusses how the teleoperation interface influences the demonstrator's performance. Finally, Section 6.6 concludes the chapter.

The work presented in this chapter has been accepted for publication at the 2025 IEEE International Conference on Robotics and Automation (ICRA).



(a) GuMi (b) Ego-GuMi

Figure 6.1: GuMi: Third-Person Perspective Interface (left), and Ego-GuMi: First-Person Perspective Interface (right). GuMi integrates the GELLO framework (Wu et al., 2023) with ABB YuMi to facilitate the teleoperation of dexterous dual-arm manipulation from a third-person viewpoint. In contrast, Ego-GuMi leverages an Apple Vision Pro headset, allowing the operator to control YuMi from an immersive and dynamic first-person perspective of the task.

6.1 PROBLEM FORMULATION

Consider a task where a robot is required to insert a lace into an eyelet (Figure 6.1). The robot is equipped with 3 RGBD cameras mounted on the two grippers and the chest. The cameras provide observation o_t at each time step $t \in \mathbb{Z}_{\geqslant 0}$. The robot provides joint states in the form of angle positions j_t at each time step. Datasets $D = \{(o_t, j_t, a_{t+1:t+k})\}_{t=1}^{N}$ are collected from expert demonstrations, where $a_{t+1:t+k}$ is the sequence of actions taken by the expert at time steps $t+1$ to $t+k$. The task is to learn a policy π that can predict the next k actions $a_{t+1:t+k}$ given the current observation o_t and joint states j_t.

6.2 TELEOPERATION INTERFACES

Out of the teleoperation interfaces available, two types can be categorised: third-person perspective (Wu et al., 2023; Zhao et al., 2024) and first-person perspective (Ding et al., 2024; Kim et al., n.d.). To compare the impact of

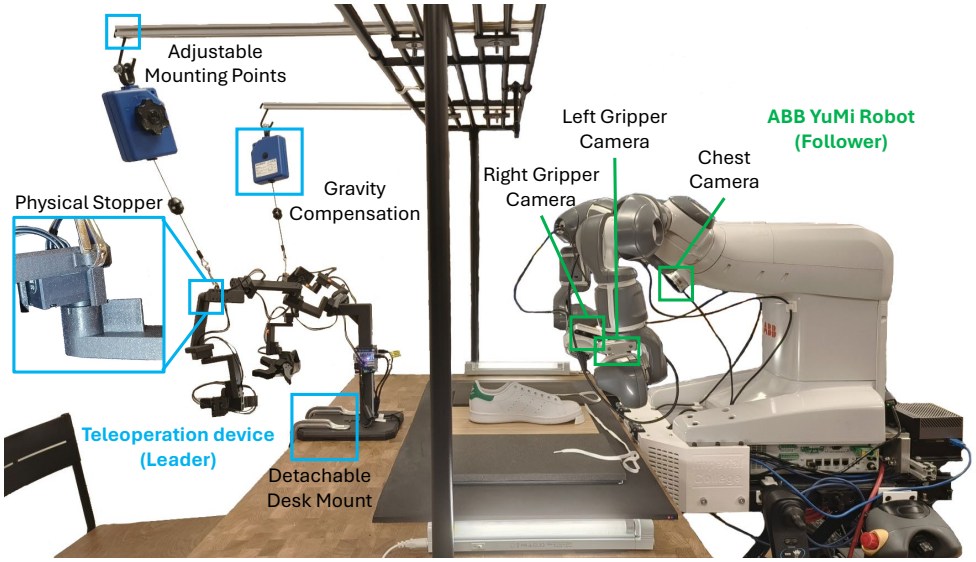


Figure 6.2: GuMi Interface. GuMi incorporates a 3D-printed kinematic model of YuMi (the leader, also referred to in this thesis as the teleoperation device), designed to interface seamlessly with the robot (the follower) for dexterous teleoperation. The teleoperation device is placed facing the same direction as the operator to facilitate teleoperation from a third-person perspective. The device is 75% scale of the actual robot and the joint states are directly mapped to the robot joints. The joints of the kinematic model are designed to have physical stoppers so that they cannot move beyond the joint limits of the robot. More detailed description of the mechanical design is provided in Appendix B

the teleoperation interface on the same task setup, this work develops two teleoperation interfaces: GuMi and Ego-GuMi.

6.2.1 *GuMi - Third-Person Perspective Interface*

This work presents an integration of the GELLO framework (Wu et al., 2023) with the dual-arm ABB YuMi robot, incorporating a custom 3D-printed kinematic model (the leader, also referred to as the teleoperation device in this thesis) designed to interface seamlessly with the robot (the follower). An illustration of this interface is shown in Figure 6.1b and Figure 6.1a. The resulting system, named GuMi, demonstrates the potential of low-cost, user-friendly teleoperation setups for executing dexterous bi-manual tasks.

The kinematic model is a 75% scale replica of the actual robot, replicating its joint configuration to enable precise one-to-one mapping with the robot joints. Physical stoppers are designed to ensure that the leader's joints remain within the joint limits of the robot so that the actions of the teleoperator are directly translated into the robot movements. Tool balancers are used

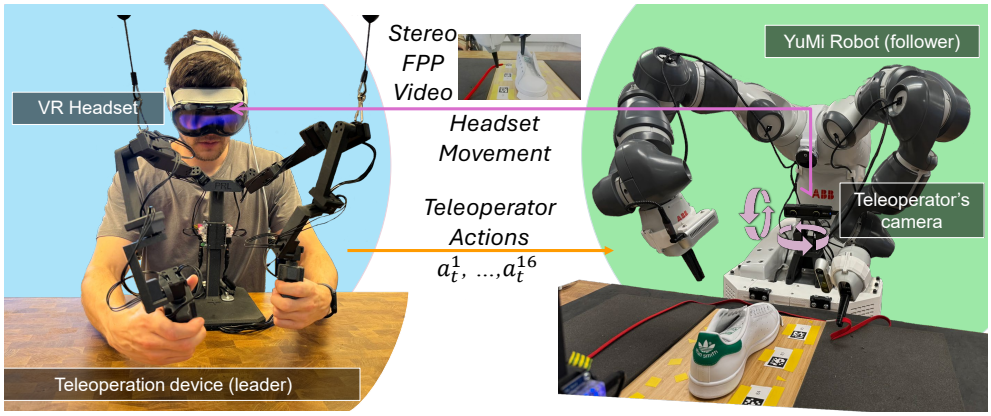


Figure 6.3: The Ego-GuMi framework facilitates first-perspective teleoperation with VR. An expert teleoperator demonstrates bi-manual lacing actions in real-time using a 3D-printed kinematic model of YuMi, while viewing a stereoscopic video feed through a VR headset (top-left). Meanwhile, YuMi perceives its environment using three cameras mounted on its arms and chest (right).

to counterbalance the weight of the kinematic model. They are positioned to separate the two arms, allowing the operator to have a clear view of the workspace. The mounting mechanisms for the tool balancers and kinematic model are adjustable to accommodate operators with varying needs. Further details on the mechanical design are provided in Appendix B.

In this setup, the kinematic model is positioned facing the same direction as the robot. The End-Effectors (EEs) of the leader moves in the same direction as the robot's EEs. The operator is positioned facing the robot and the leader, providing a third-person perspective of the task. The robot perceives its environment using three cameras mounted on its arms and chest. These camera feeds, combined with the robot's joint states and the teleoperator's actions, comprise the demonstration data. The demonstrations are later used to train a policy that learns to mimic human skills.

6.2.2 *Ego-GuMi - First-Person Perspective Interface*

To compare the impact of the teleoperation interface on the same task setup, this work extends the GuMi system with an egocentric perspective. The integrated system, termed Ego-GuMi, is illustrated in Figure 6.1b and Figure 6.3. The egocentric perspective is achieved using a Virtual Reality (VR) headset and an active sensing gimbal, which houses a stereo camera. The camera is positioned on the robot base towards the workspace to provide an immersive first-person perspective during manipulation. The camera streams

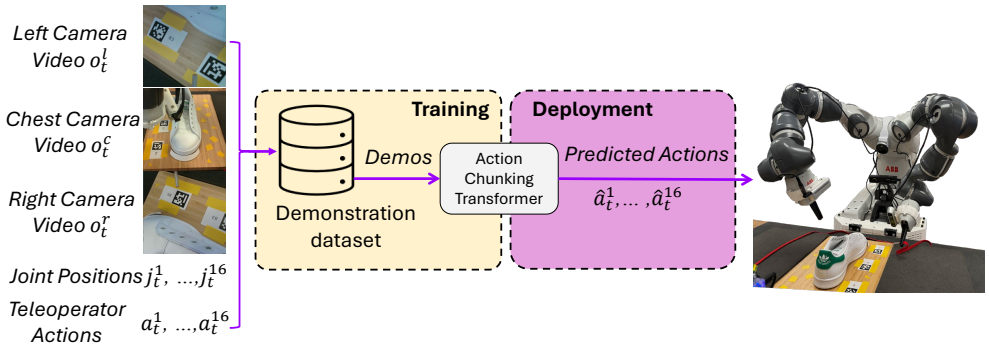


Figure 6.4: LfD framework. The camera feeds, combined with the robot's joint states and the teleoperator's actions, comprise the demonstration data. The demonstrations are used to train a policy with Action Chunking with Transformers (ACT) (Zhao et al., 2023), which learns to mimic human skills. This policy is then used to control the robot in performing the insertion task.

stereoscopic video feeds to the operator's headset, allowing the operator to view the task from the robot's perspective. The operator is positioned behind the teleoperation device, facing the virtual workspace, the same as the way the robot perceives its environment.

Long-term teleoperation tasks can cause discomfort for the operator due to the misalignment between the operator's viewpoint and the camera's perspective. The active sensing gimbal tracks the operator's head movements, which provides a dynamic and responsive viewpoint of the task. The teleoperation device is designed to have the grippers 10cm above the desk when the robot fingertips touch the desktop. This offset ensures that the operator can safely rest their arms on the desktop without risk of collision.

6.3 LEARNING SHOELACE INSERTION FROM DEMONSTRATION

This work employs ACT (Zhao et al., 2023) as the policy for learning the shoelace insertion skill from demonstrations. The policy predicts sequences of k future actions $\hat{a}_{t:t+k}$ rather than single actions, reducing the effective horizon of long trajectories collected at high frequency and therefore mitigating compounding errors in BC. ACT policy leverages a Conditional Variational Autoencoder (CVAE) to address the multi-modality of human demonstrations. The CVAE encoder, implemented as a transformer encoder, compresses the current observation $\bar{o}_t$ and target action sequence $a_{t:t+k}$ into a latent vector z to capture the variability across demonstrations. During inference, z is set to zero for deterministic decoding. The CVAE decoder is a full transformer model that reconstructs the target action sequence from the latent vector

z. The transformer encoder processes visual and proprioceptive inputs to synthesise information across modalities. The transformer decoder generates $k \times n$ action chunks, where n is the number of joints. It uses the output from the first transformer for cross-attention, with queries initialised as fixed positional embedding. The ACT policy is trained end-to-end to maximise the log-likelihood of demonstration chunks under the CVAE objective. This includes a reconstruction loss $\mathcal{L}_{\text{reconst}}$ for action prediction and KL divergence regularisation $\mathcal{L}_{\text{reg}}$ on the encoder distribution:

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_{\text{reconst}} + \beta \mathcal{L}_{\text{reg}} \\ &= \text{MSE}(\hat{a}_{t:t+k}, a_{t:t+k}) + D_{\text{KL}} \left(q_\phi(z \mid a_{t:t+k}, \bar{o}_t) \,\|\, \mathcal{N}(0, I) \right), \end{aligned} \tag{6.1}$$

where β weights the second term.

In this work, the observation includes three RGB images at a resolution of 480×480 and 16-dimensional joint positions from YuMi's arms. The images are encoded using ResNet18 backbones, then flattened and combined with learned positional embeddings. The flattened image features are concatenated with embedded joint positions and the latent z, forming the combined input sequence. In this work, k is set to 60 steps. The transformer generates a 60×16 tensor, corresponding to a 2-second sequence of 16-dimensional joint positions for YuMi's arms. Following the original ACT implementation, L1 loss is used for reconstruction, and the KL term is weighted by a hyperparameter β, which is set to 100 in this work. The model is trained for 15000 epochs, as extended training can notably improve performance even after loss plateaus. The temporal aggregation is disabled, aligning with ACT's tuning recommendations.

6.4 EXPERIMENTS

6.4.1 Hardware Setup

The ACT is trained on a computation platform with an AMD Ryzen Threadripper Pro 5965WX processor, 128GB of RAM, and an RTX 4090 GPU. During inference, the models run on a desktop computer equipped with an Intel i9 CPU with 24 cores, 32 GB of RAM, and an RTX 2080 GPU.

An ABB YuMi is used in this study with positional control and a target end-effector velocity of 400 mm/s. YuMi is augmented with two Realsense D435s cameras attached to the grippers and a Realsense L515 camera mounted on

its chest (Figure 6.3). All cameras and the robot driver run at 30Hz. The details of the robot setup are provided in Appendix A.

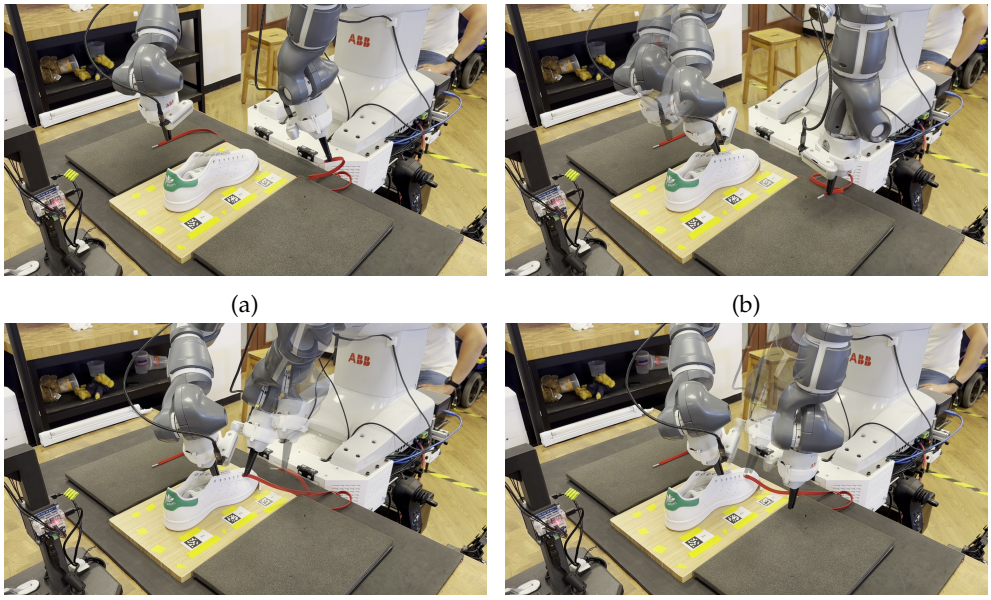
The Open-TeleVision system, a low-latency pipeline for stereo camera streaming, is used for interfacing the active sensing gimbal (Cheng et al., 2024). The end-to-end delay of this pipeline to stream the camera footage directly to the operator headset is approximately 15 ms, which is sufficiently low as real-time feedback. The setup employs a ZED Mini stereo camera to capture stereoscopic video at a resolution of 1280x720 with a frame rate of 60 fps. An Apple Vision Pro VR headset is used to receive and display the live video feeds from the cameras. Through the high-resolution headset display, low-latency processing, and precise headset tracking, Ego-GuMi provides an immersive and responsive teleoperation experience.

6.4.2 *Data Collection*

This work collects 100 demonstrations from two expert demonstrators, each specialising in one of the interfaces (GuMi or Ego-GuMi). The demonstrations are recorded to train and compare the resulting models. 100 episodes are collected from each interface, resulting in a total of 200 demonstrations. Each demonstration takes around 20s and is divided into three steps, as illustrated in Figure 6.5:

1. **Point at Eyelet with Right-Hand Camera:** The demonstrator points the right-hand camera at the target eyelet.
2. **Approach and Insert Lace with Left-Hand:** The demonstrator operates the left gripper to approach the eyelet and insert the tip of the shoelace into the eyelet.
3. **Release and Retract with Left-Hand:** Finally, the demonstrator releases the shoelace and retracts the left gripper to point at the eyelet again using the left-hand camera.

The data is recorded into *ROS Bags*. They are then synchronised and processed to extract the images and joint states. The images are cropped and resized to 480×480 pixels. Both the images and joint states were normalised before being saved as datasets. 80% of the data is used for training, and the remaining 20% is used for validation.



(a) (b)

(c) (d)

Figure 6.5: Data collection procedure: two expert demonstrators teleoperate the robot to achieve the lace eyelet insertion task using GuMi or Ego-GuMi interfaces. 100 episodes are collected from each interface. Demonstrators start recording from a comfortable configuration as in **(a)**. Three-steps are completed afterwards: **(b)** pointing at the eyelet, **(c)** approaching and inserting the lace, and **(d)** releasing and retracting.

6.4.3 *Result Analysis Metrics*

This work analyses the differences between the two interfaces from three perspectives: demonstration time, variability in demonstrations, and task performance. This section outlines the methods used to evaluate the time cost and the variability of the demonstrations.

For the analysis of the time cost of demonstrations, this work uses robust estimation, a Bayesian approach to the analysis of data from two groups (Kruschke, 2013). Further details on this methodology can be found in Chacón Quesada and Demiris (2024). The robust estimation model used to compare the time usage between interfaces is given by:

$$\begin{aligned} T_i &\sim t(\nu, \mu, \sigma) & \sigma &\sim \text{Half-Normal}(0, 1) \\ \mu &= \alpha_{ui_i} & \nu &= 1 + \nu_{-1} \\ \alpha_{ui} &\sim \text{Normal}(0, 1) & \nu_{-1} &\sim \text{Exponential}(1/29) \end{aligned} \tag{6.2}$$

where T_i denotes the list of time data, and the index i represents the row number. A t distribution is used as the likelihood for this data. This distribution is parameterised with μ, σ, and ν. The ν parameter can range continuously

from 1 to ∞. The distribution exhibits significant variation for small ν values; for ν greater than 30, it becomes essentially normal. The term $ui = \{1, 2\}$ serves as an index variable, where $ui = 1$ corresponds to GuMi, and $ui = 2$ to Ego-GuMi. Standardised T ($\mu = 0$, $\sigma = 1$) is used to facilitate assigning priors for the average effect for each interface α_{ui} and for the σ parameter, which estimates the variation across interfaces. The term ν_{-1} follows an exponential distribution with a mean of 29, ranging from 0 to ∞. To allow ν to vary from 1 to ∞, 1 is added to ν_{-1}, changing its mean to 30, providing equal opportunity for small (< 30) and larger (> 30) values (Kruschke, 2014).

The model from Equation (6.2) is fitted to the recorded data using the `ulam` interface from the `rethinking` package in R with 4 Markov chains. Multiple chains are used to verify that the joint posterior sampling worked correctly. By default, each chain consists of 1000 iterations, split equally between warmup and sampling (McElreath, 2020).

In terms of the diversity of demonstrations, expert demonstrators typically exhibit a low degree of variability in their movements as they become more proficient in performing a task through repeated practice. To assess the variability among the demonstrations provided by the demonstrators, this work computes the distances between the input command sequences. Each sequence is a 16-dimensional time series, with one dimension per joint. Due to the differences in sequence length, the sequences cannot be directly compared using Euclidean distances. This work employs Dynamic Time Warping to align the paired demonstrations (Müller, 2007). Dynamic Time Warping (DTW) provides the mapping between the sequences and measures their similarity, allowing the quantification of the variability in the input command sequences for each expert. Figure 6.6 illustrates the warping path of two demonstrations. This work leverages the DTW distance to evaluate the similarity between the input commands by the same expert and the variability of their demonstrations.

6.5 RESULTS AND DISCUSSION

This section begins with a qualitative analysis based on video recordings. The footage reveals distinct lacing strategies depending on the teleoperation interface. The expert using Ego-GuMi moves the gripper more slowly and demonstrates greater caution during insertion compared to the expert using GuMi. As a result, insertions with Ego-GuMi tend to be more accurate, with the aglet entering the eyelet directly. In contrast, insertions with GuMi often involve the aglet colliding with the eyestay before being repositioned

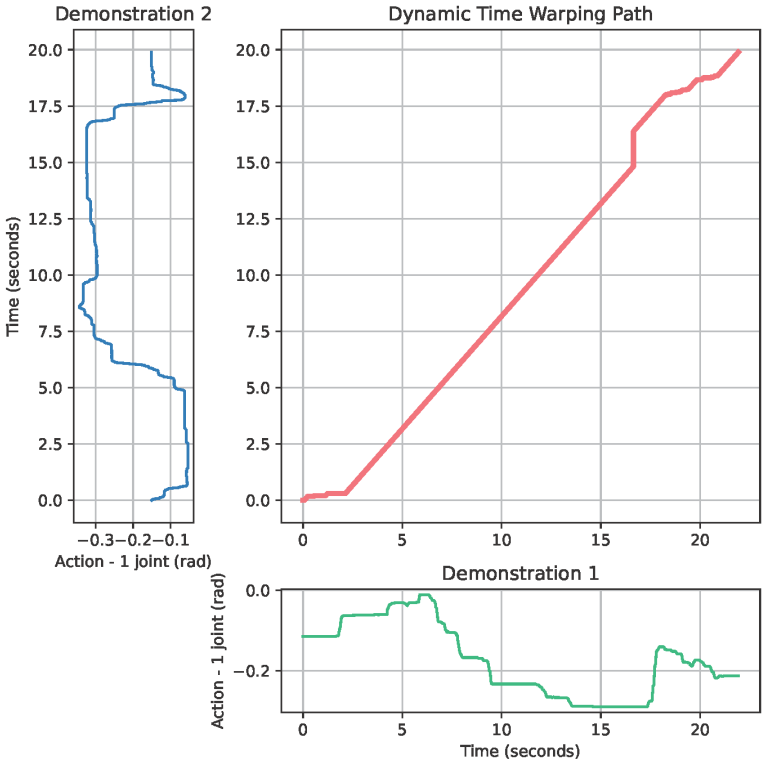


Figure 6.6: Dynamic Time Warping Analysis of Expert Demonstrations. DTW is employed to assess the variability between different expert demonstrations, which measures the accumulated Euclidean distance between input command sequences. The resulting DTW path shows the similarity between two real demonstration sequences (for one joint) provided by one of the experts. A straight line would indicate identical demonstrations, while deviations from the line reflect differences in movement patterns and execution.

and reinserted. Additionally, the Ego-GuMi expert consistently lowers the observation gripper during the insertion process, taking advantage of the unobstructed view provided by the camera feed.

From the quantitive analysis, first, the average time each expert took to finish a demonstration. The expert using GuMi took 14.5 ± 5.6s on average, while the one using Ego-GuMi took 20.3 ± 7.7s. Following model fitting using Equation (6.2), the posterior distribution is used to determine if the difference in this data is significant. Figure 6.7 reveals that the 95% Highest Density Interval (HDI) of the contrast between interface effects (α_2 - α_1) does not intercept a Region of Practical Equivalence defined from -0.1 to 0.1. Consistent with the rule outlined in Kruschke (2018), the demonstration time differed significantly between interfaces. However, Figure 6.8 shows

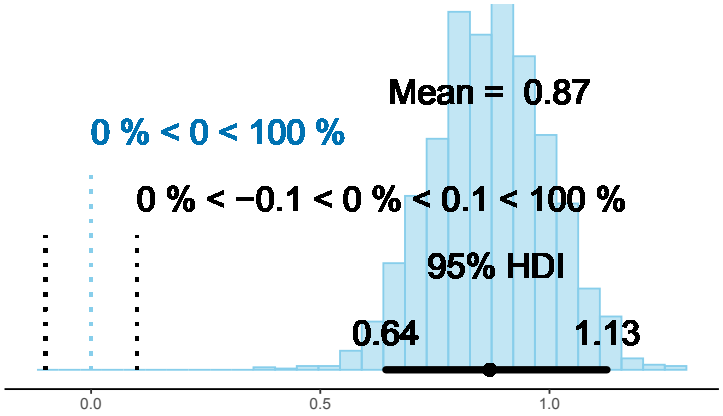


Figure 6.7: Contrast between the effect of the interfaces on the demonstration time T, i.e., (α_2 - α_1). The taller dotted lines centred at zero show the percentages of the contrast that are positive and negative. The smaller dotted lines indicate the Region of Practical Equivalence (ROPE) defined from -0.1 to 0.1 (Kruschke, 2018). Here, the percentages indicate the proportion of the distribution below, within and above the ROPE.

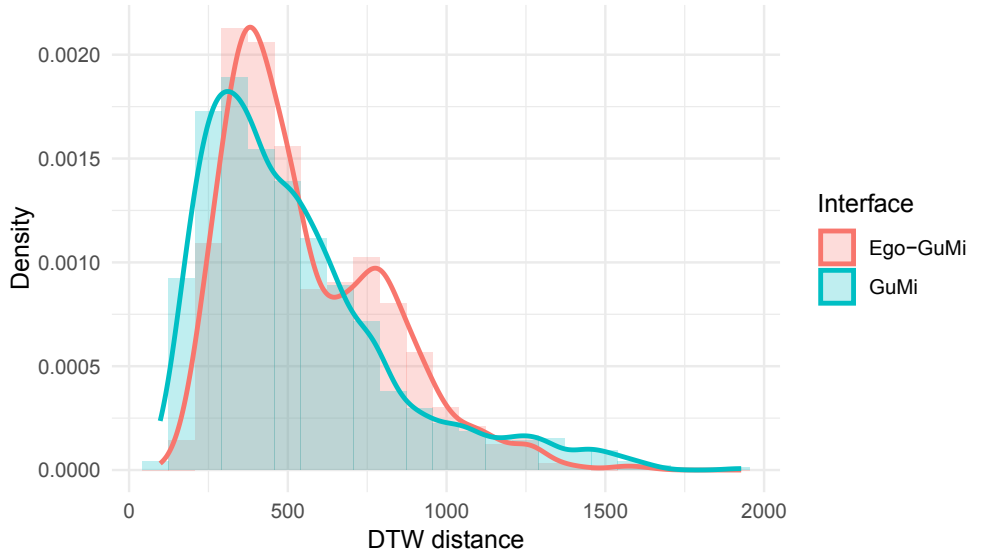


Figure 6.8: Distributions of the variability in demonstrations for both experts. For each scenario, the demonstrations are compared pairwise using DTW. Demonstrators tend to perform repetitive movements after repeated practice. The plot shows similar levels of variability in both cases, indicating similar extents of experience.

that they exhibit comparable variability in their input commands between demonstrations.

The significant difference in demonstration execution time between GuMi and Ego-GuMi interfaces may suggest that the experts' movement patterns were influenced by the visual feedback provided by each interface. Specifically,

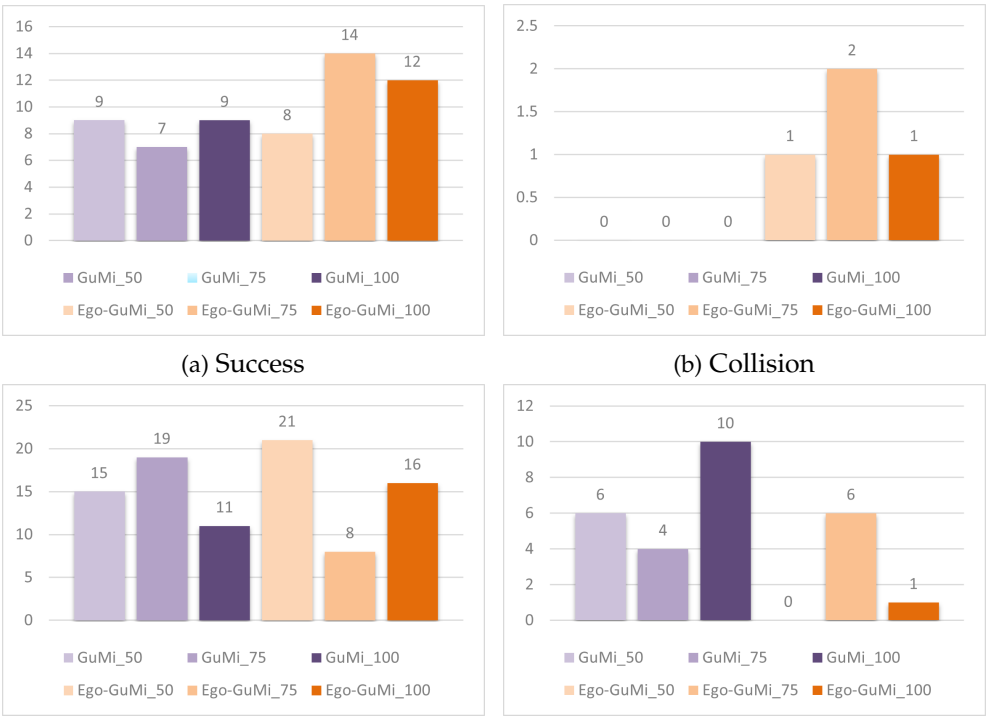
the limited 82° Horizontal Field of View (FoV) offered by the stereo camera used with Ego-GuMi may have contributed to this difference. The camera FoV is inherently more restricted than the FoV of human eyes, which allows for less contextual information to be gathered and processed before taking actions. As a result, the expert using Ego-GuMi may have been more cautious in their movements, requiring more head movements to compensate for the reduced FoV. This increased caution may have led to slower demonstrations compared to GuMi, where the expert had a more expansive FoV to inform their actions. Overall, the speed and input command analysis suggest the potential difference in the way experts perform tasks under the impact of different types of interfaces.

From the perspective of task performance, the analysis of the learning outcomes reveals that the choice of the interface impacts the Success Count (SC) of the trained policy (Table 6.1). A notably higher SC is observed with Ego-GuMi compared to GuMi across varying numbers of training episodes. This difference may stem from more consistent expert movements when using the Ego-GuMi interface. The demonstrator exhibits less hesitation with Ego-GuMi during the insertion process (step 2 in Section 6.4.2). This increased consistency likely makes it easier for the learning algorithm to replicate the expert's actions, which results in better performance. Furthermore, the policies trained with Ego-GuMi data achieve above 40% success rate when trained with at least 75 demonstrations, comparable to state-of-the-art results reported for similar tasks and architectures (Zhao et al., 2024). These findings underscore the importance of the interface design and highlight the potential benefits of using interfaces such as Ego-GuMi that provide a first-person perspective.

Table 6.1: Comparison between Ego-GuMi and GuMi interfaces on shoelace insertion task performance (N = 30): Success Count (SC), Collision Count (CC), Drop Count (DC), Time-out Count (TC). Averages for each interface are included at the end.

Metric	GuMi				Ego-GuMi			
	# of Demonstrations			Average	# of Demonstrations			Average
	50	75	100		50	75	100	
Success	9	7	9	8.3	8	14	12	11.3
Collision	0	0	0	0	1	2	1	1.3
Drop	15	19	11	15	21	8	16	15
Time-out	6	4	10	6.6	0	6	1	2.3

In addition to SC, Table 6.1 summarises Collision Count (CC), Drop Count (DC) and Time-out Count (TC) as specific aspects of the robot behaviour



(a) Success (b) Collision

(c) Drop (d) Time-out

Figure 6.9: Comparison between Ego-GuMi and GuMi interfaces on shoelace insertion task performance. Overall, the SC for Ego-GuMi is notably higher than that of GuMi. For policies trained with more than 75 and 100 Ego-GuMi episodes, the SC is above 40% higher than that with 50 episodes.

analysed. CC refers to the number of times the robot's motion stopped due to collision with the shoe or with itself, which in turn caused the torque on any joint to go beyond its allowed threshold. DC refers to the number of times the robot released the shoelace without inserting it in the eyelet first. A run is marked as a failure if it goes beyond 100s (TC). The higher CC observed with Ego-GuMi compared to GuMi may seem counterintuitive, but it can be attributed to the expert's habits and strategy during demonstrations. Specifically, the video analysis suggests the expert's tendency of bringing the grippers closer together in demonstrations, which leads to a greater frequency of collisions with the shoe, particularly when transitioning between different steps of the task (Section 6.4.2). In addition, the similar DC average seen from both interfaces suggests that this metric is less sensitive to the differences in expert demonstrations between Ego-GuMi and GuMi. Furthermore, the significantly higher TC observed for GuMi indicates that the robot faces greater difficulties completing the task within the allowed 100 seconds, potentially due to the less consistent and hesitant expert demonstrations collected with this interface.

6.6 CONCLUSIONS

This chapter presents a comparative study of two teleoperation interfaces, GuMi and Ego-GuMi, for analysing the impact of interface design on expert demonstration quality in BC for DLO manipulation. While interface effectiveness may vary depending on the task, this study highlights the importance of considering the impact of the user interface, particularly when it comes to precision-critical tasks such as SL. By evaluating the effectiveness of first-person (Ego-GuMi) and third-person (GuMi) perspective interfaces, this work aims to overcome the knowledge gap in developing more effective interfaces for BC. The findings demonstrate significant differences in demonstration execution time and consistency between the two interfaces. This research highlights the importance of considering different perspectives in designing teleoperation interfaces for DLO manipulation. By considering the impact of interface design on expert demonstration quality, one can potentially enhance the signal-to-noise ratio in the learning process, leading to more efficient and robust learning outcomes.

As per limitations, the emphasis of this study on expert demonstrators may not reflect novice user performance or learning curves. Future research should consider incorporating diverse user groups. Additionally, the impact of individual differences on the results should be considered in future studies, such as training multiple experts to perform the same demonstrations using different interfaces. This would allow for better isolation of the effects of interface design on demonstration quality and learning outcomes, providing a more robust understanding of how different interfaces can be used to improve robot learning performance.

The success rate of the trained policies in this work is less than 50%, regardless of the teleoperation perspective used. Future work could explore several directions to improve performance. One potential avenue is adopting alternative learning frameworks, such as diffusion models, which have recently shown promise in generating diverse and coherent action sequences. Increasing the number of demonstrations may also help the policy generalise better across task variations. Additionally, exploring end-effector position as the representation, rather than joint angles, could potentially provide higher-level spatial information and improve generalisability.

7

CONCLUSIONS AND FUTURE WORK

The final chapter of this thesis serves three main purposes. It begins by summarising the key contributions of the research, followed by a discussion of its limitations. Finally, it offers suggestions for potential directions for future work to address these limitations and to further the research in the field of DLO manipulation.

7.1 SUMMARY OF THESIS CONTRIBUTIONS

The main contribution of this thesis is to develop methods for the manipulation, perception and simulation of DLOs. The thesis is divided into four main parts, each addressing a different aspect of the DLO manipulation and perception.

First, Chapter 3 introduces a comprehensive system to address the novel task of robotic SL. The task of designing SL patterns is formulated as an optimisation problem, solved using a genetic algorithm. Task planning algorithms are then utilised to determine the action sequences necessary for achieving a given pattern. To enable robotic SL, a set of bi-manual action primitives is developed, supported by an active perception strategy. The system is evaluated through a series of experiments using three shoes with varying sizes, materials, and numbers of eyelets. The results highlight completion rates of 92.0%, 91.6%, and 77.5% for patterns with 6, 8, and 10 eyelets, respectively.

Chapter 4 introduces an open-source simulation environment for robot SL, designed to support the research community in developing innovative methods and advancing long-horizon robot manipulation. The chapter also presents a detailed experiment protocol and a comprehensive set of metrics for evaluating performance in bi-manual SL. The effectiveness of the benchmark is validated using a baseline approach, with results evaluated in real-world experiments involving actual robots and within the simulated environment.

In Chapter 5, the novel task of shoelace tracking in SL is tackled with the introduction of the Tracking for Shoe Lacing (TSL) algorithm. This algorithm combines probabilistic registration with physical simulation to preserve the topological relation between the shoelace and the eyelets throughout the tracking process. To evaluate the tracking performance, a sequential *H-Signature* is proposed to represent the topological state of the DLO, complemented by Dynamic Time Warping (DTW) distances to measure the tracking error. The effectiveness of TSL is evaluated against Constrained Deformable CPD 2 (CDCPD2) and Global-Local Topology Preservation (GLTP), with comparisons carried out using real-robot data, simulated data, and human demonstration data.

Finally, in Chapter 6, a customised 3D kinematic model is developed to perform demonstrations for the shoelace insertion, a subtask of SL, incorporating two teleoperation interfaces offering distinct perspectives. Furthermore, a dataset of over 100 demonstrations for each interface is collected to train manipulation policies. The thesis evaluates the results from the perspective of the completion time, demonstration diversity and the trained policy performance, exploring how the choice of teleoperation interface influences the learning outcomes of BC.

While this thesis explores a wide range of topics related to DLO manipulation, certain gaps remain that could better connect the various components of the research. Additionally, the proposed methods have their limitations, which warrant further discussion. These aspects are addressed in the following sections.

7.2 LIMITATIONS

This section examines the limitations of this thesis, including the constraints of the proposed methods and the gaps that could better integrate the different aspects of the research.

7.2.1 Failure Recovery

The SL system proposed in Chapter 3 operates under the assumption that all action primitives are executed successfully. However, in practice, failures may occur due to various factors, as outlined in Section 3.6.5. One key limitation of the system is its inability to leverage geometric information for dynamic replanning. For example, if the robot fails to insert the lace into an eyelet,

the system lacks the capability to detect this failure and adjust the action sequence accordingly.

Addressing this issue requires the integration of a perception system to monitor the outcome of each action primitive. The tracking system introduced in Chapter 5 can serve this purpose effectively. By utilising physical simulation to maintain the states of the shoelace, the tracking system can detect execution failures and anomalies in real-time, as discussed in Section 5.3.2. This enhancement would enable the system to replan dynamically and improve overall robustness.

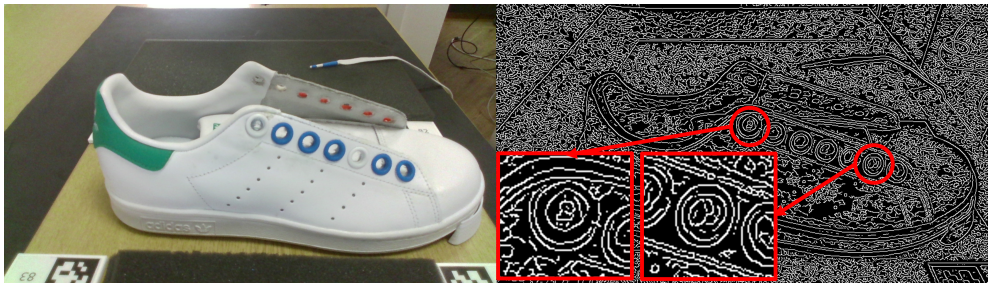
7.2.2 *Physical Parameters*

The tracking system presented in Chapter 5 operates under the assumption that the physical parameters of the shoelace are known. In practice, empirical values are often used for these parameters, which may not accurately reflect the real-world properties of the DLO. While the method prioritises preserving the topology of the DLO, its performance could potentially be enhanced with more precise physical parameter estimation.

For example, discrepancies in the stretching and bending stiffness of the shoelace can lead to overstretching in the simulated rope model, potentially causing the simulation to fail. Accurate physical parameters would not only improve the fidelity of the tracking system but also enable the physical model to predict the real-world behaviour of the DLO. This predictive capability could be leveraged to generate an estimated mask, which might accelerate the segmentation process for the DLO.

7.2.3 *Perception Of The Key Components*

The perception system outlined in Chapter 3 relies on the assumption that the aglets and eyelets are coloured and enlarged to facilitate easier detection and grasping. While this simplification is necessary for the current implementation, it limits the applicability of the system to real-world scenarios. To enhance its practicality, the system must be capable of detecting the key components of the shoelace and the shoe without any modifications. This can be achieved by training a neural network to recognise these components in their natural state. To reduce the manual effort required for eyelet annotation, edge maps can be used to train networks without manually labelling segmentation masks. As shown in Figure 7.1a, the edge maps of both coloured and plain eyelets exhibit similar structural properties. By using



(a) Edge map of coloured and plain eyelets



(b) Plain eyelet segmentation using a U-Net trained on coloured eyelet edge maps

Figure 7.1: Eyelet segmentation is performed using edge maps as input in this preliminary study. As shown in Figure 7.1a, the edge maps of both coloured and plain eyelets exhibit similar structural properties. A U-Net (Ronneberger et al., 2015) is trained using the edge maps of coloured eyelets as input and colour-thresholded masks as ground truth. The segmentation results produced by the trained network are illustrated in Figure 7.1b.

edge maps of coloured eyelets as input and masks generated through colour thresholding as groundtruth, a neural network can be trained without the need for manual labels. This thesis presents a preliminary study in which a U-Net (Ronneberger et al., 2015) is trained on coloured eyelet data and tested on plain eyelet images. As shown in Figure 7.1b, the trained network is capable of producing eyelet masks. However, the generalisability of this approach to different shoes and eyelet types requires further investigation.

A similar limitation exists in the tracking system described in Chapter 5, which assumes that the shoelace has sufficient contrast with the background for effective colour segmentation. Although addressing this challenge is beyond the current work's focus, it represents a crucial step towards building generalisable and robust manipulation systems. The tracking system should also be able to function effectively under challenging conditions, such as heavy occlusion and motion blur (Caporali et al., 2024; Yang et al., 2022*a*). One potential direction to overcome these challenges is to explore learning-based tracking approaches (Yang et al., 2022*b*).

7.3 FUTURE WORK

The previous section outlined the limitations of the proposed methods. Building on these insights, this section explores potential directions for future work aimed at addressing these limitations and advancing research in the field of DLO manipulation.

7.3.1 Planning Actions With Foundation Models

Recent advancements in Large Language Models (LLMs) and Visual Language Models (VLMs) have garnered significant attention in robotics (Brohan, Brown, Carbajal, Chebotar, Chen, Choromanski, Ding, Driess, Dubey, Finn et al., 2023; Jang et al., 2022; O'Neill et al., 2023; Shridhar et al., 2022). These models offer promising opportunities for enhancing the robustness and flexible application across different tasks and modalities (Kawaharazuka et al., 2024). For instance, the SayCan framework (Brohan, Chebotar, Finn, Hausman, Herzog, Ho, Ibarz, Irpan, Jang, Julian et al., 2023) integrates LLMs with robotic affordances, enabling robots to execute high-level natural language instructions. This approach combines the semantic reasoning capabilities of LLMs with the physical constraints and capabilities of robots, allowing them to perform complex, temporally extended tasks in real-world environments. A general approach to recovering from failures in SL could involve leveraging VLMs for action planning.

VLMs are multimodal systems capable of processing and understanding visual and textual data simultaneously. By integrating vision and language, these models can translate visual observations into natural language representations. When combined with LLMs, this information can be used to generate action plans for the SL system. Incorporating large models into the SL system could reduce the need for domain-specific knowledge, making the system more versatile and scalable.

7.3.2 Learning Physical Parameters

Numerous studies have explored methods for learning the physical parameters of Deformable Objects (DOs). For instance, Zhang and Demiris (2022) proposed a contrastive learning approach to estimate the physical parameters of garments. Their method employs a neural network to assess the similarity between the behaviour of real and simulated garments. The physical parameters are then optimised by minimising the difference between sets

of parameters that fall into the same category in the latent space. Similar techniques could be adapted to estimate the stretching and bending stiffness of shoelaces.

In another approach, Caporali et al. (2024) proposed an online method for estimating the physical parameters of DOs using a neural network model. This model is trained to replicate the dynamics of DLOs based on data generated from a Mass-Spring-Damper (MSD) model. During inference, the model parameters are iteratively refined through gradient-based optimisation to align with the observed behaviour of real DLOs. This adaptive framework enables precise shape control tasks, such as achieving target configurations, by continuously updating the model to reflect real-world dynamics.

7.3.3 Learning To Segment DLOs

Section 2.5.1 reviewed existing methods for segmenting DLOs, but a significant limitation of these approaches is the lack of robust datasets in the DLO domain. Caporali et al. (2023) proposed a weakly supervised method to segment DLOs using Virtual Reality (VR) trackers, where the user manually traces the DLO with the tracker to generate segmentation masks. While this method offers a novel approach for dataset generation, manual labelling, though simplified, remains a time-consuming and labour-intensive process. To address this challenge, Zanella et al. (2021) explored the augmentation of DLO datasets with synthetic data. However, the performance improvements achieved by segmentation models trained on synthetic datasets are often limited in real-world scenarios.

A different perspective was introduced by Chen et al. (2023), who proposed a method for segmenting plastic bags using UV-fluorescent markings applied to key areas of the bags. Labels are automatically generated by illuminating the markings with UV light, and the segmentation model, trained with RGBD images, can later perform tasks like bag opening without the fluorescent markings. This approach opens new avenues for generating segmentation datasets for DLOs. While setting up UV light sources can be complex, alternative sensor modalities, such as thermal cameras, could be explored. For example, wires that emit heat when current flows through them could be labelled using thermal imaging, providing an efficient and scalable dataset generation method.

7.3.4 *Aligning Aglets With Tactile Sensing*

One of the limitations of the system discussed in Chapter 3 is the need to align the grasped aglets. The free gripper is used to assist in orienting the aglet perpendicular to the holding gripper, which increases task completion time and reduces success rate due to possible misalignment. This issue could potentially be addressed through the use of tactile sensors. Prior work (De Gregorio et al., 2018; Wilson et al., 2023) has shown that tactile sensing can be effective for detecting the orientation of grasped DLOs and enabling accurate insertion into sockets.

For the SL task, however, integrating tactile sensors poses hardware challenges. While commercially available sensors such as GelSight and Digit have demonstrated promising capabilities, their physical size makes it difficult to insert sensor-equipped fingers into the narrow gap between eyestays. In the context of this thesis, the small payload capacity of the ABB YuMi robot further complicates fingertip sensor integration. Nonetheless, future advances in tactile sensing technology may enable their use in SL tasks.

There are two promising applications: (1) detecting the orientation of the grasped shoelace or aglet to improve insertion, and (2) tracing and pulling the shoelace after insertion to prevent twisting. While the current system can perform lacing, it does not ensure that the shoelace remains flat. Tracing is a common step in human lacing, and incorporating this behaviour could keep the shoelace flat. Related work in cable tracing (She et al., 2021) may provide a useful reference.

7.4 EPILOGUE

Manipulating DLOs with robots is a challenging problem that requires the integration of various research fields. This thesis has drawn inspiration from robotics, computer vision, topology, and machine learning to address this issue. It has made advancements in the planning, perception, and simulation of DLOs and has suggested promising directions for future work to address the limitations of the current research status. While this thesis represents only a small step towards the ultimate goal of autonomous and robust DLO manipulation, the author hopes that the work presented will inspire future research and bring the community closer to this destination.

A

ROBOT PLATFORM

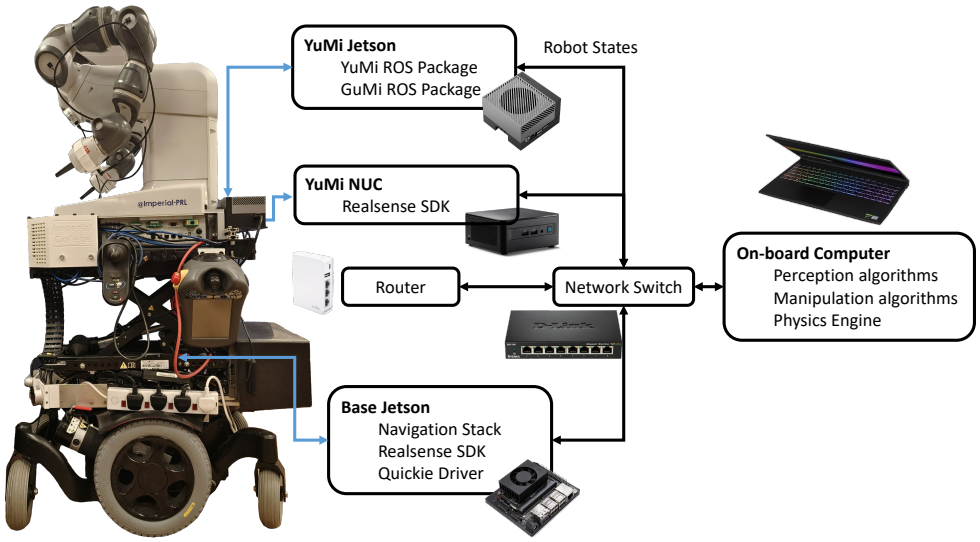


Figure A.1: **Hardware platform**. The YuMi robot is mounted on a powered wheelchair to provide additional mobility. Note: the wheelchair is not used in the work presented in this thesis. YuMi is connected to a Jetson AGX Orin, which runs the YuMi Robot Operation System (ROS) package and the associated drivers. Similarly, the mobile base is interfaced with a Jetson Xavier NX, responsible for executing all navigation-related packages and additional camera drivers. An Intel NUC hosts the remaining camera drivers. A local network facilitates communication between these computational units and the onboard computer, which manages the high-level algorithms for the platform.

The robotic platform employed in this research is illustrated in Figure A.1. The system integrates an ABB YuMi robot for manipulation tasks and a Quickie Salsa M2 Mini wheelchair serving as the mobile base. The YuMi robot is connected to a Jetson AGX Orin, which executes the YuMi ROS package and its associated drivers. Additionally, the robot's cameras are interfaced with an Intel NUC, which handles the camera drivers. The mobile base is connected to a Jetson Xavier NX, responsible for running the navigation stack and managing additional camera drivers. Communication between these

computational units is facilitated via a dedicated local network (Figure A.1). A list of the hardware components used in the robotic platform is provided in Table A.1.

A.1 ROBOT PLATFORM

The ABB YuMi (IRB 14000) is a dual-arm industrial robot, each arm offering seven degrees of freedom, specifically designed for tasks involving the assembly of lightweight parts. Each arm is equipped with a two-finger gripper at its end effector, and the robot has a combined payload capacity of 500 grams. For this work, a ROS package for YuMi, adapted from the GitHub repository yumi, is utilized to facilitate communication and programming. This package enables control of the robot in joint space and provides interfaces to operate the grippers either by applying a specified effort or positioning them precisely.

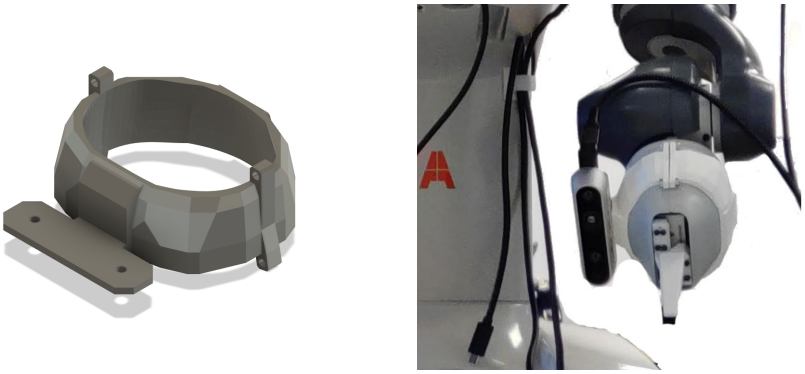
A.2 SENSORS



(a) 3D model (b) Camera Mounted

Figure A.2: **Realsense L515 Camera Mount**. The 3D printed mount is connected to YuMi with 4 screws on top and foam tape at the bottom. The position of the camera makes it free from potential collision with YuMi itself.

A RealSense L515 camera is mounted on the chest of the YuMi robot to provide a detailed view of the workspace. The L515 is a Light Detection and Ranging (LiDAR) camera capable of delivering both RGB images and point clouds. A custom-designed mount secures the camera to YuMi, as shown in Figure A.2. This carefully chosen position ensures the camera remains free from potential collisions while maintaining an unobstructed view of the manipulation target.



(a) 3D model (b) Camera Mounted

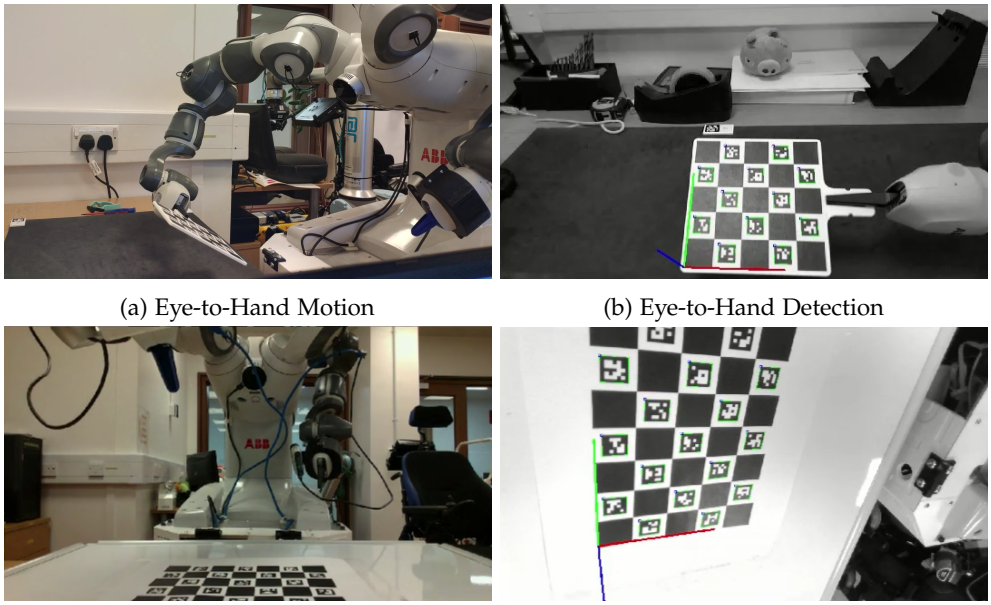
Figure A.3: **Realsense D435 Camera Mount**. The 3D-printed mount is clamped to the gripper with screws and nuts. This design makes the camera easily detachable and adjustable.

Two RealSense D435 cameras are mounted on the grippers of the YuMi robot to provide supplementary visual information. The D435 cameras are stereo cameras capable of capturing both RGB images and depth maps. These cameras are securely mounted on the grippers using a custom-designed mount, as shown in Figure A.3. This design enables easy detachment and adjustment of the cameras, offering flexibility in capturing various viewpoints during manipulation tasks.

A.3 HAND-EYE CALIBRATION

Hand-eye calibration is required to determine the transformation between the robot and the cameras. For the chest-mounted camera, calibration is performed by moving the gripper to various positions while capturing images of a custom-designed marker board, which is tailored to be easily grasped by the robot's fingers. The gripper is moved to 15 distinct positions within the workspace (Figure A.4a), and the corresponding gripper poses T_{EE}^{Robot} and marker board poses T_{Marker}^{Camera} are recorded. The transformation matrix T_{EE}^{Camera} is then computed with MoveIt Calibration (Daniilidis, 1999).

For the gripper-mounted cameras, calibration involves maintaining the marker board within the camera's view while moving the gripper to different positions (Figure A.4c). The gripper poses T_{EE}^{Robot} and marker board poses $T_{Marker}^{Gripper}$ are similarly recorded, and the transformation matrix $T_{EE}^{Gripper}$ is derived using MoveIt Calibration.



(a) Eye-to-Hand Motion (b) Eye-to-Hand Detection

(c) Eye-in-Hand Motion (d) Eye-in-Hand Detection

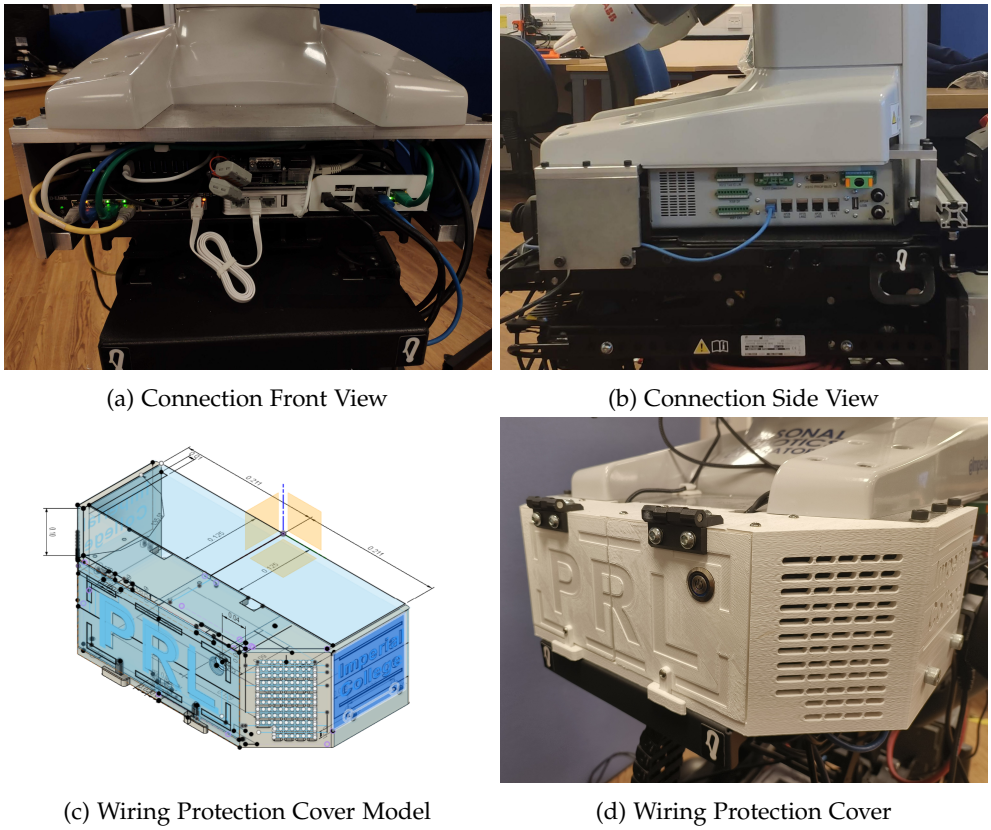
Figure A.4: **Hand-Eye Calibration**. Hand-to-Eye Calibration (Top): The gripper holds a custom-made marker board and moves to different positions in the workspace. The L515 camera captures images of the marker board, and the board poses are recorded. Hand-in-Eye Detection (Bottom): The marker board is located in the workspace, and the gripper moves to different positions while keeping the marker board in view. The gripper camera captures images of the marker board, and the gripper poses are recorded.

A.4 MOBILE PLATFORM

To enable the robot to manipulate objects over longer distances—such as tasks involving cable management in data centres—a mobile platform was developed. The base of this platform is adapted from a Quickie Salsa M2 Mini wheelchair, with its seating components removed. The YuMi robot is securely mounted onto the base using custom-designed steel connectors (Appendix A.4.1). The YuMi robot is powered by a lead-acid battery and an inverter. The full specifications of the components are listed in Table A.1. The fully assembled platform is depicted in Figure A.1.

A.4.1 Base-Robot Interface

The YuMi robot is rigidly connected to the wheelchair base using custom-designed steel connectors, as shown in Figure A.5b. These connectors are designed to be easily detachable with the removal of a few bolts and screws, allowing the robot to be separated from the base for maintenance or other



(a) Connection Front View

(b) Connection Side View

(c) Wiring Protection Cover Model

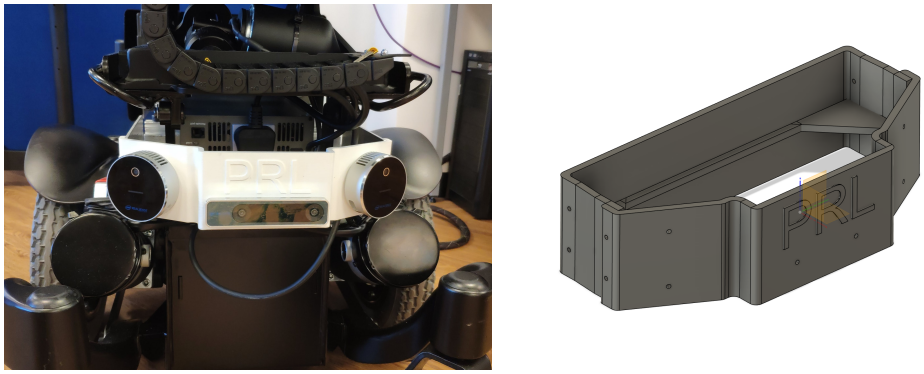
(d) Wiring Protection Cover

Figure A.5: Base-Robot Interface. The YuMi robot is rigidly connected to the wheelchair base using custom-designed steel connectors. A wiring protection cover is designed to shield the cables and wires powering the robot and the base.

purposes. The routers, switches, and a powered USE switch, as well as the connected cables and powering wires, are stored under the robot. To protect these components, a custom-designed wiring protection cover is installed, as shown in Figure A.5d.

A.4.2 Camera Module

To enable navigation for the mobile platform, suitable sensors are required to perform Simultaneous Localization and Mapping (SLAM). A RealSense T265 camera is used as the odometry source due to its accuracy and ease of use, while two RealSense L515 cameras are employed for mapping. A custom-designed and 3D-printed camera mount provides fixed transformations between the three cameras, as shown in Figure A.6. The cameras are connected to a Jetson Xavier NX, which runs the navigation stack and additional camera drivers, as shown in Figure A.1.



(a) Camera Module Installed (b) Camera Module Model

Figure A.6: **Camera module**. The 3D model of the camera module and the assembled product. The Realsense T265 provides odometry information. The 2 Realsense L515 on the sides are used for map construction.

A.4.3 The Navigation Stack

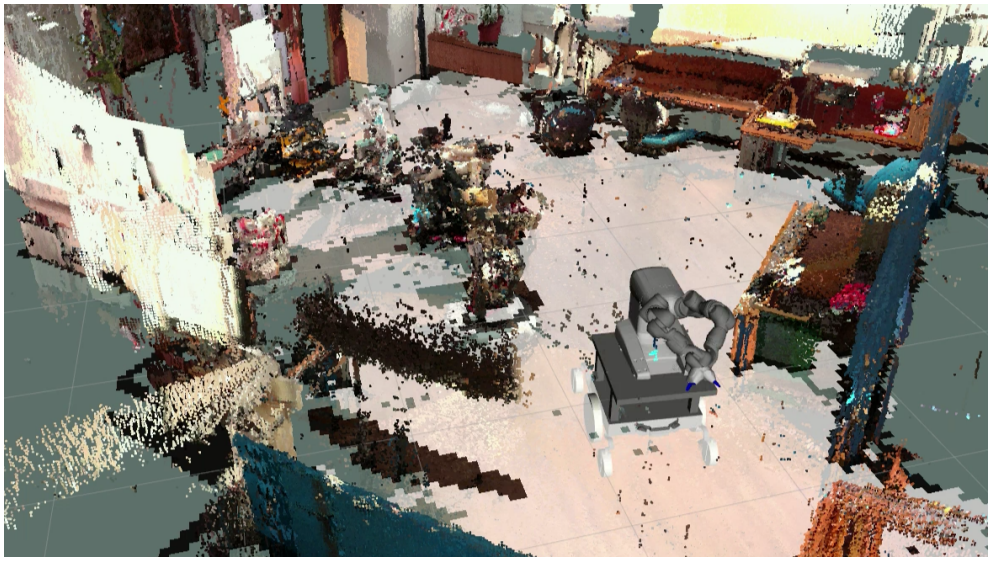


Figure A.7: **RTAB-Map with Move Base**.

With the camera module in place, the mobile base first needs to have an understanding of its surroundings. For this, RTAB-Map, a SLAM approach focused on 3D mapping, is employed. In the online mapping mode, the system navigates through unexplored areas while constructing maps of testing environments. Alternatively, the localization mode allows the robot to use a pre-constructed map to determine its position within a known environment. These modes provide the flexibility needed for both exploring new spaces and operating in previously mapped areas.

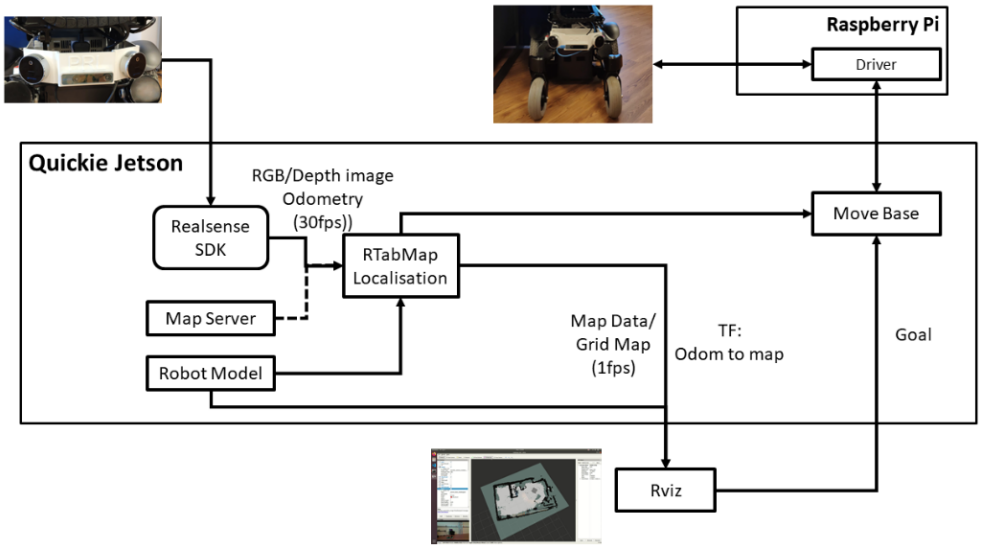


Figure A.8: **Navigation system**. The camera driver provides image data to Real-Time Appearance-Based Mapping (RTAB-Map), which generates a map of the environment. For pre-mapped environments, a map server is used instead to provide the necessary spatial information. The *Move Base* module utilizes the map and odometry data to calculate motion commands, which are then sent to the wheelchair driver for execution.

To navigate the RNet-based mobile base, the R-Net CAN library is used to send commands to the base. Control of the mobile base is managed by a Raspberry Pi, which interfaces with the R-Net control system via a CAN-Bus board. Through a web socket, the Raspberry Pi communicates with the Jetson Xavier, where control commands a computed. A simplified representation of the navigation system design is shown in Figure Figure A.8. The hardware components used for the mobile platform are listed in Table A.1.

The navigation and path planning of the mobile base are handled using the ROS *Move Base* package. This package receives a pose goal from user input and computes a sequence of command inputs required to reach the target, while avoiding obstacles in the environment. The *Move Base* package utilizes the map generated by RTAB-Map and odometry data from the RealSense T265 camera to calculate the motion commands. These commands are then sent to the driver running on a Raspberry Pi, which interfaces with the wheelchair's R-Net control system.

A.5 HARDWARE COMPONENTS

The hardware components used in the robotic platform are summarised in Table A.1.

Table A.1: Hardware Components

Component	**Description**
Robot	ABB IRB 14000 YuMi
Camera	Intel RealSense L515 x 3
Camera	Intel RealSense D435 x 2
Camera	Intel RealSense T265 x 1
Base	Sunrise Medical Wheelchair Quickie Salsa M2 Mini
Base Control	CAN-Bus Board for Raspberry Pi 3
Base Control	Raspberry Pi 3 B
Network Switch	D-Link Network Switch
Router	GL.iNet GL-AR750
USB Hub	StarTech.com 7 Port USB Hub
Battery	RS PRO Lead Acid Battery 727-0398(12V100Ah)
Inverter	Meanwell TS-700-212
Power Connection	Ring connector for M8 + 8 AWG wire (6 mm2)
Base Computer	Jetson Xavier NX
Robot Computer	Jetson AGX Orin
Sensor Computer	Intel NUC 12 Pro

B

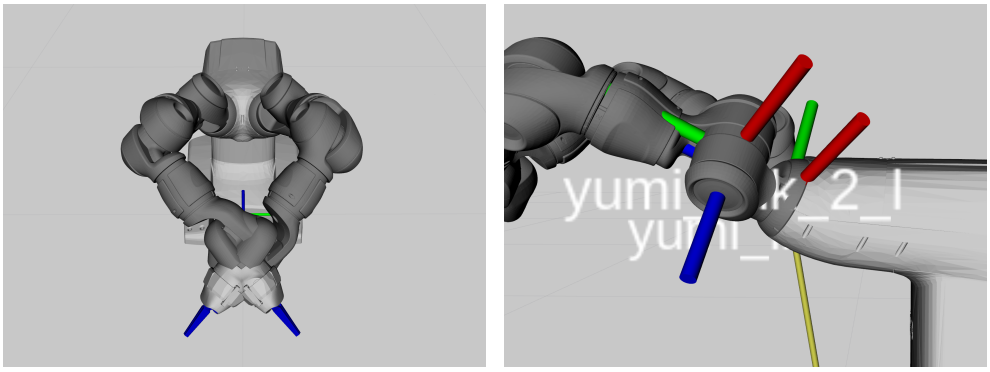
GUMI: A BI-MANUAL TELEOPERATION DEVICE

As introduced in Chapter 6, this chapter details the design and implementation of GuMi, a bi-manual teleoperation device developed for the ABB YuMi robot. The chapter begins by describing the design principles and construction of GuMi. It then provides the details of the hardware components of GuMi. Finally, it introduces the ROS interface developed for seamless integration with the teleoperation system.

B.1 DEVICE DESIGN

Designing a teleoperation device requires an analysis of the robot kinematics to determine the optimal joint configurations. This analysis serves as the foundation for designing the mechanical structure of the teleoperation device, ensuring compatibility and effective control of the robot.

B.1.1 Calculating The Joint Configuration



(a) RViz model of ABB YuMi (b) Transformation between YuMi joints

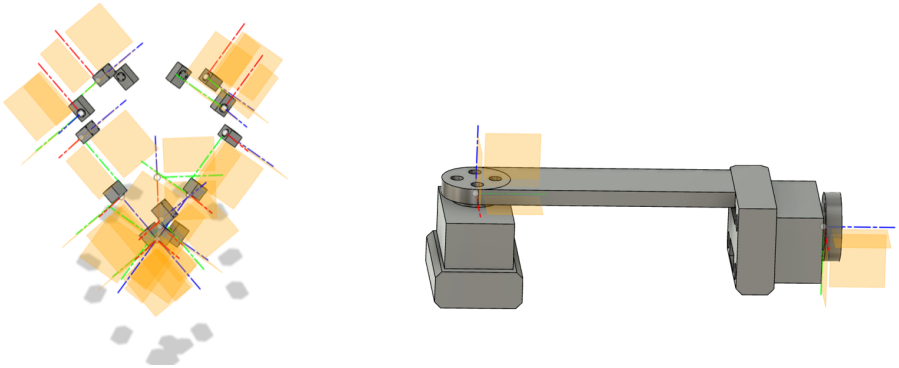
Figure B.1: Joint Configuration of ABB YuMi in RViz and the transformations between the joints.

In this section, the design process is demonstrated using ROS and RViz. The joint configuration of the teleoperation device is derived from the kinematic model of the ABB YuMi robot, as shown in Figure B.1a. By setting the joint angles to zero, the transformations between the robot joints can be calculated. An example of these transformations is provided in Figure B.1b. Setting the joint angles to zero ensures that the teleoperation device mirrors the robot configuration with all joints at their zero positions aligned. Using these transformations, the Euclidean positions of the joints can be calculated, enabling the following design of the device.

The transformations also provide the Euclidean distances between the joints. These distances, along with the size of the motors, can be used to determine the scale factor for the teleoperation device. The joint positions can then be scaled accordingly, ensuring that the device movements correspond accurately to the robot kinematic configuration.

B.1.2 Mechanical Design

Once the joint positions are calculated, the mechanical design of the teleoperation device can be created using any Computer-Aided Design (CAD) software. In this section, the mechanical design of GuMi is demonstrated using AutoDesk Fusion 360.

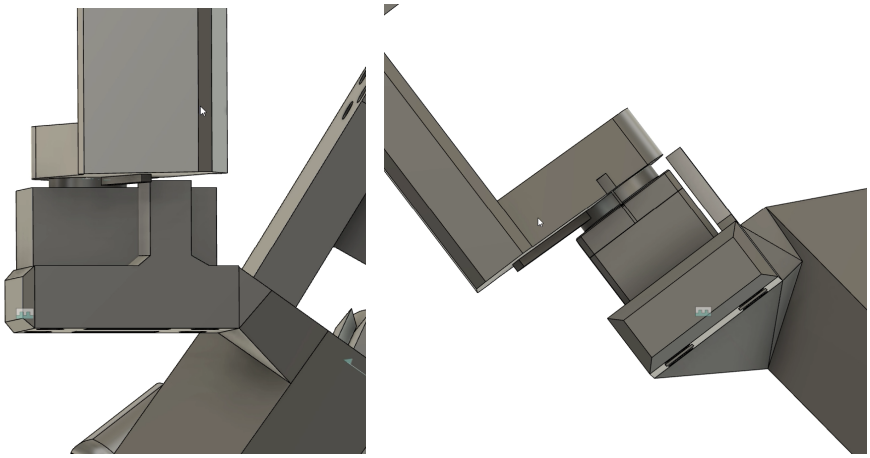


(a) Motors positioned (b) Motor connection

Figure B.2: Motor positioning and connection in GuMi. The motors are positioned according to the calculated joint positions.

Figure B.2a illustrates the placement of the motor models in the CAD software based on the calculated joint positions. The configuration of the motor positions demonstrates the resemblance to the configuration of the actual robot, as shown in Figure B.1a, ensuring consistency between the teleoperation device and the robot it emulates. Following the positioning of

the motors, the design of mechanical connections between them can be carried out. These connections are critical for transmitting motion and maintaining structural integrity. An example of such a mechanical connection is provided in Figure B.2b, highlighting an example of the integration of the components in the design.

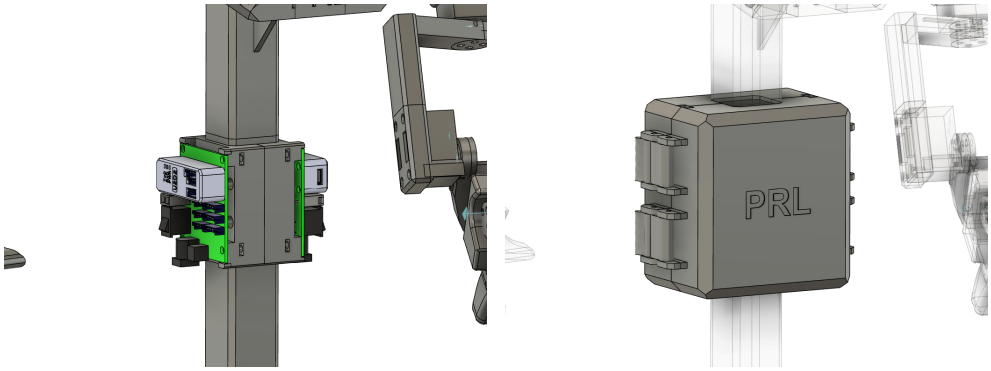


(a) Motor stopper (b) Motor alignment marker

Figure B.3: Motor stopper and alignment in GuMi. The stopper prevents the motor from rotating more than the joint limits. The alignment provides references for the motor positions.

Some additional features can be added to the mechanical design when creating the connections. During teleoperation, the robot needs to replicate the configuration of the teleoperation device. Without physical constraints, the motors can be rotated more than the joint limits, which can cause damage to the robot or prevent the robot from performing the teleoperation commands. To this end, Stoppers can be added to the motor connections to keep the motor angles within the joint limits. Figure B.3a shows an example of a motor stopper. At the initial stages, the teleoperation device typically needs to be set to align with the robot configuration. This can be difficult since the teleoperation device does not have the same appearance as the robot. To properly align the teleoperation device with the real robot, alignment markers can also be added to the motors to provide references for the motor positions. This can be achieved with small intrusions on the motor connections. Figure B.3b shows an example of a motor alignment.

Although the electrical connections involve low voltage (5V), to prevent potential risks of contacting sensitive skin areas (e.g. open wounds), or sparks in the case of short circuits, the power circuits are protected with a wiring box (Figure B.4). The box is designed to be clamped onto the main pillar. It



(a) GuMi Circuits (b) GuMi Wiring Box

Figure B.4: GuMi Power Circuit Protection. Although the electrical connections involve low voltage (5V), the power circuits are protected with a box to prevent potential risks. The box is designed to be easily opened for maintenance.

can be easily removed from the teleoperation device with the clips from both sides of the box for maintenance or other purposes.

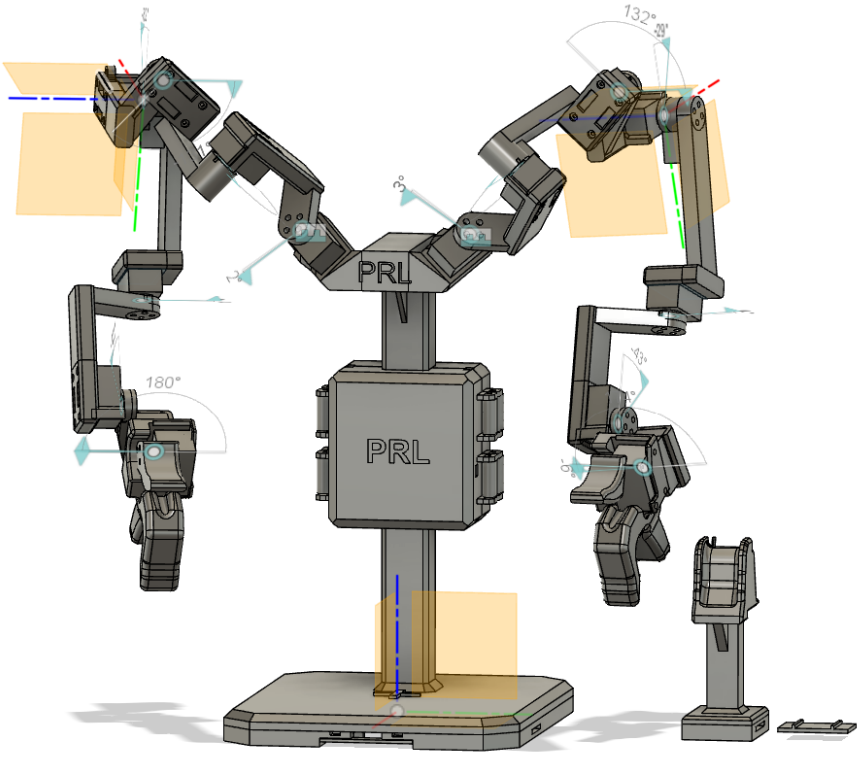


Figure B.5: GuMi model. The supports are designed to allow the device to be mounted on a table or a stand with a clamp. The right side shows a rest support for the grippers.

Once the mechanical connections are designed, the teleoperation device is mostly completed. The supports for the teleoperation device need to be considered in this stage. Common supports allow the device to be mounted on a table or a stand with a clamp. Figure B.5 shows the final model of GuMi.

B.1.3 Gravity Compensation

Overall there are two types of gravity compensation methods: cheaper rubber bands (Wu et al., 2023) and more expensive tool balancers (Zhao et al., 2024). Also, the rubber bands are more flexible and can be easily replaced, it is difficult to adjust the tension of the rubber bands properly for all scenarios, especially when the robot has a high degree of freedom. The tool balancer on the other hand, although restricts the movement of the teleoperation device and is also not a perfect solution, can be adjusted to the weight of the tool and it is easier to set up.

B.2 HARDWARE COMPONENTS



Figure B.6: GuMi fully assembled. The device is 75% scale of the actual robot.

The model can be then 3D printed and assembled. Figure B.6 shows the final product of GuMi. The device is 75% scale of the actual robot. It is connected to an IKEA kitchen desk with two clamps. The gravity compensation is provided by a tool balancer, which is attached to the shelf above the desk through 2 RS PRO 20x20mm Aluminium Profile Struct.

The hardware components of GuMi are listed in Table B.1. The servo motors are equipped with high-resolution encoders which provide measurements of the joint angles. The motors are connected with Dynamixel X3P Robot Cables. The port sequence of the motors does not matter. The U2D2 is connected to the computer through a powered USB hub.

Table B.1: Hardware components of GuMi

Component	Description
Joint Motors	Dynamixel XL330-M288-T
Gripper Motors	Dynamixel XL330-M077-T
Motor Controller	ROBOTIS - U2D2
Power Supply	RS PRO 24W AC/DC Adapter 5V DR Output
Gravity Compensation	Aeromotive RB2 Tool Balancer (0.5-0.9kg)
Fixation	Kennedy 200mm F-Clamp
3D Printer	Prusa MK4IS
Filament	Prusa PLA Galaxy Black

B.3 CALCULATING MOTOR OFFSETS

The motor readings are not always zero when the assembled links are at zero angles. To compensate for this, the motor offsets need to be calculated. This is achieved by setting the motor angles to zero using the alignment markers and measuring the motor readings. The offsets are stored in the interface of the teleoperation device and are subtracted from the motor readings.

B.4 GUMI-ROS INTERFACE

One of the most common ways to interface a teleoperation device with a robot is through ROS. Inspired by Zhao et al. (2023), in the setup of GUMI, the teleoperation device is connected to the robot through ROS positions controllers. The ROS interface of GUMI publishes the current configuration of the teleoperation device with the states of the grippers at 30Hz. The commands are subscribed by the robot position controller which is then sent to the robot.

Different from the ROS interface of the robot described in Appendix A.1. For teleoperation, a YuMi ROS package ABB Robot Driver and a RobotWare Add-In State Machine are used to communicate with YuMi and programme its actions. ABB Externally Guided Motion (EGM) is used to control the robot. It allows the robot controller to communicate with the computer through

User Datagram Protocol (UDP) connection. The default maximum speed deviation is set to 180 degrees per second and motion supervision is disabled.

C

SIMULATION WITH UNITY3D

During the development of the robot simulation environment, Unity interfaces were created to communicate with ROS packages and Python scripts. In this chapter, the Unity interfaces for robot simulation are presented. The chapter is divided into two sections. The first section presents the Unity interface to communicate with ROS packages. The second section presents high-speed the Unity interface to communicate with Python scripts.

C.1 UNITY INTERFACE FOR ROBOT SIMULATION

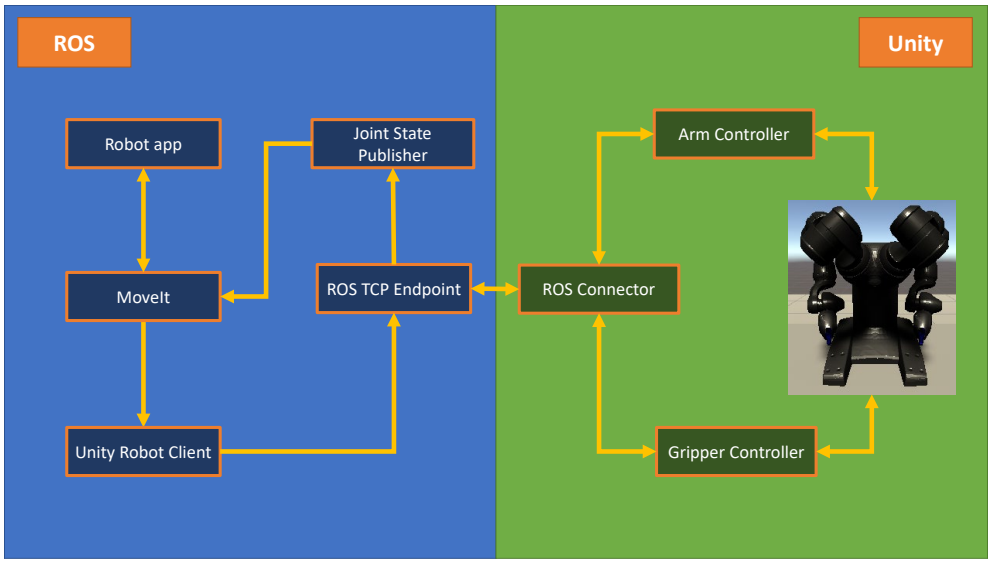


Figure C.1: Unity ROS Interface Structure. The robot app contains the main process of the robot behaviour. It sends motion requests to MoveIt which plans the motion and sends the motion commands to Unity for execution. The Unity interface communicates with ROS through a TCP/IP connection.

Unity developers have been trying to integrate Unity with ROS for robot simulation and motion planning. ROS-TCP-Connector is a Unity package that allows Unity to communicate with ROS. The package uses TCP/IP to send

and receive messages between Unity and ROS. On top of the communication package, Unity developers also created Unity Robotics Hub to facilitate the development of robot simulation in Unity. The package provides a set of tools and examples to use MoveIt for robot motion planning in Unity.

However, the logic that Unity Robotics Hub is not very intuitive for robot developers. The package sets MoveIt as a planning server and all the planning requests and main procedures are written in Unity. A robot programmer might want instead use Unity only as a simulation environment that accepts motion commands and sends state information to ROS. As a result, this work presents an alternative Unity interface for robot simulation that allows users to treat Unity purely as a simulator [1].

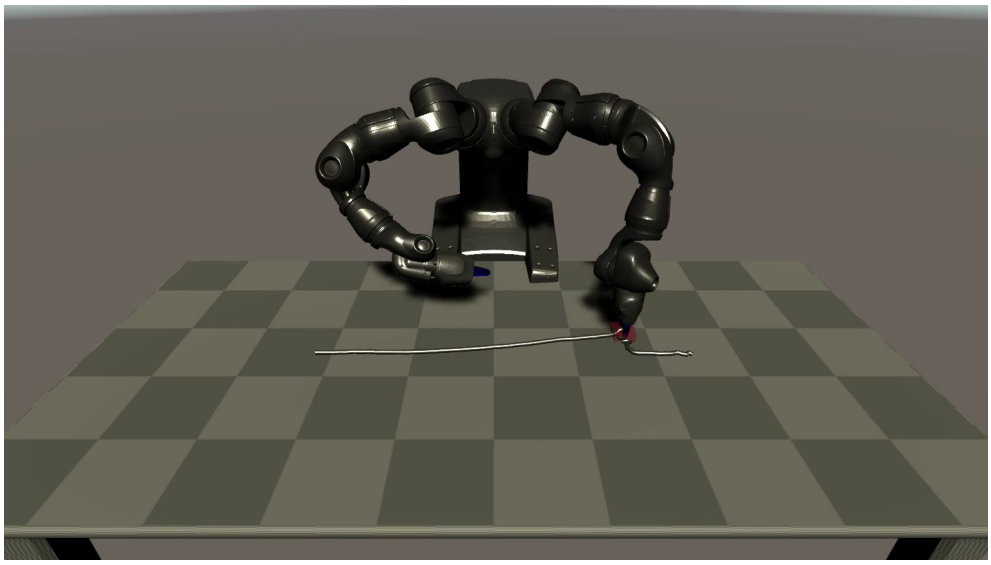


Figure C.2: Unity ROS Interface Demo. The robot arm motion is planned with MoveIt. The demo moves the robot arm to grasp a simulated rope and drop it to a target position.

The proposed Unity interface for robot simulation is shown in Figure C.1. The robot app contains the main process of the robot behaviour. It sends motion requests to MoveIt which plans the motion and sends the motion commands to the Unity Robot Client. Unity Robot Client is a ROS Hardware interface that communicates with Unity through a TCP/IP connection. The Unity interface receives the motion commands and sends them to the robot simulation in Unity for execution. The arm and gripper states are simultaneously published back to ROS for monitoring and control. Figure C.2 shows a demo of the Unity ROS interface. The robot arm motion is planned with

1 This interface is open-source on https://github.com/HainingLuo/UnityRoboticsDemo.

MoveIt. The demo moves the robot arm to grasp a simulated rope and drag it to a target position.

C.2 FAST COMMUNICATION BETWEEN UNITY AND PYTHON WITH GRPC

In some applications such as Reinforcement Learning (RL), the communication between Unity and Python needs to be fast. Using ROS communication in this case may not be the most optimal solution. Google Remote Procedure Call (gRPC) is a high-speed communication protocol that can be used to communicate between Unity and Python. The gRPC package is available in both C# and Python. However, due to the lack of support on HTTP/2 in Unity, it has been difficult to use gRPC to communicate with Python.

From the Python or C++ side, the gRPC server can be set up easily by installing the gRPC package and defining gRPC service in proto files. However, on the Unity side, the most recent available gRPC package is 1.14.1. The package can be downloaded and imported into Unity. An example of the gRPC client in Unity is the ml-agent package developed by Unity. However, it is not very straightforward to use the package for general purposes.

One important note is that Unity can only be used as gRPC clients. The nature of Unity as a game engine makes it difficult to handle concurrent connections and high-throughput traffic. It is recommended to use Unity as a client and Python as a server in gRPC communication.

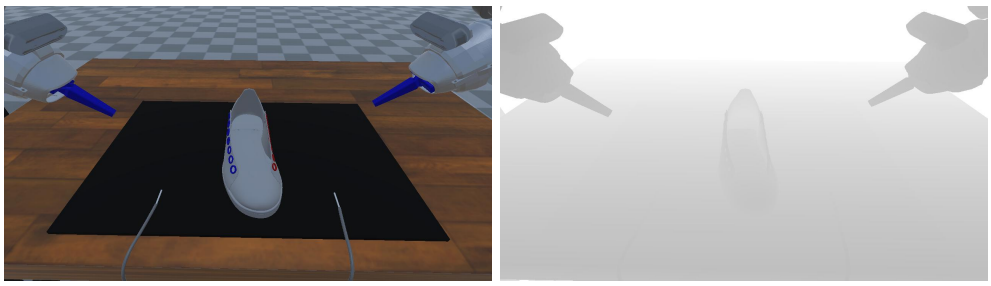
C.3 SIMULATING CAMERAS IN UNITY

Unity cameras are used to render simulation scenes from different perspectives. The cameras have many parameters that can be adjusted to achieve different rendering effects. This thesis presents a camera package that supports both RGB and depth cameras in Unity. The Unity side of the package is implemented with C# scripts and shaders. The Python side of the package is implemented with Python scripts. The package is released as a part of the UniLace project.

C.3.1 Shader For Depth Cameras

Although commonly used in robotics, depth cameras are not natively supported in Unity. The machine learning package ml-agent developed by Unity does not support depth cameras as well. This work develops a custom shader

for depth cameras in Unity and transfers the depth images to Python as byte arrays. Supporting scripts on the Python side can convert the byte arrays to depth images for further processing.



(a) RGB Camera (b) Depth Camera

Figure C.3: RGB and Depth Camera in Unity. The depth camera is rendered with a custom shader.

In the rendering process, Unity first decides the objects visible in the camera view. Those objects occluded or out of the camera view are culled. The rendering pipeline then calculates the 2D scene coordinates and the pixel values, considering the lighting and shadows. In this process, shaders define the appearance of the objects, such as the texture, reflection, transparency, or complex visual effects.

A shader in Unity commonly includes two parts: vertex shader and fragment shader. The vertex shader calculates the 2D scene coordinates of the objects. The fragment shader calculates the pixel values of the objects based on the output of the vertex shader. In this work, the vertex shader calculates the depth values of the objects with respect to the camera. The fragment shader then outputs the depth values as pixel values.

Real depth cameras often have noises in the depth images. The noises are added to the depth images in the shader to simulate the real depth cameras. The shader generates random Gaussian noises based on the pixel locations and the depth values. The variance of this noise is controlled by a parameter in the shader. A demo of the RGB and depth cameras in Unity is shown in Figure C.3.

C.3.2 Camera Parameters

The camera parameters can be adjusted in the Unity editor. The parameters define the intrinsic and extrinsic properties of the cameras. This section presents a set of camera parameters that are commonly used. The open-

source repository also supports modifying these parameters in the YAML config file.

1. **Resolution**: This defines the resolution of the camera image. The resolution of the image affects the quality and detail of the captured image. It is important to note that the visible range of a camera is dictated by the Field of View (FoV), as well as Focal Length and Sensor Size.
2. **FoV**: The field of view of the camera in degrees. The FoV is the extent of the observable world that is seen given any moment.
3. **Focus Distance**: The distance from the camera to the point of focus.
4. **Focus Length**: The distance between the lens and the image sensor.
5. **Sensor Size**: The size of the camera sensor. The sensor size affects the visible range of the camera.
6. **Lens Shift**: The lens shift of the camera. The lens shift is used to correct the perspective distortion.
7. **Clipping Depth**: The near and far clipping planes of the camera. The near-clipping plane is the closest distance that the camera can render. The far-clipping plane is the farthest distance that the camera can render.

Apart from the above parameters, the package also supports adjusting the depth noise standard deviation and the frame rate of the camera. The depth noise standard deviation is set to 0 by default. This parameter can be adjusted according to a physical camera. For example, the Intel RealSense L515 has a depth error rate of 0.25%, which can be converted to a standard deviation of 0.0025. The frame rate of the camera heavily affects the performance of the simulation. This should be adjusted according to the computational resources available.

C.3.3 Transferring Images Between Unity And ROS

The images are transferred from Unity to ROS as byte arrays and stored in *Image* messages. The images are then converted to other formats for further processing. It is important to note that the images are coded upside down in Unity and need to be flipped before other conversions.

BIBLIOGRAPHY

Akbulut, S. and King, H. (1981), 'All knots are algebraic', *Commentarii Mathematici Helvetici* **56**, 339–351. (page 32).

Almaghout, K., Cherubini, A. and Klimchik, A. (2024), 'Robotic co-manipulation of deformable linear objects for large deformation tasks', *Robotics and Autonomous Systems* p. 104652. (page 33).

Bäckström, C. and Nebel, B. (1995), 'Complexity results for sas+ planning', *Computational Intelligence* **11**(4), 625–655. (page 33).

Baksic, P., Courtecuisse, H., Duriez, C. and Bayle, B. (2020), Robotic needle insertion in moving soft tissues using constraint-based inverse finite element simulation, *in* '2020 IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 2407–2413. (pages 29, 33).

Bender, J., Müller, M., Otaduy, M. A., Teschner, M. and Macklin, M. (2014), A survey on position-based simulation methods in computer graphics, *in* 'Computer graphics forum', Vol. 33, Wiley Online Library, pp. 228–251. (page 34).

Bhattacharya, S., Likhachev, M. and Kumar, V. (2011), Identification and representation of homotopy classes of trajectories for search-based path planning in 3d., *in* 'Robotics: Science and Systems', pp. 27–30. (page 39).

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choromanski, K., Ding, T., Driess, D., Dubey, A., Finn, C. and others (2023), 'Rt-2: Vision-language-action models transfer web knowledge to robotic control', *arXiv preprint arXiv:2307.15818* . (page 115).

Brohan, A., Chebotar, Y., Finn, C., Hausman, K., Herzog, A., Ho, D., Ibarz, J., Irpan, A., Jang, E., Julian, R. and others (2023), Do as i can, not as i say: Grounding language in robotic affordances, *in* 'Conference on robot learning', PMLR, pp. 287–318. (page 115).

Calli, B., Walsman, A., Singh, A., Srinivasa, S., Abbeel, P. and Dollar, A. M. (2015), 'Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set', *IEEE Robotics & Automation Magazine* **22**(3), 36–52. (pages 21, 67).

Cao, M., Cao, K., Yuan, S., Nguyen, T.-M. and Xie, L. (2023), 'Neptune: nonentangling trajectory planning for multiple tethered unmanned vehicles', *IEEE Transactions on Robotics* **39**(4), 2786–2804. (page 39).

Caporali, A., Galassi, K., Zanella, R. and Palli, G. (2022), 'Fastdlo: Fast deformable linear objects instance segmentation', *IEEE Robotics and Automation Letters* **7**(4), 9075–9082. (page 37).

Caporali, A., Kicki, P., Galassi, K., Zanella, R., Walas, K. and Palli, G. (2024), 'Deformable linear objects manipulation with online model parameters estimation', *IEEE Robotics and Automation Letters* . (pages 76, 114, 116).

Caporali, A., Pantano, M., Janisch, L., Regulin, D., Palli, G. and Lee, D. (2023), 'A weakly supervised semi-automatic image labeling approach for deformable linear objects', *IEEE Robotics and Automation Letters* **8**(2), 1013–1020. (page 116).

Chacón Quesada, R. and Demiris, Y. (2024), 'Multi-dimensional evaluation of an augmented reality head-mounted display user interface for controlling legged manipulators', *ACM Transactions on Human-Robot Interaction* **13**(2), 1–37. (page 104).

Chatzilygeroudis, K., Fichera, B., Lauzana, I., Bu, F., Yao, K., Khadivar, F. and Billard, A. (2020), 'Benchmark for bimanual robotic manipulation of semi-deformable objects', *IEEE Robotics and Automation Letters* **5**(2), 2443–2450. (pages 21, 36).

Chen, L. Y., Shi, B., Seita, D., Cheng, R., Kollar, T., Held, D. and Goldberg, K. (2023), Autobag: Learning to open plastic bags and insert objects, *in* '2023 IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 3918–3925. (page 116).

Chen, S., Xu, Y., Yu, C., Li, L., Ma, X., Xu, Z. and Hsu, D. (2022), Daxbench: Benchmarking deformable object manipulation with differentiable physics, *in* 'The Eleventh International Conference on Learning Representations'. (page 37).

Cheng, X., Li, J., Yang, S., Yang, G. and Wang, X. (2024), 'Open-television: teleoperation with immersive active visual feedback', *arXiv preprint arXiv:2407.01512* . Retrieved from https://doi.org/10.48550/arXiv.2407.01512 (pages 43, 103).

Cherubini, A., Leitner, J., Ortenzi, V. and Corke, P. (2018), Towards vision-based manipulation of plastic materials, *in* '2018 IEEE/RSJ International

Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 485–490. (page 40).

Chi, C. and Berenson, D. (2019), 'Occlusion-robust deformable object tracking without physics simulation. in 2019 ieee/rsj int. conf. on intelligent robots and systems (iros)'. (pages 22, 29, 38).

Chui, H. and Rangarajan, A. (2000), A feature registration framework using mixture models, *in* 'Proceedings IEEE workshop on mathematical methods in biomedical image analysis. MMBIA-2000 (Cat. No. PR00737)', IEEE, pp. 190–197. (page 38).

Clayton-Jones, M. C. (2021), 'Nurse-delivered shoe-lacing intervention: Effect on comfort and toe pressures for active community-dwelling adults (age 65+)'. (page 20).

Clegg, A., Yu, W., Erickson, Z., Tan, J., Liu, C. K. and Turk, G. (2017), Learning to navigate cloth using haptics, *in* '2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 2799–2805. (page 35).

Community, B. O. (2018), *Blender - a 3D modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam. Retrieved from `http://www.blender.org` (page 34).

Coumans, E. and Bai, Y. (2017), 'Pybullet, a python module for physics simulation in robotics, games and machine learning'. (page 34).

Coumans, E. and others (2013), 'Bullet physics library', *Open source: bullet-physics. org* **15**(49), 5. (page 34).

Daniilidis, K. (1999), 'Hand-eye calibration using dual quaternions', *The International Journal of Robotics Research* **18**(3), 286–298. (page 123).

Darvish, K., Penco, L., Ramos, J., Cisneros, R., Pratt, J., Yoshida, E., Ivaldi, S. and Pucci, D. (2023), 'Teleoperation of humanoid robots: A survey', *IEEE Transactions on Robotics* **39**(3), 1706–1727. (page 41).

David, W., Howe, A., Knoblock, C., McDermott, I. D., Ram, A., Veloso, M., Weld, D., Sri, D. W., Barrett, A., Christianson, D. and others (1998), 'Pddl| the planning domain definition language', *Technical Report, Tech. Rep.* . (page 33).

De Gregorio, D., Zanella, R., Palli, G., Pirozzi, S. and Melchiorri, C. (2018), 'Integration of robotic vision and tactile sensing for wire-terminal insertion

tasks', *IEEE Transactions on Automation Science and Engineering* **16**(2), 585–598. (pages 19, 29, 117).

Deb, K. (2000), 'An efficient constraint handling method for genetic algorithms', *Computer methods in applied mechanics and engineering* **186**(2-4), 311–338. (page 51).

Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T. (2002), 'A fast and elitist multiobjective genetic algorithm: Nsga-ii', *IEEE transactions on evolutionary computation* **6**(2), 182–197. (page 51).

Ding, R., Qin, Y., Zhu, J., Jia, C., Yang, S., Yang, R., Qi, X. and Wang, X. (2024), 'Bunny-visionpro: Real-time bimanual dexterous teleoperation for imitation learning', *CoRR* . (pages 43, 98).

Dwivedi, V., Chauhan, T., Saxena, S. and Agrawal, P. (2012), 'Travelling salesman problem using genetic algorithm', *IJCA proceedings on development of reliable information systems, techniques and related issues (DRISTI 2012)* **1**(1), 25. (page 51).

Erez, T., Tassa, Y. and Todorov, E. (2015), Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx, *in* '2015 IEEE international conference on robotics and automation (ICRA)', IEEE, pp. 4397–4404. (page 34).

Erickson, Z., Gangaram, V., Kapusta, A., Liu, C. K. and Kemp, C. C. (2020), Assistive gym: A physics simulation framework for assistive robotics, *in* '2020 IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 10169–10176. (pages 21, 34, 36).

Ficuciello, F., Migliozzi, A., Coevoet, E., Petit, A. and Duriez, C. (2018), Fem-based deformation control for dexterous manipulation of 3d soft objects, *in* '2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 4007–4013. (pages 33, 35).

Fikes, R. E. and Nilsson, N. J. (1971), 'Strips: A new approach to the application of theorem proving to problem solving', *Artificial intelligence* **2**(3-4), 189–208. (page 33).

Garcia-Camacho, I., Borràs, J., Calli, B., Norton, A. and Alenyà, G. (2022), 'Household cloth object set: Fostering benchmarking in deformable object manipulation', *IEEE Robotics and Automation Letters* **7**(3), 5866–5873. (page 36).

Garcia-Camacho, I., Lippi, M., Welle, M. C., Yin, H., Antonova, R., Varava, A., Borras, J., Torras, C., Marino, A., Alenya, G. and others (2020), 'Benchmarking bimanual cloth manipulation', *IEEE Robotics and Automation Letters* **5**(2), 1111–1118. (pages 21, 36).

Ge, S., Fan, G. and Ding, M. (2014), Non-rigid point set registration with global-local topology preservation, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops', pp. 245–251. (pages 38, 82).

Grannen, J., Sundaresan, P., Thananjeyan, B., Ichnowski, J., Balakrishna, A., Viswanath, V., Laskey, M., Gonzalez, J. E. and Goldberg, K. (2020), Learning robot policies for untangling dense knots in linear deformable structures, *in* 'Conference on Robot Learning (CoRL)'. (pages 29, 32, 33).

Guo, H., Wu, F., Qin, Y., Li, R., Li, K. and Li, K. (2023), 'Recent trends in task and motion planning for robotics: A survey', *ACM Computing Surveys* **55**(13s), 1–36. (page 32).

Hagen, M. and Hennig, E. M. (2009), 'Effects of different shoe-lacing patterns on the biomechanics of running shoes', *Journal of Sports Sciences* **27**(3), 267–275. (page 20).

Haslum, P., Lipovetzky, N., Magazzeni, D., Muise, C., Brachman, R., Rossi, F. and Stone, P. (2019), *An introduction to the planning domain definition language*, Vol. 13, Springer. (pages 31, 33).

Hernández, E., Carreras, M., Antich, J., Ridao, P. and Ortiz, A. (2011), A topologically guided path planner for an auv using homotopy classes, *in* '2011 IEEE International Conference on Robotics and Automation', IEEE, pp. 2337–2343. (page 39).

Hernandez, E., Carreras, M. and Ridao, P. (2015), 'A comparison of homotopic path planning algorithms for robotic applications', *Robotics and Autonomous Systems* **64**, 44–58. (page 39).

Hu, Y., Anderson, L., Li, T.-M., Sun, Q., Carr, N., Ragan-Kelley, J. and Durand, F. (2019), 'Difftaichi: Differentiable programming for physical simulation', *arXiv preprint arXiv:1910.00935* . (page 77).

Hu, Y., Liu, J., Spielberg, A., Tenenbaum, J. B., Freeman, W. T., Wu, J., Rus, D. and Matusik, W. (2019), Chainqueen: A real-time differentiable physical simulator for soft robotics, *in* '2019 International conference on robotics and automation (ICRA)', IEEE, pp. 6265–6271. (page 77).

Huang, I., Narang, Y., Eppner, C., Sundaralingam, B., Macklin, M., Bajcsy, R., Hermans, T. and Fox, D. (2022), 'Defgraspsim: Physics-based simulation of grasp outcomes for 3d deformable objects', *IEEE Robotics and Automation Letters* **7**(3), 6274–6281. (page 36).

Huang, X., Chen, D., Guo, Y., Jiang, X. and Liu, Y. (2023), 'Untangling multiple deformable linear objects in unknown quantities with complex backgrounds', *IEEE Transactions on Automation Science and Engineering* **21**(1), 671–683. (pages 29, 32).

Huang, Z., Hu, Y., Du, T., Zhou, S., Su, H., Tenenbaum, J. B. and Gan, C. (2021), Plasticinelab: A soft-body manipulation benchmark with differentiable physics, *in* 'Ninth International Conference on Learning Representations (ICLR)'. (page 21).

Hulin, T., Hertkorn, K., Kremer, P., Schätzle, S., Artigas, J., Sagardia, M., Zacharias, F. and Preusche, C. (2011), The dlr bimanual haptic device with optimized workspace, *in* '2011 IEEE International Conference on Robotics and Automation', IEEE, pp. 3441–3442. Retrieved from https://doi.org/10.1109/ICRA.2011.5980066 (page 42).

Hussein, A., Gaber, M. M., Elyan, E. and Jayne, C. (2017), 'Imitation learning: A survey of learning methods', *ACM Computing Surveys (CSUR)* **50**(2), 1–35. (page 40).

Iyer, A., Peng, Z., Dai, Y., Guzey, I., Haldar, S., Chintala, S. and Pinto, L. (2024), 'Open teach: A versatile teleoperation system for robotic manipulation', *arXiv preprint arXiv:2403.07870* . Retrieved from https://doi.org/10.48550/arXiv.2403.07870 (page 42).

Jang, E., Irpan, A., Khansari, M., Kappler, D., Ebert, F., Lynch, C., Levine, S. and Finn, C. (2022), Bc-z: Zero-shot task generalization with robotic imitation learning, *in* 'Conference on Robot Learning', PMLR, pp. 991–1002. (page 115).

Jia, B., Pan, Z., Hu, Z., Pan, J. and Manocha, D. (2019), 'Cloth manipulation using random-forest-based imitation learning', *IEEE Robotics and Automation Letters* **4**(2), 2086–2093. (page 40).

Kawaharazuka, K., Matsushima, T., Gambardella, A., Guo, J., Paxton, C. and Zeng, A. (2024), 'Real-world robot applications of foundation models: A review', *Advanced Robotics* **38**(18), 1232–1254. (page 115).

Keipour, A., Bandari, M. and Schaal, S. (2022), 'Deformable one-dimensional object detection for routing and manipulation', *IEEE Robotics and Automation Letters* **7**(2), 4329–4336. (page 88).

Kim, J. W., Zhao, T. Z., Schmidgall, S., Deguet, A., Kobilarov, M., Finn, C. and Krieger, A. (2024), 'Surgical robot transformer (srt): Imitation learning for surgical tasks', *arXiv preprint arXiv:2407.12998* . (page 42).

Kim, J. W., Zhao, T. Z., Schmidgall, S., Deguet, A., Kobilarov, M., Finn, C. and Krieger, A. (n.d.), Surgical robot transformer (srt): Imitation learning for surgical tasks, *in* '8th Annual Conference on Robot Learning'. (page 98).

Kruschke, J. (2014), *Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan*, Elsevier. Retrieved from https://www.sciencedirect.com/book/9780124058880 (page 105).

Kruschke, J. K. (2013), 'Bayesian estimation supersedes the t test', *Journal of Experimental Psychology: General* **142**(2), 573. Retrieved from https://doi.org/10.1037/a0029146 (page 104).

Kruschke, J. K. (2018), 'Rejecting or accepting parameter values in Bayesian estimation', *Advances in methods and practices in psychological science* **1**(2), 270–280. (pages 106, 107).

Kudoh, S., Gomi, T., Katano, R., Tomizawa, T. and Suehiro, T. (2015), In-air knotting of rope by a dual-arm multi-finger robot, *in* '2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 6202–6207. (page 41).

Laezza, R., Gieselmann, R., Pokorny, F. T. and Karayiannidis, Y. (2021), Reform: A robot learning sandbox for deformable linear object manipulation, *in* '2021 IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 4717–4723. (page 36).

Lagriffoul, F., Dantam, N. T., Garrett, C., Akbari, A., Srivastava, S. and Kavraki, L. E. (2018), 'Platform-independent benchmarks for task and motion planning', *IEEE Robotics and Automation Letters* **3**(4), 3765–3772. (page 31).

Larranaga, P., Kuijpers, C. M., Murga, R. H. and Yurramendi, Y. (1996), 'Learning bayesian network structures by searching for the best ordering with genetic algorithms', *IEEE transactions on systems, man, and cybernetics-part A: systems and humans* **26**(4), 487–493. (page 51).

Lee, A. X., Lu, H., Gupta, A., Levine, S. and Abbeel, P. (2015), Learning force-based manipulation of deformable objects from multiple demonstrations, *in* '2015 IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 177–184. (page 41).

Lee, Y., Hu, E. S. and Lim, J. J. (2021), Ikea furniture assembly environment for long-horizon complex manipulation tasks, *in* '2021 IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 6343–6349. (page 21).

Liang, W., Xie, J., Wang, Z., Tan, J. and Ma, X. (2024), 'Constrained behavior cloning for robotic learning', *arXiv preprint arXiv:2408.10568* . Retrieved from https://doi.org/10.48550/arXiv.2408.10568 (pages 23, 41, 97).

Lim, V., Huang, H., Chen, L. Y., Wang, J., Ichnowski, J., Seita, D., Laskey, M. and Goldberg, K. (2022), Real2sim2real: Self-supervised learning of physical single-step dynamic actions for planar robot casting, *in* '2022 International Conference on Robotics and Automation (ICRA)', IEEE, pp. 8282–8289. (page 77).

Lin, X., Qi, C., Zhang, Y., Huang, Z., Fragkiadaki, K., Li, Y., Gan, C. and Held, D. (2023), Planning with spatial-temporal abstraction from point clouds for deformable object manipulation, *in* 'Conference on Robot Learning', PMLR, pp. 1640–1651. (page 21).

Lin, X., Wang, Y., Olkin, J. and Held, D. (2021), Softgym: Benchmarking deep reinforcement learning for deformable object manipulation, *in* 'Conference on Robot Learning', PMLR, pp. 432–448. (pages 21, 36).

Lui, W. H. and Saxena, A. (2013), Tangled: Learning to untangle ropes with rgb-d perception, *in* '2013 IEEE/RSJ International Conference on Intelligent Robots and Systems', IEEE, pp. 837–844. (page 32).

Luo, H. and Demiris, Y. (2023), Bi-manual robot shoe lacing, *in* '2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 8772–8775. (pages 45, 89).

Luo, H. and Demiris, Y. (2024), 'Benchmarking and simulating bimanual robot shoe lacing', *IEEE Robotics and Automation Letters* . Retrieved from https://doi.org/10.1109/LRA.2024.3440101 (pages 63, 89).

Luo, J., Xu, C., Geng, X., Feng, G., Fang, K., Tan, L., Schaal, S. and Levine, S. (2024), 'Multi-stage cable routing through hierarchical imitation learning', *IEEE Transactions on Robotics* . (page 41).

Lv, K., Yu, M., Pu, Y., Jiang, X., Huang, G. and Li, X. (2023), Learning to estimate 3-d states of deformable linear objects from single-frame occluded point clouds, *in* '2023 IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 7119–7125. (page 88).

Makoviychuk, V., Wawrzyniak, L., Guo, Y., Lu, M., Storey, K., Macklin, M., Hoeller, D., Rudin, N., Allshire, A., Handa, A. and others (n.d.), Isaac gym: High performance gpu based physics simulation for robot learning, *in* 'Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track'. (page 36).

Matas, J., James, S. and Davison, A. J. (2018), Sim-to-real reinforcement learning for deformable object manipulation, *in* 'Conference on Robot Learning', PMLR, pp. 734–743. (page 34).

McConachie, D., Dobson, A., Ruan, M. and Berenson, D. (2020), 'Manipulating deformable objects by interleaving prediction, planning, and control', *The International Journal of Robotics Research* **39**(8), 957–982. (page 34).

McElreath, R. (2020), *Statistical rethinking: A Bayesian course with examples in R and Stan*, Chapman and Hall/CRC. (page 105).

McKendy, K. M., Watanabe, Y., Bilgic, E., Enani, G., Munshi, A., Lee, L., Feldman, L. S., Fried, G. M. and Vassiliou, M. C. (2017), 'Establishing meaningful benchmarks: the development of a formative feedback tool for advanced laparoscopic suturing', *Surgical endoscopy* **31**, 5057–5065. (page 37).

Menz, H. B., Dufour, A. B., Casey, V. A., Riskowski, J. L., McLean, R. R., Katz, P. and Hannan, M. T. (2013), 'Foot pain and mobility limitations in older adults: the framingham foot study', *Journals of Gerontology Series A: Biomedical Sciences and Medical Sciences* **68**(10), 1281–1285. (page 20).

Minaee, S., Boykov, Y., Porikli, F., Plaza, A., Kehtarnavaz, N. and Terzopoulos, D. (2021), 'Image segmentation using deep learning: A survey', *IEEE transactions on pattern analysis and machine intelligence* **44**(7), 3523–3542. (page 37).

Mitrano, P. and Berenson, D. (2024), 'The grasp loop signature: A topological representation for manipulation planning with ropes and cables', *arXiv preprint arXiv:2403.01611* . (page 39).

Moll, M. and Kavraki, L. E. (2006), 'Path planning for deformable linear objects', *IEEE Transactions on Robotics* **22**(4), 625–636. (page 31).

Müller, M. (2007), 'Dynamic time warping', *Information retrieval for music and motion* pp. 69–84. Retrieved from https://doi.org/10.1007/978-3-540-74048-3_4 (page 105).

Munawar, A., Wu, J. Y., Fischer, G. S., Taylor, R. H. and Kazanzides, P. (2022), 'Open simulation environment for learning and practice of robot-assisted surgical suturing', *IEEE Robotics and Automation Letters* **7**(2), 3843–3850. (page 36).

Myronenko, A. and Song, X. (2010), 'Point set registration: Coherent point drift', *IEEE transactions on pattern analysis and machine intelligence* **32**(12), 2262–2275. (page 38).

Nair, A., Chen, D., Agrawal, P., Isola, P., Abbeel, P., Malik, J. and Levine, S. (2017), Combining self-supervised learning and imitation for vision-based rope manipulation, *in* '2017 IEEE international conference on robotics and automation (ICRA)', IEEE, pp. 2146–2153. (page 31).

O'Neill, A., Rehman, A., Gupta, A., Maddukuri, A., Gupta, A., Padalkar, A., Lee, A., Pooley, A., Gupta, A., Mandlekar, A. and others (2023), 'Open x-embodiment: Robotic learning datasets and rt-x models', *arXiv preprint arXiv:2310.08864* . (page 115).

Petit, A., Lippiello, V., Fontanelli, G. A. and Siciliano, B. (2017), 'Tracking elastic deformable objects with an rgb-d sensor for a pizza chef robot', *Robotics and Autonomous Systems* **88**, 187–201. (page 35).

Pignat, E. and Calinon, S. (2017), 'Learning adaptive dressing assistance from human demonstration', *Robotics and Autonomous Systems* **93**, 61–75. (page 40).

Polster, B. (2002), 'What is the best way to lace your shoes?', *Nature* **420**(6915), 476–476. (pages 20, 31, 48).

Polster, B. (2006), *The shoelace book: a mathematical guide to the best (and worst) ways to lace your shoes*, number 24, American Mathematical Soc. (pages 31, 48, 50).

Qin, Y., Yang, W., Huang, B., Van Wyk, K., Su, H., Wang, X., Chao, Y.-W. and Fox, D. (2023), 'Anyteleop: A general vision-based dexterous robot arm-hand teleoperation system', *arXiv preprint arXiv:2307.04577* . Retrieved from https://doi.org/10.48550/arXiv.2307.04577 (page 42).

Raison, N., Ahmed, K., Fossati, N., Buffi, N., Mottrie, A., Dasgupta, P. and Van Der Poel, H. (2017), 'Competency based training in robotic surgery: benchmark scores for virtual reality robotic simulation', *Bju international* **119**(5), 804–811. (page 37).

Rambow, M., Schauß, T., Buss, M. and Hirche, S. (2012), Autonomous manipulation of deformable objects based on teleoperated demonstrations, *in* '2012 IEEE/RSJ International Conference on Intelligent Robots and Systems', IEEE, pp. 2809–2814. (page 41).

Redon, S., Kheddar, A. and Coquillart, S. (2002), Fast continuous collision detection between rigid bodies, *in* 'Computer graphics forum', Vol. 21, Wiley Online Library, pp. 279–287. (pages 21, 88).

Richter, S. and Westphal, M. (2010), 'The lama planner: Guiding cost-based anytime planning with landmarks', *Journal of Artificial Intelligence Research* **39**, 127–177. (pages 32, 72).

Ronneberger, O., Fischer, P. and Brox, T. (2015), U-net: Convolutional networks for biomedical image segmentation, *in* 'Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18', Springer, pp. 234–241. (page 114).

Roweis, S. T. and Saul, L. K. (2000), 'Nonlinear dimensionality reduction by locally linear embedding', *science* **290**(5500), 2323–2326. (page 38).

Russell, S. J. and Norvig, P. (2016), *Artificial intelligence: a modern approach*, Pearson. (pages 31, 51).

Saha, M. and Isto, P. (2007), 'Manipulation planning for deformable linear objects', *IEEE Transactions on Robotics* **23**(6), 1141–1150. (page 32).

Sánchez, D., Wan, W. and Harada, K. (2020), 'Tethered tool manipulation planning with cable maneuvering', *IEEE Robotics and Automation Letters* **5**(2), 2777–2784. (page 19).

Sanchez, J., Corrales, J.-A., Bouzgarrou, B.-C. and Mezouar, Y. (2018), 'Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey', *The International Journal of Robotics Research* **37**(7), 688–716. (pages 19, 29, 33, 34).

Schettino, V. and Demiris, Y. (2019), Inference of user-intention in remote robot wheelchair assistance using multimodal interfaces, *in* '2019 IEEE/RSJ

International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 4600–4606. (page 41).

Schulman, J., Gupta, A., Venkatesan, S., Tayson-Frederick, M. and Abbeel, P. (2013), A case study of trajectory transfer through non-rigid registration for a simplified suturing scenario, *in* '2013 IEEE/RSJ International Conference on Intelligent Robots and Systems', IEEE, pp. 4111–4117. (page 41).

Schulman, J., Lee, A., Ho, J. and Abbeel, P. (2013), Tracking deformable objects with point clouds, *in* '2013 IEEE International Conference on Robotics and Automation', IEEE, pp. 1130–1137. (page 33).

Schwarz, M., Lenz, C., Memmesheimer, R., Pätzold, B., Rochow, A., Schreiber, M. and Behnke, S. (2023), Robust immersive telepresence and mobile telemanipulation: Nimbro wins ana avatar xprize finals, *in* '2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)', IEEE, pp. 1–8. Retrieved from https://doi.org/10.1109/Humanoids57100.2023.10375179 (page 42).

Seita, D., Florence, P., Tompson, J., Coumans, E., Sindhwani, V., Goldberg, K. and Zeng, A. (2021), Learning to rearrange deformable cables, fabrics, and bags with goal-conditioned transporter networks, *in* '2021 IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 4568–4575. (page 36).

Seita, D., Ganapathi, A., Hoque, R., Hwang, M., Cen, E., Tanwani, A. K., Balakrishna, A., Thananjeyan, B., Ichnowski, J., Jamali, N. and others (2020), Deep imitation learning of sequential fabric smoothing from an algorithmic supervisor, *in* '2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 9651–9658. (page 33).

Seo, M., Han, S., Sim, K., Bang, S. H., Gonzalez, C., Sentis, L. and Zhu, Y. (2023), Deep imitation learning for humanoid loco-manipulation through human teleoperation, *in* '2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)', IEEE, pp. 1–8. Retrieved from https://doi.org/10.1109/Humanoids57100.2023.10375203 (page 42).

She, Y., Wang, S., Dong, S., Sunil, N., Rodriguez, A. and Adelson, E. (2021), 'Cable manipulation with a tactile-reactive gripper', *The International Journal of Robotics Research* **40**(12-14), 1385–1401. (pages 29, 117).

Shridhar, M., Manuelli, L. and Fox, D. (2022), Cliport: What and where pathways for robotic manipulation, *in* 'Conference on robot learning', PMLR, pp. 894–906. (page 115).

Si, W., Wang, N. and Yang, C. (2021), 'A review on manipulation skill acquisition through teleoperation-based learning from demonstration', *Cognitive Computation and Systems* **3**(1), 1–16. Retrieved from https://doi.org/10.1049/ccs2.12005 (pages 43, 97).

Sin, F. S., Schroeder, D. and Barbič, J. (2013), Vega: non-linear fem deformable object simulator, *in* 'Computer Graphics Forum', Vol. 32, Wiley Online Library, pp. 36–48. (page 33).

Sivakumar, A., Shaw, K. and Pathak, D. (2022), 'Robotic telekinesis: Learning a robotic hand imitator by watching humans on youtube', *arXiv preprint arXiv:2202.10448* . Retrieved from https://doi.org/10.48550/arXiv.2202.10448 (page 42).

Solomon AI and 3D Vision (2017), 'World's first automated shoe lacing machine'. https://www.youtube.com/watch?v=1oeXw3awuaM (page 30).

Studio, V. M. (2016), 'Obi physics', https://obi.virtualmethodstudio.com/. (page 35).

Sudry, M., Jurgenson, T., Tamar, A. and Karpas, E. (2023), Hierarchical planning for rope manipulation using knot theory and a learned inverse model, *in* 'Conference on Robot Learning', PMLR, pp. 1596–1609. (page 32).

Sundaresan, P., Grannen, J., Thananjeyan, B., Balakrishna, A., Ichnowski, J., Novoseller, E., Hwang, M., Laskey, M., Gonzalez, J. E. and Goldberg, K. (2021), Untangling dense non-planar knots by learning manipulation features and recovery policies, *in* '17th Robotics: Science and Systems, RSS 2021', MIT Press Journals. (page 32).

Sundaresan, P., Grannen, J., Thananjeyan, B., Balakrishna, A., Laskey, M., Stone, K., Gonzalez, J. E. and Goldberg, K. (2020), Learning rope manipulation policies using dense object descriptors trained on synthetic depth data, *in* '2020 IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 9411–9418. (pages 29, 32).

Suzuki, K., Kanamura, M., Suga, Y., Mori, H. and Ogata, T. (2021), In-air knotting of rope using dual-arm robot based on deep learning, *in* '2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 6724–6731. (page 29).

Tagliabue, E., Pore, A., Dall'Alba, D., Magnabosco, E., Piccinelli, M. and Fiorini, P. (2020), Soft tissue simulation environment to learn manipulation

tasks in autonomous robotic surgery, *in* '2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 3261–3266. (pages 21, 36).

Tandon, S., Javdani, S., O'Brien, J. F. and Abbeel, P. (n.d.), 'Motion planning for deformable one-dimensional objects'. (page 31).

Tang, T., Fan, Y., Lin, H.-C. and Tomizuka, M. (2017), State estimation for deformable objects by point registration and dynamic simulation, *in* '2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 2427–2433. (pages 22, 29, 38).

Tang, T. and Tomizuka, M. (2022), 'Track deformable objects from point clouds with structure preserved registration', *The International Journal of Robotics Research* **41**(6), 599–614. (pages 22, 33, 38, 79).

Technologies, U. (2005), 'Unity engine', https://unity.com/. (page 35).

Thananjeyan, B., Kerr, J., Huang, H., Gonzalez, J. E. and Goldberg, K. (2022), 'All you need is luv: Unsupervised collection of labeled images using invisible uv fluorescent indicators', *arXiv preprint arXiv:2203.04566* . (page 37).

Todorov, E., Erez, T. and Tassa, Y. (2012), Mujoco: A physics engine for model-based control, *in* '2012 IEEE/RSJ international conference on intelligent robots and systems', IEEE, pp. 5026–5033. (page 34).

Tonnesen, D. L. (1998), *Dynamically coupled particle systems for geometric modeling, reconstruction, and animation.*, University of Toronto. (page 34).

Trinkaus, E. and Shang, H. (2008), 'Anatomical evidence for the antiquity of human footwear: Tianyuan and sunghir', *Journal of Archaeological Science* **35**(7), 1928–1933. (page 30).

Turchetti, G., Palla, I., Pierotti, F. and Cuschieri, A. (2012), 'Economic evaluation of da vinci-assisted robotic surgery: a systematic review', *Surgical endoscopy* **26**, 598–606. (page 42).

Vaswani, A. (2017), 'Attention is all you need', *Advances in Neural Information Processing Systems* . (page 41).

Viswanath, V., Shivakumar, K., Kerr, J., Thananjeyan, B., Novoseller, E., Ichnowski, J., Escontrela, A., Laskey, M., Gonzalez, J. E. and Goldberg, K. (2022), 'Autonomously untangling long cables', *arXiv preprint arXiv:2207.07813* . (pages 29, 32).

Wakamatsu, H., Arai, E. and Hirai, S. (2006), 'Knotting/unknotting manipulation of deformable linear objects', *The International Journal of Robotics Research* **25**(4), 371–395. (page 29).

Wakamatsu, H., Tsumaya, A., Arai, E. and Hirai, S. (2004), Planning of one-handed knotting/raveling manipulation of linear objects, *in* 'IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004', Vol. 2, IEEE, pp. 1719–1725. (page 32).

Wang, C., Zhang, Y., Zhang, X., Wu, Z., Zhu, X., Jin, S., Tang, T. and Tomizuka, M. (2022), 'Offline-online learning of deformation model for cable manipulation with graph neural networks', *IEEE Robotics and Automation Letters* **7**(2), 5544–5551. (page 19).

Wang, Q., McCarthy, R., Bulens, D. C., Sanchez, F. R., McGuinness, K., O'Connor, N. E. and Redmond, S. J. (2023), 'Identifying expert behavior in offline training datasets improves behavioral cloning of robotic manipulation policies', *IEEE Robotics and Automation Letters* . (page 41).

Wang, T. and Yamakawa, Y. (2022), 'Real-time occlusion-robust deformable linear object tracking with model-based gaussian mixture model', *Frontiers in Neurorobotics* **16**, 886068. (pages 29, 38).

Wang, W., Berenson, D. and Balkcom, D. (2015), An online method for tight-tolerance insertion tasks for string and rope, *in* '2015 IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 2488–2495. (pages 19, 29, 39).

Wang, Y., McConachie, D. and Berenson, D. (2021), Tracking partially-occluded deformable objects while enforcing geometric constraints, *in* '2021 IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 14199–14205. (pages 22, 29, 37, 39, 79).

Wang, Y., Wei, X., Liu, F., Chen, J., Zhou, Y., Shen, W., Fishman, E. K. and Yuille, A. L. (2020), Deep distance transform for tubular structure segmentation in ct scans, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition', pp. 3833–3842. (page 37).

Weng, Z., Paus, F., Varava, A., Yin, H., Asfour, T. and Kragic, D. (2021), Graph-based task-specific prediction models for interactions between deformable and rigid objects, *in* '2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 5741–5748. (page 35).

Westwood, J. and others (2007), 'Sofa–an open source framework for medical simulation', *Medicine Meets Virtual Reality 15: In Vivo, in Vitro, in Silico: Designing the Next in Medicine* **125**, 13. (page 35).

Wilson, A., Jiang, H., Lian, W. and Yuan, W. (2023), Cable routing and assembly using tactile-driven motion primitives, *in* '2023 IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 10408–10414. (page 117).

Wu, P., Shentu, Y., Yi, Z., Lin, X. and Abbeel, P. (2023), 'Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators', *arXiv preprint arXiv:2309.13037* . Retrieved from https://doi.org/10.48550/arXiv.2309.13037 (pages 42, 98, 99, 133).

Xiang, J., Dinkel, H., Zhao, H., Gao, N., Coltin, B., Smith, T. and Bretl, T. (2023), 'Trackdlo: Tracking deformable linear objects under occlusion with motion coherence', *IEEE Robotics and Automation Letters* . (pages 22, 29, 39, 88).

Yan, M., Li, G., Zhu, Y. and Bohg, J. (2020), Learning topological motion primitives for knot planning, *in* '2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 9457–9464. (pages 29, 32, 35, 40, 41).

Yan, M., Zhu, Y., Jin, N. and Bohg, J. (2020), 'Self-supervised learning of state estimation for manipulating deformable linear objects', *IEEE robotics and automation letters* **5**(2), 2372–2379. (page 33).

Yan, W., Vangipuram, A., Abbeel, P. and Pinto, L. (2021), Learning predictive representations for deformable objects using contrastive estimation, *in* 'Conference on Robot Learning', PMLR, pp. 564–574. (page 36).

Yang, P.-C., Sasaki, K., Suzuki, K., Kase, K., Sugano, S. and Ogata, T. (2016), 'Repeatable folding task by humanoid robot worker using deep learning', *IEEE Robotics and Automation Letters* **2**(2), 397–403. (page 40).

Yang, T., Liu, J., Wang, Y. and Xiong, R. (2023), Self-entanglement-free tethered path planning for non-particle differential-driven robot, *in* '2023 IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 7816–7822. (page 39).

Yang, Y., Stork, J. A. and Stoyanov, T. (2022*a*), 'Learning differentiable dynamics models for shape control of deformable linear objects', *Robotics and Autonomous Systems* **158**, 104258. (pages 19, 37, 114).

Yang, Y., Stork, J. A. and Stoyanov, T. (2022*b*), 'Particle filters in latent space for robust deformable linear object tracking', *IEEE Robotics and Automation Letters* **7**(4), 12577–12584. (page 114).

Yin, H., Varava, A. and Kragic, D. (2021), 'Modeling, learning, perception, and control methods for deformable object manipulation', *Science Robotics* **6**(54), eabd8803. (pages 19, 23, 29, 36, 40, 41).

You, Y., Shao, L., Migimatsu, T. and Bohg, J. (2021), Omnihang: Learning to hang arbitrary objects using contact point correspondences and neural collision estimation, *in* '2021 IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 5921–5927. (page 36).

Yu, M., Lv, K., Zhong, H., Song, S. and Li, X. (2022), 'Global model learning for large deformation control of elastic deformable linear objects: An efficient and adaptive approach', *IEEE Transactions on Robotics* **39**(1), 417–436. (page 35).

Yu, M., Zhong, H. and Li, X. (2022), Shape control of deformable linear objects with offline and online learning of local linear deformation models, *in* '2022 International Conference on Robotics and Automation (ICRA)', IEEE, pp. 1337–1343. (page 40).

Yuan, J. and Cheriyadat, A. M. (2016), 'Image feature based gps trace filtering for road network generation and road segmentation', *Machine Vision and Applications* **27**, 1–12. (page 37).

Zanella, R., Caporali, A., Tadaka, K., De Gregorio, D. and Palli, G. (2021), Auto-generated wires dataset for semantic segmentation with domain-independence, *in* '2021 International conference on computer, control and robotics (ICCCR)', IEEE, pp. 292–298. (pages 37, 116).

Zare, M., Kebria, P. M., Khosravi, A. and Nahavandi, S. (2024), 'A survey of imitation learning: Algorithms, recent developments, and challenges', *IEEE Transactions on Cybernetics* . (page 40).

Zhang, F. and Demiris, Y. (2022), 'Learning garment manipulation policies toward robot-assisted dressing', *Science robotics* **7**(65), eabm6010. (page 115).

Zhang, X., Li, Z., Gao, X., Jin, D. and Li, J. (2021), 'Channel attention in lidar-camera fusion for lane line segmentation', *Pattern Recognition* **118**, 108020. (page 37).

Zhao, T. Z., Kumar, V., Levine, S. and Finn, C. (2023), 'Learning fine-grained bimanual manipulation with low-cost hardware', *arXiv preprint arXiv:2304.13705* . Retrieved from https://doi.org/10.48550/arXiv.2304.13705 (pages 42, 101, 134).

Zhao, T. Z., Tompson, J., Driess, D., Florence, P., Ghasemipour, S. K. S., Finn, C. and Wahid, A. (2024), Aloha unleashed: A simple recipe for robot dexterity, *in* '8th Annual Conference on Robot Learning'. (pages 41, 98, 108, 133).

Zhaole, S., Zhou, H., Nanbo, L., Chen, L., Zhu, J. and Fisher, R. B. (2023), 'A robust deformable linear object perception pipeline in 3d: From segmentation to reconstruction', *IEEE Robotics and Automation Letters* **9**(1), 843–850. (pages 22, 37, 38).

Zhou, P., Qi, J., Duan, A., Huo, S., Wu, Z. and Navarro-Alarcon, D. (2024), 'Imitating tool-based garment folding from a single visual observation using hand-object graph dynamics', *IEEE Transactions on Industrial Informatics* . (page 36).

Zhou, P., Zhu, J., Huo, S. and Navarro-Alarcon, D. (2021), 'Lasesom: A latent and semantic representation framework for soft object manipulation', *IEEE Robotics and Automation Letters* **6**(3), 5381–5388. (page 37).

Zhu, J., Cherubini, A., Dune, C., Navarro-Alarcon, D., Alambeigi, F., Berenson, D., Ficuciello, F. and others (2022), 'Challenges and outlook in robotic manipulation of deformable objects', *IEEE Robotics & Automation Magazine* **29**(3), 67–77. (pages 22, 29, 37).

Zhu, J., Navarro, B., Fraisse, P., Crosnier, A. and Cherubini, A. (2018), Dual-arm robotic manipulation of flexible cables, *in* '2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 479–484. (pages 19, 31).

Zhu, J., Navarro, B., Passama, R., Fraisse, P., Crosnier, A. and Cherubini, A. (2019), 'Robotic manipulation planning for shaping deformable linear objects withenvironmental contacts', *IEEE Robotics and Automation Letters* **5**(1), 16–23. (page 22).

Zhu, Y., Joshi, A., Stone, P. and Zhu, Y. (2023), Viola: Imitation learning for vision-based manipulation with object proposal priors, *in* 'Conference on Robot Learning', PMLR, pp. 1199–1210. (page 42).