

Imperial College London
Dyson School of Design Engineering

Kinematic-Model-Free Online Learning for Robot Motion Control

Ahmad AlAttar

Submitted in part fulfilment of the requirements
for the degree of Doctor of Philosophy at
Imperial College London, 2023

The Statement of Originality

I hereby declare that this thesis and the work reported herein was composed by and originated entirely from me. Information derived from the published and unpublished work of others has been acknowledged in the text and references are given in the list of sources.

Ahmad AlAttar (2023)

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution-Non Commercial 4.0 International Licence (CC BY-NC). Under this licence, you may copy and redistribute the material in any medium or format. You may also create and distribute modified versions of the work. This is on the condition that: you credit the author and do not use it, or any derivative works, for a commercial purpose. 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. Where a work has been adapted, you should indicate that the work has been changed and describe those changes. 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.

Abstract

Controlling robots has been a thriving field of research since their emergence. Traditionally, robots are controlled using joint space to task space transformations based on kinematic and dynamic models. However, unconventional robots like soft, morphing, malleable, transforming, and evolving ones pose challenges for accurate modeling and control. Controllers in literature typically fall into two categories: model-based and model-free.

Kinematic-model-free control offers a promising solution to overcome challenges faced by conventional controllers. This method doesn't require prior knowledge of the robot's kinematic or dynamic models. Instead, it gathers data through exploratory actuations, building a volatile local linear model to approximate the robot's behavior and estimate actuation signals for target movement. This thesis extends the state-of-the-art kinematic-model-free controller to tackle various problems such as tip orientation and pose control, robot configuration control, obstacle avoidance, gravity compensation, and continuous control.

Initially applied to planar rigid robots, the controller was adapted for soft continuum robots, demonstrating its effectiveness through both simulation and physical experimentation. Quaternion and dual-quaternion mathematics were employed for position, orientation, and pose control. A dual-quaternionic kinematic-model-free multi-point controller was introduced to resolve redundancy in hyper-redundant manipulators, validated through simulation tests.

Integration with model-predictive control enabled path planning and obstacle avoidance, validated through physical experimentation on a planar rigid robot manipulator with virtual obstacles. Modifications allowed the controller to run continuously at higher speeds, outperforming conventional methods in the face of kinematic or dynamic changes.

The kinematic-model-free controller showcased versatility across different robot types, both in simulation and physical experiments. This research pushed the boundaries of the kinematic-model-free controller and validated its effectiveness through rigorous testing.

Acknowledgements

During the course of my studies, many people encouraged me, supported me, guided me, and inspired me. It is impossible to thank all of them here. However, I'd like to take the chance to mention some of them.

I'd like to first and foremost thank His Excellency Omar Al Olama for the engorgement and support he gave me to pursue my PhD. The Ministry of Education of the UAE have supported my studied from my bachelor's in Canada all the way up to my master's and PhD in the UK. I'd like to thank them for the scholarships they have provided me. I'd also like to thank the UAE's embassy and cultural attaché in Canada and the UK for their support.

I'd like to thank my supervisor Dr. Petar Kormushev for his support and guidance throughout my PhD. I'd like to thank my colleagues at the Robot Intelligence Lab at the Dyson School of Design Engineering for making great memories, Roni Permana Saputra, Digby Chappell, Francesco Cursi, and Ke Wang. I'd also like to thank all the professors and peers I collaborated with, Dr. Ian Walker and his PhD student and Dr. Federico Renda and his student Ikhlas Ben Hmida.

Thanks to Khalifa AlQama for giving me the opportunity to conduct research and physical experimentation at Dubai Future Labs. I'd like give a special thanks to my mentor, Dr. Tarek Taha, who continuously supported me in all my endeavours. I'd like to thank all of my colleagues there for their support during my PhD, Omar AlHammadi, Mohammad Bin Shabib, Erhamah AlSuwaidi, Maryam Bu Humaid, Fatma Bin Kharbash, Maimoona Behandy, Hessa AlEassa, Meera AlRoumi, Maitha AlQaydi, Sultan Bin Sulaiman, and everyone else there. I'd also like to thank my interns who helped with running some experiments: Kushek Balasundra, Omar AlGouhary, and Yahya AlHaj.

Although it is impossible to mention all the friends I awe gratitude to, I would like to thank my friends from Canada, Mohammad Azarbar, Faray Omer Shahid, Abdulrehman Dumyati and his family, and Abdullah Babili, who encouraged me to pursue a PhD. I would also like to thank my friends from London for their supporting during my studies, Sheikha AlMazrouri, Mohammad AlMazrouri, Beshayer AlRum, and Noora AlSerkal. Not to forget my friends from

Dubai, Ali Asmar, Sharif Nami, and Faisal Al Ghaith and my friends from the art world, Musab AlHammadi, Mohammad AlHammadi, and Latifa Saeed. I would also like to thank my friend Khalid AlAwadi and his dog Atlas for allowing me to use their office space to write my thesis.

And mostly, I would like to thank my family, my aunt, my uncle, my grandparents, my wife, and my little princesses for standing by my side during the ups and downs during my studies and for believing in me throughout the whole journey. Thank you.

To the sun and sand that burned me, to everyone and no one, to the stars that lit my nights, to
you, and myself.

The future belongs to those who can imagine it, design it, and execute it. It isn't something you await, but rather create.

- His Highness Sheikh Mohammed bin Rashid Al Maktoum

Contents

Abstract	3
Acknowledgements	4
1 Introduction	16
1.1 Contributions	18
1.2 Outline	18
References	20
2 Kinematic-Model-Free Control	23
2.1 Related Work	23
2.2 KMF Position Control	25
2.3 KMF Orientation Control	30
2.4 KMF Pose Control	32
2.5 KMF Tendon-Driven Continuum Robot Control	36
2.6 KMF Pneumatic Continuum Robot Control	38
2.7 Results	40
2.7.1 Simulated Rigid Robot Control	40
2.7.2 Simulated Continuum Robot Control	41
2.7.3 Physical Rigid Robot Control	52
2.7.4 Physical Continuum Robot Control	52
2.8 Discussion	63
References	65
3 Quaternionic Kinematic-Model-Free Control	68
3.1 Related Work	69
3.2 Mathematical Background	70
3.3 KMF Quaternionic Position Control	77
3.4 KMF Quaternionic Orientation Control	78
3.5 KMF Dual-Quaternionic Pose Control	80
3.6 KMF Multi-Point Pose Control	81
3.7 Results	86
3.7.1 Single-Point Pose Control	86
3.7.2 Multi-Point Pose Control	91
3.8 Discussion	99
References	102

4	Kinematic-Model-Free Obstacle Avoidance	105
4.1	Related Work	106
4.2	KMF-PC Control	107
4.3	Results	111
4.3.1	Reaching Using KMF and KMF-PC	112
4.3.2	Obstacle Avoidance Using KMF-PC	112
4.3.3	Adaptability of KMF-PC	115
4.4	Discussion	116
	References	119
5	Continuous Kinematic-Model-Free Control	123
5.1	Related Work	124
5.2	Continuous KMF Control	126
5.3	Results	130
5.3.1	Simulated Results	130
5.3.2	Real-world Experimentation Results	132
5.4	Discussion	161
	References	167
6	Conclusion	171
A	Mathematics	175

List of Tables

1.1	Contributions of kinematic-model-free control research.	19
3.1	Robot specifications.	86
3.2	Target poses.	87
3.3	Denavit–Hartenberg parameters of redundant robot.	92
3.4	Reaching steps for different degrees of freedom enabled.	95
3.5	Control points placement.	96
3.6	Steps required to complete three reaching tasks using three arrangements of control points.	98

List of Figures

2.1	Flowchart of simple KMF position control.	26
2.2	Unlike the conventional controllers in (a) and (b), the kinematic-model-free controller in (c) does not use encoder feedback or joint angle feedback during control. Instead, the controller using exteroception to track the robot's end effector. Also, the kinematic-model-free controller does not have any prior knowledge of the robot's kinematic or dynamic models [?].	27
2.3	Example actuation primitives with different parameters: duration d_p and magnitude τ_p [?].	28
2.4	Flowchart of simple KMF orientation control.	31
2.5	Flowchart of simple KMF pose control.	33
2.6	Pose path planning space	35
2.7	a) A 2-DoF robot manipulator reaching a target point in 2D space; b) A 3-DoF robot manipulator reaching a target point in 3D space.	42
2.8	a) A 3-DoF robot manipulator reaching a target pose in 2D space; b) a 3-DoF robot manipulator tracing a square while keeping the end effector perpendicular to the path followed.	43
2.9	Overview plot of the design of the continuum robot used for the simulation tests along with a table detailing the actuator locations.	45
2.10	(a) Manipulator tracing a circle in 3D space. The controller performs both tip-position control and gravity compensation. The starting position is marked by 1 and the ending position is marked by 4. Marks 2 and 3 are almost at the circle top and bottom positions; (b) Change in actuation strengths, $\Delta\tau(t)$, added to overall actuation signal for control; (c) Actuation strengths, $\tau(t)$, used to control each actuator.	47
2.11	(a) Manipulator tip moving to the target position denoted as a blue point in 3D space. The controller performs both tip-position control and gravity compensation in this test too. The starting and ending positions are marked by 1 and 4 respectively. Point 2 shows the manipulator configuration prior to the actuator's failure, point 3 shows the step right after the actuator's failure. (b) Change in actuation strengths, $\Delta\tau(t)$, added to overall actuation signal for control; (c) Actuation strengths, $\tau(t)$, used to control each actuator.	48
2.12	(a) Robot moving to a target position. The starting and ending positions are marked by 1 and 2 respectively. Point 2 shows that the robot reached the target position by simultaneous actuator strength and actuator slope control. (b) actuator slopes; (c) Actuation strengths, $\tau(t)$, used to control each actuator. . .	49

2.13	(a) Manipulator tip moving to a target position. The starting and ending positions are marked by 1 and 2 respectively. Point 2 shows that the manipulator reaches the target position by simultaneous actuator strength control and shrinking. (b) Change in robot length, L , for shrinking; (c) Actuation strengths, $\tau(t)$, used to control each actuator.	53
2.14	(a) Manipulator tip moving to a target position. The starting and ending positions are marked by 1 and 2 respectively. Point 2 shows that the manipulator reaches the target position through simultaneous actuator strength control and growth. (b) Change in robot length, L , for growing; (c) Actuation strengths, $\tau(t)$, used to control each actuator.	54
2.15	Target reaching using the Panda manipulator constrained to planar motion with only 2 DoFs enabled.	55
2.16	OctArm pneumatically-actuated manipulator.	57
2.17	Line path trajectory.	58
2.18	Lemniscate path trajectory.	58
2.19	Line path followed by robot.	59
2.20	Lemniscate path followed by robot.	60
2.21	(a) End-effector and target positions of 3-DoF robot while following line path. (b) Measured end-effector position of 3-DoF robot while following line path. . . .	60
2.22	(a) End-effector and target positions of 3-DoF robot while following lemniscate path. (b) Measured end-effector position of 3-DoF robot while following lemniscate path.	61
2.23	(a) End-effector and target positions of 4-DoF robot while following line path. (b) Measured end-effector position of 4-DoF robot while following line path. . . .	61
2.24	(a) End-effector and target positions of 4-DoF robot while following lemniscate path. (b) Measured end-effector position of 4-DoF robot while following lemniscate path.	62
2.25	(a) End-effector and target positions of 6-DoF robot while following line path. (b) Measured end-effector position of 6-DoF robot while following line path. . . .	63
3.1	Flowchart of KMF Multi-Point control.	83
3.2	The KMF-DQ controller actuates the robot's joints to perform a set of reaching tasks to draw out a polygon in 3D space with each vertex having a pre-specified pose within the robot's reachable workspace. The bottom left three plots show the orthogonal views of the robot performing the experiment. The bottom right three plots show the final resulting trajectory of the end effector. The robot moves towards the next target only after reaching the current target.	88
3.3	The control signals sent to the actuators generated by the controller. These actuation signals correspond to the first reaching task where the end effector moves towards target 1.	89
3.4	The joint positions of the three joints. These data correspond to the first reaching task where the end effector moves towards target 1.	90
3.5	Distance of the end-effector pose to the target pose for each reaching task. The pose distance approaches zero as the end effector reaches the targets.	91
3.6	The simulated 9-degrees-of-freedom continuum robot.	92

3.7	The controller performed the task of reaching, bringing a single control point at the end effector to a desired target pose. As more joints were enabled, the performance was degraded. In particular, in the 2-degrees-of-freedom case, the robot reaches the target in 21 steps, while, in the 9-degrees-of-freedom case, the robot reaches the target in 150 steps.	94
3.8	The three arrangements of the control points (CP's) along the robot's kinematic chain. In the first arrangement, only one control point was used which was placed at the end effector. In the second and third arrangements, two and three equally spaced control points were used, respectively.	96
3.9	The three reaching tasks to be performed. The translucent pose is the initial configuration from which the robot is to actuate towards the opaque pose. . . .	97
3.10	The controller performed three reaching tasks. In all 3 tasks, the controller converged faster when using more control points (CPs).	97
3.11	These joint positions correspond to reaching task 1. The larger jitters are due to exploratory behavior, which is triggered to collect more information about the robot's local kinodynamics.	98
3.12	In this simulation run, the controller attempted to actuate the robot to a back arching configuration. It achieved to reach this configuration within 150 steps. .	99
4.1	Simulated and physical experimentation of KMF control and the proposed KMF-PC in the absence of any obstacle. Comparing simulation results (a) and (b), both controllers performed similar, reaching the target in a semi-straight line. As for the physical experiments, we notice that the KMF-PC took a slightly more curved path to reach the target position. Nonetheless, both KMF and KMF-PC performed similarly well, achieving the reaching task within 30 steps.	113
4.2	Simulation results of simultaneous reaching and obstacle avoidance using KMF-PC on simulated 2-DOF, 3-DOF, and 4-DOF manipulator robots. It is noted in the 4-DOF case, where the complexity is relatively higher, more exploratory motion is triggered which caused the end effector to penetrate the virtual obstacle.	114
4.3	Physical experiment results of KMF-PC reaching and obstacle avoidance. In both cases, the end effector reached close to the obstacle. As such, obstacle inflation is recommended. In the single obstacle scenario, the robot reached the target in 21 steps. In the two obstacles scenario, it reached the target in 30 steps.	115
4.4	Physical experiment results of reaching and obstacle avoidance whilst modifying the end effector during execution. When the end effector is altered, the controller triggers an exploration phase and resumes reaching and obstacle avoidance right after. The modifications include shortening the end effector, rigid and soft extensions, and offsetting the end effector.	117
5.1	Flowchart of continuous kinematic-model-free control.	128
5.2	A planar 2-DoF robot manipulator following a circular path using continuous kinematic-model-free control.	131
5.3	A planar 2-DoF robot manipulator that reaches a target point while compensating for gravity. The direction of gravity is changed as indicated on top of each plot every few steps. The beginning of the change is marked by a blue circle in the path taken by the robot. Despite the changes in the direction of the force of gravity, the controller manages to reach the target position marked by a red circle.	133
5.4	A planar 2-DoF physical robot manipulator using Hebi X5-9 actuators.	135

5.5	Target reaching using continuous kinematic-model-free control. The controller reached the target position with a 20 Hz or higher control loop frequency.	136
5.6	End-effector distance to target over time for various control loop frequencies of the continuous kinematic-model-free controller tested.	137
5.7	Target reaching using conventional control. The controller reached the target position with all control loop frequencies tested.	138
5.8	End-effector distance to target over time for various control loop frequencies of the conventional controller tested.	139
5.9	A comparison between continuous kinematic-model-free control and conventional control for target reaching with a control loop frequency of 40 Hz.	140
5.10	Robot manipulator with end-effector extension.	142
5.11	Target reaching with end-effector extension using continuous kinematic-model-free control. The controller reached the target position with a 20 Hz or higher control loop frequency.	143
5.12	Target reaching with end-effector extension using conventional control. The controller did not reach the target position for all control loop frequencies tested.	144
5.13	A comparison between continuous kinematic-model-free control and conventional control for target reaching with end-effector extension and a control loop frequency of 40 Hz.	145
5.14	Robot manipulator with end-effector offset.	147
5.15	Target reaching with end-effector offset using continuous kinematic-model-free control. The controller reached the target position with a 30 Hz or higher control loop frequency.	148
5.16	Target reaching with end-effector offset using conventional control. The controller did not reach the target position for all control loop frequencies tested.	149
5.17	A comparison between continuous kinematic-model-free control and conventional control for target reaching with end-effector offset and a control loop frequency of 40 Hz.	150
5.18	Robot manipulator with actuators order switched.	152
5.19	Target reaching with switched actuators using continuous kinematic-model-free control. The controller reached the target position with a 20 Hz or higher control loop frequency.	153
5.20	Target reaching with switched actuators using conventional control. The controller did not reach the target position for all control loop frequencies tested.	154
5.21	A comparison between continuous kinematic-model-free control and conventional control for target reaching with added weight and a control loop frequency of 40 Hz.	155
5.22	Robot manipulator with added weight.	157
5.23	Target reaching with added weight using continuous kinematic-model-free control. The controller reached the target position with a 40 Hz or higher control loop frequency.	158
5.24	Target reaching with added weight using conventional control. The controller sagged below the target position for all control loop frequencies tested.	159
5.25	A comparison between continuous kinematic-model-free control and conventional control for target reaching with added weight and a control loop frequency of 40 Hz.	160
5.26	Robot manipulator with tilting ramp.	162
5.27	Target reaching with tilting ramp using continuous kinematic-model-free control. The controller reached the target position with a 20 Hz or higher control loop frequency.	163

5.28	Target reaching with tilting ramp using conventional control. The controller failed to reach the target position for all control loop frequencies tested.	164
5.29	A comparison between continuous kinematic-model-free control and conventional control for target reaching with tilting ramp and a control loop frequency of 40 Hz.	165

Chapter 1

Introduction

Since the emergence of modern robots in the early 1950s, robot control became a thriving field of research as interest grew [?]. Conventional control of robotic systems, such as manipulators, is based on the kinematics and dynamics of rigid bodies that use joint-space to task-space transformations to calculate their poses [?]. This approach leaves the controller heavily dependent on the robot's configuration, which assumes a static arrangement of joints and links. This kinematic-model-based approach has remained the prevalent control method for over half a century [?].

Conventional robot controllers require both the kinematic and dynamic models of the robot in order to perform accurate control [?]. However, soft robots, morphing robots, malleable robots, transforming robots, and evolving robots can be challenging to model accurately and control. Tremendous research has been done in the area controls for soft robots. Most of the controllers found in literature can be categorized into two main classes: Model-based controllers, which rely on models describing the system's motion for controller derivation; and Model-free controllers, which use data-driven and empirical methods to do the same [?].

Model-based controllers have been the predominant method of control for robot manipulation. The Jacobian of a robot manipulator that relates the joint velocities to the end-effector velocity is calculated prior to control [?] and is used to control the robot with forward and inverse kine-

matics [?]. Using such models imposes many limitations [?] and assumes the Denavit-Hartenberg parameters, that define the robots kinematics, to be static. An example of model-based control is visual servoing that uses an external camera to calculate the required end-effector velocity to reach a desired target [?]. In soft robot control, the Piece-wise Constant Curvature (PCC) assumption, which discretizes the profile of the manipulator into constant curvature sections, and the is used in closed-loop controllers [?]. The Geometric Variable Strain Model (GVS), a strain based discretization of the continuous Cosserat model, is also used to develop a controller for soft robots [?]. Moreover, quasi-static models are used to control soft robot tip position [?]. This type of control makes geometric simplifications and assumptions which can be inaccurate in the presence of unmodeled forces, such as gravity. A model-based controller is limited by the inherent inaccuracies and singularities of the model it follows [?].

On the other hand, model-free approaches for control do not require the robot's kinematic or dynamic models a priori for control. Approximating a Jacobian model is one method for model-free control [?]. Dynamic parameters estimation [?], motor babbling [?], machine learning [?], and reinforcement learning [?] have all been used for robot control. In soft robot control neural networks [?] or deep reinforcement learning [?] were used to learn to control a soft robot's tip position. In general, data-driven approaches provide a promising method for control learning for soft robots. However, these control methods require either the development of a model or the training of one prior to control.

The Kinematic-model-free control detailed in this thesis is an approach that is able to control a robot's end-effector position without any prior model data or knowledge [?]. The approach works by sending pseudorandom actuation signals to robot actuators and sampling end-effector displacements which are then used to estimate actuation force signals that bring the end-effector towards a desired target position through generating volatile local linear models [?]. This controller is explored and studied in detail, pushing the boundaries of model-free control for robot motion learning. The contributions of this work and the outline of this thesis are below.

1.1 Contributions

The research detailed in this thesis extends the state-of-the-art kinematic-model-free controller in many fronts with plentiful tests and experimentation on different robots, both rigid and soft robots. Table 1.1 summarizes the contributions of this work and where the findings and results were disseminated.

1.2 Outline

There are six chapters in this thesis with each chapter covering a specific contribution(s) apart from the first and last chapters that introduce and conclude the thesis:

- **Chapter 1** introduces the research domain and focus of this work and highlights the contributions and outline of the thesis;
- **Chapter 2** covers the kinematic-model-free controller for soft continuum robot planar and spatial tip-position control;
- **Chapter 3** presents the quaternionic kinematic-model-free position and orientation controllers and the dual-quaternionic single-point and multi-point controllers;
- **Chapter 4** details kinematic-model-free position control with obstacle avoidance using model-free predictive-control;
- **Chapter 5** presents the continuous kinematic-model-free controller for planar target reaching with gravity compensation;
- **Chapter 6** concludes the thesis by summarizing the research outcomes, results and findings, and future research directions.

Table 1.1: Contributions of kinematic-model-free control research.

Contributions	Description	Dissemination
KMF end-effector planar orientation control	Orientation control of a planar serial manipulator using angular control parameters	-
KMF end-effector planar position control of soft continuum robot	Tip-position control of a growing soft continuum robot with thread-like actuators in 3D space	[?]
KMF end-effector spatial position control of soft continuum robot	Planar tip-position control of a pneumatically-actuated soft continuum robot	[?]
KMF quaternionic end-effector position/orientation control	End-effector position/orientation control using quaternionic representation	-
KMF dual-quaternionic end-effector pose control	End-effector pose control using dual-quaternions in 3D space	[?]
KMF dual-quaternionic multi-point pose control	Multi-point control for robot configuration using dual-quaternions in 3D space	[?]
KMF obstacle avoidance	End-effector position control and path planning with obstacle avoidance using model-free predictive-control	[?]
KMF continuous end-effector position control	Planar position control of robot end-effector with continuous control for gravity compensation	-

References

- [1] Ahmad AlAttar, Ikhlas Ben Hmida, Federico Renda, and Petar Kormushev. Kinematic-model-free tip position control of reconfigurable and growing soft continuum robots. In *2023 IEEE International Conference on Soft Robotics (RoboSoft)*. IEEE, 2023.
- [2] Ahmad AlAttar, Digby Chappell, and Petar Kormushev. Kinematic-model-free predictive control for robotic manipulator target reaching with obstacle avoidance. *Frontiers in Robotics and AI*, 9:7, 2022.
- [3] Ahmad AlAttar, Francesco Cursi, and Petar Kormushev. Kinematic-model-free redundancy resolution using multi-point tracking and control for robot manipulation. *Applied Sciences*, 11(11):4746, 2021.
- [4] Ahmad AlAttar and Petar Kormushev. Kinematic-model-free orientation control for robot manipulation using locally weighted dual quaternions. *Robotics*, 9(4):76, 2020.
- [5] Tatsuya Aoki, Tomoaki Nakamura, and Takayuki Nagai. Learning of motor control from motor babbling. *IFAC-PapersOnLine*, 49(19):154–158, 2016.
- [6] Chien-Chern Cheah, Chao Liu, and Jean-Jacques E Slotine. Adaptive jacobian tracking control of robots with uncertainties in kinematic, dynamic and actuator models. *IEEE transactions on automatic control*, 51(6):1024–1029, 2006.
- [7] Hyungsuck Cho. *Opto-mechatronic systems handbook: techniques and applications*. CRC press, 2002.
- [8] Francesco Cursi, George P Mylonas, and Petar Kormushev. Adaptive kinematic modelling for multiobjective control of a redundant surgical robotic tool. *Robotics*, 9(3):68, 2020.
- [9] Cosimo Della Santina, Robert K Katzschmann, Antonio Bicchi, and Daniela Rus. Model-based dynamic feedback control of a planar soft robot: trajectory tracking and interaction with the environment. *The International Journal of Robotics Research*, 39(4):490–513, 2020.

- [10] Chase Frazelle, Ian Walker, Ahmad AlAttar, and Petar Kormushev. Kinematic-model-free control for space operations with continuum manipulators. In *2021 IEEE Aerospace Conference (50100)*, pages 1–11. IEEE, 2021.
- [11] Thomas George Thuruthel, Yasmin Ansari, Egidio Falotico, and Cecilia Laschi. Control strategies for soft robotic manipulators: A survey. *Soft robotics*, 5(2):149–163, 2018.
- [12] Ashitava Ghosal. *Robotics: fundamental concepts and analysis*. Oxford university press, 2006.
- [13] Yao Huang and Jianbo Su. Visual servoing of nonholonomic mobile robots: A review and a novel perspective. *IEEE Access*, 7:134968–134977, 2019.
- [14] Petar Kormushev, Sylvain Calinon, and Darwin Caldwell. Reinforcement learning in robotics: Applications and real-world challenges. *Robotics*, 2(3):122–148, 2013.
- [15] Petar Kormushev, Yiannis Demiris, and Darwin G Caldwell. Encoderless position control of a two-link robot manipulator. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 943–949. IEEE, 2015.
- [16] Petar Kormushev, Yiannis Demiris, and Darwin G Caldwell. Kinematic-free position control of a 2-dof planar robot arm. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5518–5525. IEEE, 2015.
- [17] Gaofeng Li, Dezhen Song, Shan Xu, Lei Sun, and Jingtai Liu. Kinematic-free orientation control for a deformable manipulator based on the geodesic in rotation group $so(3)$. *IEEE Robotics and Automation Letters*, 3(3):2432–2438, 2018.
- [18] Anup Teejo Mathew, Ikhlas Mohamed Ben Hmida, Costanza Armanini, Frederic Boyer, and Federico Renda. Sorosim: A matlab toolbox for hybrid rigid-soft robots based on the geometric variable-strain approach. *IEEE Robotics and Automation Magazine*, pages 2–18, 2022.

- [19] Ryota Morimoto, Satoshi Nishikawa, Ryuma Niiyama, and Yasuo Kuniyoshi. Model-free reinforcement learning with ensemble for a soft continuum robot arm. In *2021 IEEE 4th International Conference on Soft Robotics (RoboSoft)*, pages 141–148. IEEE, 2021.
- [20] Saeed B Niku. *Introduction to robotics: analysis, control, applications*. John Wiley & Sons, 2020.
- [21] MIH Nour, Jasmin Ooi, and KY Chan. Fuzzy logic control vs. conventional pid control of an inverted pendulum robot. In *2007 International Conference on Intelligent and Advanced Systems*, pages 209–214. IEEE, 2007.
- [22] Mark W Spong and Masayuki Fujita. Control in robotics. *The Impact of Control Technology: Overview, Success Stories, and Research Challenges*, 2011.
- [23] Cem Tutcu, Bora A Baydere, Seref K Talas, and Evren Samur. Quasi-static modeling of a novel growing soft-continuum robot. *The International Journal of Robotics Research*, 40(1):86–98, 2021.
- [24] Robert J Webster III and Bryan A Jones. Design and kinematic modeling of constant curvature continuum robots: A review. *The International Journal of Robotics Research*, 29(13):1661–1683, 2010.
- [25] Jun Wu, Jinsong Wang, and Zheng You. An overview of dynamic parameter identification of robots. *Robotics and computer-integrated manufacturing*, 26(5):414–419, 2010.
- [26] Qingxiang Zhao, Jiewen Lai, Kaicheng Huang, Xiaobing Hu, and Henry K Chu. Shape estimation and control of a soft continuum robot under external payloads. *IEEE/ASME Transactions on Mechatronics*, 2021.

Chapter 2

Kinematic-Model-Free Control

2.1 Related Work

Rigid robots are comprised of a set number of joints connected by rigid links. Rigid robots, although more straightforward to model and control, is not suitable for cluttered environments or where collisions occur that may damage the robot and control.

Soft robots have grown in popularity in the recent decade as an alternative to rigid robots, this is primarily due to their light weight, inherent compliance, and fault tolerance [?]. Soft Robots have highly varying designs and functionalities based on the field of application. Continuum robots are a class of soft robots with long and highly-deformable bodies, inspired by octopus tentacles and elephant trunks. These robots can be deployed as alternatives to traditional robotic arms in applications where delicate objects are being handled, or human-robot interaction is required. Further research into this class of robots has provided adaptable designs that are capable of actuator re-routing, or growth to adapt to varying target tip positions.

Re-configurable robots with actuator re-routing capabilities allow for a higher level of adaptability and an increased range of motion. This is explored in [?] where a continuum robot with actuators routed through disks attached to its backbone is presented. The rotation of the robot's disks facilitates the re-routing of actuators to form various helical paths. This results in

improved motion capabilities and different trajectories. Another case where actuator re-routing is explored is [?]. This work demonstrates the use of different paths to route parallel actuators in order to achieve different actuation trajectories.

Growing robots are another highly adaptable class that draws heavily from nature, where plants such as vines use growth as a means of navigation based on environmental stimuli [?]. In general, growth is the extension of the body through the tip, while maintaining no relative movement between the remaining body and the environment; i.e. without friction [?]. Growing robots that are capable of drastic changes in length have been investigated by researchers. One such robot is the growing robot in [?] which can change in length considerably relative to other similar robots through tip-extrusion. Researchers in [?, ?] developed a root-like growing robot that grows using an additive layering technique. These growing robots can be used in search and rescue as well as inspection applications.

A great deal of research has been done in the area of soft robot controllers, a challenging, but necessary undertaking that facilitates the practical use of these adaptable systems. Most of the controllers found in literature can be categorized into two main classes: Model-based controllers, which rely on the analytical model describing the system's motion for controller derivation; and Model-free controllers, which use data-driven and empirical methods to do the same [?]. These controllers are then further classified into sets that deal with either dynamic or inverse-kinematic control.

Research done on Model-based approaches for control includes: [?], where a closed-loop controller is introduced for dynamic trajectory tracking and surface following based on a Piece-wise Constant Curvature (PCC) assumption, which discretizes the profile of the manipulator into constant curvature sections; [?], where a controller is developed using the Geometric Variable Strain Model (GVS), a strain based discretization of the continuous Cosserat model; [?], where researchers calculate control parameters based on a quasi-static model to control the tip-position of a growing soft robot; or [?], researchers controlled a continuum robot using a variable curva-

ture kinematic modeling approach with parameters identification. This type of control makes geometric simplifications and assumptions which can be inaccurate in the presence of unmodeled forces, such as gravity. A model-based controller is limited by the inherent inaccuracies and singularities of the model it follows, an example of this would be the kinematic singularity of the PCC kinematics in the straight pose [?].

Alternatively, a model-free approach can be seen in: [?], where researchers performed shape reconstruction and used neural networks to learn to control the robot's tip position or in [?] where a continuum robot's tip position is controlled using deep reinforcement learning. Data-driven approaches in general provide a promising method for control learning for soft continuum robots. However, the control methods mentioned earlier require either the development of a model or the training of one prior to control.

2.2 KMF Position Control

Earlier KMF controllers, the Encoderless [?] and Kinematic-Free [?] controllers, are discrete controllers that control a manipulator's end-effector position using actuation primitive command signals. A high-level flowchart of the approach is shown in Figure 2.1. This approach assumes that the locally combined kinematics and dynamics, hereon refer to as "kinodynamics", of the robot can be learnt from sending pseudo-randomly generated actuation control signals to joints and observing their effect on the end effector of the robot. After gathering sufficient actuation-observation pairs, the local kinodynamics can be approximated and the controller can estimate the actuation primitive signal required to move the end effector towards a desired target position. After each actuation, the resulting and desired effects on the end effector are compared. If the difference is significant, a new exploratory phase is triggered to gather more information about the local kinodynamics.

The main components of the flowchart are as follows:

- A decision is made whether the end effector has reached the target position and the

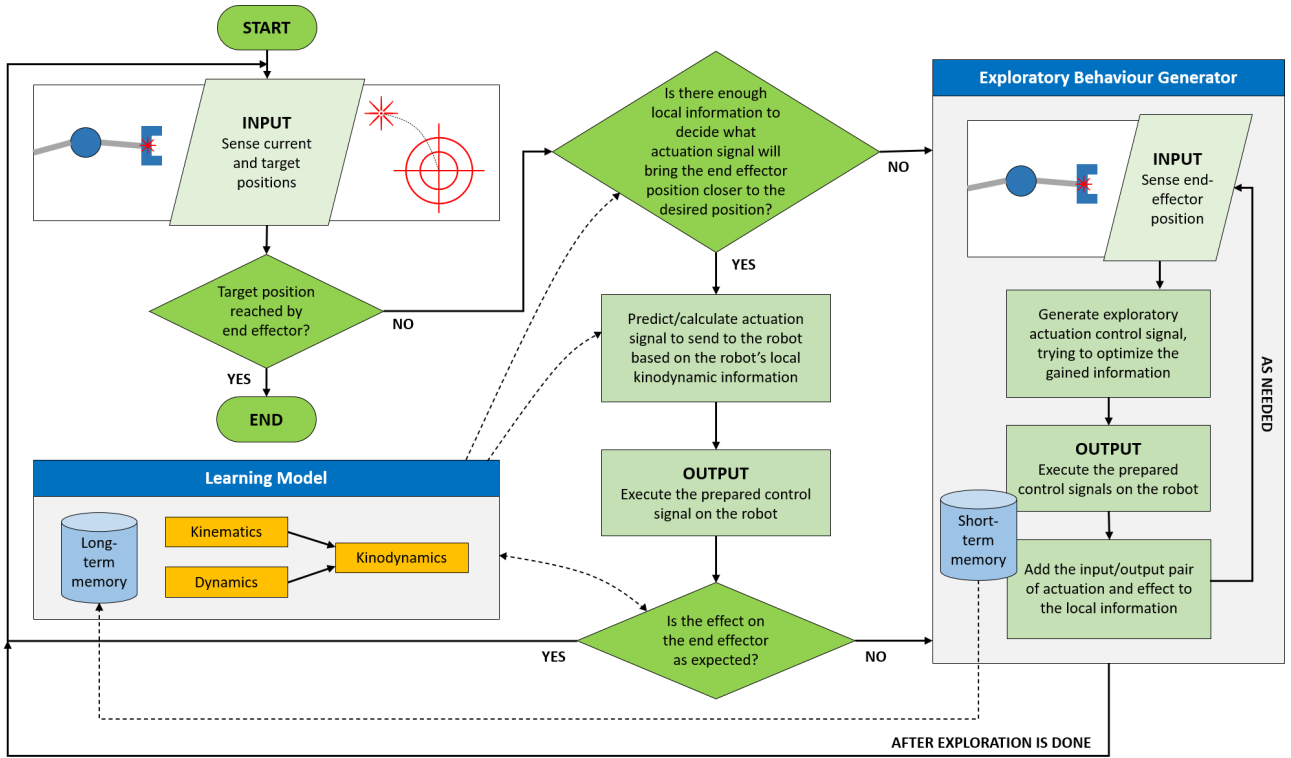


Figure 2.1: Flowchart of simple KMF position control.

controller should stop or further actuation is required.

- If there isn't sufficient information about the local kinodynamics, more information is gathered by triggering an exploratory phase.
- Using the available kinodynamics information, the controller predicts/estimates the required actuation to move the end effector towards a given reference position.
- The calculated actuation is executed on the robot without any encoder feedback.
- After each actuation, the resulting and desired effect on the end effector are compared. If the difference is significant, more information is gathered about the local kinodynamics by triggering an exploratory phase.
- In the exploratory phase, pseudo-random actuation primitives that could vary in magnitude and duration are generated.
- The generator has a short-term memory which helps generate fewer primitives while optimizing the gained information.

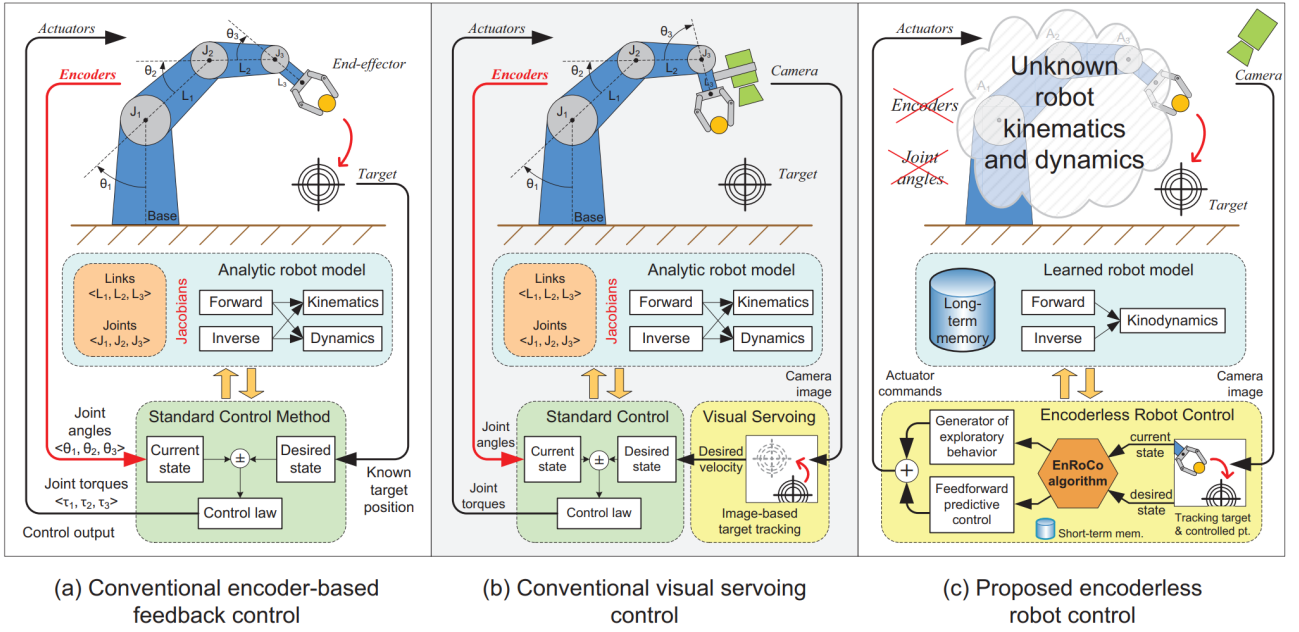


Figure 2.2: Unlike the conventional controllers in (a) and (b), the kinematic-model-free controller in (c) does not use encoder feedback or joint angle feedback during control. Instead, the controller using exteroception to track the robot's end effector. Also, the kinematic-model-free controller does not have any prior knowledge of the robot's kinematic or dynamic models [?].

While the controller is running, the actuation primitives and their resulting effects are recorded in the long-term memory which is used for calculating new predictions. The differences between kinematic-model-free control and other control methods are shown in Figure 2.2.

Actuation Primitives

Actuation primitives produce control signals $\tau(t)$, which could be torques, voltages, currents, etc. These signals are sent to the actuators to move the robot. An actuation primitive is defined as a function of time:

$$\tau(t) = \begin{cases} \tau_p & \text{if } t \in [t_0, t_0 + \frac{d_p}{2}) \\ -\tau_p & \text{if } t \in [\frac{d_p}{2}, t_0 + d_p) \\ 0 & \text{if } t \in (-\infty, t_0) \cup (t_0 + d_p, \infty) \end{cases} \quad (2.1)$$

where t_0 is the starting time, τ_p defines the magnitude of the actuation primitive, and d_p defines the duration of the primitive. See Figure 2.3. The controller sends the control signal generated by the actuation primitives to the actuators of the robot to move the end

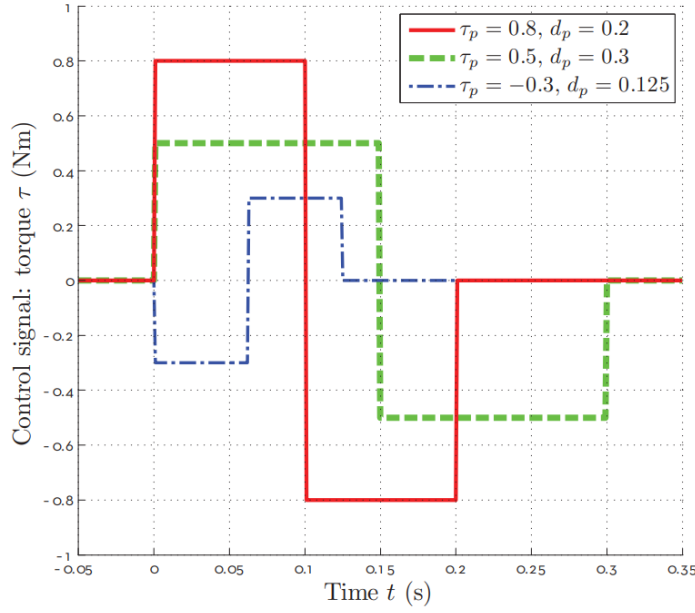


Figure 2.3: Example actuation primitives with different parameters: duration d_p and magnitude τ_p [?].

effector towards the target reference position. The actuation signals are sent sequentially and synchronously to the robot.

Exploratory Behaviour

Whenever there is insufficient information about the local kinodynamics, at start and whenever the robot behaves unpredictably, an exploratory phase is triggered. The exploratory phase executes pseudo-random actuation primitives on the robot. A short-term memory, D_s , helps optimally select actuation primitives to send that maximize information gain. In the implementations of [?] and [?], the short-term memory was used to insure the orthogonality of the actuation primitives. After each movement, actuation-observation pairs are stored in the long-term memory, D_l .

Actuation Prediction

Let $\hat{p}$ be the desired actuation primitive whose parameters $\tau_p(\hat{p})$ are to be estimated to actuate

the end effector towards the target position:

$$b_1 = \begin{bmatrix} \tau_p^1(\hat{p}) \\ \tau_p^2(\hat{p}) \\ \vdots \\ \tau_p^m(\hat{p}) \end{bmatrix} \quad (2.2)$$

where $\tau_p^j(\hat{p})$ is the magnitude of the desired actuation primitive $\hat{p}$ on actuator j . The principle notion behind KMF control is to represent the desired primitive as a linear combination of kNN primitives recorded in D_l . The distance is calculated from the current end-effector position to the starting point of each primitive. The linear combination is expressed as follows:

$$A_1 w = b_1 \quad (2.3)$$

where $w = [w_1, w_2, \dots, w_k]^T$ is the vector of unknown weights and matrix A_1 consists of the kNN actuation primitives:

$$A_1 = \begin{bmatrix} \tau_p^1(p_1) & \tau_p^1(p_2) & \cdots & \tau_p^1(p_k) \\ \tau_p^2(p_1) & \tau_p^2(p_2) & \cdots & \tau_p^2(p_k) \\ \vdots & \vdots & \ddots & \vdots \\ \tau_p^m(p_1) & \tau_p^m(p_2) & \cdots & \tau_p^m(p_k) \end{bmatrix} \quad (2.4)$$

where $\tau_p^j(p_i)$ is the magnitude of the actuation primitive i on actuator j .

Locally-Weighted Learning

During exploration, actuation primitives and their corresponding effects on the end effector are recorded. At any step during reaching, the controller finds the kNN tuples of action primitives and their corresponding effects. The observation matrix is defined as follows:

$$A_2 = \begin{bmatrix} \Delta x(p_1) & \Delta x(p_2) & \cdots & \Delta x(p_k) \\ \Delta y(p_1) & \Delta y(p_2) & \cdots & \Delta y(p_k) \\ \Delta z(p_1) & \Delta z(p_2) & \cdots & \Delta z(p_k) \end{bmatrix} \quad (2.5)$$

where $\Delta i(p_j)$ is the displacement effect on the end effector along the i -axis as a result of exerting actuation p_j . Let the desired effect be expressed as follows:

$$b_2 = \begin{bmatrix} \Delta x(\hat{p}) \\ \Delta y(\hat{p}) \\ \Delta z(\hat{p}) \end{bmatrix} \quad (2.6)$$

The desired effect, b_2 , can be expressed as a linear combination of the kNN observed effects, A_2 , as follows:

$$A_2 w = b_2 \quad (2.7)$$

Note that many solutions might exist for w since A_2 might not be full rank. To resolve this issue, least squares regression is used to find the smallest squared vector b_2 that is a solution to the equation. This calculated value can then be used in equation (2.3) to find the desired actuation primitive that will actuate the end effector towards the desired target position.

Remarks

Note that the order of actuators does not matter. The controller does not know which actuators corresponds to which joint. Since the controller does not use any prior model of the robot and makes no assumption about the actuators and their relationship to the joints, this makes the controller agnostic to kinematic changes in the robot. This is one major advantage over conventional controllers. Despite their adaptability, this form of the controller is limited to position control, completely disregarding orientation.

2.3 KMF Orientation Control

Although orientation control on its own has little-to-no useful applications, the mathematical framework is presented here for completeness and the concepts in this section will lay the foundation on top of which simultaneous position and orientation, or pose, control will be derived in the next section. Figure 2.4 shows a high-level flowchart of the approach.

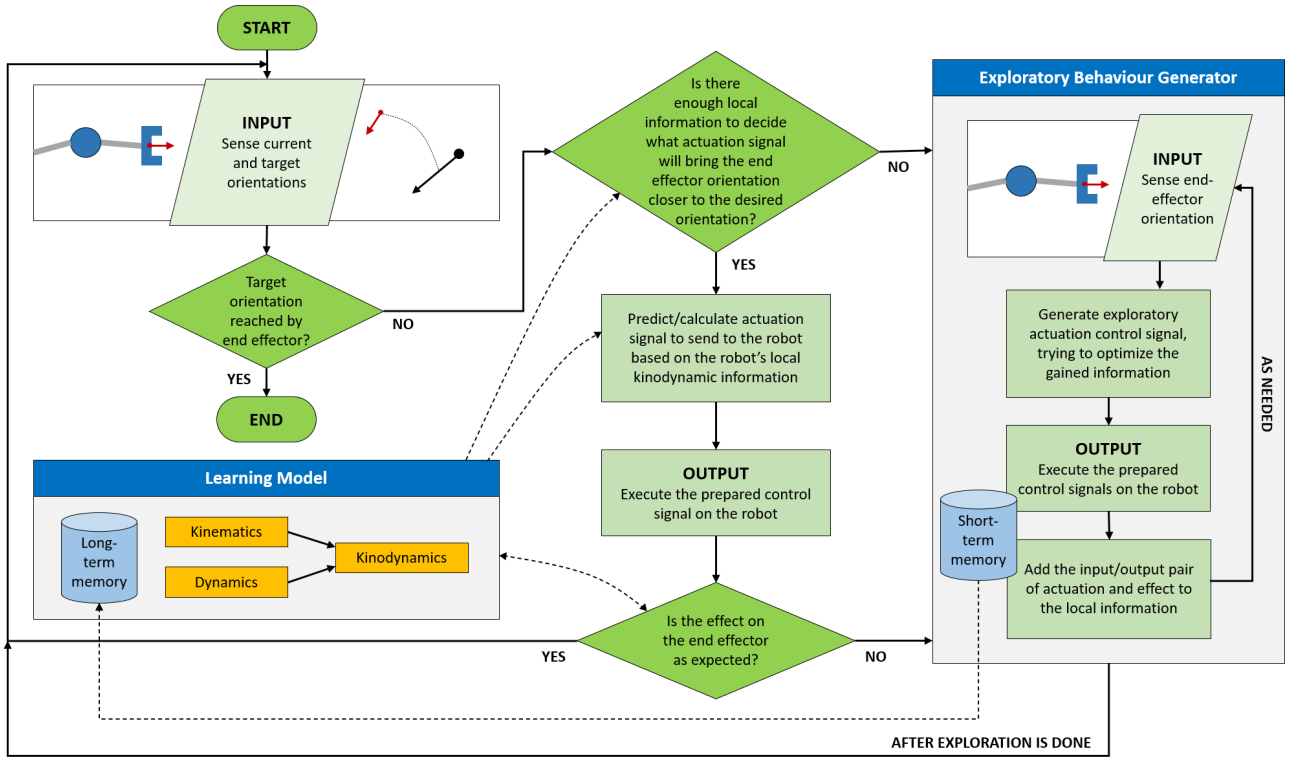


Figure 2.4: Flowchart of simple KMF orientation control.

There are several representations for orientation. For example, rotation matrices, Euler angles, quaternions, etc. In KMF orientation control for planar robots, orientation of the end effector is simply represented by a single variable θ . The observation matrix, in this case a 1D vector, would simply be a collection of the kNN rotational displacements:

$$A_2 = [\Delta\theta(p_1), \Delta\theta(p_2), \dots, \Delta\theta(p_k)] \quad (2.8)$$

each of which corresponds to a primitive in the actuation matrix:

$$A_1 = \begin{bmatrix} \tau_p^1(p_1) & \tau_p^1(p_2) & \cdots & \tau_p^1(p_k) \\ \tau_p^2(p_1) & \tau_p^2(p_2) & \cdots & \tau_p^2(p_k) \\ \vdots & \vdots & \ddots & \vdots \\ \tau_p^m(p_1) & \tau_p^m(p_2) & \cdots & \tau_p^m(p_k) \end{bmatrix} \quad (2.9)$$

where $\tau_p^j(p_i)$ is the magnitude of the actuation primitive i on actuator j . At any step, the

controller attempts to estimate an actuation primitive:

$$b_1 = \begin{bmatrix} \tau_p^1(\hat{p}) \\ \tau_p^2(\hat{p}) \\ \vdots \\ \tau_p^m(\hat{p}) \end{bmatrix} \quad (2.10)$$

where $\tau_p^j(\hat{p})$ is the magnitude of the desired actuation primitive $\hat{p}$ on actuator j to move the end effector towards the desired goal orientation:

$$b_2 = \Delta\theta(\hat{p}) \quad (2.11)$$

The following local linear model is then solved for the unknown weights:

$$A_2 w = b_2 \quad (2.12)$$

where the weights, $w = [w_1, w_2, \dots, w_k]^T$ is used to estimate the actuation primitive through the following local linear model:

$$A_1 w = b_1 \quad (2.13)$$

This controller is capable of adjusting the orientation of a robot's end effector and performing simple rotational reaching tasks. However, position control is completely disregarded, limiting the application of this controller. Moreover, this controller is only applicable to planar motion since only a single value is used to represent the end effector's orientation.

2.4 KMF Pose Control

Most robot manipulation tasks require simultaneous position and orientation, or pose, control. The simple KMF pose controller is capable of controlling the pose of the end effector of planar redundant robots. Figure 2.5 shows the high-level flowchart of the controller.

The robot tries to estimate some actuation primitive to move the end effector towards a

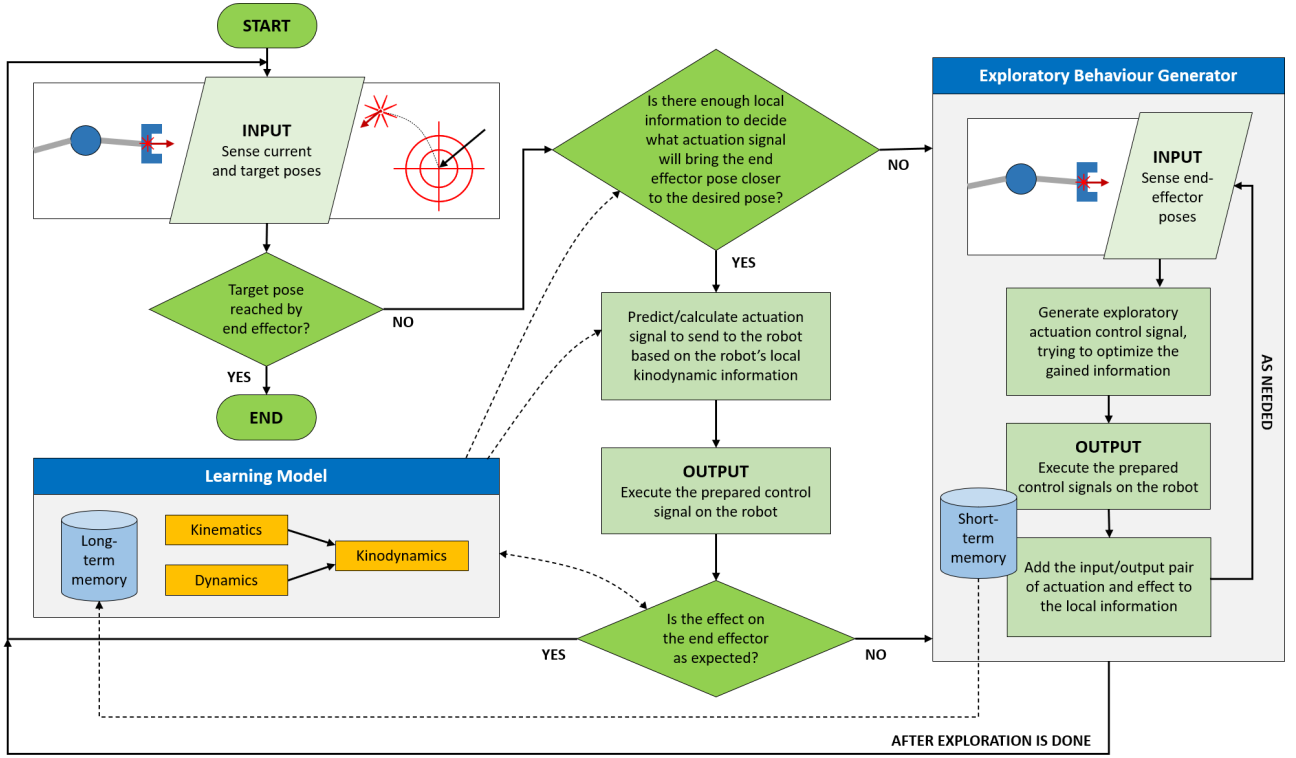


Figure 2.5: Flowchart of simple KMF pose control.

desired goal pose $[x_t, y_t, \theta_t]^T$:

$$b_1 = \begin{bmatrix} \tau_p^1(\hat{p}) \\ \tau_p^2(\hat{p}) \\ \vdots \\ \tau_p^m(\hat{p}) \end{bmatrix} \quad (2.14)$$

where $\tau_p^j(\hat{p})$ is the magnitude of the desired actuation primitive $\hat{p}$ on actuator j . The required primitive $\hat{p}$ is a linear combination of the kNN primitives that have been previously executed and recorded. The kNN distance is calculated from the end effector's current position to the starting position of each primitive p_i . The linear combination of the kNN primitives can be expressed in matrix form as follows:

$$A_1 w = b_1 \quad (2.15)$$

where $w = [w_1, w_2, \dots, w_k]^T$ is a set of unknown weights and the A_1 contains the primitives

of the kNN:

$$A_1 = \begin{bmatrix} \tau_p^1(p_1) & \tau_p^1(p_2) & \cdots & \tau_p^1(p_k) \\ \tau_p^2(p_1) & \tau_p^2(p_2) & \cdots & \tau_p^2(p_k) \\ \vdots & \vdots & \ddots & \vdots \\ \tau_p^m(p_1) & \tau_p^m(p_2) & \cdots & \tau_p^m(p_k) \end{bmatrix} \quad (2.16)$$

where $\tau_p^j(p_i)$ is the magnitude of the actuation primitive i on actuator j . To find the unknown weights w , the information recorded about the outcomes of each of the kNN primitives is used:

$$A_2 = \begin{bmatrix} \Delta x(p_1) & \Delta x(p_2) & \cdots & \Delta x(p_k) \\ \Delta y(p_1) & \Delta y(p_2) & \cdots & \Delta y(p_k) \\ \Delta \theta(p_1) & \Delta \theta(p_2) & \cdots & \Delta \theta(p_k) \end{bmatrix} \quad (2.17)$$

where $[\Delta x(p_i), \Delta y(p_i), \Delta \theta(p_i)]^T$ is the relative displacement of the end effector corresponding to primitive p_i . Next, the current end-effector and target poses are used to calculate the relative displacement required. This will actuate the end effector towards the target pose. The relative displacement of the end-effector is as follows:

$$b_2 = \begin{bmatrix} \Delta x(\hat{p}) \\ \Delta y(\hat{p}) \\ \Delta \theta(\hat{p}) \end{bmatrix} \quad (2.18)$$

The weights w are solved for using the following expression:

$$A_2 w = b_2 \quad (2.19)$$

which is then used to find the required primitive to move the end effector towards the target.

Orientation wrapping

The key challenge with orientation control is handling the non-linear orientation wrapping that occurs at $\pm\pi$. Three solutions were identified: quaternion representation, angle representation change, and angle range switching.

Quaternion representation Angles can be represented using quaternions $ai + bj + ck + w$. The four numbers a , b , c , and d increases the complexity of the path planning space to 6D, $\{x, y, a, b, c, w\}$. Although it will resolve the orientation wrapping problem, it requires excessive computation for simple planar control.

Orientation representation change In orientation representation change, orientation is represented by two numbers, a and b where $a = \sin(\theta)$ and $b = \cos(\theta)$ where θ is a 1D manifold represented in a 2D space as a and b . Thus, making the path planning space 4D $\{x, y, a, b\}$. The KMF controller will attempt to find a straight line in this 4D space to the target $[x_t, y_t, a_t, b_t]$. This approach will also increase the complexity of unnecessarily for a simple planar control case.

Orientation range switching In orientation range switching, orientation is represented by a single number θ measured from the positive x-axis. Thus, the controller will find a straight line path to the target $[x, y, \theta]$ in the 3D space $\{x, y, \theta\}$, shown in Figure 2.6.

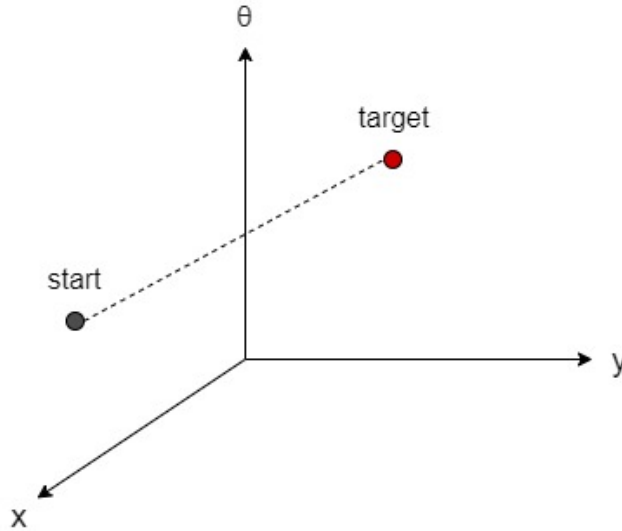


Figure 2.6: Pose path planning space

Based on the orientation of the current end effector θ and the target θ_t , the orientation range is switched between $[-\pi, +\pi]$ and $[0, 2\pi]$. The range switching technique follows this algorithm:

```

 $x = \operatorname{argmin}[range_{[-\pi, +\pi]}(\theta_t - \theta), range_{[0, 2\pi]}(\theta_t - \theta)]$ 

IF  $x == range_{[-\pi, +\pi]}(\theta_t - \theta)$  :

    switch range to  $[-\pi, +\pi]$ 

ELSE :

    switch range to  $[0, 2\pi]$ 
    
```

This method insures that the discontinuity at $\pm\pi$ is avoided. This approach is most suitable for the planar robot case because the angle is represented by a single number, keeping the path planning space 3D.

2.5 KMF Tendon-Driven Continuum Robot Control

The kinematic-model-free controller generates and learns online local linear models that it uses for actuating the robot's tip position, on the other hand, other data-driven approaches such as those discussed in the introduction, either use offline learning or learn global models. The quasi-static assumption employed by the kinematic-model-free controller allows the control of a robot manipulator through actuation. Actuation primitives produce control signals that are used for actuating the robot. The controller initiates a random exploration where samples are gathered. The random exploration process is achieved by sending randomly-generated control signals $\tau(t)$ to the robot actuators in the following form:

$$\tau(t) = \tau(t-1) + \Delta\tau(t) \quad (2.20)$$

where $\tau(t-1)$ is the previous actuation sent, $\tau(t-1) = 0$ for $t = 0$. Taking into account previous actuation primitives, gravity compensation is achieved, as opposed to previous work

done in kinematic-model-free control where simply $\tau(t) = \Delta\tau(t)$. Note that:

$$\Delta\tau(t) = \begin{cases} \tau_p & t \in [t_0, t_0 + d_p) \\ 0 & \text{otherwise} \end{cases}. \quad (2.21)$$

where τ_p is the actuation magnitude and d_p is the duration of the actuation signal. After each actuation, the end-effector displacement, $\psi_p = [\Delta x, \Delta y, \Delta z]$, is sampled, forming tuples of the form $\{\tau_p, \psi_p\}$. Actuators are stored in the actuation matrix:

$$A_1 = \begin{bmatrix} \tau_p^1(p_1) & \tau_p^1(p_2) & \cdots & \tau_p^1(p_k) \\ \tau_p^2(p_1) & \tau_p^2(p_2) & \cdots & \tau_p^2(p_k) \\ \vdots & \vdots & \ddots & \vdots \\ \tau_p^n(p_1) & \tau_p^n(p_2) & \cdots & \tau_p^n(p_k) \end{bmatrix} \quad (2.22)$$

where $\tau_p^i(p_j)$ is the j^{th} actuation signal on actuator i , n is the robot's degrees of freedom that are controlled, and k is the k -nearest-neighbor (k NN) samples to the robot's current end-effector position. The corresponding end-effector displacements are stored in the observation matrix:

$$A_2 = \begin{bmatrix} \Delta x(p_1) & \Delta x(p_2) & \cdots & \Delta x(p_k) \\ \Delta y(p_1) & \Delta y(p_2) & \cdots & \Delta y(p_k) \\ \Delta z(p_1) & \Delta z(p_2) & \cdots & \Delta z(p_k) \end{bmatrix} \quad (2.23)$$

where $\Delta i(\hat{p})$ is the displacement of the robot's end-effector along the i^{th} direction in Cartesian space. After the completion of the exploratory phase, the controller calculates the required actuation signal to move the robot's end effector towards via points, $[x_i, y_i, z_i]$, that lead to the final target position $[x_t, y_t, z_t]$. The actuation signal is in the form:

$$b_1 = \begin{bmatrix} \tau_p^1(\hat{p}) \\ \tau_p^2(\hat{p}) \\ \vdots \\ \tau_p^n(\hat{p}) \end{bmatrix} \quad (2.24)$$

where $\hat{p}$ is the actuation primitive desired that will be estimated as a linear combination of

other actuation primitives previously sampled. This actuation signal will cause the following displacement to the robot's end effector:

$$b_2 = \begin{bmatrix} \Delta x(\hat{p}) \\ \Delta y(\hat{p}) \\ \Delta z(\hat{p}) \end{bmatrix} \quad (2.25)$$

which will move the end effector towards the target position; i.e. it is the displacement between two consecutive via points that leads the robot's end effector towards the final target position. Estimating b_1 starts by generating a volatile local linear model as follows:

$$A_2 w = b_2 \quad (2.26)$$

where $w = [w_1, w_2, \dots, w_k]^T$ is the vector of unknown weights. The weights are solved for using least squares regression and are then plugged into the following:

$$A_1 w = b_1 \quad (2.27)$$

to yield an estimate for b_1 which is sent to the robot. Should any discrepancy occur between the desired end-effector displacement and actual displacement, a new exploratory phase is triggered. It is important to note that the local linear models are volatile and are regenerated at every step.

2.6 KMF Pneumatic Continuum Robot Control

Continuum robots can be pneumatically controlled using McKibben muscles, which extend along their lengths when pressurized. Placing several muscles in parallel can create a section in the continuum robot. Pressurizing the muscles in a section by the same amount will cause an extension. Bending, on the other hand, can be achieved differentially actuating the muscles. The constant-curvature model can be used to describe the kinematics of each section using 3 values: bending magnitudes u and v and arc length s . The bending values u and v represent

the bending along the local x-axis and y-axis, respectively. The combined magnitudes gives the overall section bending magnitude and the direction can be found as follows:

$$\phi = \tan^{-1} \left(\frac{v}{u} \right) \quad (2.28)$$

To achieve proper motion in continuum robots, other muscles have to be relaxed in an amount proportional to the muscle being pressurized. For fixed length continuum robots, differential actuation must be applied. The pressure value for muscle i in a 3-muscle section of the continuum robot can be related to the parameters bending parameters and arc-length as follows:

$$p_1 = \int \tau_s(t)dt - d \int \tau_v(t)dt \quad (2.29)$$

$$p_2 = \int \tau_s(t)dt + \frac{d}{2} \int \tau_v(t)dt + \frac{\sqrt{3}d}{2} \int \tau_u(t)dt \quad (2.30)$$

$$p_3 = \int \tau_s(t)dt + \frac{d}{2} \int \tau_v(t)dt - \frac{\sqrt{3}d}{2} \int \tau_u(t)dt \quad (2.31)$$

where τ_u , τ_v , and τ_s are the primitives mapped to the kinematic values u , v , and s , respectively, and can be used as actuation primitives for the KMF controller. The actuation vector can then be written as:

$$b_1 = \begin{bmatrix} \tau_u(\hat{p}) \\ \tau_v(\hat{p}) \\ \tau_s(\hat{p}) \end{bmatrix} \quad (2.32)$$

that is calculated using the actuation matrix, A_1 , observation matrix, A_2 , and displacement

vector, b_2 :

$$A_1 = \begin{bmatrix} \tau_u(p_1) & \tau_u(p_2) & \cdots & \tau_u(p_k) \\ \tau_v(p_1) & \tau_v(p_2) & \cdots & \tau_v(p_k) \\ \tau_s(p_1) & \tau_s(p_2) & \cdots & \tau_s(p_k) \end{bmatrix} \quad (2.33)$$

$$A_2 = \begin{bmatrix} \Delta x(p_1) & \Delta x(p_2) & \cdots & \Delta x(p_k) \\ \Delta y(p_1) & \Delta y(p_2) & \cdots & \Delta y(p_k) \\ \Delta z(p_1) & \Delta z(p_2) & \cdots & \Delta z(p_k) \end{bmatrix} \quad (2.34)$$

and

$$b_2 = \begin{bmatrix} \Delta x(\hat{p}) \\ \Delta y(\hat{p}) \\ \Delta z(\hat{p}) \end{bmatrix} \quad (2.35)$$

The following volatile local linear model is formulated

$$A_2 x = b_2 \quad (2.36)$$

and solved for x and plugged into the following volatile local linear model

$$A_1 x = b_1 \quad (2.37)$$

which is solved for b_1 to find the actuation signal sent to the robot. Any discrepancy between the expected and observed behavior will trigger an exploratory phase.

2.7 Results

2.7.1 Simulated Rigid Robot Control

The kinematic-model-free controller was tested on a simulated 2-DoF robot manipulator in 2D space and a simulated 3-DoF robot manipulator in 3D space, which were developed using

MATLAB's Robotics toolbox [?]. For both robots, the task was to reach a target point. See Figure 2.7. In both cases, the robot starts with an exploration phase and then calculates actuation primitives that brings the end effector towards the target point.

Furthermore, simultaneous position and orientation control was demonstrated in simulation using a 3-DoF planar robot manipulator. See Figure 2.8. The robot first had to reach a target pose in 2D space. Secondly, the robot had to trace a square path while keeping the robot's end effector perpendicular to the path. Once again, the robot starts with an exploration phase and then begins to move towards the target point(s). Whenever a discrepancy between the expected and actual end-effector displacement is measured, another exploration phase is triggered.

2.7.2 Simulated Continuum Robot Control

In order to simulate the behavior of soft robots under various loading conditions the SoRoSim toolbox [?], is used. SoRoSim is a MATLAB toolbox based on the Geometric Variable Strain (GVS) approach introduced by Renda et al. [?] in statics and Boyer et al.[?] in dynamics. The GVS approach is based on a variable strain parametrization of soft links represented by Cosserat rods, 1D, slender rods that can bend, twist, stretch, and shear. SoRoSim directly extends rigid robot modeling techniques to soft and hybrid systems, while maintaining a high level of accuracy, by exploiting the high fidelity of the Cosserat Rod approach. The simplifying assumptions of the GVS approach minimize the number of DoFs required to represent systems and provide a geometrically exact framework leading to accurate, fast, and computationally less expensive results.

The toolbox enables users to define and manipulate links (rigid or soft) and robotic systems (linkages). GUIs assist in the definition and assembly of links as well as the definition of DoFs and constraints. A variety of typical external forces and actuation inputs are handled by GUIs, which provides a black-box experience, allowing users from all backgrounds to easily use the toolbox to perform static and dynamic analyses. SoRoSim can be used alongside built-in functions, add-ons such as the Optimization Toolbox, and other custom codes to further facilitate the

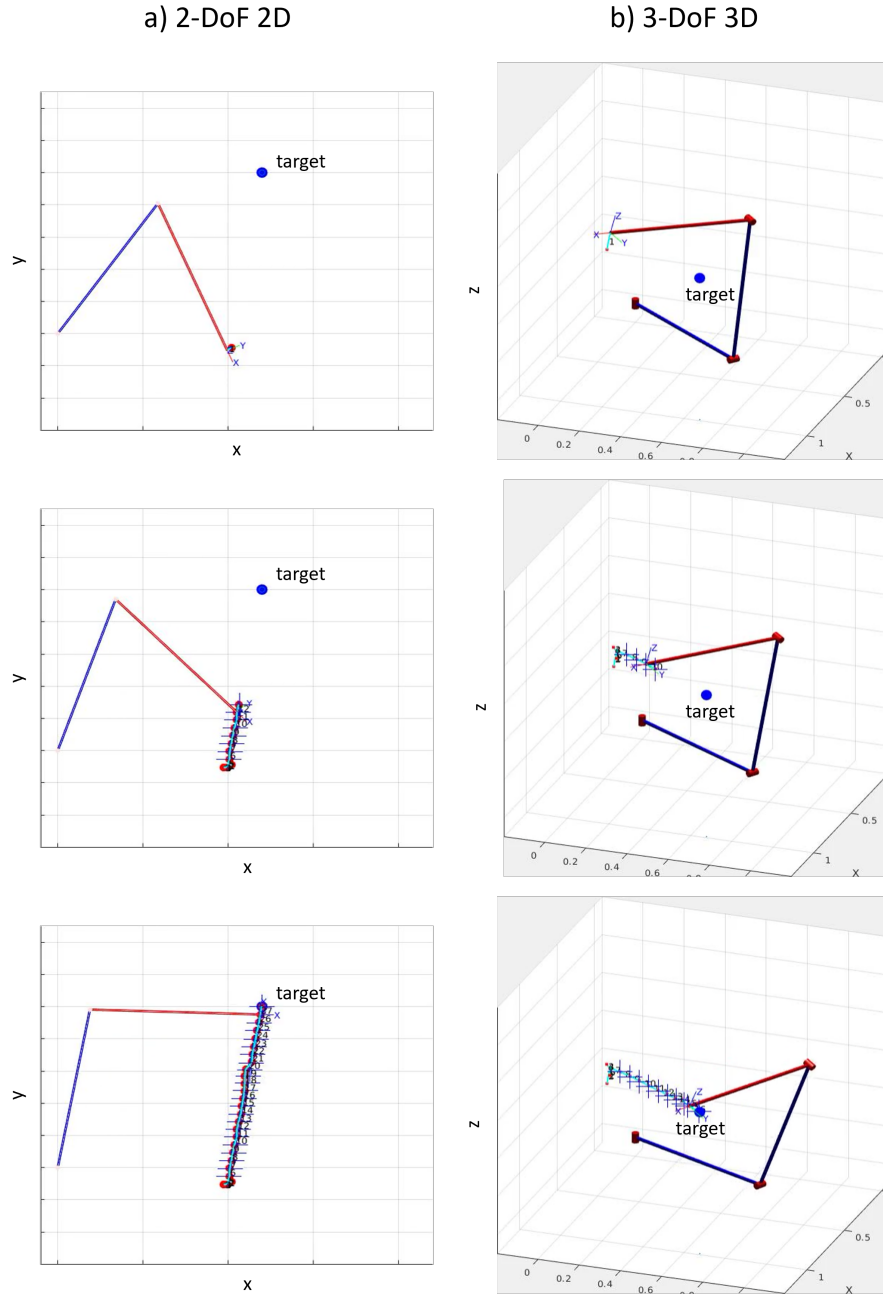


Figure 2.7: a) A 2-DoF robot manipulator reaching a target point in 2D space; b) A 3-DoF robot manipulator reaching a target point in 3D space.

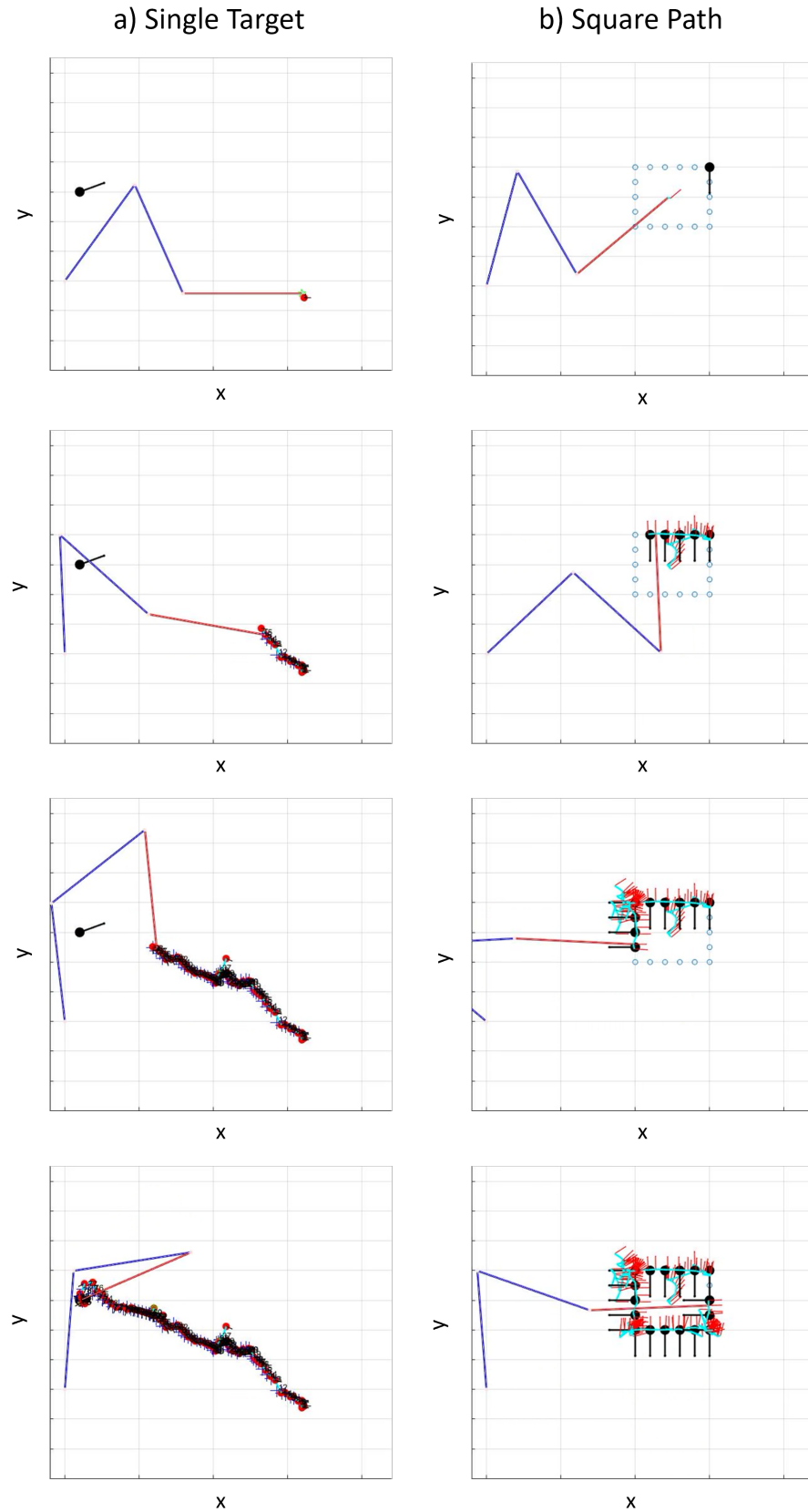


Figure 2.8: a) A 3-DoF robot manipulator reaching a target pose in 2D space; b) a 3-DoF robot manipulator tracing a square while keeping the end effector perpendicular to the path followed.

analysis and control of robotic systems.

Four tests were run to validate the merits of deploying the kinematic-model-free controller on a continuum soft robot: 1) tracing a circle with the robot's tip; 2) reaching a target point with sudden actuator failures; 3) target reaching with simultaneous actuator strength and actuator configuration control; and 4) target reaching with actuator strength control and active growth.

The robot used for the simulations is 0.58 m long, with a linearly varying radius that goes from $r_{base} = 0.015$ m to $r_{tip} = 0.008$ m. The material properties used were $E = 50$ MPa and $\nu = 0.5$. Six parallel actuators were routed as detailed in the table shown in Figure 2.9 and used to actuate the robot. It is important to note that gravity is on in all simulation tests and that all target positions lie within the robot's reachable workspace.

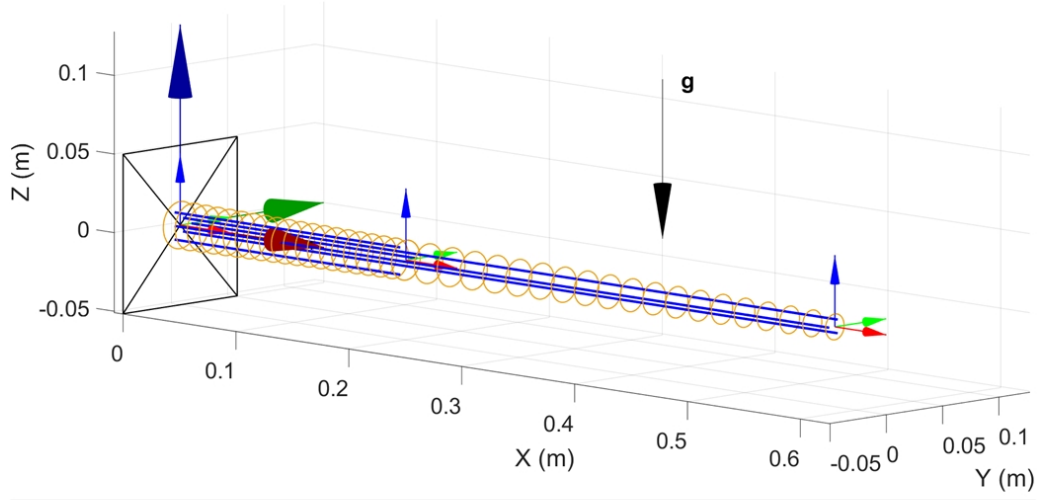
Tracing a Circle

In the first test, the controller attempts to trace a circle with its tip. The circle is defined as follows:

$$r(t) = \begin{bmatrix} 0.5 \\ 0.15\cos((2\pi/5)t) \\ 0.15\sin((2\pi/5)t) \end{bmatrix} \quad (2.38)$$

The robot starts from an extended configuration, which is the static equilibrium solution to the gravitational load, where actuator strengths are zero and the tip is positioned at $[x, y, z] = [0.580, 0, -0.017]$. Gravity is enabled in the simulator with the acceleration due to gravity defined as $[g_x, g_y, g_z] = [0, 0, -9.81]$.

Target points from the circular trajectory were used to calculate desired actuation signals for the robot. The controller starts off by going through an exploratory phase to collect samples. It then uses the closest ($kNN = 4$) samples to calculate the next desired actuation. In Figure 2.10(a), the simulation result of tracing a circle is shown. The controller managed to trace the



Actuator Number	Actuator Length	Y – Coordinate	Z – Coordinate
1	0.20	d_1	0
2	0.20	$\frac{-d_1}{2}$	$\frac{d_1\sqrt{3}}{2}$
3	0.20	$\frac{-d_1}{2}$	$\frac{-d_1\sqrt{3}}{2}$
4	0.58	$\frac{d_2}{2}$	$\frac{-d_2\sqrt{3}}{2}$
5	0.58	$\frac{d_2}{2}$	$\frac{d_2\sqrt{3}}{2}$
6	0.58	d_2	0

$$d_1 = 0.01 \text{ and } d_2 = 0.005 \text{ m}$$

Figure 2.9: Overview plot of the design of the continuum robot used for the simulation tests along with a table detailing the actuator locations.

circle in 3D space in 305 steps.

The robot tracks the circular trajectory similar to [?], which traces a circle with the same radius as (2.38). However, the slight movements off the circular trajectory are due to exploratory behavior being re-triggered due to discrepancies measured between the desired and actual displacements. Furthermore, it is worth noting that in [?], gravitational forces are not considered but using the kinematic-free controller's modified actuation signal in equation 5.8 the controller easily compensates for gravity. In Figure 2.10(b), the delta actuation strengths calculated by the controller are shown which are accumulated to yield the final actuation sent to the robot as shown in Figure 2.10(c). We also note the discrete shape of the signal since the controller at hand is a discrete-time controller that runs in discrete steps. However, relative to all previous work, the signal is more continuous in the sense that incremental changes are added to the current signal, as defined by Equation (1), which is a contribution that enabled proper gravity compensation to be achieved. Furthermore, despite having infinite DoFs, all six actuators in the robot are independent and alter the end-effector's tip position equally, as opposed to the work done in [?] where the actuators were in series and affected one another.

Actuator Failure

In the second test, the controller attempts to reach a point in 3D space where a sudden failure occurs to a subset of the actuators, reducing the robot's degrees of actuation. The target point which the manipulator attempts to move the tip towards is $[0.426, -0.177, 0.065]$. actuators 1 and 4 are disabled at step 25 of the control to simulate a failure in their actuation and demonstrate the effectiveness of kinematic-model-free control in overcoming this issue and guiding the tip to the target position despite the reduced number of actuators.

The controller actuates the tip to within 1 mm from the target. After the initial exploratory phase, the tip begins to move towards the target in a straight trajectory until the actuators are disabled in step 25. Point 2 in Figure 2.11(a) shows step 24 prior to actuator failure. We observe a large dip away from the expected trajectory due to the change in actuators in step 25. This triggers a new exploration phase at point 3, which enables the controller to guide the

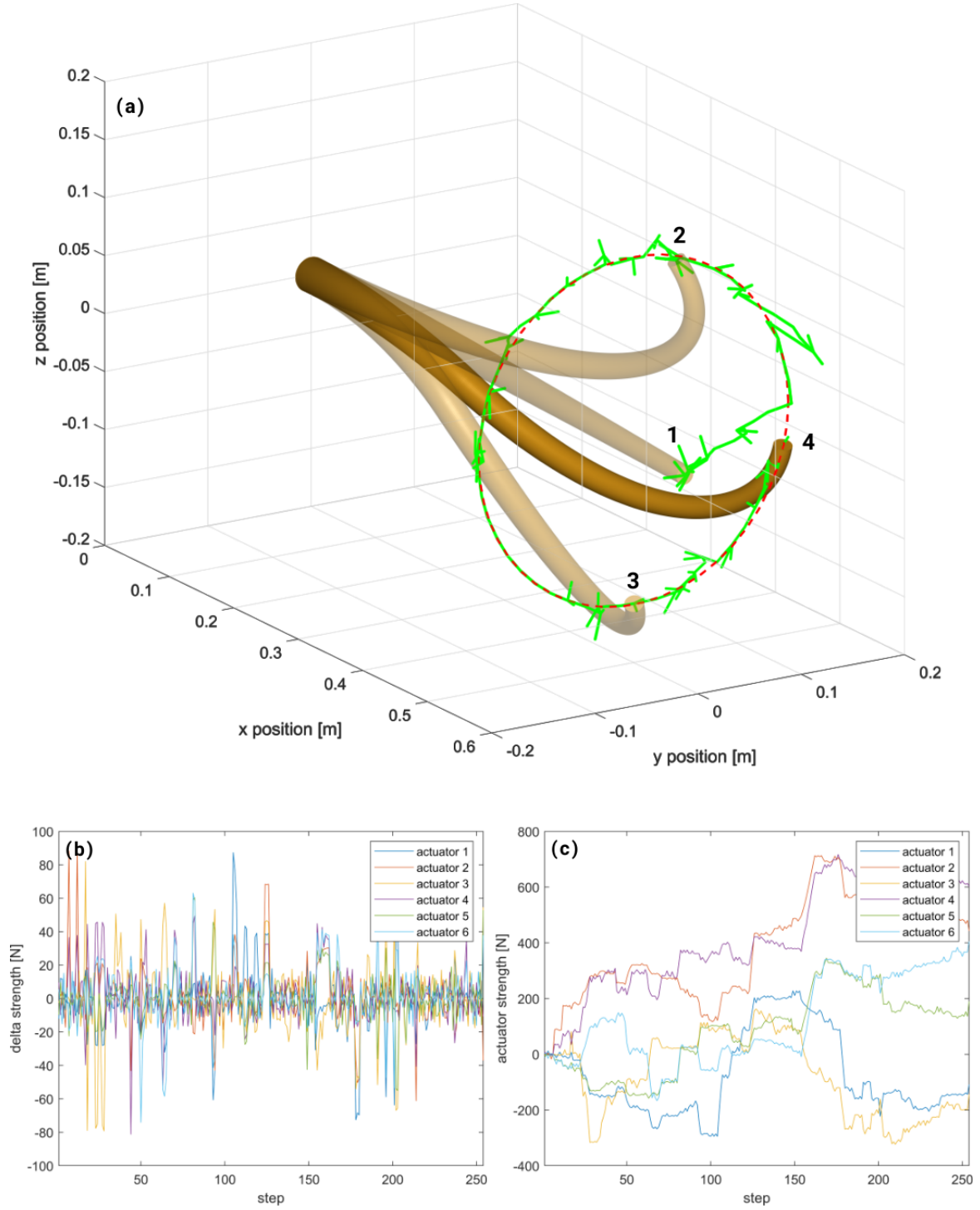


Figure 2.10: (a) Manipulator tracing a circle in 3D space. The controller performs both tip-position control and gravity compensation. The starting position is marked by 1 and the ending position is marked by 4. Marks 2 and 3 are almost at the circle top and bottom positions; (b) Change in actuation strengths, $\Delta\tau(t)$, added to overall actuation signal for control; (c) Actuation strengths, $\tau(t)$, used to control each actuator.

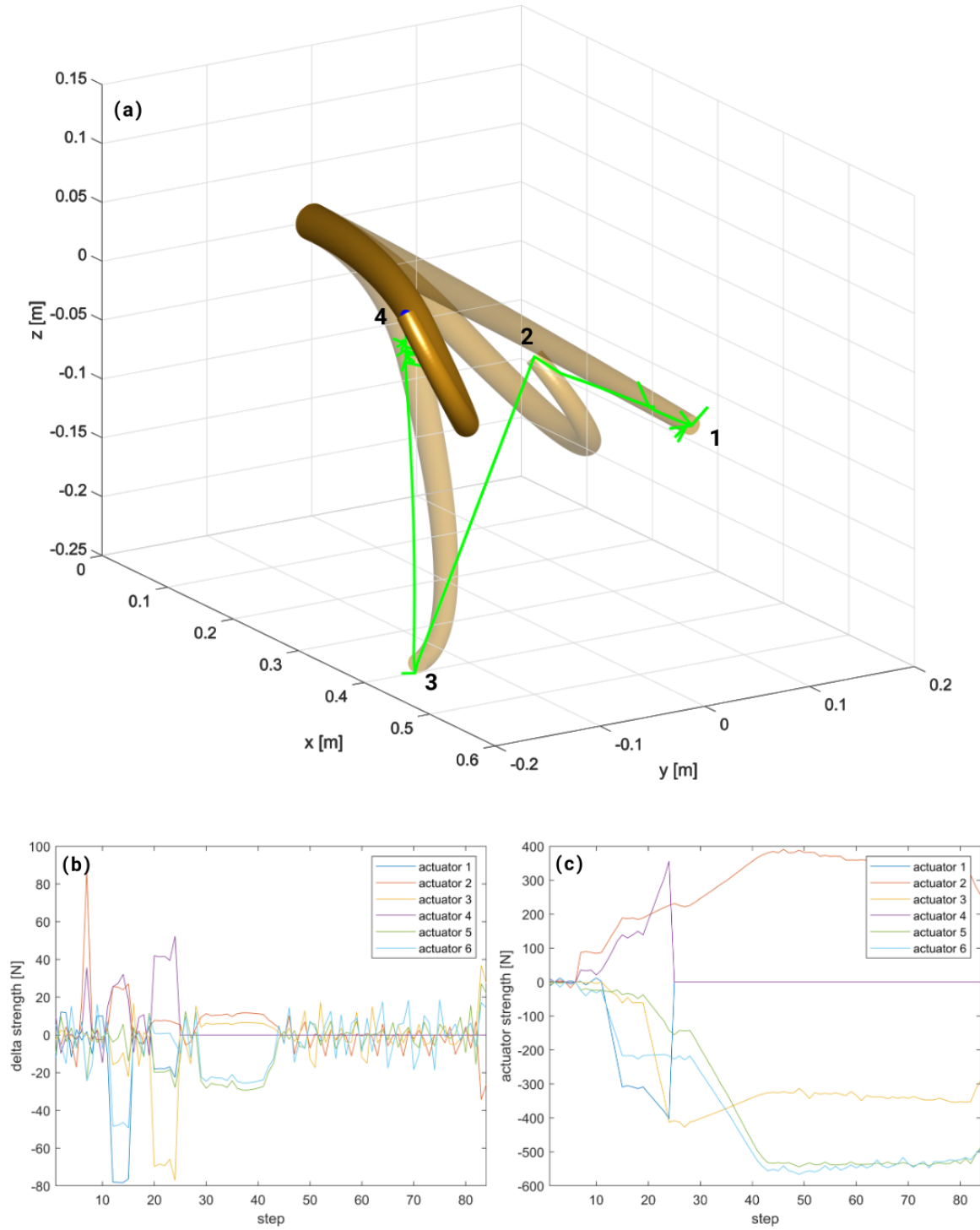


Figure 2.11: (a) Manipulator tip moving to the target position denoted as a blue point in 3D space. The controller performs both tip-position control and gravity compensation in this test too. The starting and ending positions are marked by 1 and 4 respectively. Point 2 shows the manipulator configuration prior to the actuator's failure, point 3 shows the step right after the actuator's failure. (b) Change in actuation strengths, $\Delta\tau(t)$, added to overall actuation signal for control; (c) Actuation strengths, $\tau(t)$, used to control each actuator.

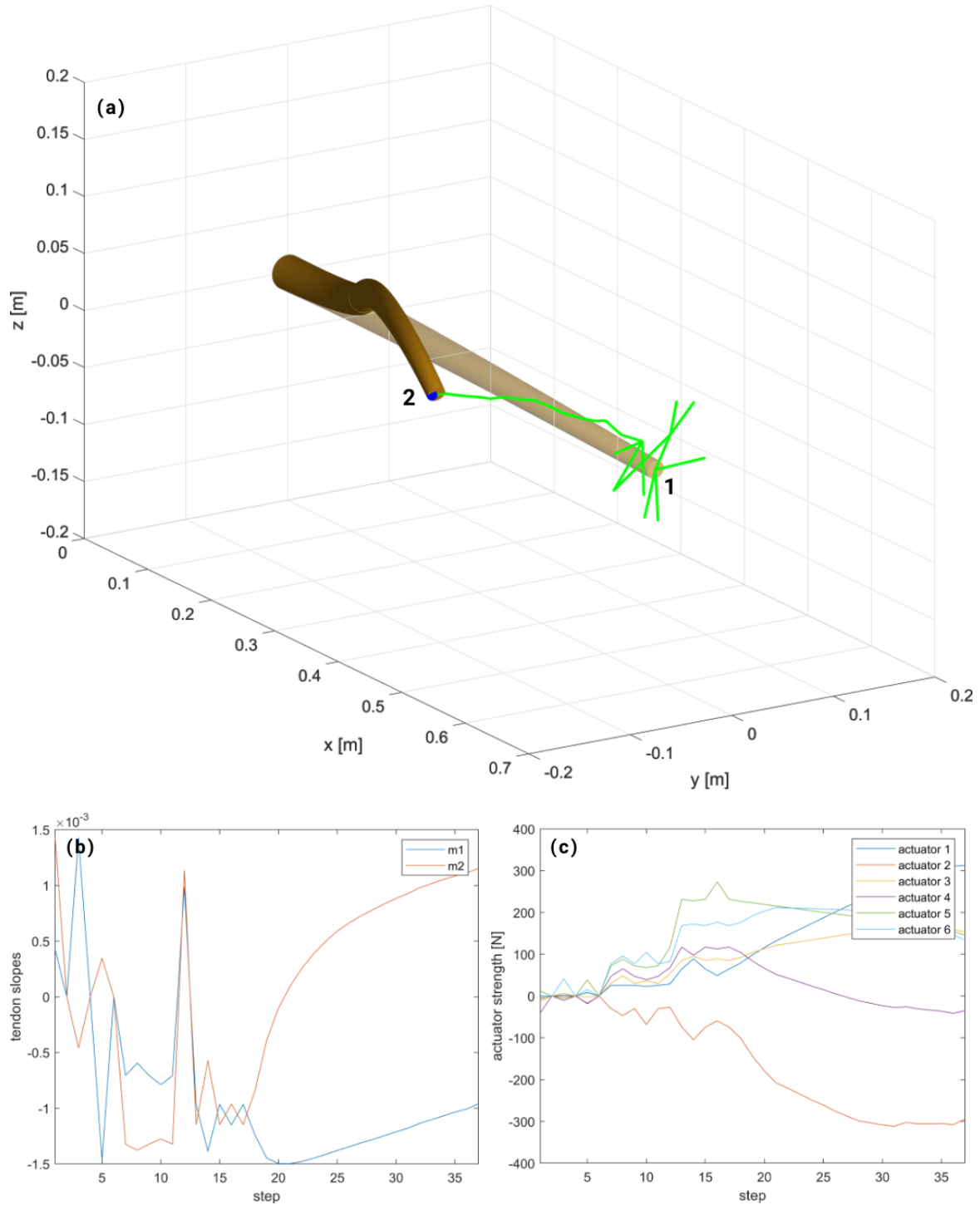


Figure 2.12: (a) Robot moving to a target position. The starting and ending positions are marked by 1 and 2 respectively. Point 2 shows that the robot reached the target position by simultaneous actuator strength and actuator slope control. (b) actuator slopes; (c) Actuation strengths, $\tau(t)$, used to control each actuator.

robot tip to the target position in a total of 84 steps.

The δ actuation strength and actuation strength plots show the input of actuators 1 and 4 becoming zero as the controller reaches step 25 to indicate that they do not contribute to the actual actuation after step 25.

Reconfiguring Robot

To explore the control of a re-configurable robot we use the same robot from tests 1 and 2 but with a constant radius of 0.015 m. In the third test, the robot performs another single-target reaching task with gravity compensation. The actuation primitives in this case include 8 parameters, instead of 6. Six parameters are for actuator strengths and two parameters are for actuator slopes, m_1 and m_2 . actuators 1, 2, and 3 use slope m_1 and actuators 4, 5, and 6 use slope m_2 . The controller actively controls both the actuator strengths and actuator slopes in order to reach the desired target position. The actuation matrix has the following form:

$$A_1 = \begin{bmatrix} \tau_p^1(p_1) & \tau_p^1(p_2) & \tau_p^1(p_3) & \tau_p^1(p_4) \\ \vdots & \vdots & \vdots & \vdots \\ \tau_p^6(p_1) & \tau_p^6(p_2) & \tau_p^6(p_3) & \tau_p^6(p_4) \\ \hline m_p^1(p_1) & m_p^1(p_2) & m_p^1(p_3) & m_p^1(p_4) \\ m_p^2(p_1) & m_p^2(p_2) & m_p^2(p_3) & m_p^2(p_4) \end{bmatrix} \quad (2.39)$$

for $kNN = 4$. The desired actuation is then calculated:

$$b_1 = [A_1^T K]^T w \quad (2.40)$$

where K is a diagonal weighting matrix to weigh the two types of parameters in the actuation matrix differently, which was experimentally determined to be $diag(1, 1, 1, 1, 1, 1, 10^{-4}, 10^{-4})$.

For this test we limit the actuator slope values to the range $-0.015 \leq m \leq 0.015$ which

ensures that the actuators do not leave the physical boundaries of the manipulator. The target point set was $[0.526, -0.187, 0.065]$. The controller varies the slope values along with the actuator strengths to find the best combination that brings the tip within the specified range of the target point. The plot in Figure 2.12(b) shows the slope values for each group of actuators. The values obtained for m_1 and m_2 when the robot's tip converged to the target position were -0.0015 and 0.0003 , respectively. The plot shown in Figure 2.12(c) shows the various actuator strength values obtained at each trial in the simulation. The robot tip reaches the target position after 27 trials.

The ability to re-route the actuators according to online data allows the controller to optimize their location and reach the target point quicker than if the actuator routes were fixed. A simulation run for the starting actuator routes showed that 49 trials are needed to reach the target point. Re-routing the actuators also results in different task spaces, which could enable the manipulator to reach points that were not previously within its range.

Growing Robot

A length-changing robot that grows and shrinks is controlled to perform target reaching. The length, L , of the robot is altered by l such that $L + l$. The controller is not limited to a specific geometry but for this case we use a constant r to avoid the tip radius going to zero or becoming negative as it grows at the tip. The actuation matrix is defined as:

$$A_1 = \begin{bmatrix} \tau_p^1(p_1) & \tau_p^1(p_2) & \tau_p^1(p_3) & \tau_p^1(p_4) \\ \vdots & \vdots & \vdots & \vdots \\ \tau_p^6(p_1) & \tau_p^6(p_2) & \tau_p^6(p_3) & \tau_p^6(p_4) \\ \hline l_p(p_1) & l_p(p_2) & l_p(p_3) & l_p(p_4) \end{bmatrix} \quad (2.41)$$

where the desired actuation is calculated following Equation (11) with K experimentally determined to be $\text{diag}(1, 1, 1, 1, 1, 1, 10^{-3})$.

The robot starts with a tip position of $[0.58, 0, -0.017]$ and a length of 0.58 m. In the first try, the robot is to reach the target tip position $[0.4, -0.1, 0]$, which requires the robot to shrink in length. The results are shown in Figure 2.13. The robot manages to reach the target point in 15 steps and reaches a length of 0.413 m.

In the second try, the robot had to reach the target tip position $[x, y, z] = [0.7, 0.1, 0]$, which required the robot to grow in length. The results are shown in Figure 2.14. The robot successfully reaches the target point in 23 steps with a length of 0.712 m.

2.7.3 Physical Rigid Robot Control

Physical experimentation of the kinematic-model-free position controller was done using Franka Emika's Panda manipulator [?]. The robot's joints were constrained, leaving only 2 joints moving freely. The robot's configuration was constrained to move in a plane to eliminate the effects of gravity. The Vive tracker is an active tracker used for motion capture of rigid bodies. A Vive tracker was placed on the robot's end effector to measure the robot's end-effector displacement. Another Vive tracker was used to mark the target position the controller needs to reach.

Two target positions were tested. See Figure 2.15. The robot starts with an exploration phase. The robot then calculates actuation primitives that will bring the robot's end-effector to the target position. Whenever a discrepancy between the desired and actual displacements is measured, another exploratory phase is triggered. In both tests, the controller managed to actuate the robot's end-effector towards the desired target position.

2.7.4 Physical Continuum Robot Control

The OctArm, shown in Figure 2.16, is a pneumatically-actuated continuum robot with 3 sections, each section having 3 muscles, providing a 9-DoF manipulator. The OctArm has a length extension range from 1.07 m to 1.38 m. The muscles can also be pressurized together, reducing

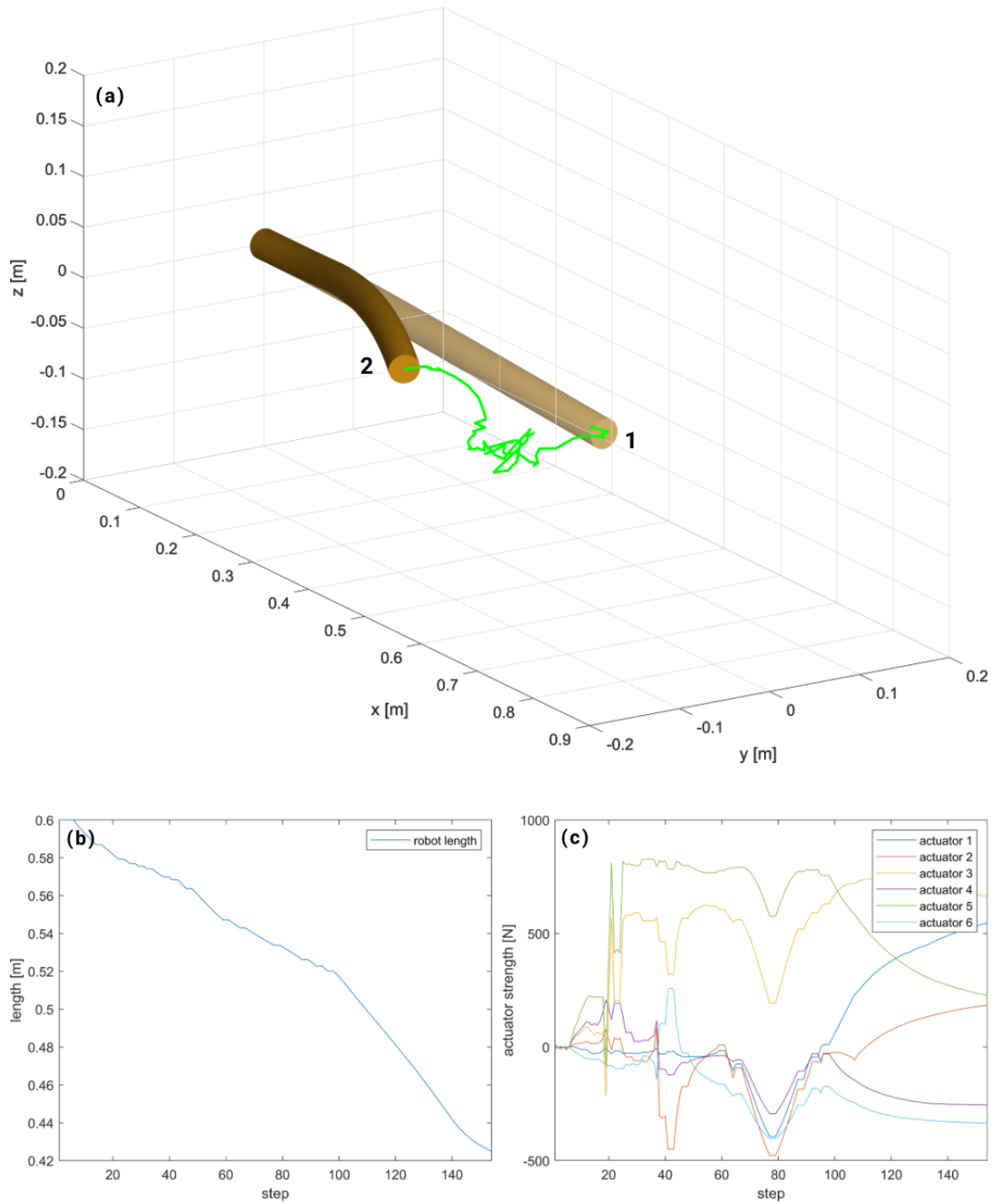


Figure 2.13: (a) Manipulator tip moving to a target position. The starting and ending positions are marked by 1 and 2 respectively. Point 2 shows that the manipulator reaches the target position by simultaneous actuator strength control and shrinking. (b) Change in robot length, L , for shrinking; (c) Actuation strengths, $\tau(t)$, used to control each actuator.

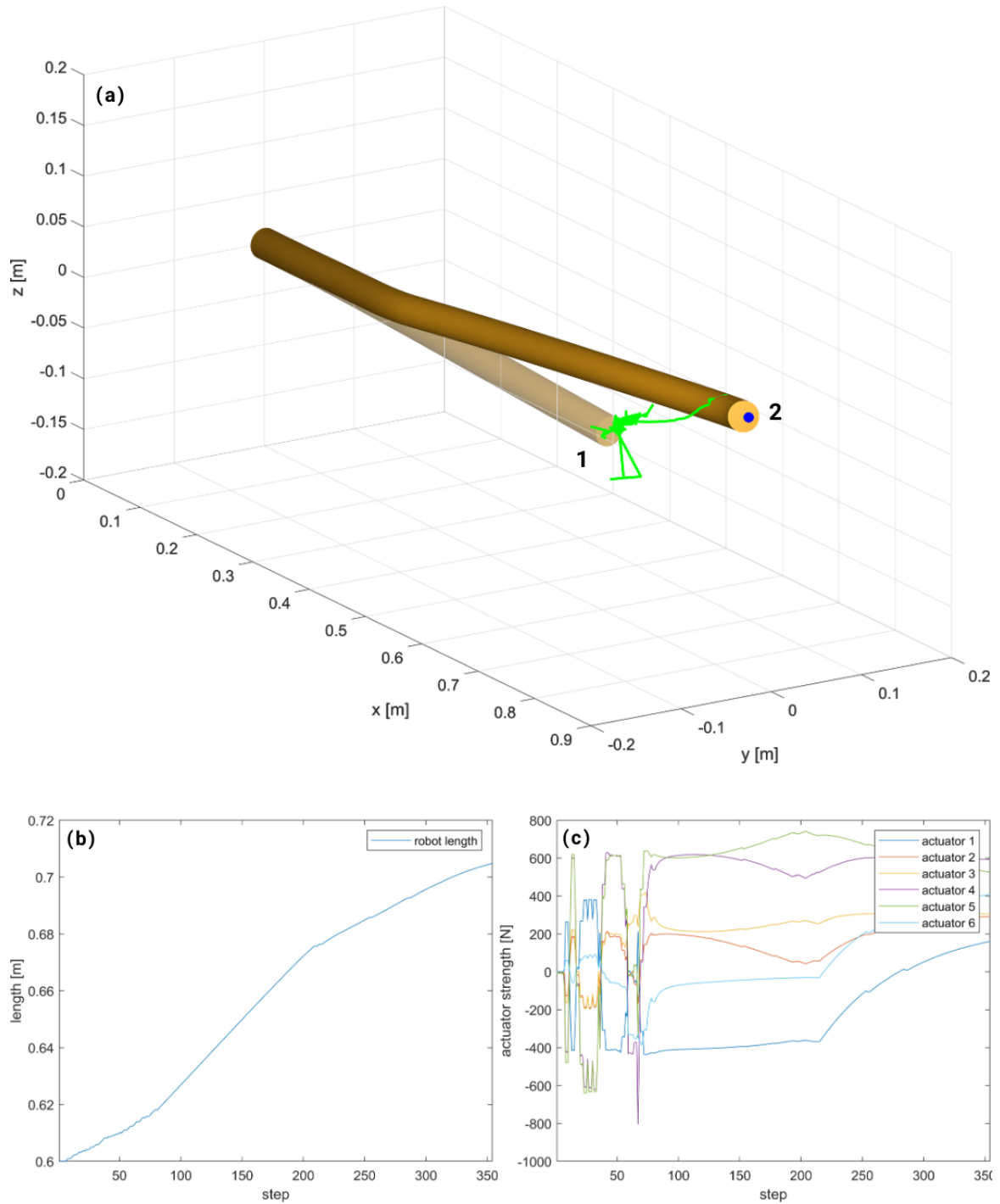


Figure 2.14: (a) Manipulator tip moving to a target position. The starting and ending positions are marked by 1 and 2 respectively. Point 2 shows that the manipulator reaches the target position through simultaneous actuator strength control and growth. (b) Change in robot length, L , for growing; (c) Actuation strengths, $\tau(t)$, used to control each actuator.

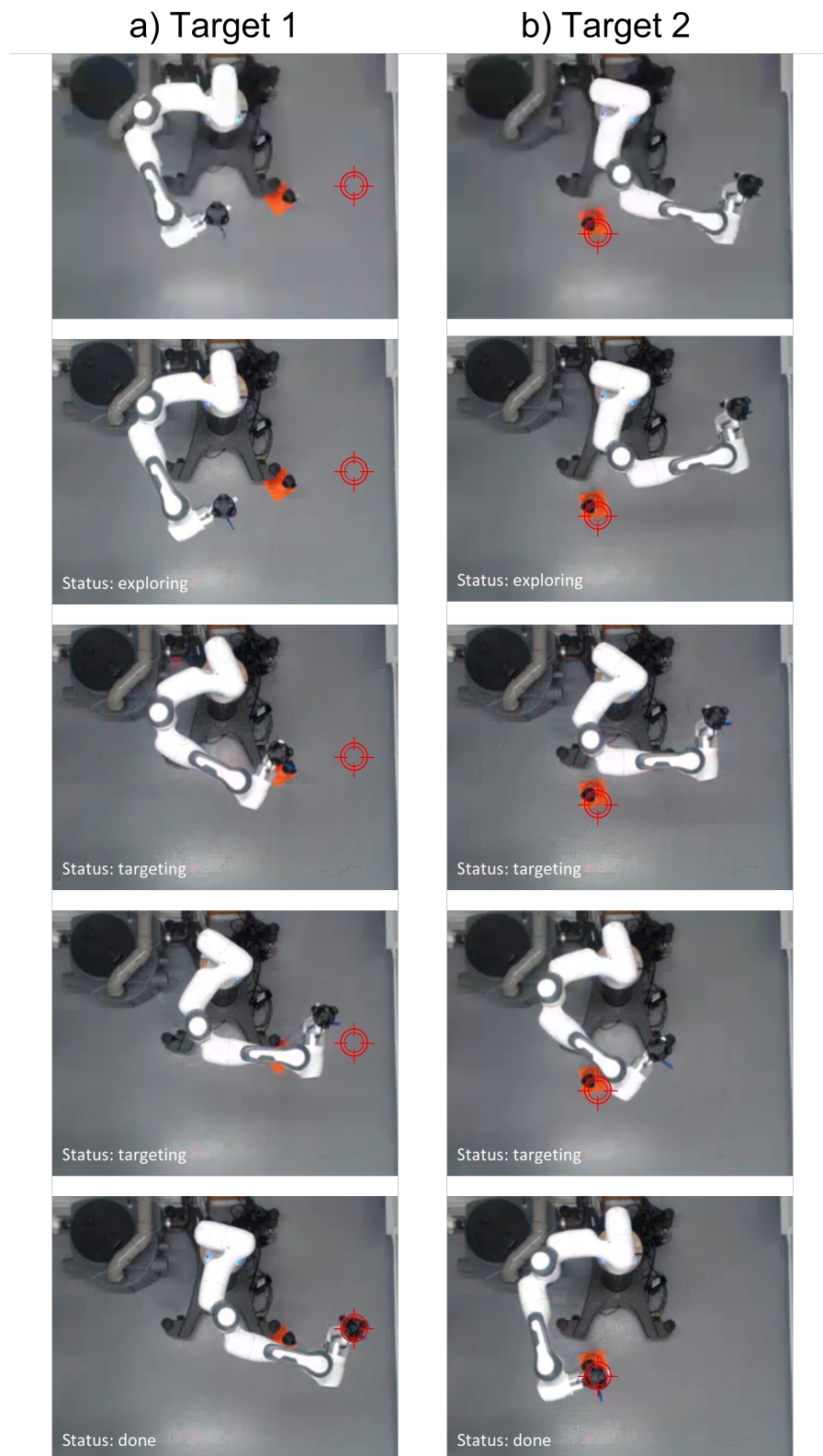


Figure 2.15: Target reaching using the Panda manipulator constrained to planar motion with only 2 DoFs enabled.

the degrees of freedom of the robot.

Two experiments are conducted for different degrees of freedom of the OctArm. In the first experiment, the robot follows a straight line path that runs parallel to the x-axis in the taskspace. The path is comprised of 9 points spaced evenly between $[-0.2, 0.95, -0.85]$ and $[0.2, 0.95, 0.85]$ in the camera's frame. See Figure 2.17. Initially, the robot starts at $x = 0.2$, follows the line, then moves back to the start of the line. In the second experiment, the robot follows a planar lemniscate path on the xz-plane. The path is comprised of 19 points that are spaced almost 5 cm apart. See Figure 2.18. For both experiments, the robot's tip position must be within 2 cm of a point and is paused for 3 seconds (for the robot to come to rest and reading clear results) before moving to the next point. Tracking of the robot's end effector is done through the use of a Kinect sensor with hue filtering. See Figure 2.19 and 2.20. The KMF controller is implemented in MATLAB. A Simulink model is responsible for integrating the actuation primitives into pressure values that actuates the OctArm.

3-DoF Control

In the 3-DoF case, the OctArm is treated as a single section; i.e. muscle i of each section is pressurized simultaneously for $i = 1, 2, 3$. The actuation matrix is then:

$$b_1 = \begin{bmatrix} \tau_u(\hat{p}) \\ \tau_v(\hat{p}) \\ \tau_s(\hat{p}) \end{bmatrix} \quad (2.42)$$

The 3-DoF continuum robot completed the line path within 3 mins with minimal error or stepping outside the 2 cm limit. The time is significant given the 3 second pause in place between actuations. See Figure 2.21. The robot also completed the lemniscate path, however with more discrepancies and explorations. The robot manages to converge back to the desired points to complete the path. See Figure 2.22. This can be improved by having more dense points in the path to follow or by having more exploration in certain regions.

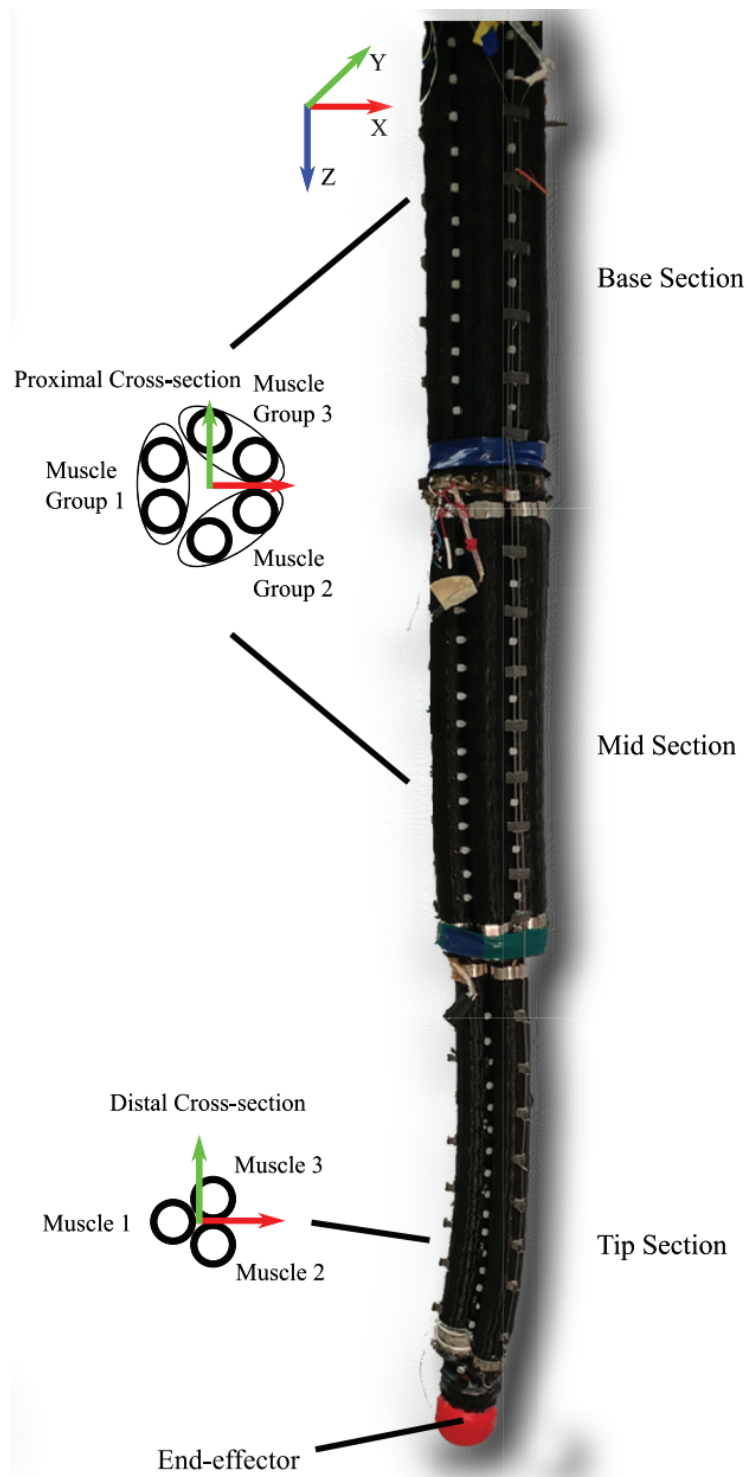


Figure 2.16: OctArm pneumatically-actuated manipulator.

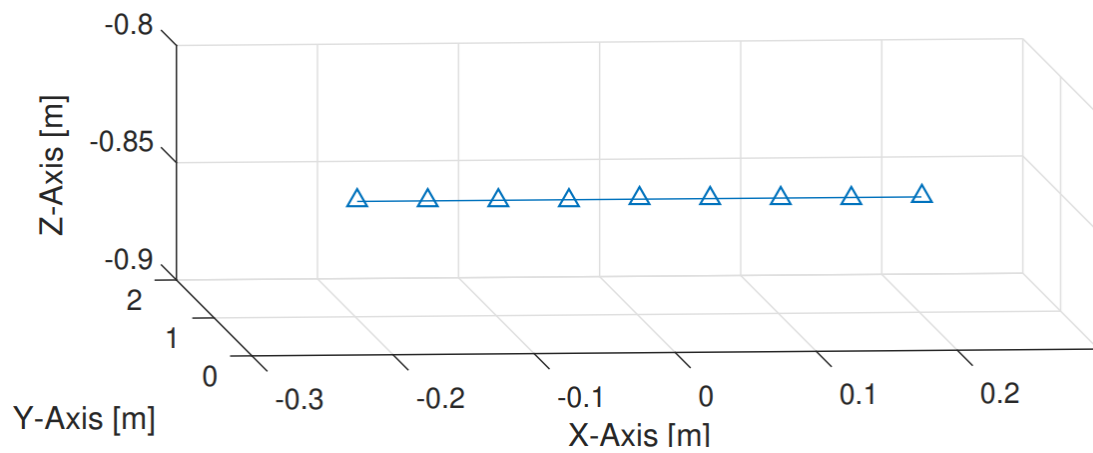


Figure 2.17: Line path trajectory.

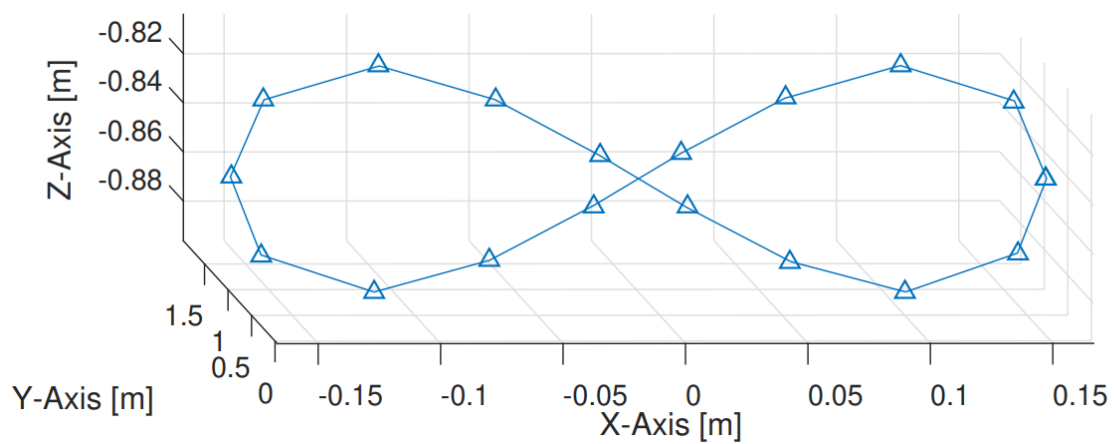


Figure 2.18: Lemniscate path trajectory.

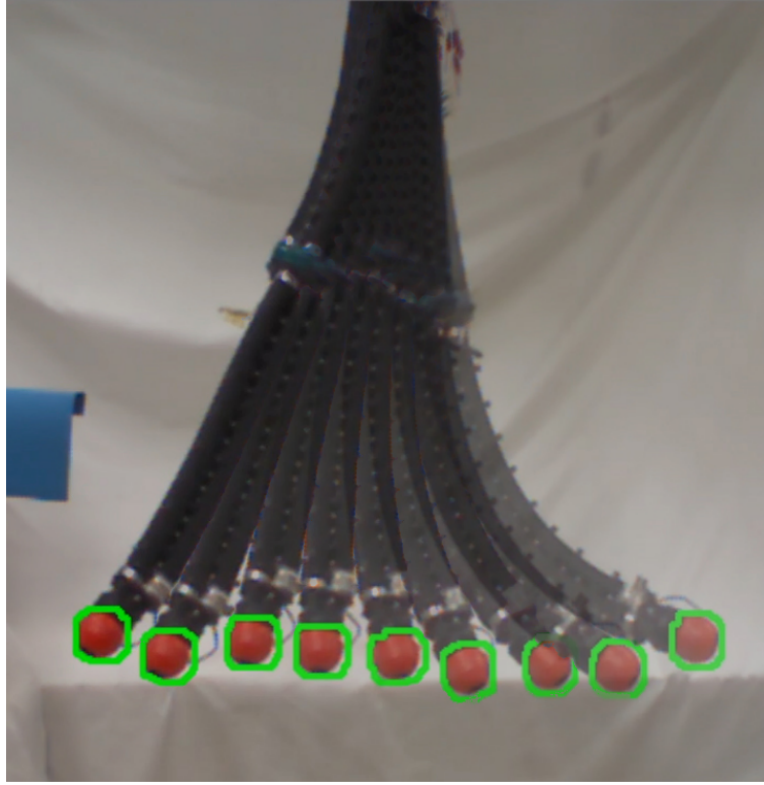


Figure 2.19: Line path followed by robot.

4-DoF Control

In the 4-DoF case, the base and mid sections of the OctArm are treated as a single section. The robot here is treated as a non-extensible robot with each section initialized to its mid-range extension and differentially controlled using the parameters u and v only. The actuation matrix is then:

$$b_1 = \begin{bmatrix} \tau_{u_b}(\hat{p}) \\ \tau_{v_b}(\hat{p}) \\ \tau_{u_t}(\hat{p}) \\ \tau_{v_t}(\hat{p}) \end{bmatrix} \quad (2.43)$$

The addition of another degree of freedom did not hinder the performance of the controller as can be seen in Figures 2.23 and 2.24. However, more exploratory behaviour was triggered to gather sufficient many samples to complete the paths. The completion times of both the 3-DoF and 4-DoF cases are similar.

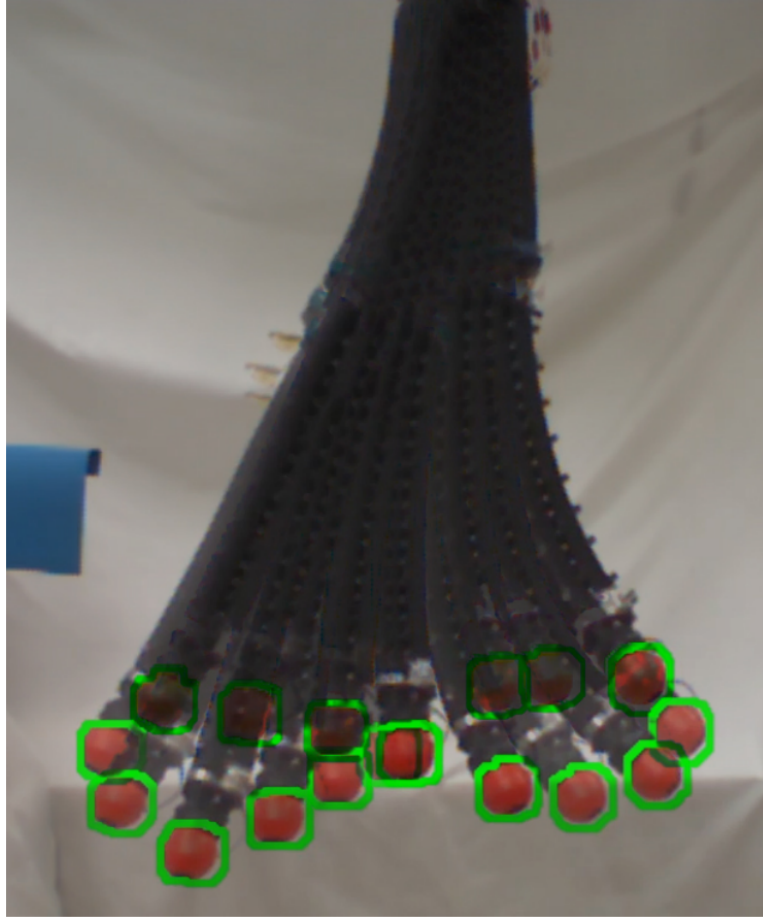


Figure 2.20: Lemniscate path followed by robot.

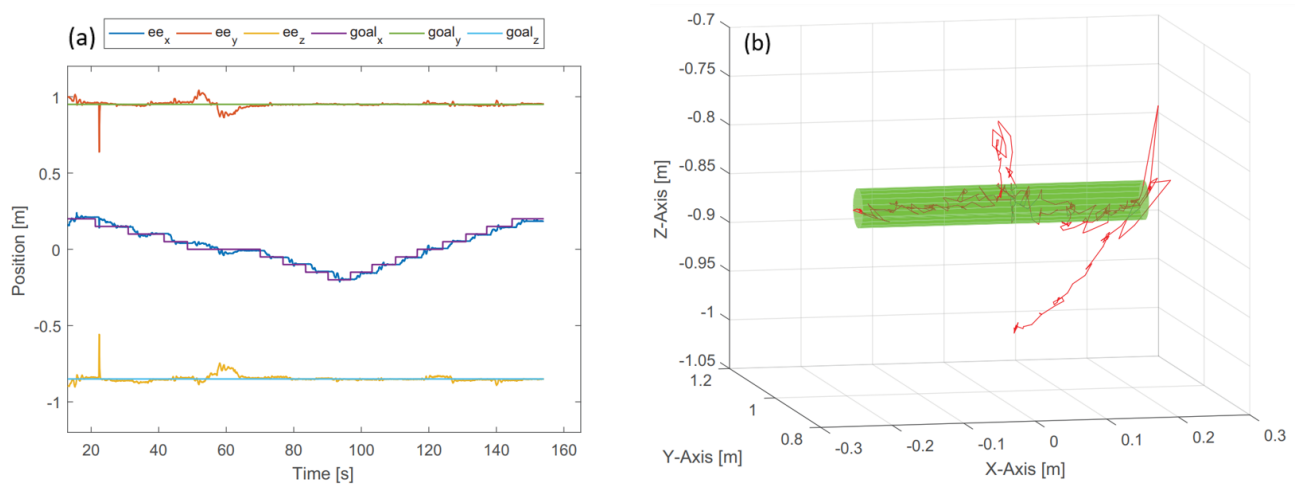


Figure 2.21: (a) End-effector and target positions of 3-DoF robot while following line path. (b) Measured end-effector position of 3-DoF robot while following line path.

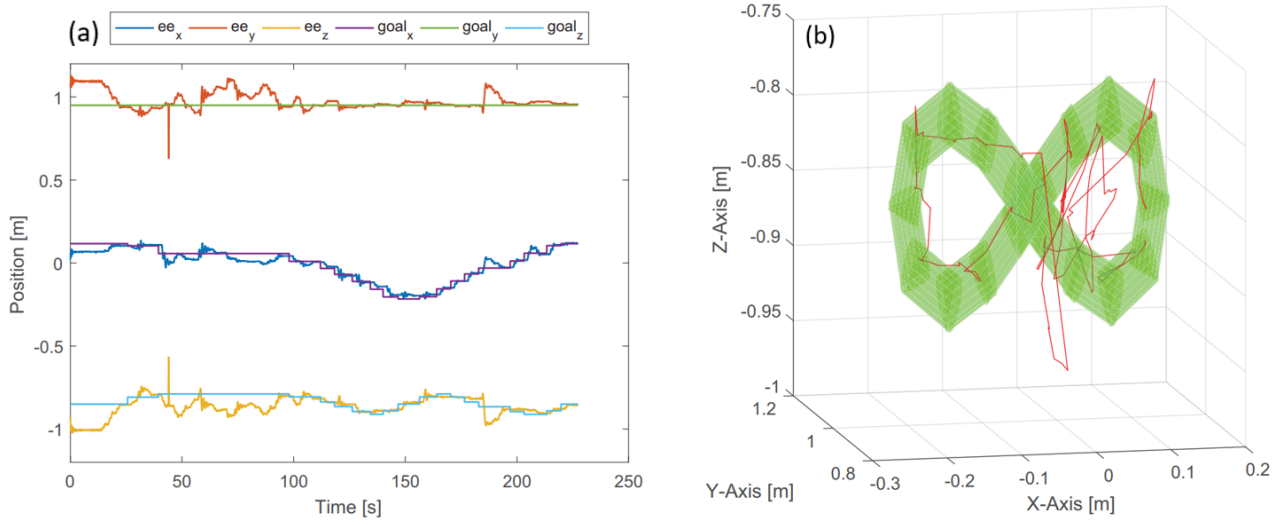


Figure 2.22: (a) End-effector and target positions of 3-DoF robot while following lemniscate path. (b) Measured end-effector position of 3-DoF robot while following lemniscate path.

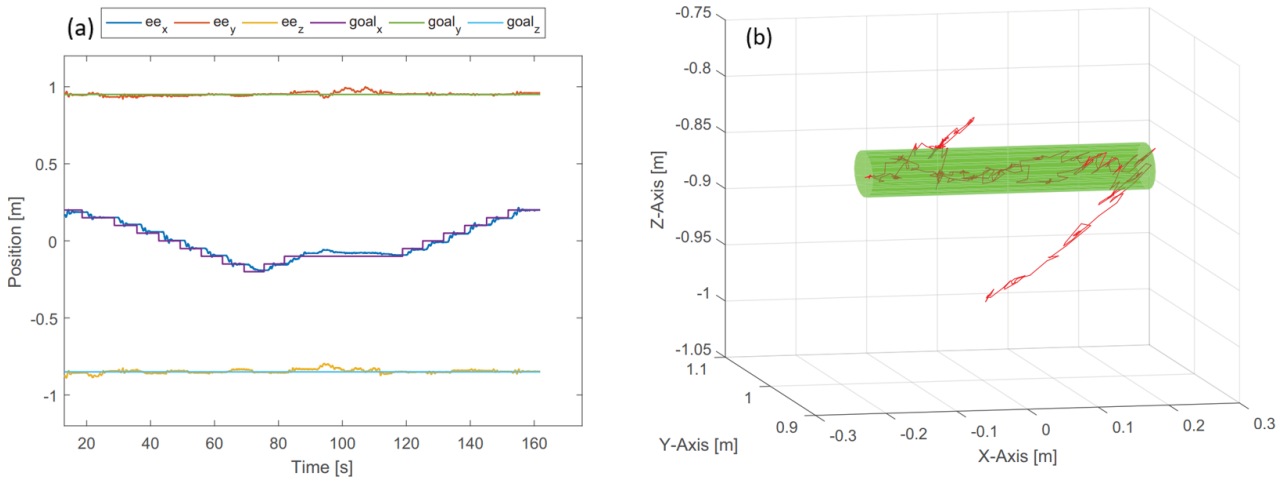


Figure 2.23: (a) End-effector and target positions of 4-DoF robot while following line path. (b) Measured end-effector position of 4-DoF robot while following line path.

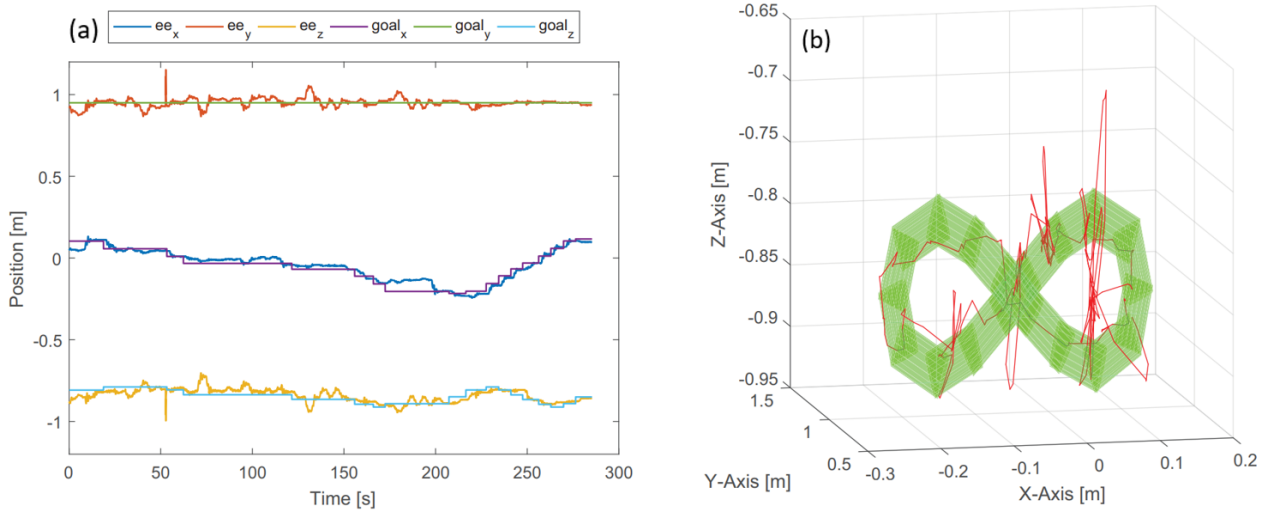


Figure 2.24: (a) End-effector and target positions of 4-DoF robot while following lemniscate path. (b) Measured end-effector position of 4-DoF robot while following lemniscate path.

6-DoF Control

In the 6-DoF case, each section is controlled separately with its own parameters, u and v . Similar to the 4-DoF case, the robot is treated as non-extensible and is controlled differentially. The actuation matrix is then:

$$b_1 = \begin{bmatrix} \tau_{u_b}(\hat{p}) \\ \tau_{v_b}(\hat{p}) \\ \tau_{u_m}(\hat{p}) \\ \tau_{v_m}(\hat{p}) \\ \tau_{u_t}(\hat{p}) \\ \tau_{v_t}(\hat{p}) \end{bmatrix} \quad (2.44)$$

The added degrees of freedom was expected to impact the performance of the controller. This can be noticed from the trajectory of the robot shown in Figure 2.25. The completion time took 3-4 times longer than the previous cases, with most of the time being wasted in explorations and inaccurate actuations. Given the difficulty of completing the line path, attempting the lemniscate path was forgone.

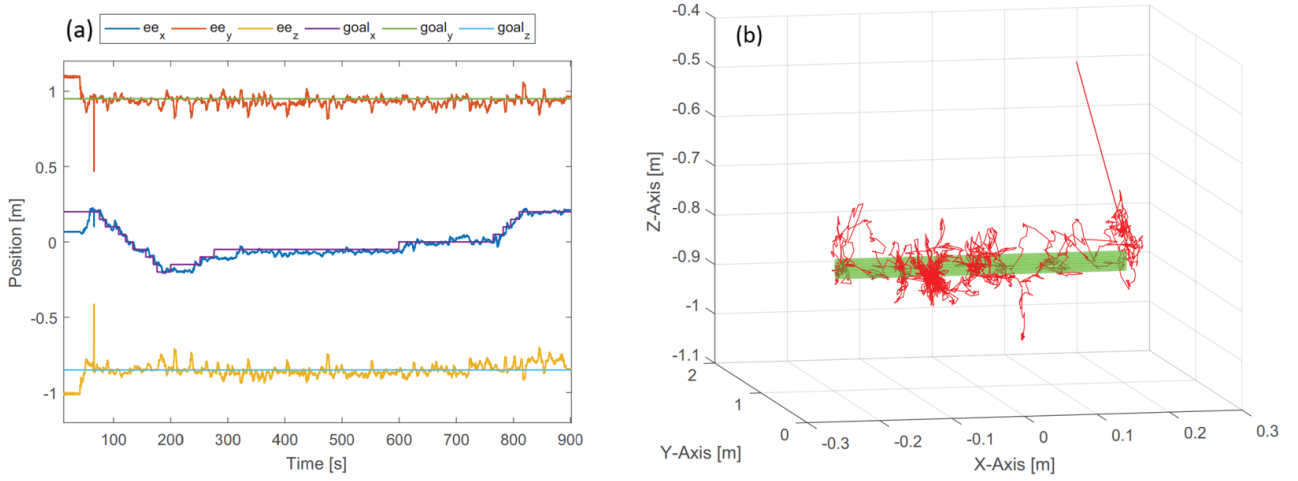


Figure 2.25: (a) End-effector and target positions of 6-DoF robot while following line path. (b) Measured end-effector position of 6-DoF robot while following line path.

2.8 Discussion

The kinematic-model-free controller is capable of controlling the robot's end-effector position and orientation without any prior knowledge of the robot's kinematic or dynamic models. The controller was tested on a simulated planar 2-DoF rigid robot that performed target reaching and a 3-DoF robot in 3D space. The controller was also tested on a simulated planar 3-DoF robot to perform simultaneous position and orientation control of the end effector. Physical experimentation on a planar robot manipulator was done on a constrained manipulator with only 2-DoFs enabled. The robot successfully performed target reaching using kinematic-model-free position control of the end effector.

On the other hand, soft continuum robots are widely used and researched due to their advantages over their rigid counterparts. However, the inherent compliance of these systems results in infinite DoFs which makes controlling them a challenge. Accurate models and controllers are required that also take into account external forces.

The kinematic-model-free controller was tested on a threadlike actuator-driven soft continuum robot in 3D space with gravity compensation. The controller was validated through simulation tests where the robot performed target reaching and followed a desired trajectory.

The controller's robustness and adaptability were tested by cutting a subset of the actuators during execution. Furthermore, kinematic-model-free control methods were used to control a robot with re-configurable actuators and a growing robot, where the robot managed to reach desired target tip positions via simultaneous control of actuator strengths and actuator slopes or robot length.

The kinematic-model-free controller was also deployed on a physical continuum robot for planar tip position control. The 3-section pneumatically-actuated robot traversed two planar paths, a line and a lemniscate path. When increasing the degrees of freedom of the robot enabled for control, the performance suffers from excessive exploratory behaviour. The robot, however, manages to perform the tasks in the 3-DoF and 4-DoF cases.

References

- [1] Ahmad AlAttar, Francesco Cursi, and Petar Kormushev. Kinematic-model-free redundancy resolution using multi-point tracking and control for robot manipulation. *Applied Sciences*, 11(11):4746, 2021.
- [2] Jose Barreiros, Kevin W O’Brien, Samantha Hong, Michael F Xiao, Ho-Jung Yang, and Robert F Shepherd. Configurable tendon routing in a 3d-printed soft actuator for improved locomotion in a multi-legged robot. In *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*, pages 94–101. IEEE, 2019.
- [3] Frédéric Boyer, Vincent Lebastard, Fabien Candelier, and Federico Renda. Dynamics of continuum and soft robots: A strain parameterization based approach. *IEEE Transactions on Robotics*, 37(3):847–863, 2020.
- [4] Peter Corke. *Robotics, vision and control: fundamental algorithms in MATLAB® second, completely revised*, volume 118. Springer, 2017.
- [5] Cosimo Della Santina, Robert K Katzschmann, Antonio Bicchi, and Daniela Rus. Model-based dynamic feedback control of a planar soft robot: trajectory tracking and interaction with the environment. *The International Journal of Robotics Research*, 39(4):490–513, 2020.
- [6] Franka Emika GmbH. Panda arm, 2023.
- [7] Thomas George Thuruthel, Yasmin Ansari, Egidio Falotico, and Cecilia Laschi. Control strategies for soft robotic manipulators: A survey. *Soft robotics*, 5(2):149–163, 2018.
- [8] Reinhard M Grassmann, Priyanka Rao, Quentin Peyron, and Jessica Burgner-Kahrs. Fas—a fully actuated segment for tendon-driven continuum robots. *Frontiers in Robotics and AI*, 9, 2022.
- [9] Elliot W Hawkes, Laura H Blumenschein, Joseph D Greer, and Allison M Okamura. A soft robot that navigates its environment through growth. *Science Robotics*, 2(8):eaan3028, 2017.

- [10] Xinjia Huang, Jiang Zou, and Guoying Gu. Kinematic modeling and control of variable curvature soft continuum robots. *IEEE/ASME Transactions on Mechatronics*, 26(6):3175–3185, 2021.
- [11] Petar Kormushev, Yiannis Demiris, and Darwin G Caldwell. Encoderless position control of a two-link robot manipulator. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 943–949. IEEE, 2015.
- [12] Petar Kormushev, Yiannis Demiris, and Darwin G Caldwell. Kinematic-free position control of a 2-dof planar robot arm. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5518–5525. IEEE, 2015.
- [13] Cecilia Laschi, Barbara Mazzolai, and Matteo Cianchetti. Soft robotics: Technologies and systems pushing the boundaries of robot abilities. *Science robotics*, 1(1):eaah3690, 2016.
- [14] Anup Teejo Mathew, Ikhlas Mohamed Ben Hmida, Costanza Armanini, Frederic Boyer, and Federico Renda. Sorosim: A matlab toolbox for hybrid rigid-soft robots based on the geometric variable-strain approach. *IEEE Robotics and Automation Magazine*, pages 2–18, 2022.
- [15] Ryota Morimoto, Satoshi Nishikawa, Ryuma Niiyama, and Yasuo Kuniyoshi. Model-free reinforcement learning with ensemble for a soft continuum robot arm. In *2021 IEEE 4th International Conference on Soft Robotics (RoboSoft)*, pages 141–148. IEEE, 2021.
- [16] F. Renda, C. Armanini, V. Lebastard, F. Candelier, and F. Boyer. A geometric variable-strain approach for static modeling of soft manipulators with tendon and fluidic actuation. *IEEE Robotics and Automation Letters*, 5(3):4006–4013, 2020.
- [17] Federico Renda, Costanza Armanini, Anup Mathew, and Frederic Boyer. Geometrically-exact inverse kinematic control of soft manipulators with general threadlike actuators’ routing. *IEEE Robotics and Automation Letters*, 7(3):7311–7318, 2022.

- [18] Ali Sadeghi, Alessio Mondini, and Barbara Mazzolai. Toward self-growing soft robots inspired by plant roots and based on additive manufacturing technologies. *Soft robotics*, 4(3):211–223, 2017.
- [19] Ali Sadeghi, Alice Tonazzini, Liyana Popova, and Barbara Mazzolai. A novel growing device inspired by plant root soil penetration behaviors. *PloS one*, 9(2):e90139, 2014.
- [20] Amir Sanati Nezhad and Anja Geitmann. The cellular mechanics of an invasive lifestyle. *Journal of experimental botany*, 64(15):4709–4728, 2013.
- [21] Cem Tutcu, Bora A Baydere, Seref K Talas, and Evren Samur. Quasi-static modeling of a novel growing soft-continuum robot. *The International Journal of Robotics Research*, 40(1):86–98, 2021.
- [22] Kevin C Vaughn, Andrew J Bowling, et al. 1 biology and physiology of vines. *Horticultural reviews*, 38:1, 2011.
- [23] Robert J Webster III and Bryan A Jones. Design and kinematic modeling of constant curvature continuum robots: A review. *The International Journal of Robotics Research*, 29(13):1661–1683, 2010.
- [24] Qingxiang Zhao, Jiewen Lai, Kaicheng Huang, Xiaobing Hu, and Henry K Chu. Shape estimation and control of a soft continuum robot under external payloads. *IEEE/ASME Transactions on Mechatronics*, 2021.

Chapter 3

Quaternionic Kinematic-Model-Free Control

The encoderless robot controller [?] and kinematic-free controller [?] successfully controlled a robot manipulator’s end effector in task space without requiring any prior kinematic model of the robot by learning local linear models. However, these controllers only managed to control the position of a two-degrees-of-freedom planar robot, leaving orientation control unsolved.

Orientation control imposes challenges due to the discontinuities or singularities that exist in the orientation spectrum, such as the gimbal lock in Euler angles or angle wrapping that occurs after a full rotation. Without resolving these issues, the robot will misinterpret movements when crossing the point of discontinuity or can take a longer route to reach the desired target, causing the robot to behave undesirably and inefficiently. Quaternions, discovered by Hamilton around 1844 [?], are an extension to the complex number system and are able to represent rotations without suffering from discontinuities. Furthermore, dual quaternions (DQs), introduced by Clifford around 1873 [?], provide a unified representation of a rigid link’s pose, both position and orientation. In the dual-quaternionic KMF controller, linear combinations of dual quaternions are used to learn a local kinematics and dynamics model of the robot which allows for simultaneous tip position and orientation control in 3D space.

However, this controller underperforms, taking longer than required to converge to the desired pose, when higher-degrees-of-freedom serial robots are to be controlled, leaving redundancies unresolved, as well. The kinematic-model-free multi-point controller presented in this chapter enables the control of hyper-redundant serial robot manipulators and resolving redundancies in these manipulators by tracking and controlling multiple points along the kinematic chain of such robots.

3.1 Related Work

Model-based controllers have been the predominant method of control for robot manipulation. The Jacobian of a robot, which relates the joint velocities to the end-effector velocity, is calculated prior to control [?], after which it is controlled using forward and inverse kinematics [?]. Using such models imposes many limitations [?] and assumes the Denavit-Hartenberg parameters, that define the robots kinematics, to be static. One example of model-based control is visual servoing that uses an external camera to calculate the required end-effector velocity to reach a desired target [?]. However, this assumption falls apart in the case of soft robots, when robots are damaged, or when robots transform or evolve in any way, such as altering the end effector by changing its gripper. Therefore, model-based conventional controllers are accurate but not adaptable.

Other researchers performed model-free control by approximating a Jacobian that is later used to actuate a robot. Researchers in Reference [?] controlled a deformable manipulator and in Reference [?] controlled a soft manipulator by approximating its Jacobian model. Other researchers controlled a dynamically-undefined robot by first estimating the robot's dynamic parameters and then controlled a robot using conventional methods [?]. Learning methods, such as motor babbling, that start with no predefined model of a robot's kinodynamics are used to approximate a model to control robots [?]. Other researchers use machine learning [?] and reinforcement learning algorithms [?] to control robots.

On the other hand, researchers in [?] directly controlled current motors to actuate an unmanned underwater vehicle by means of deterministic artificial intelligence where uncertainties are ignored by making use of first principles. Researchers in Reference [?] controlled a SCARA robot's position through motor voltages using extended state observer to estimate system dynamics. Other researchers studied hybrid model-free and model-based approaches to increase the performance of robot navigation [?]. However, most of these learning approaches require encoder feedback for estimating the robot's pose.

The model-learning approach presented in our preliminary work in [?, ?] are KMF controllers because they start with no prior knowledge about the robot's configuration and learn a model without requiring kinematic information about the robot's joint position during control. The earlier KMF controllers only addressed position control of a 2-DOF planar robot, leaving orientation control and control of serial manipulators of higher degrees-of-freedom unsolved.

3.2 Mathematical Background

Quaternions

Quaternions are an extension to the complex number system used to represent rotations, first described by Hamilton around 1844 [?]. A quaternion $\mathbf{q}$ is the sum of a scalar s and a vector $\mathbf{v} = (q_1, q_2, q_3)$:

$$\mathbf{q} = s + \mathbf{v} = s + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k} \quad (3.1)$$

where $\mathbf{i}, \mathbf{j}, \mathbf{k}$ are unit vectors pointing along the x-,y-,z-axes. The addition and subtraction of quaternions are element-wise:

$$\mathbf{p} \pm \mathbf{q} = (s_p \pm s_q) + (v_p \pm v_q) = (p_0 \pm q_0) + (p_1 \pm q_1)\mathbf{i} + (p_2 \pm q_2)\mathbf{j} + (p_3 \pm q_3)\mathbf{k} \quad (3.2)$$

The conjugate of a quaternion $\mathbf{q}$ is defined as:

$$\mathbf{q}^* = s - \mathbf{v} = s - q_1\mathbf{i} - q_2\mathbf{j} - q_3\mathbf{k} \quad (3.3)$$

from which the norm of a quaternion can be calculated as $\|\mathbf{q}\| = \sqrt{\mathbf{q}^*\mathbf{q}}$. A quaternion $\mathbf{q}$ is unit when $s^2 + q_1^2 + q_2^2 + q_3^2 = 1$ or simply $\|\mathbf{q}\| = 1$. The vec_4 operator maps the elements of a quaternion $\mathbf{q}$ to a 4-dimensional vector (s, q_1, q_2, q_3) . A detailed examination of quaternions can be found in [?].

Dual Quaternions

Dual quaternions, introduced by Clifford [?], are an extension to the dual number system used to represent rigid transformations by unifying rotational and translational information into a single mathematical object. A dual quaternion $\hat{\mathbf{q}}$ is the sum of two quaternions:

$$\hat{\mathbf{q}} = \mathbf{p} + \epsilon\mathbf{q} \quad (3.4)$$

where $\mathbf{p} = s_p + \mathbf{v}_p$ is the real part and $\mathbf{q} = s_q + \mathbf{v}_q$ is the dual part. The dual element ϵ is nilpotent, $\epsilon^2 = 0$. The addition and subtraction of two dual quaternions is performed separately on the real and dual parts:

$$\hat{\mathbf{q}}_1 \pm \hat{\mathbf{q}}_2 = (\mathbf{p}_1 \pm \mathbf{p}_2) + \epsilon(\mathbf{q}_1 \pm \mathbf{q}_2) \quad (3.5)$$

The conjugate of a dual quaternion is $\hat{\mathbf{q}}^* = \mathbf{p}^* + \epsilon\mathbf{q}^*$. The norm of a dual quaternion is defined as $\|\hat{\mathbf{q}}\| = \sqrt{\hat{\mathbf{q}}^*\hat{\mathbf{q}}}$. A dual quaternion $\hat{\mathbf{q}}$ is unit when the real part is unit and is perpendicular to the dual part, $s_p^2 + p_1^2 + p_2^2 + p_3^2 = 1$ and $s_p s_q + p_1 q_1 + p_2 q_2 + p_3 q_3 = 0$ or simply when $\|\hat{\mathbf{q}}\| = 1$ and $\langle \mathbf{p}, \mathbf{q} \rangle = 0$. The vec_8 operator maps the elements of a dual quaternion $\hat{\mathbf{q}}$ to an 8-dimensional vector $(s_p, p_1, p_2, p_3, s_q, q_1, q_2, q_3)$.

Rigid Transformation

A rigid transformation can be represented using a dual quaternion:

$$\hat{\mathbf{q}} = \mathbf{r} + \frac{\epsilon}{2}\mathbf{t}\mathbf{r} \quad (3.6)$$

where $\mathbf{r}$ is the rotation quaternion and $\mathbf{t}$ is the translation quaternion [?]. From a unit dual quaternion $\mathbf{p} + \epsilon\mathbf{q}$, the rotation and translation quaternions are derived as: $\mathbf{r} = \mathbf{p}$ and $\mathbf{t} = 2\mathbf{q}\mathbf{p}^*$, where $\mathbf{r}$ is the rotation quaternion and $\mathbf{t}$ is the translation quaternion. For the translation quaternion, the scalar part is always zero, $t = 0 + \mathbf{v}_t = \mathbf{v}_t$. The dual quaternion transformation represents a pure rotation when the dual part is zero, $\mathbf{q} = 0$.

Interpolation

Interpolation between two 3D rotations represented by two quaternions is achieved using Spherical Linear Interpolation (SLERP) which was introduced by Shoemake in 1985 [?]. This method of interpolation is popular because it has the following desirable properties: constant speed, shortest path, and coordinate system invariant [?]. Let $\Phi(\mathbf{p}, \mathbf{q}, t)$ denote an interpolation between $\mathbf{p}$ and $\mathbf{q}$, where $t \in [0, 1]$. The interpolated quaternion is then

$$\mathbf{m} = \Phi(\mathbf{p}, \mathbf{q}, t) = \mathbf{p}(\mathbf{p}^*\mathbf{q})^t \quad (3.7)$$

Screw Linear Interpolation (ScLERP) is a generalization of SLERP that extends it to interpolate between dual quaternions and is defined similarly,

$$\hat{\mathbf{m}} = \Phi(\hat{\mathbf{p}}, \hat{\mathbf{q}}, t) = \hat{\mathbf{p}}(\hat{\mathbf{p}}^*\hat{\mathbf{q}})^t \quad (3.8)$$

where $t \in [0, 1]$.

Relativity

The relative rotational displacement, $\mathbf{b}_{rel}$, between two quaternions, $\mathbf{p}$ and $\mathbf{q}$, can be easily

found as:

$$Qrel(\mathbf{p}, \mathbf{q}) = \mathbf{b}_{rel} = \mathbf{p}^* \mathbf{q} \quad (3.9)$$

Finding the relative dual quaternion, $\hat{\mathbf{b}}_{rel}$, between two dual quaternions, $\hat{\mathbf{p}}$ and $\hat{\mathbf{q}}$, is not as straightforward. The dual quaternions are split into their rotational and translational quaternions and then recombined after calculating the relative rotation and translation as follows:

$$DQrel(\hat{\mathbf{p}}, \hat{\mathbf{q}}) = \hat{\mathbf{b}}_{rel} = \mathbf{r}_1^* \mathbf{r}_2 + \frac{\epsilon}{2}(\mathbf{t}_1^* + \mathbf{t}_2) \mathbf{r}_1^* \mathbf{r}_2 \quad (3.10)$$

Note that $\hat{\mathbf{b}}_{rel}$ must be a unit dual quaternion and normalized if not.

Distance

Any quaternionic KMF approach requires a distance metrics between two quaternions or dual quaternions which will provide a measure of how close the end effector is to the target pose that reduces to zero as the end effector approaches the target. The *vec4* and *Qrel* functions are used to find the distance between two quaternions as follows:

$$Qdist(\mathbf{q}_1, \mathbf{q}_2) = \| \text{vec4}(\mathbf{0}) - \text{vec4}(Qrel(\mathbf{q}_1, \mathbf{q}_2)) \| \quad (3.11)$$

where $\mathbf{0}$ is the null quaternion. This measure of quaternionic distance satisfies the following three properties:

$$Qdist(\mathbf{p}, \mathbf{q}) = 0 \iff \mathbf{p} = \mathbf{q} \quad (3.12)$$

$$Qdist(\mathbf{p}, \mathbf{q}) = Qdist(\mathbf{q}, \mathbf{p}) \quad (3.13)$$

$$Qdist(\mathbf{p}, \mathbf{q}) \leq Qdist(\mathbf{p}, \mathbf{z}) + Qdist(\mathbf{z}, \mathbf{q}) \quad (3.14)$$

Note that $Qdist \rightarrow 0$ as $\mathbf{q}_1 \rightarrow \mathbf{q}_2$, indicating that the end effector has reached the target

and to end execution. As for dual quaternions, the $vec8$ and $DQrel$ functions are used to find the distance between two dual quaternions as follows:

$$DQdist(\hat{\mathbf{q}}_1, \hat{\mathbf{q}}_2) = \| vec8(\hat{\mathbf{0}}) - vec8(DQrel(\hat{\mathbf{q}}_1, \hat{\mathbf{q}}_2)) \| \quad (3.15)$$

where $\hat{\mathbf{0}}$ is the null dual quaternion. This measure of distance satisfies the following three properties:

$$DQdist(\hat{\mathbf{p}}, \hat{\mathbf{q}}) = 0 \iff \hat{\mathbf{p}} = \hat{\mathbf{q}} \quad (3.16)$$

$$DQdist(\hat{\mathbf{p}}, \hat{\mathbf{q}}) = DQdist(\hat{\mathbf{q}}, \hat{\mathbf{p}}) \quad (3.17)$$

$$DQdist(\hat{\mathbf{p}}, \hat{\mathbf{q}}) \leq DQdist(\hat{\mathbf{p}}, \hat{\mathbf{z}}) + DQdist(\hat{\mathbf{z}}, \hat{\mathbf{q}}) \quad (3.18)$$

Although unifying both translational and rotational distances into a single distance metrics can be an ambiguous measure of distance between poses, it can be seen that $DQdist \rightarrow 0$ as $\hat{\mathbf{q}}_1 \rightarrow \hat{\mathbf{q}}_2$, which is suitable for our control purposes; to know when the end effector has reached the target and to end execution.

Scaling

Vectors are scaled by simply multiplying them by a constant. For scaling quaternions by some constant c :

$$Qscale(\mathbf{q}, c) = \mathbf{q}^c \quad (3.19)$$

However to develop a similar concept of scaling by some constant c for dual quaternions, dual quaternions must be broken into their translational and rotational quaternions and then recombined after scaling as follows:

$$DQscale(\hat{\mathbf{q}}, c) = \hat{\mathbf{s}}_{scale} = \mathbf{r}^c + \frac{\epsilon c}{2} \mathbf{tr}^c \quad (3.20)$$

where the exponential of some quaternion $\mathbf{q}^c = \exp(\ln(\mathbf{q})c)$. The definitions of $\exp$ and $\ln$ can be found in [?]. Note that $\hat{\mathbf{s}}_{scale}$ must be a unit dual quaternion and normalized if not. For example, $c = 2$ would result in a dual quaternion composed of twice the rotation and translation.

Linear Combination

To find a weighted sum of quaternions $\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_n$ using weights $w_1, w_2, \dots, w_n$, the type of quaternion must first be determined; whether it is a translational or rotational quaternion. For translational quaternions, $\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_n$, the weighted sum is found as follows:

$$QLC_t(\mathbf{t}_1, \dots, \mathbf{t}_n, w_1, \dots, w_n) = w_1\mathbf{t}_1 + w_2\mathbf{t}_2 + \dots + w_n\mathbf{t}_n \quad (3.21)$$

or more compactly:

$$QLC_t(\mathbf{t}_1, \dots, \mathbf{t}_n, w_1, \dots, w_n) = \sum_{i=1}^n w_i\mathbf{t}_i \quad (3.22)$$

On the other hand, for rotational quaternions, $\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_n$, the weighted sum is found as follows:

$$QLC_r(\mathbf{r}_1, \dots, \mathbf{r}_n, w_1, \dots, w_n) = \mathbf{r}_1^{w_1}\mathbf{r}_2^{w_2}\dots\mathbf{r}_n^{w_n} \quad (3.23)$$

which be written more compactly:

$$QLC_r(\mathbf{r}_1, \dots, \mathbf{r}_n, w_1, \dots, w_n) = \prod_{i=1}^n \mathbf{r}_i^{w_i} \quad (3.24)$$

As for dual quaternions, to find a weighted sum of $\hat{\mathbf{q}}_1, \hat{\mathbf{q}}_2, \dots, \hat{\mathbf{q}}_n$ using weights $w_1, w_2, \dots, w_n$, the dual quaternions are decomposed into their rotational quaternions, $\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_n$, and translational quaternions, $\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_n$. The weighted sum of the rotational quaternions is as follows:

$$\mathbf{r}' = \mathbf{r}_1^{w_1}\mathbf{r}_2^{w_2}\dots\mathbf{r}_n^{w_n} \quad (3.25)$$

which be written more compactly:

$$\mathbf{r}' = \prod_{i=1}^n \mathbf{r}_i^{w_i} \quad (3.26)$$

The weighted sum of the translational quaternions is calculated as follows:

$$\mathbf{t}' = w_1 \mathbf{t}_1 + w_2 \mathbf{t}_2 + \cdots + w_n \mathbf{t}_n \quad (3.27)$$

or more compactly:

$$\mathbf{t}' = \sum_{i=1}^n w_i \mathbf{t}_i \quad (3.28)$$

As a result, the weighted sum of the dual quaternions is computed as:

$$DQLC(\hat{\mathbf{q}}_1, \cdots, \hat{\mathbf{q}}_n, w_1, \cdots, w_n) = \mathbf{r}' + \frac{\epsilon}{2} \mathbf{t}' \mathbf{r}' \quad (3.29)$$

$$= \prod_{i=1}^n \mathbf{r}_i^{w_i} + \frac{\epsilon}{2} \sum_{i=1}^n w_i \mathbf{t}_i \prod_{i=1}^n \mathbf{r}_i^{w_i} \quad (3.30)$$

Regression

Regression can be seen as the inverse problem of calculating the weighted sum. The functions QLC and $Qdist$ are used to find the regression coefficients, $w = [w_1, w_2, \cdots, w_n]$, of the quaternions $\boldsymbol{\psi} = [\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_n]$ that yield some desired quaternion $\mathbf{q}_d$:

$$Qreg(\mathbf{q}_d, \boldsymbol{\psi}) = \min_w Qdist(\mathbf{q}_d, \mathbf{q}_r) \quad (3.31)$$

where

$$\mathbf{q}_r = QLC(\boldsymbol{\psi}, w) \quad (3.32)$$

Where QLC_t or QLC_r is used according to the type of quaternion. As for dual quaternions, the functions $DQLC$ and $DQdist$ are used to find the regression coefficients, $w = [w_1, w_2, \cdots, w_n]$, of the dual quaternions $\hat{\boldsymbol{\psi}} = [\hat{\mathbf{q}}_1, \hat{\mathbf{q}}_2, \cdots, \hat{\mathbf{q}}_n]$ that yield some desired dual

quaternion $\hat{\mathbf{q}}_d$:

$$DQreg(\hat{\mathbf{q}}_d, \hat{\boldsymbol{\psi}}) = \min_w DQdist(\hat{\mathbf{q}}_d, \hat{\mathbf{q}}_r) \quad (3.33)$$

where

$$\hat{\mathbf{q}}_r = DQLC(\hat{\boldsymbol{\psi}}, w) \quad (3.34)$$

Both $Qreg$ and $DQreg$ can be optimized using some optimization algorithm. In our implementation of the KMF-DQ, MATLAB's FMINCON optimization algorithm was used.

3.3 KMF Quaternionic Position Control

The position of a robot's end effector can be represented using a quaternion:

$$\mathbf{p} = 0 + x\mathbf{i} + y\mathbf{j} + z\mathbf{k} \quad (3.35)$$

where x, y, z are the positions of the end effector along the x -, y -, z - axes. The controller tries to bring the end effector to some target position, $\mathbf{p}_t$. The controller performs the reaching task by reaching intermediate positions, $\mathbf{p}_i$, along the way which are found using LERP interpolation between the end effector's current position, $\mathbf{p}_c$, and the target position, $\mathbf{p}_t$. At each step, the controller estimates an actuation primitive, b , to move the end effector towards the target position:

$$b = \begin{bmatrix} \tau_p^1(\hat{p}) \\ \tau_p^2(\hat{p}) \\ \vdots \\ \tau_p^m(\hat{p}) \end{bmatrix} \quad (3.36)$$

where $\tau_p^j(\hat{p})$ is the magnitude of the desired actuation primitive $\hat{p}$ on actuator j . It is

expressed as a linear combination of the kNN actuation primitives:

$$Aw = b \quad (3.37)$$

where $w = [w_1, w_2, \dots, w_k]^T$ is a set of unknown weights and the A contains the primitives of the kNN:

$$A = \begin{bmatrix} \tau_p^1(p_1) & \tau_p^1(p_2) & \cdots & \tau_p^1(p_k) \\ \tau_p^2(p_1) & \tau_p^2(p_2) & \cdots & \tau_p^2(p_k) \\ \vdots & \vdots & \ddots & \vdots \\ \tau_p^m(p_1) & \tau_p^m(p_2) & \cdots & \tau_p^m(p_k) \end{bmatrix} \quad (3.38)$$

The corresponding effect of each of the kNN actuation primitives forms the observation matrix:

$$\boldsymbol{\psi} = [\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_k] \quad (3.39)$$

The desired actuation primitive, b , is found by substituting in the weights from solving the following:

$$QReg(\mathbf{p}_i, \boldsymbol{\psi}) = \min_w QDist(\mathbf{p}_i, QLC(\boldsymbol{\psi}, w)) \quad (3.40)$$

After each step, the resulting effect is compared to the expected effect. Should there be any significant discrepancy, indicating that the local kinodynamics are not well captured within the local linear model, a new exploratory phase is triggered. The algorithm terminates when $QDist < \sigma$, where σ is the termination accuracy.

3.4 KMF Quaternionic Orientation Control

The orientation of a robot's end effector can be represented using a quaternion:

$$\mathbf{q} = w + a\mathbf{i} + b\mathbf{j} + c\mathbf{k} \quad (3.41)$$

The controller attempts to actuate the robot's end effector towards some desired orientation, $\mathbf{q}_t$. The controller reaches intermediate orientations, $\mathbf{q}_i$, along the way which are found using LERP interpolation between the end effector's current orientation, $\mathbf{q}_c$, and the target orientation, $\mathbf{q}_t$. At each step, the controller estimates an actuation primitive, b , to actuate the robot's end effector towards the desired target orientation:

$$b = \begin{bmatrix} \tau_p^1(\hat{p}) \\ \tau_p^2(\hat{p}) \\ \vdots \\ \tau_p^m(\hat{p}) \end{bmatrix} \quad (3.42)$$

where $\tau_p^j(\hat{p})$ is the magnitude of the desired actuation primitive $\hat{p}$ on actuator j . It is expressed as a linear combination of the kNN actuation primitives:

$$Aw = b \quad (3.43)$$

where $w = [w_1, w_2, \dots, w_k]^T$ is a set of unknown weights and the A contains the primitives of the kNN:

$$A = \begin{bmatrix} \tau_p^1(p_1) & \tau_p^1(p_2) & \cdots & \tau_p^1(p_k) \\ \tau_p^2(p_1) & \tau_p^2(p_2) & \cdots & \tau_p^2(p_k) \\ \vdots & \vdots & \ddots & \vdots \\ \tau_p^m(p_1) & \tau_p^m(p_2) & \cdots & \tau_p^m(p_k) \end{bmatrix} \quad (3.44)$$

The corresponding effect of each of the kNN actuation primitives forms the observation matrix:

$$\psi = [\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_k] \quad (3.45)$$

The desired actuation primitive, b , is found by substituting in the weights from solving the

following:

$$QReg(\mathbf{q}_i, \boldsymbol{\psi}) = \min_w QDist(\mathbf{q}_i, QLC(\boldsymbol{\psi}, w)) \quad (3.46)$$

After the completion of each step, the resulting effect on the end effector is compared to the desired effect. In the case of having significant discrepancy, which indicates that the local kinodynamics are not well captured within the local linear model, a new exploratory phase is triggered. The algorithm terminates when $QDist < \sigma$, where σ is the termination accuracy.

3.5 KMF Dual-Quaternionic Pose Control

At the core of the idea lies the assumption that it is possible to obtain information about the local kinodynamics of the robotic manipulator, which is gathered by generating pseudo-random actuation control signals and observing their effects on the robot's end effector.

When the end effector reaches the target, the controller ends the execution. Otherwise, the end effector is to reach a set of intermediate targets $\hat{\mathbf{q}}_i$ that lead to the final desired target pose, $\hat{\mathbf{q}}_t$, in discrete steps. These intermediate targets are found using ScLERP interpolation between the end effector's current pose, $\hat{\mathbf{q}}_c$, and the desired target pose, $\hat{\mathbf{q}}_t$. At each step, the controller tries to estimate an actuation primitive, $\mathbf{b}$, to move the end effector towards the desired goal pose:

$$\mathbf{b} = \begin{bmatrix} \tau_p^1(\hat{p}) \\ \tau_p^2(\hat{p}) \\ \vdots \\ \tau_p^m(\hat{p}) \end{bmatrix} \quad (3.47)$$

where $\tau_p^j(\hat{p})$ is the magnitude of the desired actuation primitive $\hat{p}$ on actuator j . After each actuation, the end-effector displacement is recorded as an observation, which is calculated using the $DQrel$ function. The kNN observation dual quaternions form the observation vector:

$$\boldsymbol{\psi} = [\hat{\mathbf{o}}_1, \dots, \hat{\mathbf{o}}_k] \quad (3.48)$$

These dual quaternions contain the rotational and translational displacements caused by the executed actuation primitives. The actuation primitives corresponding to these observations form the actuation matrix:

$$A = \begin{bmatrix} \tau_p^1(p_1) & \tau_p^1(p_2) & \cdots & \tau_p^1(p_k) \\ \tau_p^2(p_1) & \tau_p^2(p_2) & \cdots & \tau_p^2(p_k) \\ \vdots & \vdots & \ddots & \vdots \\ \tau_p^m(p_1) & \tau_p^m(p_2) & \cdots & \tau_p^m(p_k) \end{bmatrix} \quad (3.49)$$

The desired actuation primitive, $\mathbf{b}$, is assumed to be a linear combination of the kNN actuation primitives:

$$Aw = b \quad (3.50)$$

where $w = [w_1, w_2, \dots, w_k]^T$ is a set of unknown weights. These weights are found by solving the local regression of the observation vector:

$$DQreg(\hat{\mathbf{q}}_i, \boldsymbol{\psi}) = \min_w DQdist(\hat{\mathbf{q}}_i, DQLC(\boldsymbol{\psi}, w)) \quad (3.51)$$

With the weights, the desired actuation is then computed using Equation (2.1). After the completion of each finite actuation, the resulting effect on the end effector's state is compared with the anticipated effect. If the difference is significant, this means that the local linear model approximation of the local kinodynamics was not known precisely enough, which will trigger a new exploratory phase. The controller escapes the exploratory phase once it has done a set number of explorations. Lastly, the algorithm terminates when $DQdist < \sigma$, where σ is the termination accuracy.

3.6 KMF Multi-Point Pose Control

After specifying the control points that are to be controlled via markers or trackers, the controller first goes through an actuators discovery phase. Each actuator, $j \in \mathbb{J}$, is moved one after another

but in no particular order, where $\mathbb{J}$ is the set of all actuators. For each control point $i \in \mathbb{I}$, the actuators that can contribute to displacing it are assigned to that point, where $\mathbb{I}$ is the set of all control points. Thus, actuators could be assigned to multiple control points. The control point that can be displaced by all actuators, most probably being the end effector of the robot, is assigned an index $i = n$, where n is the cardinality of set $\mathbb{I}$. The discovery phase produces the following Boolean matrix:

$$S = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{bmatrix} = \begin{bmatrix} \delta_1^1 & \delta_2^1 & \cdots & \delta_m^1 \\ \delta_1^2 & \delta_2^2 & \cdots & \delta_m^2 \\ \vdots & \vdots & \ddots & \vdots \\ \delta_1^n & \delta_2^n & \cdots & \delta_m^n \end{bmatrix} \quad (3.52)$$

where m is the cardinality of set $\mathbb{J}$. The function above is defined as: $\delta_j^i = 1$ if actuator j can displace control point i and $\delta_j^i = 0$ otherwise. To find the number of actuators that contribute to some control point i , the dot product can be used $a_i = s_i \cdot s_i$. A vector of the number of such actuators can be defined as $a = [a_1, a_2, \dots, a_n]$.

The KMF Multi-Point controller, just like any other KMF controller, assumes that the local kinodynamics information of the manipulator is attainable. This information is attained by generating pseudo-random actuation signals to actuate the robot joints and observing their result on the robot's control point(s). A high-level diagram summarizing the KMF Multi-Point approach is shown in Figure 3.1.

The actuation primitives produce signals $\tau(t)$ which are sent to each of the robot's actuators and is formulated in (2.1). Note that actuation primitives could be motor torques, motor currents, joint positions, etc. The controller tracks the poses of the control points, $\hat{\mathbf{q}}_i$, using dual quaternion representation which yield the controlled points vector:

$$\hat{\phi} = [\hat{\mathbf{q}}_1, \dots, \hat{\mathbf{q}}_n] \quad (3.53)$$

The controller attempts to bring the points $\hat{\mathbf{q}}_i$ towards desired target poses $\hat{\mathbf{d}}_i^f$, yielding the

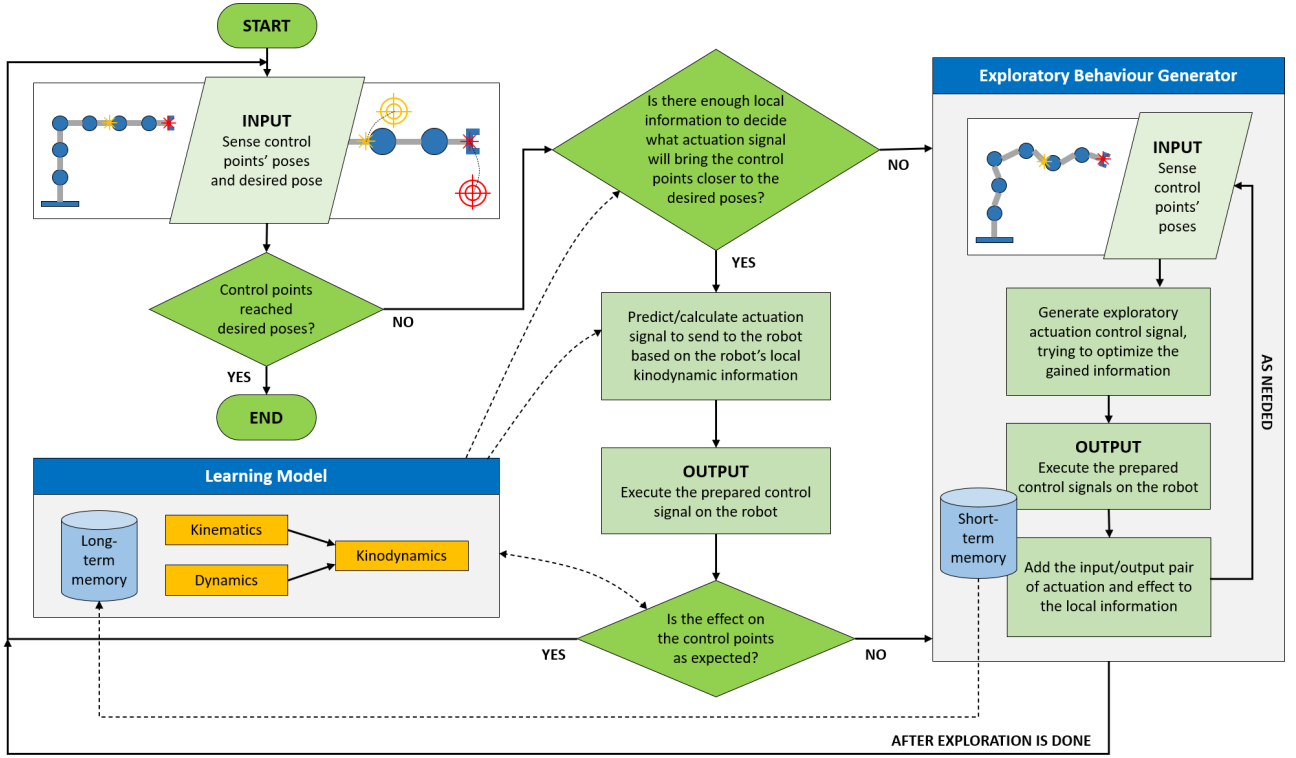


Figure 3.1: Flowchart of KMF Multi-Point control.

desired poses vector as follows:

$$\hat{\gamma} = [\hat{\mathbf{d}}_i^f, \dots, \hat{\mathbf{d}}_n^f] \quad (3.54)$$

By controlling multiple points, redundancies can be easily resolved by specifying the desired poses of other points along the robot's serial chain. The controller terminates when $\hat{\phi} = \hat{\gamma}$; i.e. when the poses of the controlled points of interest coincide with their corresponding desired poses. This condition can be relaxed by allow the controlled points' poses to reach within some deviation σ of their respective desired target poses. Otherwise, the controller is to actuate the control points towards a set of intermediate poses $\hat{\mathbf{d}}_i^t$ that lead to the final desired target poses $\hat{\mathbf{d}}_i^f$ in discrete steps. The intermediate poses are found using ScLERP interpolation between the current poses of the controlled points, $\hat{\mathbf{q}}_i$, and desired target poses, $\hat{\mathbf{d}}_i^f$. At each step, the KMF Multi-Point controller estimates actuation primitives, b_i , to move each controlled point $i \in \mathbb{I}$

towards their respective desired poses:

$$b_i = \begin{bmatrix} \tau_p^1(\hat{p}) \\ \tau_p^2(\hat{p}) \\ \vdots \\ \tau_p^m(\hat{p}) \end{bmatrix} \quad (3.55)$$

where $\tau_p^j(\hat{p})$ is the magnitude of the actuation primitive of actuator j . After each step where the robot joints have been actuated, the displacement of the control points are recorded as observations, which are calculated using the *DQrel* function. The nearest k observation dual quaternions to each point $i \in \mathbb{I}$ form the observation matrix:

$$\hat{\psi} = \begin{bmatrix} \hat{\psi}_1 \\ \hat{\psi}_2 \\ \vdots \\ \hat{\psi}_n \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{o}}_1^1 & \hat{\mathbf{o}}_2^1 & \dots & \hat{\mathbf{o}}_k^1 \\ \hat{\mathbf{o}}_1^2 & \hat{\mathbf{o}}_2^2 & \dots & \hat{\mathbf{o}}_k^2 \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\mathbf{o}}_1^n & \hat{\mathbf{o}}_2^n & \dots & \hat{\mathbf{o}}_k^n \end{bmatrix} \quad (3.56)$$

where $\hat{\mathbf{o}}_k^i$ is the k^{th} closest observation dual quaternion of controlled point i . These dual quaternions contain the rotational and translational displacements of the control points caused by the executed actuation primitives. The actuation primitives corresponding to these observations form the actuation matrix:

$$A_i = \begin{bmatrix} \tau_p^1(p_1) & \tau_p^1(p_2) & \dots & \tau_p^1(p_k) \\ \tau_p^2(p_1) & \tau_p^2(p_2) & \dots & \tau_p^2(p_k) \\ \vdots & \vdots & \ddots & \vdots \\ \tau_p^{a_i}(p_1) & \tau_p^{a_i}(p_2) & \dots & \tau_p^{a_i}(p_k) \end{bmatrix} \quad (3.57)$$

The desired actuation primitives, b_i , is assumed to be a linear combination of the k nearest actuation primitives such that:

$$A_i w_i = b_i \quad (3.58)$$

where $w_i = [w_1^i, w_2^i, \dots, w_k^i]^T$ is a set of unknown weights. These weights are found by solving

the local regression of the observation matrix:

$$DQreg(\hat{\mathbf{d}}_i^t, \hat{\boldsymbol{\psi}}_i) = \min_{w_i} DQdist(\hat{\mathbf{d}}_i^t, DQLC(\hat{\boldsymbol{\psi}}_i, w_i)) \quad (3.59)$$

With the weights for each control point, the desired actuation is then computed using Equation (3.58) for each control point. The respective actuation matrix is used in the calculation. After the primitive for each control point has been obtained, the following averaging is done to dampen and stir the actuation of joints that contribute to the displacement of the control points:

$$b'_i = \frac{\sum_{r=1}^{r=i} \eta_r^i Z(b_r)}{\sum_{r=1}^{r=i} \eta_r^i} \quad (3.60)$$

where the operator Z adds zero padding to a vector to keep the length equal to m and the averaging weights matrix is a lower triangular matrix:

$$H = \begin{bmatrix} \eta_1^1 & 0 & 0 & 0 & \cdots & 0 \\ \eta_1^2 & \eta_2^2 & 0 & 0 & \cdots & 0 \\ \eta_1^3 & \eta_2^3 & \eta_3^3 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \eta_1^n & \eta_2^n & \eta_3^n & \eta_4^n & \ddots & \eta_n^n \end{bmatrix} \quad (3.61)$$

In the case that the averaging weights for each control point is normalized, Equation (3.60) would simply be:

$$b'_i = \sum_{r=1}^{r=i} \eta_i^r Z(b_r) \quad (3.62)$$

After each discrete actuation, the resulting displacement on the pose of each control point is compared to the anticipated effect. If the difference is significant, a new exploratory phase will be triggered to collect more data about the robot's local kinodynamics as shown in Figure 3.1. The controller leaves the exploratory phase once it has completed a predefined number of explorations. The controller terminates once all control point poses have reached the desired poses or close enough to them, $DQdist(\hat{\mathbf{q}}, \hat{\mathbf{d}}) < \sigma$.

3.7 Results

3.7.1 Single-Point Pose Control

A dynamic simulation of a 3-DOF robot was performed using MATLAB’s Robotics Toolbox [?] and DQ Robotics Toolbox [?]. The kinematic and dynamic specifications of the robot used for the experiments are shown in Table 3.1.

Table 3.1: Robot specifications.

Link	Length (m)	Mass (kg)
L_1	0.50	1.00
L_2	0.50	1.00
L_3	0.50	1.00

The information in Table 3.1 is not given to the KMF-DQ controller. The controller was implemented as described previously using $k = 4$ nearest neighbours comprising of actuation primitives and their corresponding observations. Since this experiment was conducted in simulation, there was no need to use a camera to obtain the end-effector’s state. Instead, this was achieved internally from the simulator. We note that gravity is neglected in the simulation to keep the focus on simultaneous position and orientation control, deferring gravity compensation until future research.

The experiment involves a set of reaching tasks. The controller must move the robot’s end effector to four target poses, drawing a polygon in 3D space. The position and orientation of the four target poses are listed in Table 3.2. The starting position of the robot was $(-0.85059, 1.2148, -0.12941)$ and orientation was $(-1.5708, 0.9599, 2.8798)$.

Table 3.2: Target poses.

Target	Position	Orientation
1	(−0.6827, 0.9525, 0.3565)	(−1.5710, 1.3350, −2.3476)
2	(−0.1410, 1.1470, 0.3830)	(1.5706, 1.3089, 0.8729)
3	(0.1216, 1.4198, 0.1710)	(1.5709, 1.3090, 0.3489)
4	(−0.8506, 1.2148, −0.1294)	(−1.5709, 0.9599, 2.8797)

The orientations are specified here in Euler XYZ angles for simplicity. The poses were selected to be within the robot’s reachable workspace. The results are shown in Figure 3.2 in all three orthogonal views. The KMF-DQ controller generates intermediate targets to follow the yellow trajectory. The robot successfully reaches all four target poses, drawing the polygon in 3D space. These results prove that the local kinodynamic information of the robot is sufficient to create a local model that can be used to actuate the end effector towards a given target. Note that the specified target poses must lie within the robot’s reachable workspace, otherwise a higher termination tolerance, σ , must be set.

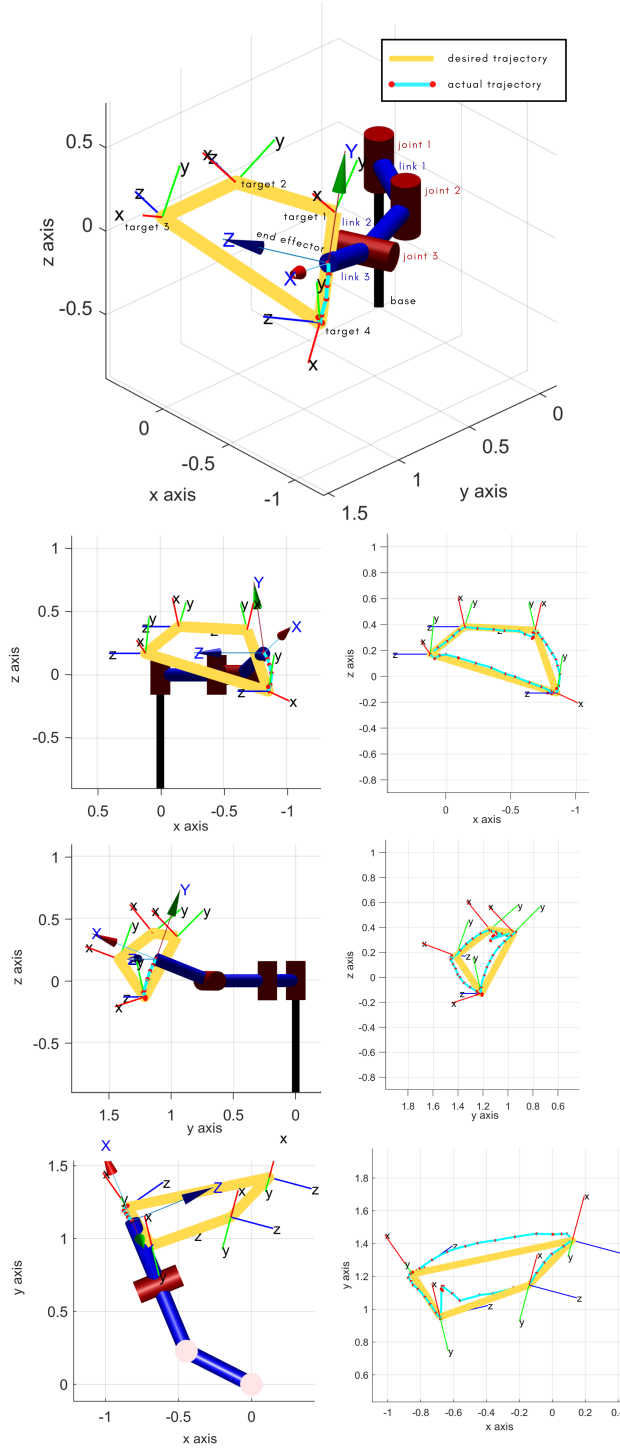


Figure 3.2: The KMF-DQ controller actuates the robot’s joints to perform a set of reaching tasks to draw out a polygon in 3D space with each vertex having a pre-specified pose within the robot’s reachable workspace. The bottom left three plots show the orthogonal views of the robot performing the experiment. The bottom right three plots show the final resulting trajectory of the end effector. The robot moves towards the next target only after reaching the current target.

To demonstrate the actuation primitives generated by the controller, Figure 3.3 shows the sequence of actuated primitives of the first reaching task (moving towards target 1). Since the KMF-DQ is a discrete controller, the figure shows the discrete primitives sent to the actuators. We note that as the end effector reaches the target, lower torques are sent to the actuators. The joint position of the three joints corresponding to the first reaching task is shown in Figure 3.4 to demonstrate the resulting displacements in the joints.

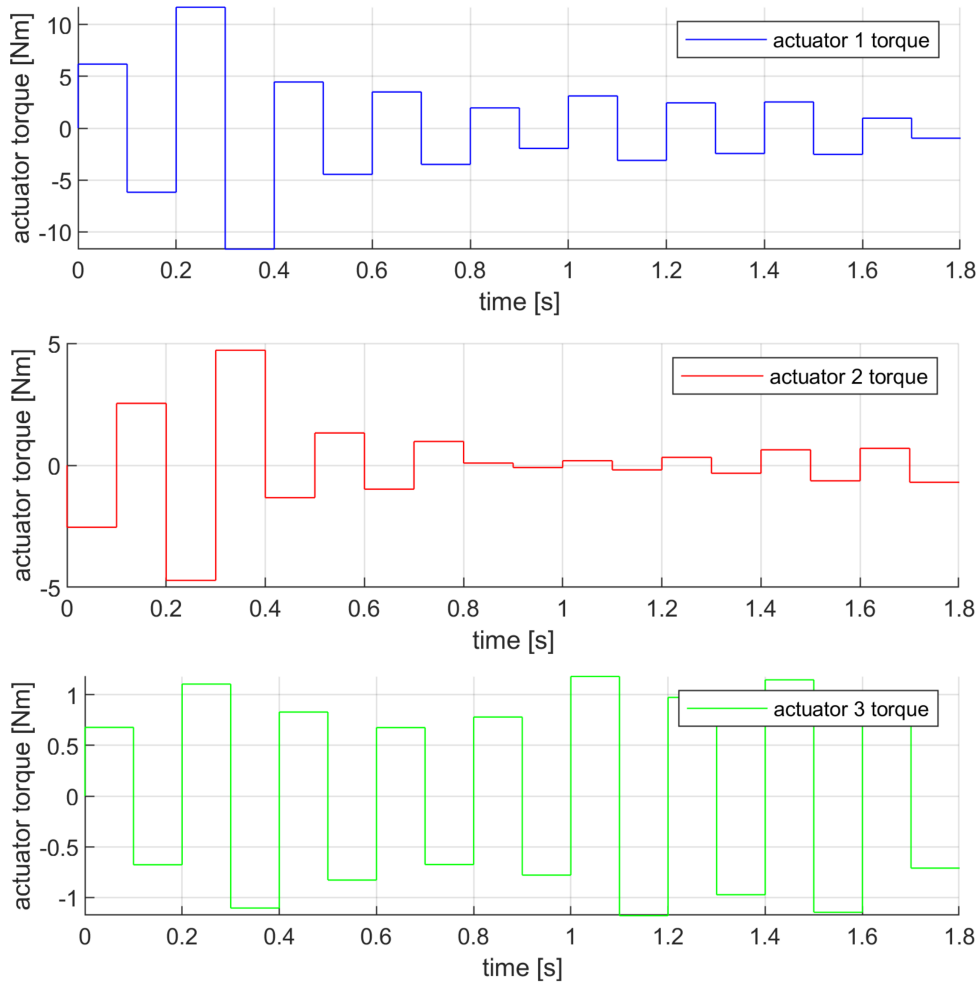


Figure 3.3: The control signals sent to the actuators generated by the controller. These actuation signals correspond to the first reaching task where the end effector moves towards target 1.

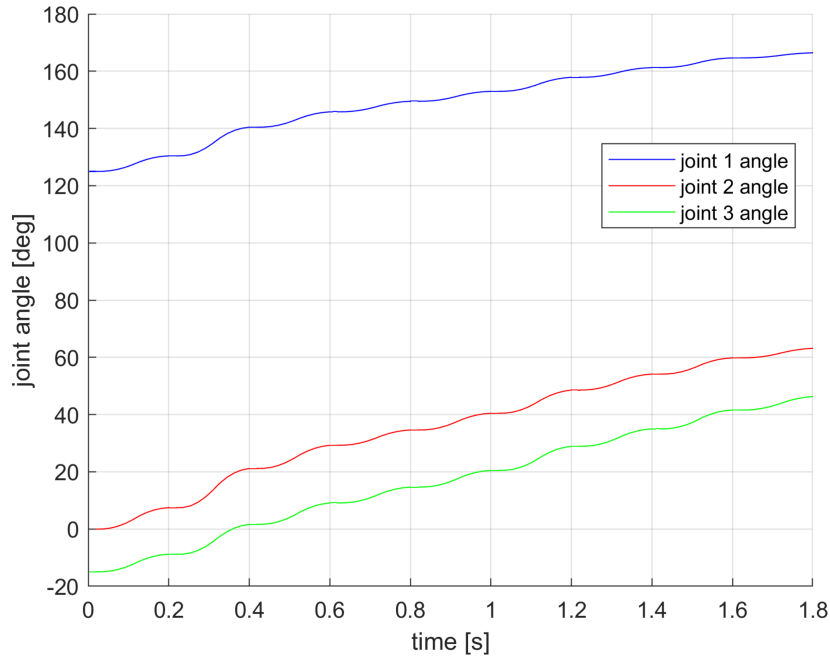


Figure 3.4: The joint positions of the three joints. These data correspond to the first reaching task where the end effector moves towards target 1.

We can see that the joint angles of the three joints vary smoothly with no sudden jerks. Furthermore, Figure 3.5 shows the pose distance of the end effector to the targets using the $DQdist$ function. We note that the pose distance approaches zero as the end effector approaches the targets.

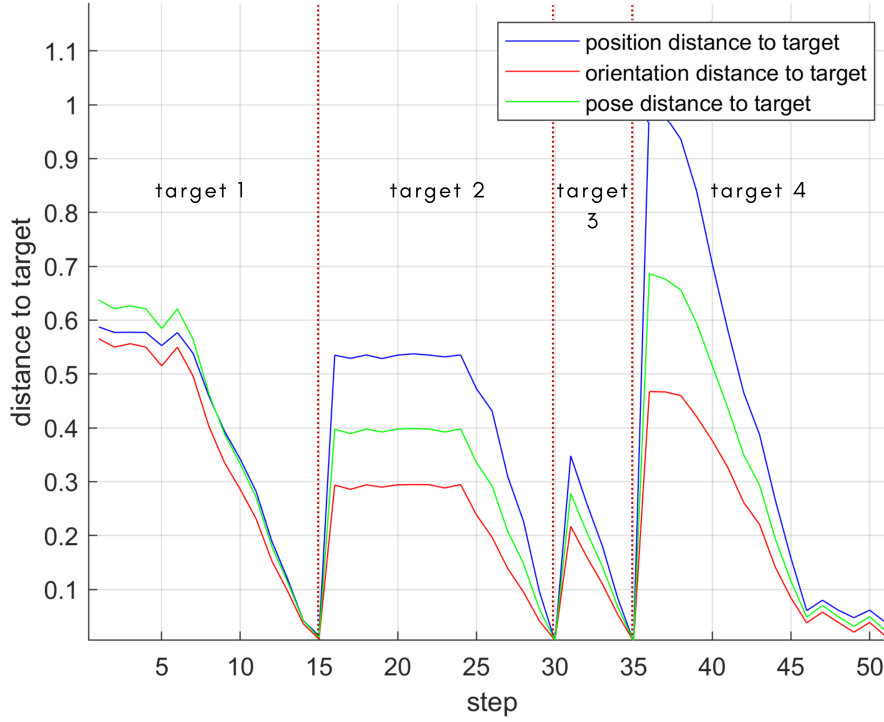


Figure 3.5: Distance of the end-effector pose to the target pose for each reaching task. The pose distance approaches zero as the end effector reaches the targets.

The simulation demonstrated convergence to the desired targets. The controller’s robustness was measured by running the simulation 100 times, each with a random starting robot configuration and a randomly placed target pose within the robot’s reachable workspace that is achievable within 100 steps. The target poses lie within the robot’s achievable configuration space. It was found that 87% of the time, the controller successfully performed the reaching task. In the remaining 13%, the controller did not converge to solutions that actuate the robot towards the targets. These failures can occur because the optimizer gets stuck in a local minimum, which will yield inaccurate results for the weights and thus the calculated actuation primitives.

3.7.2 Multi-Point Pose Control

The controller is tested on a simulated redundant 9-degrees-of-freedom continuum robot, shown in Figure 3.6.

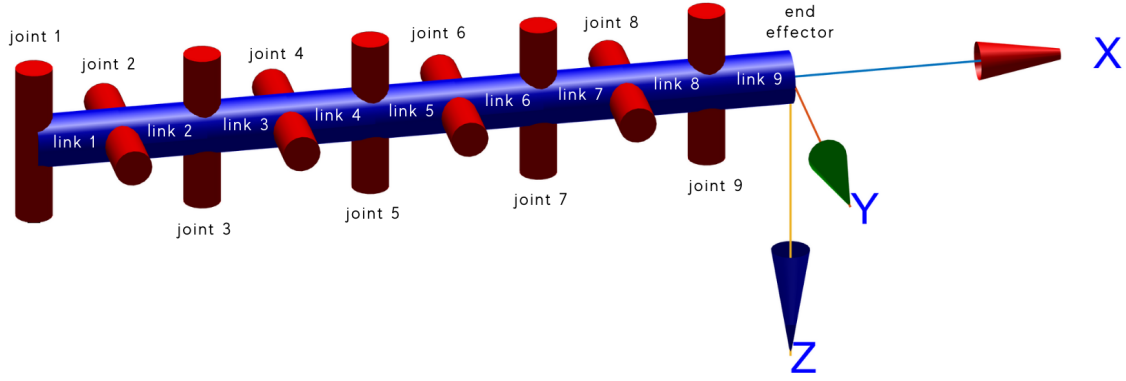


Figure 3.6: The simulated 9-degrees-of-freedom continuum robot.

The robot was simulated using Matlab’s Robotics Toolbox [?] and DQ Robotics Toolbox [?]. The actuation primitives used were joint position command signals to drive the control points towards the target poses. The Denavit–Hartenberg parameters of the robot is listed in Table 3.3.

Table 3.3: Denavit–Hartenberg parameters of redundant robot.

Link	d_i	θ_i	a_i	α_i
1	0	0	0.05	1.5708
2	0	0	0.05	1.5708
3	0	0	0.05	−1.5708
4	0	0	0.05	1.5708
5	0	0	0.05	−1.5708
6	0	0	0.05	1.5708
7	0	0	0.05	−1.5708
8	0	0	0.05	1.5708
9	0	0	0.05	0

The controller was programmed as described previously using $k = 4$ nearest neighbors that comprise of actuation primitive and observation pairs. Since the experiments were conducted in simulation, a tracking camera was not needed to determine the poses of the control points.

In addition, actuators discovery was also omitted in the simulation. Instead, the simulator provided these information directly. Furthermore, gravity was omitted in the simulation to keep the focus on multi-point pose control, deferring gravity compensation to future work.

Three experiments were performed to study the capabilities of the proposed kinematic-model-free multi-point controller. The first experiment involved end-effector pose control for the continuum robot with increasing number of degrees of freedom enabled. This experiment shows the degrading performance of the controller when increasing the number of degrees of freedom using a single control point. In the second experiment, the 9-degrees-of-freedom continuum robot was controlled using 1, 2, and 3 control point(s). This experiment highlights the capability of the kinematic-model-free multi-point controller to control robots of high degrees of freedom and resolve redundancies. Lastly, the robustness of the kinematic-model-free multi-point controller was measured by executing the simulation experiment 100 times with randomly-generated configurations.

Redundant Robot Single-Point Pose Control

In the first experiment, a single control point situated at the end effector of the continuum robot was used to actuate the robot towards a desired target pose. Initially, only the last 2 joints of the robot were enabled. Gradually, an extra joint was enabled, increasing the robot's degrees of freedom. The robot began from the home pose where the control-point's task-space position was $(0, 0, 0.4500)$, and orientation was $(0, 0, 3.1416)$. Orientations in this paper are specified in ZYX Euler angle representation for simplicity. The target position was $(0.4218, 0.0240, 0.0605)$ and orientation was $(0.6202, -0.6009, 2.7578)$. The results are shown in Figure 3.7 and tabulated in Table 3.4.

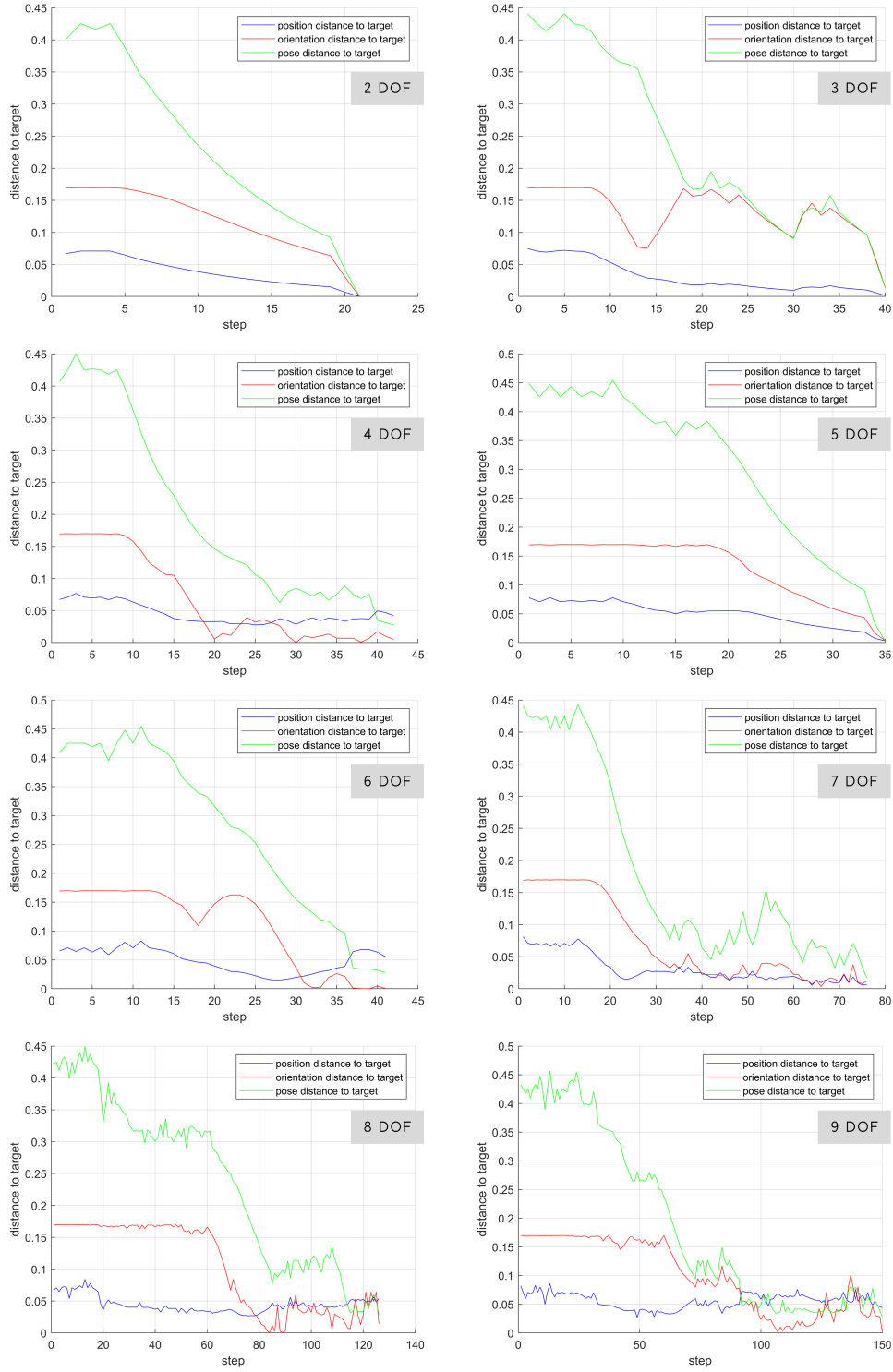


Figure 3.7: The controller performed the task of reaching, bringing a single control point at the end effector to a desired target pose. As more joints were enabled, the performance was degraded. In particular, in the 2-degrees-of-freedom case, the robot reaches the target in 21 steps, while, in the 9-degrees-of-freedom case, the robot reaches the target in 150 steps.

Table 3.4: Reaching steps for different degrees of freedom enabled.

Degrees of Freedom	Steps to Reach
2	21 steps
3	40 steps
4	42 steps
5	35 steps
6	41 steps
7	76 steps
8	127 steps
9	150 steps

In the simple 2-degrees-of-freedom case, the control point converged to the target pose within 25 steps. When increasing the degrees of freedom, it is noted that the controller takes longer to converge. Although it is evidently possible to reach the target pose within 25 steps, the added redundancies introduced additional nonlinearities that require excessive exploratory behavior. In the case where all 9 joints of the robot were enabled, the control point reached the target pose in 150 steps, a significant decrease in performance. With the stopping condition ($DQdist < \sigma$) applied, it can be seen that the controller can terminate before fully reaching the target pose.

Redundant Robot Multi-Point Pose Control

In this experiment, multiple control points were added to the continuum robot. In the first arrangement, only a single control point was added that was situated at the end effector. In the second arrangement, 2 control points were added, one at the end effector and one at the end of link 5. Finally, in the third arrangement, the 3 control points were added that were evenly distributed throughout the robot's kinematic chain. The placements of the control point for each arrangement are summarized in Table 3.5.

Table 3.5: Control points placement.

Arrangement	Control Point 1	Control Point 2	Control Point 3
1	end effector	-	-
2	end of link 5	end effector	-
3	end of link 3	end of link 6	end effector

These arrangements are depicted in Figure 3.8. Three reaching tasks were preformed with each arrangement. All desired poses were within the robot's configuration space. See Figure 3.9. The reaching tasks increased in complexity in terms of dexterity required. We note that arrangement 1 corresponds to kinematic-model-free control of the end-effector pose. Thus, this experiment also serves as a comparison with the kinematic-model-free orientation controller in Reference [?]. The results are shown in Figure 3.10 and tabulated in Table 3.6.

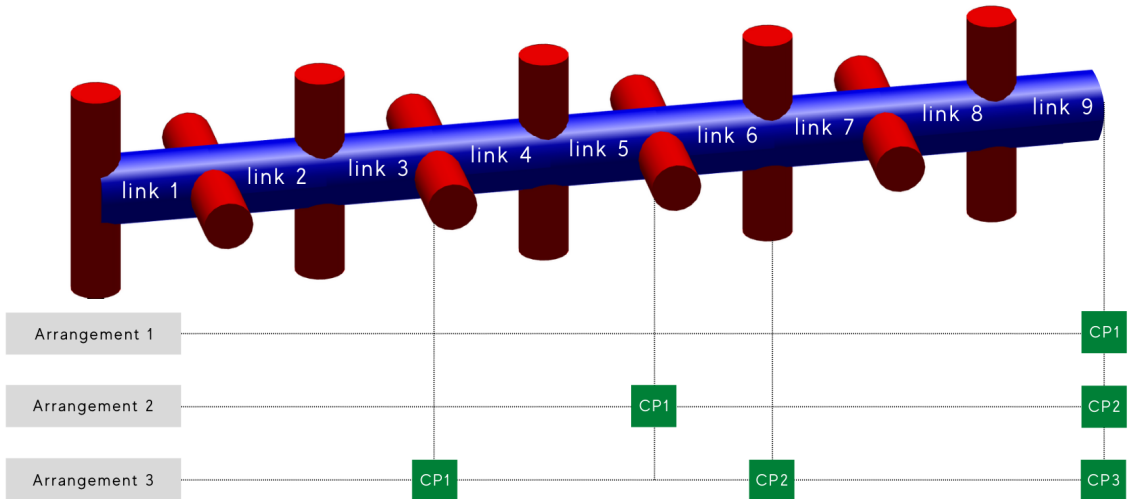


Figure 3.8: The three arrangements of the control points (CP's) along the robot's kinematic chain. In the first arrangement, only one control point was used which was placed at the end effector. In the second and third arrangements, two and three equally spaced control points were used, respectively.

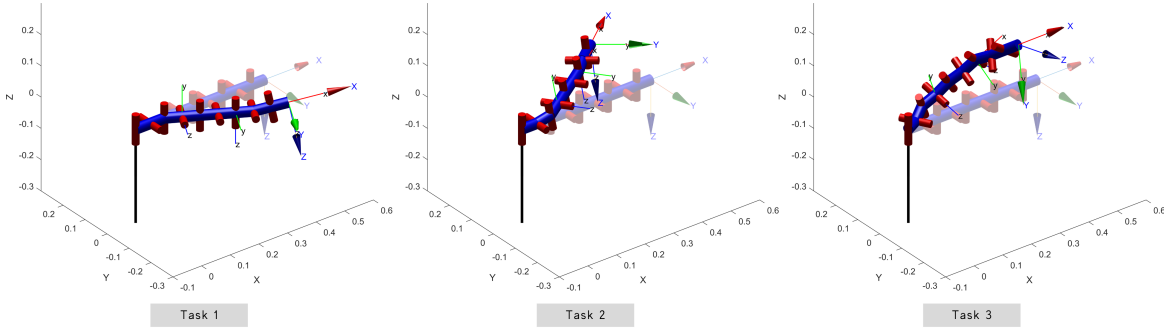


Figure 3.9: The three reaching tasks to be performed. The translucent pose is the initial configuration from which the robot is to actuate towards the opaque pose.

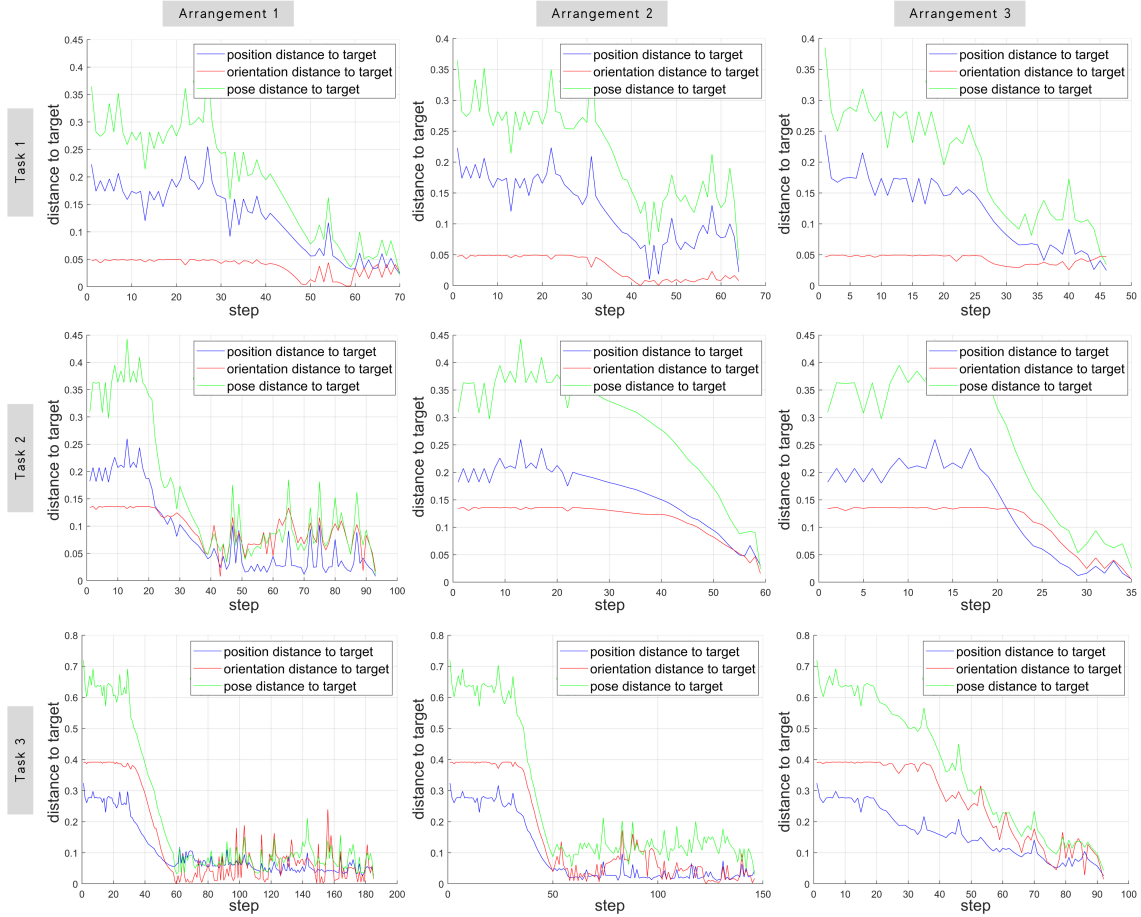


Figure 3.10: The controller performed three reaching tasks. In all 3 tasks, the controller converged faster when using more control points (CPs).

Only the pose distance of the control point situated at the end effector is shown. It can be seen that the controller converges faster to the desired pose when using more control points situated along the robot's kinematic chain. In task 1, we note that a 66% improvement in convergence speed when using 3 control points. In task 2 and 3, there was a 38% and 50% improvement was measured, respectively. These results indicate that adding more control points

Table 3.6: Steps required to complete three reaching tasks using three arrangements of control points.

Task	Arrangement		
	1	2	3
1	70 steps	64 steps	46 steps
2	93 steps	59 steps	35 steps
3	185 steps	147 steps	92 steps

will improve convergence and resolve redundancies in hyper-redundant systems. As for accuracy, the termination criteria was set to $DQdist \leq 0.03$, achieving a close enough reaching accuracy.

Furthermore, to demonstrate the joints' position during an execution, the joint position are shown in Figure 3.11. The larger jitters in joint positions correspond to exploratory behavior, which is triggered whenever there is deficiency in local kinodynamic information.

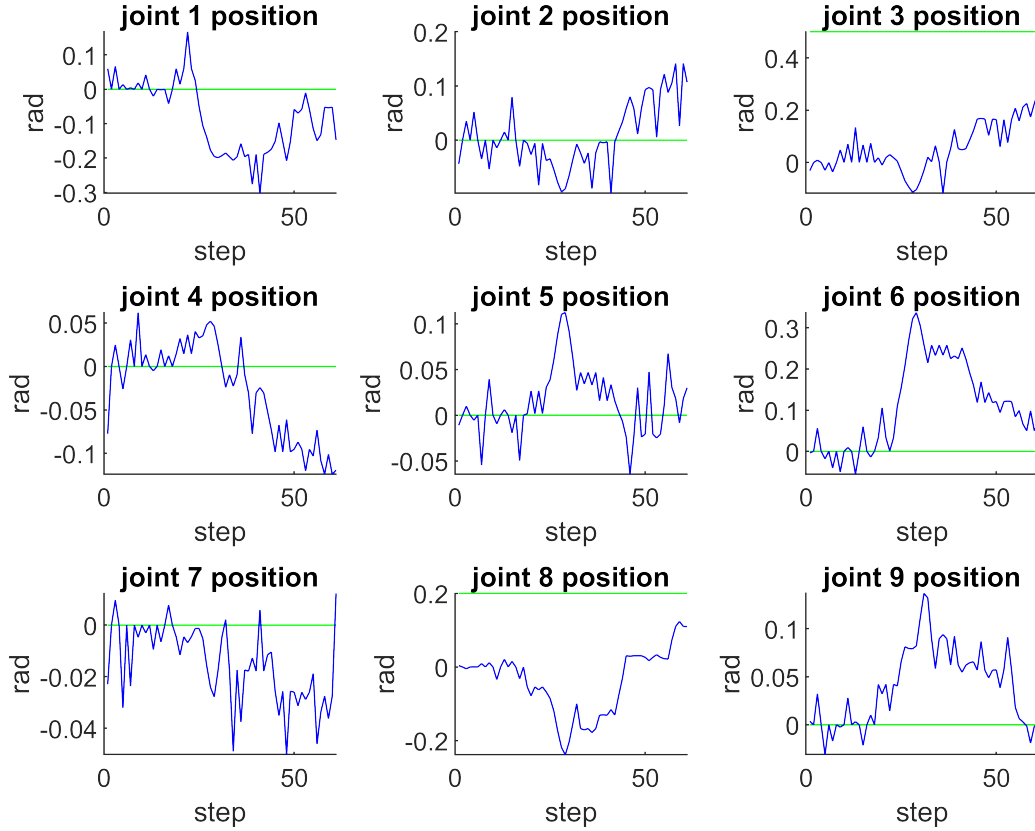


Figure 3.11: These joint positions correspond to reaching task 1. The larger jitters are due to exploratory behavior, which is triggered to collect more information about the robot's local kinodynamics.

The robustness of the algorithm was measured by executing the simulation 100 times. Each time, the robot must reach a randomly-generated pose that is within its configura-

tion space and it reachable within 150 steps. If the controller takes longer, the simulation is restarted. Three control points were used. One interesting configuration is shown in Figure 3.12. The controller managed to achieve this bent arch configuration within 150 steps.

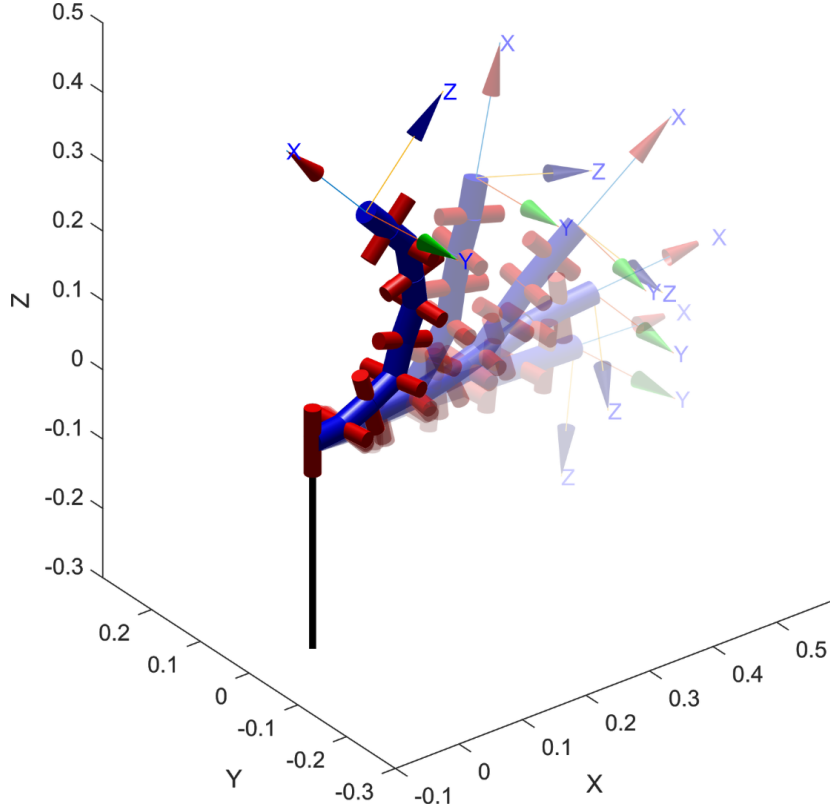


Figure 3.12: In this simulation run, the controller attempted to actuate the robot to a back arching configuration. It achieved to reach this configuration within 150 steps.

It was observed that 82% of the runs, the controller manages to reach to the target pose. However, in 18% of the runs, the controller did not successfully converge to the solutions that brought the robot's end effector to the desired poses. This can occur when the optimization is stuck in a local minimum, which results in an inaccurate actuation signal that does not optimally move the control points towards the target poses.

3.8 Discussion

This chapter presents a model-free controller for robot manipulation using locally weighted combinations of dual quaternions. A kinematic-model-free control for controlling the pose of multiple points along the kinematic chain of a continuum robot which is able to resolve

redundancies and converge faster to a desired configuration is also presented. The proposed controller is able to resolve redundancies and converge faster to a desired configuration. The approach does not require any joint angle information or prior knowledge about the robot's kinematics and dynamics. The controller works by executing random actuation primitives and perceiving their resulting effects on the end effector. These pieces of information are then used to build a local kinodynamic model. An actuation primitive is calculated using dual quaternion regression, which is executed on the robot to move the end effector towards a given target. This chapter also presents the dual quaternion distance, relative dual quaternion, and dual quaternion regression functions used for building a local kinodynamic model.

Simulation experiments were conducted in which a manipulator performed a set of reaching tasks. The simulation results show that the robot can reach a set of reference poses, successfully performing simultaneous position and orientation control. For the multi-point KMF controller, it was noted that when more control points were used, the controller performed better and converged faster to the desired target robot pose. Controlling multiple points along a robot did not only resolves redundancies, but proved to converge faster to the desired pose.

For the single-point KMF controller, the robustness of the approach was tested by running the simulation 100 times with random target poses that lie within the robot's reachable workspace, which are achievable by some configuration of the robot. The controller managed to successfully reach the target poses 87% of the time within 100 steps. While for the multi-point KMF controller, the controller succeeded in reaching the target poses within 150 steps 82% of the times.

The proposed approach looks promising and can be used in applications where simultaneous translational and rotational reaching is required. This approach can be used for new robot designs where a model is not present and where flexibility is favoured over precision and speed. With regards to future work, scalability to higher-degrees-of-freedom robot manipulators, continuous control, and practical real-world implementation issues such as joint limits and friction remain open problems that will be investigated in future research. Also, key issues that

hindered the immediate experimentation of the proposed controller to physical experiments, such as gravity compensation and friction compensation, must be tackled before the experiments are performed since the current controller does not take into account these physical phenomena.

References

- [1] Bruno Vilhena Adorno and Murilo Marques Marinho. Dq robotics: A library for robot modeling and control. *IEEE Robotics & Automation Magazine*, 28(3):102–116, 2020.
- [2] Tatsuya Aoki, Tomoaki Nakamura, and Takayuki Nagai. Learning of motor control from motor babbling. *IFAC-PapersOnLine*, 49(19):154–158, 2016.
- [3] Chien-Chern Cheah, Chao Liu, and Jean-Jacques E Slotine. Adaptive jacobian tracking control of robots with uncertainties in kinematic, dynamic and actuator models. *IEEE transactions on automatic control*, 51(6):1024–1029, 2006.
- [4] Clifford. Preliminary Sketch of Biquaternions. *Proceedings of the London Mathematical Society*, s1-4(1):381–395, 1871.
- [5] Peter Corke. *Robotics, vision and control: fundamental algorithms in MATLAB® second, completely revised*, volume 118. Springer, 2017.
- [6] Peter I Corke and Oussama Khatib. *Robotics, vision and control: fundamental algorithms in MATLAB*, volume 73. Springer, 2011.
- [7] Francesco Cursi, George P Mylonas, and Petar Kormushev. Adaptive kinematic modelling for multiobjective control of a redundant surgical robotic tool. *Robotics*, 9(3):68, 2020.
- [8] Rémi Dromnelle, Erwan Renaudo, Guillaume Pourcel, Raja Chatila, Benoît Girard, and Mehdi Khamassi. How to reduce computation time while sparing performance during robot navigation? a neuro-inspired architecture for autonomous shifting between model-based and model-free learning. In *Conference on Biomimetic and Biohybrid Systems*, pages 68–79. Springer, 2020.
- [9] Ashitava Ghosal. *Robotics: fundamental concepts and analysis*. Oxford university press, 2006.
- [10] Ron Goldman. Understanding quaternions. *Graphical models*, 73(2):21–49, 2011.

- [11] William Rowan Hamilton. On quaternions; or on a new system of imaginaries in algebra. *Philosophical magazine*, 25(3):489–495, 1844.
- [12] Da-Peng Han, Qing Wei, and Ze-Xiang Li. Kinematic control of free rigid bodies using dual quaternions. *International Journal of Automation and Computing*, 5(3):319–324, 2008.
- [13] Yao Huang and Jianbo Su. Visual servoing of nonholonomic mobile robots: A review and a novel perspective. *IEEE Access*, 7:134968–134977, 2019.
- [14] Yusong Jin, Yufei Wang, Xiaotong Chen, Zhanchi Wang, Xinghua Liu, Hao Jiang, and Xiaoping Chen. Model-less feedback control for soft manipulators. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2916–2922. IEEE, 2017.
- [15] Ladislav Kavan, Steven Collins, Carol O’Sullivan, and Jiri Zara. Dual quaternions for rigid transformation blending. *Trinity College Dublin, Tech. Rep. TCD-CS-2006-46*, 2006.
- [16] Petar Kormushev, Sylvain Calinon, and Darwin Caldwell. Reinforcement learning in robotics: Applications and real-world challenges. *Robotics*, 2(3):122–148, 2013.
- [17] Petar Kormushev, Yiannis Demiris, and Darwin G Caldwell. Encoderless position control of a two-link robot manipulator. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 943–949. IEEE, 2015.
- [18] Petar Kormushev, Yiannis Demiris, and Darwin G Caldwell. Kinematic-free position control of a 2-dof planar robot arm. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5518–5525. IEEE, 2015.
- [19] Gaofeng Li, Dezhen Song, Shan Xu, Lei Sun, and Jingtai Liu. Kinematic-free orientation control for a deformable manipulator based on the geodesic in rotation group $so(3)$. *IEEE Robotics and Automation Letters*, 3(3):2432–2438, 2018.
- [20] MIH Nour, Jasmin Ooi, and KY Chan. Fuzzy logic control vs. conventional pid control of an inverted pendulum robot. In *2007 International Conference on Intelligent and Advanced Systems*, pages 209–214. IEEE, 2007.

- [21] Amir Saleki and Mohammad Mehdi Fateh. Model-free control of electrically driven robot manipulators using an extended state observer. *Computers & Electrical Engineering*, 87:106768, 2020.
- [22] Timothy Sands. Development of deterministic artificial intelligence for unmanned underwater vehicles (uuv). *Journal of Marine Science and Engineering*, 8(8):578, 2020.
- [23] Simo Särkkä. Notes on quaternions. *Internal Technical Document, Helsinki University of Technology*, 2007.
- [24] Ken Shoemake. Animating rotation with quaternion curves. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 245–254, 1985.
- [25] Jun Wu, Jinsong Wang, and Zheng You. An overview of dynamic parameter identification of robots. *Robotics and computer-integrated manufacturing*, 26(5):414–419, 2010.

Chapter 4

Kinematic-Model-Free Obstacle Avoidance

Researchers over the past few decades studied optimal control, especially for applications where systems dynamics and constraints require proper handling [?]. Model Predictive Control (MPC) is an advance control method widely used in academia and industry [?]. MPCs are widely used in robotics due to their ability to handle constraints and run in real-time [?]. In general, MPCs require a system model prior to control in order to optimize the controller over a finite horizon [?] which can be challenging to obtain or maintain for certain types of robots (e.g. soft robots, continuum robots, transforming robots, etc.) [?]. The kinematic-model-free (KMF) controller is capable of controlling robot manipulators without any previous knowledge of the robot's kinematic structure or dynamic properties. The controller works by building local linear models that are used to perform reaching tasks [?]. However, the KMF controller previously have not been applied to high-level tasks such as trajectory planning and obstacle avoidance as they are concerned with low-level control. Along with the merits of MPCs, integrated together, they can perform high-level tasks. In this chapter, a kinematic-model-free predictive controller (KMF-PC), that is capable of performing end-effector reaching and obstacle avoidance for robot manipulators, is presented.

4.1 Related Work

MPC is a control method that uses a system model to predict future behavior which can be sensitive to prediction errors, especially in the case of nonlinear plants. To mitigate such sensitivities, adaptive model predictive control, where the plant model and nominal conditions are continuously adapted, is used [?]. MPCs gained popularity due to its ability to handle constraints and its capacity to run in real-time, which allowed it to be widely used in robotics [?]. MPCs have been used for path planning and collision avoidance of road vehicles [?] and multiple micro aerial vehicles [?]. [?] controlled a manipulator to perform reaching tasks while avoiding a dynamic obstacle. [?] used a nonlinear-MPC to control N robots for cooperative manipulation of objects whilst avoiding obstacles. [?] performed obstacle avoidance of mobile manipulators using an MPC.

Researchers in [?] used MPC for trajectory tracking and obstacle avoidance of mobile robots. In [?], an MPC was used for controlling a robot manipulator that avoids fixed and human obstacles. In [?], researchers used MPC for dynamic target tracking and obstacle avoidance for a person-following robot. Researchers in [?] used MPC for continuous mobile manipulation and collision avoidance. Other researchers used MPC for controlling an autonomous vehicle and performing obstacle avoidance [?, ?]. Researchers in [?] used a MPC for controlling a dog robot for helping vision-impaired people to navigate around obstacles.

In general, MPCs require a system model for predicting the state of the system over a finite horizon [?]. This imposes that the system to be controlled and its dynamic model must be known prior to control. Furthermore, nonlinear models can burden the controller with heavy computational requirements [?]. This can be challenging in robotics as some robots, such as soft and continuum robots, are not straightforward to model accurately, while others might change, adapt, morph, transform, or evolve, such as robots that change their kinematic structure or dynamic properties during movement or robots that get accidentally damaged or bent [?].

On the other hand, KMF control methods can overcome these challenges as these control methods do not require any prior knowledge of the robot's kinematic or dynamic models. Kinematic-model-free controllers work by gathering information through short exploratory actuations, building a local linear model that approximates the robot's local kinematic and dynamic behaviour, and estimating an actuation signal that moves the robot towards a given reference target. Previously, KMF was demonstrated to perform position control of a robot manipulator capable of performing target-reaching tasks [?, ?]. The controller was then experimented on a pneumatically-actuated soft continuum robot [?] for space applications. Then, a pose (position and orientation) controller was presented that controls a robot's end-effector pose using locally weighed dual quaternions [?]. Finally, the degrees-of-freedom was scaled up to be controlled by tracking and controlling multiple points along the robot's kinematic chain [?].

Although model-free predictive control methods were previously studied [?, ?, ?], online kinematic-model-free predictive control for robot manipulators is an unexplored area of research up to our knowledge.

4.2 KMF-PC Control

This section formulates the KMF-PC controller. The nonlinear model of a rigid-body robot manipulator can be obtained using the following general dynamic equation:

$$M(q(t)\ddot{q} + c(q, \dot{q}) + g(q(t)) + \tau_f(t) = \tau(t) \quad (4.1)$$

where q are the joint angles, $\dot{q}$ are the joint velocities, $\ddot{q}$ are the joint accelerations, M is the inertia matrix, c is the Coriolis vector, g is the gravity vector, τ_f is the friction torque, and τ is the applied torque.

For a 2-DOF robot manipulator, let $\hat{p}$ be the desired actuation primitive whose parameters

$\tau_p(\hat{p})$ are to be estimated to actuate the end effector towards the target position:

$$b_1 = \begin{bmatrix} \tau_p^1(\hat{p}) \\ \tau_p^2(\hat{p}) \end{bmatrix} \quad (4.2)$$

Following the kinematic-model-free approach, b_1 can be approximated as a linear combination of actuation primitives as follows:

$$A_1 w = b_1 \quad (4.3)$$

where $w = [w_1, w_2, \dots, w_k]^T$ is the vector of unknown weights and A_1 , the actuation matrix, is defined as follows:

$$A_1 = \begin{bmatrix} \tau_p^1(p_1) & \tau_p^1(p_2) & \dots & \tau_p^1(p_k) \\ \tau_p^2(p_1) & \tau_p^2(p_2) & \dots & \tau_p^2(p_k) \end{bmatrix} \quad (4.4)$$

During exploration, the controller applies pseudo-random actuation primitives with the following curve:

$$\tau(t) = \begin{cases} \tau_p & t \in [t_0, t_0 + \frac{d_p}{2}) \\ 0 & otherwise \end{cases} \quad (4.5)$$

The resulting end-effector displacements are recorded into the observation matrix:

$$A_2 = \begin{bmatrix} \Delta x(p_1) & \Delta x(p_2) & \dots & \Delta x(p_k) \\ \Delta y(p_1) & \Delta y(p_2) & \dots & \Delta y(p_k) \\ \Delta z(p_1) & \Delta z(p_2) & \dots & \Delta z(p_k) \end{bmatrix} \quad (4.6)$$

where $\Delta i(p_j)$ is the displacement effect on the end effector along the i -axis as a result of exerting actuation p_j . In the planar case, $\Delta z(p_j) = 0$. Let the desired effect be expressed as

follows:

$$b_2 = \begin{bmatrix} \Delta x(\hat{p}) \\ \Delta y(\hat{p}) \\ \Delta z(\hat{p}) \end{bmatrix} \quad (4.7)$$

The desired effect, b_2 , can be expressed as a linear combination of the kNN observed effects, A_2 , as follows:

$$A_2 w = b_2 \quad (4.8)$$

Note that many solutions might exist for w since A_2 might not be full rank. To resolve this issue, least squares regression is used to find the smallest squared vector b_2 that is a solution to the equation. This calculated value can then be used in equation (4.3) to find the desired actuation primitive that will actuate the end effector towards the desired target position. After every actuation, the actual and desired end-effector positions are compared. In the case of significant discrepancy, another exploratory phase is triggered.

The system model of linear MPC in state space form is as follows:

$$x(k+1) = Ax(k) + Bu(k) \quad (4.9)$$

$$y(k) = Cx(k) + Du(k) \quad (4.10)$$

where matrices A , B , C , and D can vary with time and:

- k : current control time index
- x : system state variables
- u : manipulable inputs
- y : system outputs

In general, a system model is used to predict the future states of a dynamic system by the MPC to generate a control signal that minimizes some cost function over a predefined prediction horizon [?]. The cost function optimizes the control variables to achieve some desirable reference which results in a sequence of N optimal actuations, of which the MPC only applies the first [?]. This process is then repeated after shifting the horizon forward. If the system model is linear, analytical solutions to the MPC can be used for extremely fast computation. In the case of rigid-body robots, the dynamic model shown in (A.18) is integrated in time, and becomes extremely non-linear as more joints are added, making MPC challenging. The problem is often simplified by linearizing the system dynamics, however this requires an accurate model of the system; something that is not possible for unknown robots.

To make use of linear MPCs, a state space formulation for KMF control is required. The actuation equation in KMF control can be premultiplied by the pseudoinverse of A_1 , the actuation matrix:

$$w = A_1^\dagger b_1. \quad (4.11)$$

This can then be substituted into the observation equation to yield the following:

$$b_2 = A_2 A_1^\dagger b_1 \quad (4.12)$$

Since $\Delta x(k+1) = b_2$ and $u(k) = b_1$, we can easily notice that:

$$\Delta x(k+1) = Bu(k) \quad (4.13)$$

where

$$B = A_2 A_1^\dagger \quad (4.14)$$

The cost function is defined as follows:

$$l(x, u) = \|x_{des} - x_u\| + \|u\| \quad (4.15)$$

where x_{des} is the desired state of the system. Formally, the optimization problem is expressed as follows:

$$\min_u \quad J_N(x_0, u) = \sum_{k=0}^{N-1} l(x(k), u(k)) \quad (4.16)$$

$$s.t. \quad \Delta x(k+1) = Bu(k) \quad (4.17)$$

$$x(0) = x_0 \quad (4.18)$$

$$\|u\| \leq u_{max} \quad (4.19)$$

$$(x_1(k) - a)^2 + (x_2(k) - b)^2 \geq r^2 \quad (4.20)$$

where x_0 is the initial state of the system, u_{max} is the maximum torque to be applied, $x_i(k)$ is the i^{th} system state variable, and $\{a, b, r\}$ are the circular constraint parameters of the obstacle. Since the local linear model built is volatile and is only used for the next actuation, a linear MPC suffices. As with all KMF controllers, stability and convergence are not guaranteed due to their exploratory nature.

4.3 Results

In this section, simulation and physical experimentation results are presented and discussed. In the first experiment, a simple reaching task on a simulated 2-DOF robot and a physical 2-DOF robot was performed using classical KMF controller and the proposed KMF-PC. In the second experiment, reaching tasks on a 2-DOF, 3-DOF, and 4-DOF simulated robot as well as the 2-DOF physical robot was performed with one and two circular obstacles. In the third experiment, the adaptability of the KMF-PC is demonstrated by modifying the 2-DOF robot's kinematic structure midway. All the robots are planar, restricting the motion to the horizontal plane, to neglect the effects of gravity.

The simulation robots are simulated on MATLAB using Peter Corke’s Robotics Toolbox [?]. The KMF-PC makes use of CasADi [?]. End-effector position was easily obtained using toolbox functions. On the other hand, the physical robot is comprised of aluminium links and Hebi X-Series actuators, (X5-9) capable of delivering 9 Nm of continuous torque, which are controlled using torque signals [?]. The end effector of the robot is tracked using a calibrated webcam and a green marker. Direct pixel coordinates is used for determining the end-effector position for physical experiments. Virtual obstacles are used to emulate a circular obstacle that is to be avoided.

4.3.1 Reaching Using KMF and KMF-PC

In this experiment, regular KMF and the proposed KMF-PC are used to perform a simple reaching task on a simulated and a physical 2-DOF robot in the absence of any obstacle. Figure 4.1 shows both the simulated and physical experimentation results of running both controllers.

It is evident that, in the absence of any obstacle, both controllers performed similarly and were able to reach the target position in less than 30 steps.

4.3.2 Obstacle Avoidance Using KMF-PC

In the second experiment, obstacle avoidance is performed on a 2-DOF, 3-DOF, and 4-DOF simulated robot and the 2-DOF physical robot. The results of simulations are summarized in Figure 4.2. The KMF-PC is capable of avoiding obstacles along the reaching path. Despite the increase in complexity when an extra DOF is added, the controller successfully performs the reaching task whilst avoiding the obstacles (2-DOF and 3-DOF). It is also noted, due to the exploratory nature of the KMF approach, should the controller trigger an exploratory phase near any obstacle, this might cause collision with the obstacle (evident in the 4-DOF case). As such, it is recommended to inflate the obstacles to avoid any collision due to exploratory behaviour.

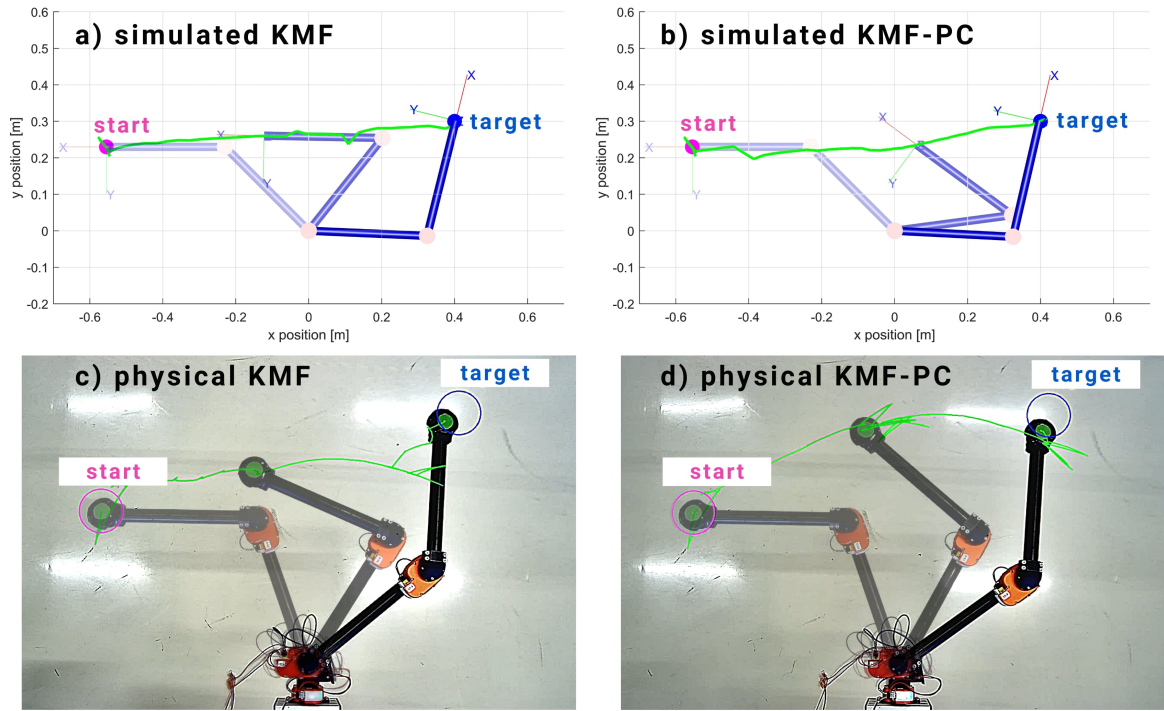


Figure 4.1: Simulated and physical experimentation of KMF control and the proposed KMF-PC in the absence of any obstacle. Comparing simulation results (a) and (b), both controllers performed similar, reaching the target in a semi-straight line. As for the physical experiments, we notice that the KMF-PC took a slightly more curved path to reach the target position. Nonetheless, both KMF and KMF-PC performed similarly well, achieving the reaching task within 30 steps.

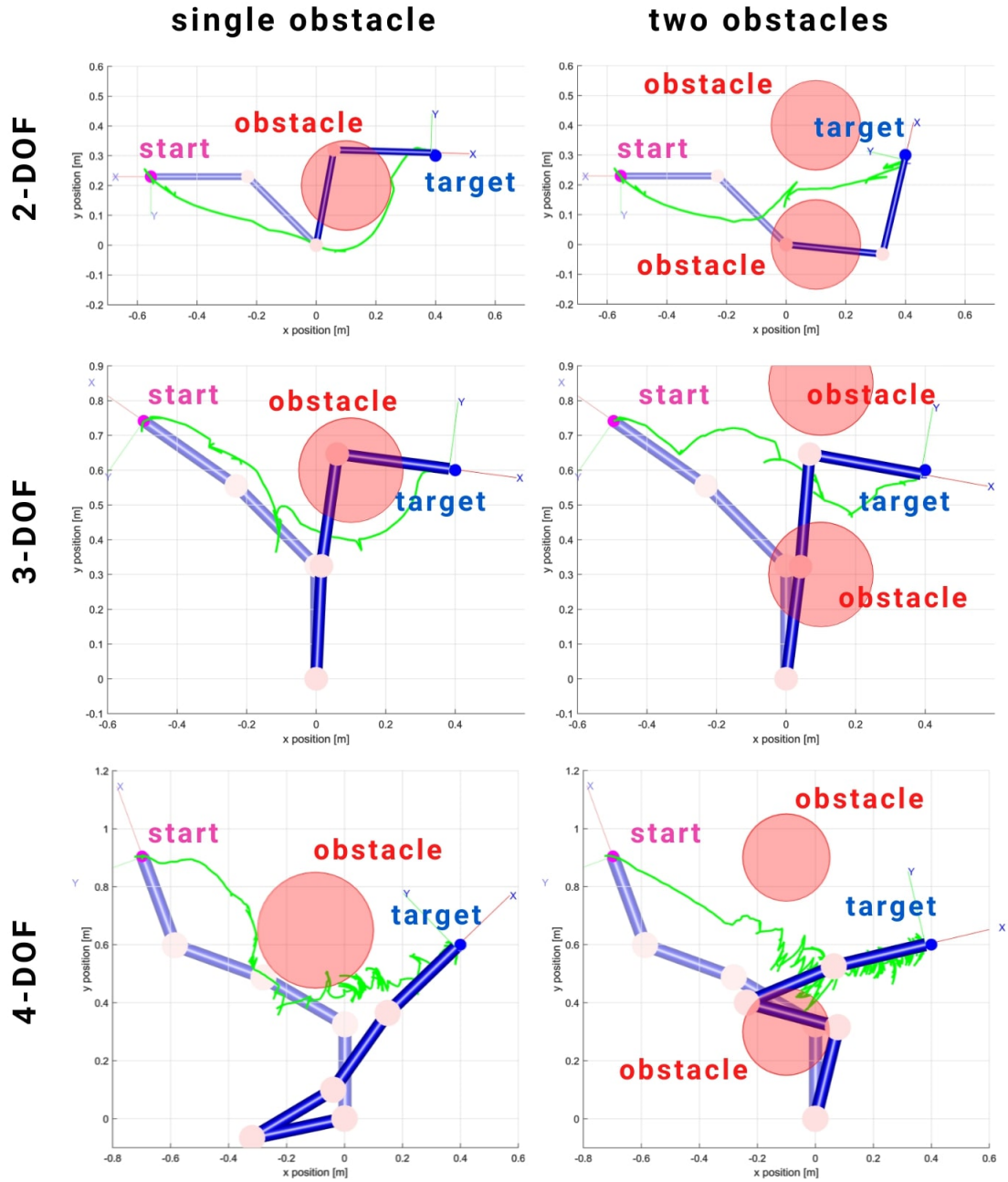


Figure 4.2: Simulation results of simultaneous reaching and obstacle avoidance using KMF-PC on simulated 2-DOF, 3-DOF, and 4-DOF manipulator robots. It is noted in the 4-DOF case, where the complexity is relatively higher, more exploratory motion is triggered which caused the end effector to penetrate the virtual obstacle.

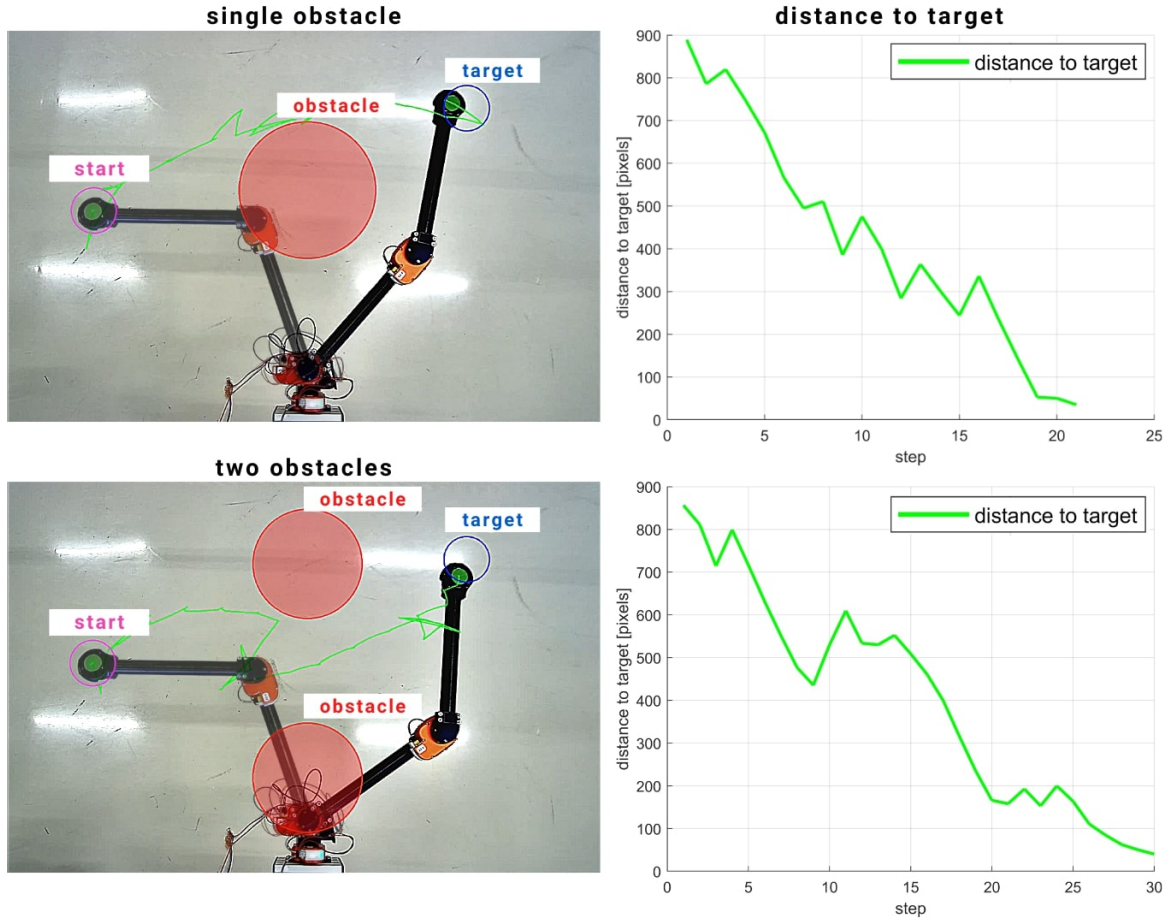


Figure 4.3: Physical experiment results of KMF-PC reaching and obstacle avoidance. In both cases, the end effector reached close to the obstacle. As such, obstacle inflation is recommended. In the single obstacle scenario, the robot reached the target in 21 steps. In the two obstacles scenario, it reached the target in 30 steps.

As for the physical experiments, the results are shown in Figure 4.3. The KMF-PC controller successfully performed the reaching task whilst avoiding the obstacles. In both cases, single and two obstacles, the KMF-PC controller brought the instances where the robot came close to the obstacle(s). As such, inflating the obstacles is recommended.

When one obstacle was presented, the controller took 21 steps to reach the target end-effector position. When two obstacles were in the way, the controller took 30 steps to reach the target.

4.3.3 Adaptability of KMF-PC

In the third experiment, the adaptability of the proposed controller is demonstrated through a series of tests in which the kinematic structure of the physical robot is modified midway

the reaching task. The results are summarized in Figure 4.4. First, the robot’s second link is shortened by moving the marker down the link. Second, the link is extended by adding a rigid cardboard to the last link. Third, the end effector is extended using a soft link. Fourth, the end effector is offset.

The end effector is altered by pausing the execution temporarily and manually modifying the end effector. After the modification, the execution is resumed. In all cases, when the execution is resumed, the controller goes into an exploratory phase which is triggered due to the discrepancy measured between the predicted and actual state of the end effector. The controller, then after, resumes moving toward the target position. In all cases, the robot managed to reach the target within 50 steps.

4.4 Discussion

In this chapter, a KMF controller is presented that is capable of obstacle avoidance, using an MPC approach, without any prior knowledge of the robot’s model. KMF controllers make no assumption about the robot’s structure nor the environment. The formulation of the KMF-PC was derived in this paper, showing how obstacle avoidance can be achieved without prior knowledge of the robot’s dynamics or kinematics. Using this approach to allow MPC to be applied to unknown robots enables a greater level of control to be achieved, with minimal computational complexity. Both simulated and physical experiments were conducted. Running the KMF-PC controller in the absence of any obstacle behaved similar to the classical KMF controller. Single and double obstacle avoidance was tested in simulation using 2-DOF, 3-DOF, and 4-DOF robots and on a physical 2-DOF robot. It is found that due to the exploratory nature of KMF control, any exploratory actuations trigger near any obstacle might cause the robot’s end effector to penetrate or collide with the obstacle. As such, obstacle inflation is recommended. Furthermore, the adaptability of the KMF-PC was demonstrated through a series of tests where the end effector was modified halfway during execution. In all experiments, the robot managed to perform simultaneous reaching and obstacle avoidance, reaching the target

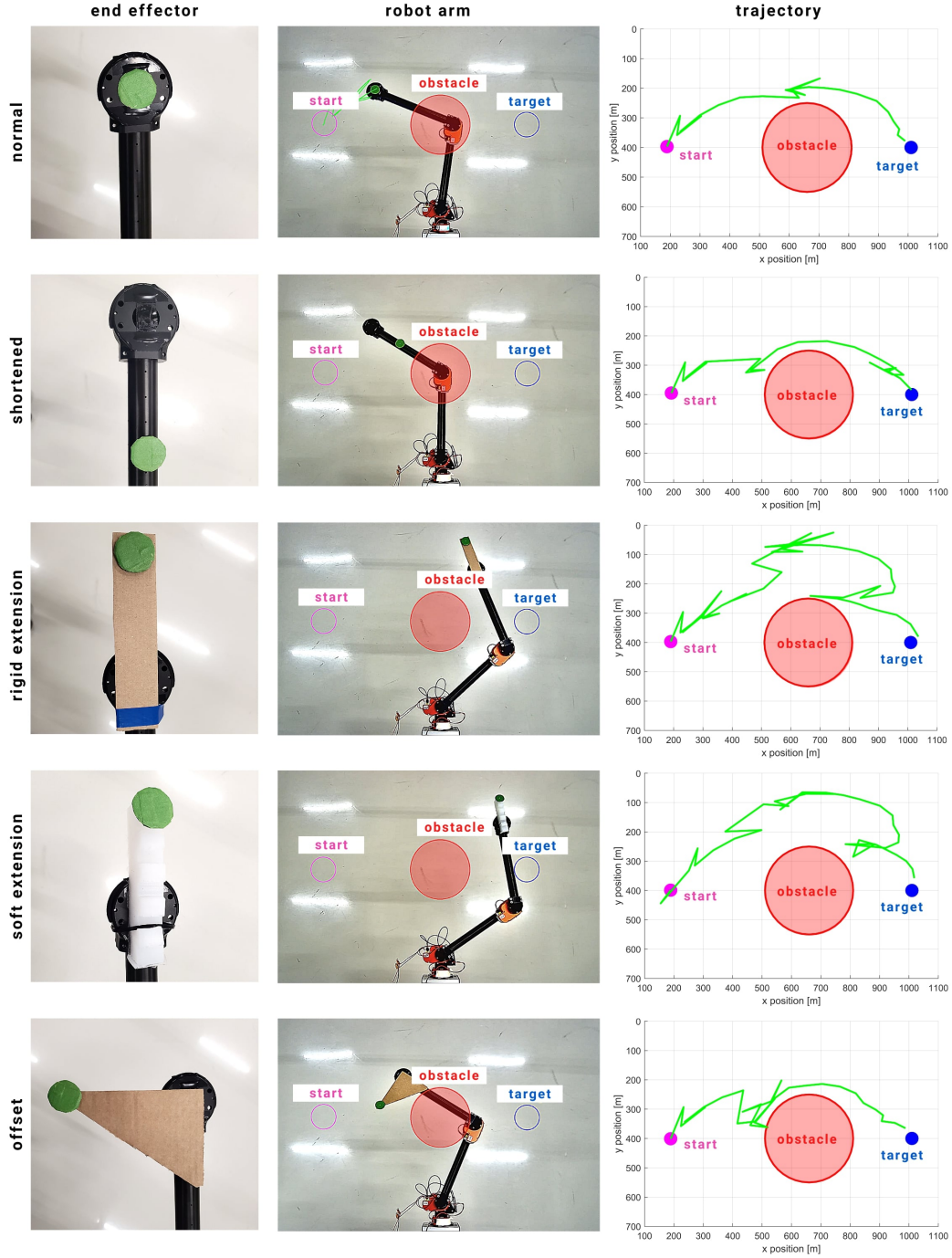


Figure 4.4: Physical experiment results of reaching and obstacle avoidance whilst modifying the end effector during execution. When the end effector is altered, the controller triggers an exploration phase and resumes reaching and obstacle avoidance right after. The modifications include shortening the end effector, rigid and soft extensions, and offsetting the end effector.

position in less than 50 steps. At the current state, KMF should be applied to robot control problems where flexibility and adaptability is favoured over accuracy and speed. Future work includes obstacle-aware exploratory actuations that take into account the obstacles to avoid collisions.

References

- [1] Abdul Afram and Farrokh Janabi-Sharifi. Theory and applications of hvac control systems—a review of model predictive control (mpc). *Building and Environment*, 72:343–355, 2014.
- [2] Ahmad AlAttar, Francesco Cursi, and Petar Kormushev. Kinematic-model-free redundancy resolution using multi-point tracking and control for robot manipulation. *Applied Sciences*, 11(11):4746, 2021.
- [3] Ahmad AlAttar and Petar Kormushev. Kinematic-model-free orientation control for robot manipulation using locally weighted dual quaternions. *Robotics*, 9(4):76, 2020.
- [4] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.
- [5] Avijit Ashe and K Madhava Krishna. Dynamic target tracking & collision avoidance behaviour of person following robot using model predictive control. In *2020 24th International Conference on System Theory, Control and Computing (ICSTCC)*, pages 660–666. IEEE, 2020.
- [6] Peter Corke. *Robotics, vision and control: fundamental algorithms in MATLAB® second, completely revised*, volume 118. Springer, 2017.
- [7] Phuong Nam Dao, Hong Quang Nguyen, Thanh Long Nguyen, and Xuan Sinh Mai. Finite horizon robust nonlinear model predictive control for wheeled mobile robots. *Mathematical Problems in Engineering*, 2021, 2021.
- [8] Michael G Forbes, Rohit S Patwardhan, Hamza Hamadah, and R Bhushan Gopaluni. Model predictive control in industry: Challenges and opportunities. *IFAC-PapersOnLine*, 48(8):531–538, 2015.
- [9] Alberto Franco and Vitor Santos. Short-term path planning with multiple moving obstacle avoidance based on adaptive mpc. In *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 1–7. IEEE, 2019.

- [10] Chase Frazelle, Ian Walker, Ahmad AlAttar, and Petar Kormushev. Kinematic-model-free control for space operations with continuum manipulators. In *2021 IEEE Aerospace Conference (50100)*, pages 1–11. IEEE, 2021.
- [11] Hebi Robotics. Hebi robotics x-series actuator, 2023.
- [12] Jie Ji, Amir Khajepour, Wael William Melek, and Yanjun Huang. Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints. *IEEE Transactions on Vehicular Technology*, 66(2):952–964, 2016.
- [13] Mina Kamel, Javier Alonso-Mora, Roland Siegwart, and Juan Nieto. Robust collision avoidance for multiple micro aerial vehicles using nonlinear model predictive control. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 236–243. IEEE, 2017.
- [14] Mahyar Khosravi, Masoume Amirbande, Davood A Khaburi, Marco Rivera, Jose Riveros, Jose Rodriguez, Abolfazl Vahedi, and Patrick Wheeler. Review of model predictive control strategies for matrix converters. *IET Power Electronics*, 12(12):3021–3032, 2019.
- [15] Petar Kormushev, Yiannis Demiris, and Darwin G Caldwell. Encoderless position control of a two-link robot manipulator. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 943–949. IEEE, 2015.
- [16] Petar Kormushev, Yiannis Demiris, and Darwin G Caldwell. Kinematic-free position control of a 2-dof planar robot arm. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5518–5525. IEEE, 2015.
- [17] Maximilian Krämer, Christoph Rösmann, Frank Hoffmann, and Torsten Bertram. Model predictive control of a collaborative manipulator considering dynamic obstacles. *Optimal Control Applications and Methods*, 41(4):1211–1232, 2020.
- [18] Hongran Li and Shigeru Yamamoto. A model-free predictive control method based on polynomial regression. In *2016 SICE International Symposium on Control Systems (ISCS)*, pages 1–6. IEEE, 2016.

- [19] Jianhua Li, Jianfeng Sun, Liqun Liu, and Jiasheng Xu. Model predictive control for the tracking of autonomous mobile robot combined with a local path planning. *Measurement and Control*, page 00202940211043070, 2021.
- [20] Wei Li and Rong Xiong. Dynamical obstacle avoidance of task-constrained mobile manipulation using model predictive control. *IEEE Access*, 7:88301–88311, 2019.
- [21] Alexandres Nikou, Christos Verginis, Shahab Heshmati-Alamdari, and Dimos V Dimarogonas. A nonlinear model predictive control scheme for cooperative manipulation with singularity and collision avoidance. In *2017 25th Mediterranean Conference on Control and Automation (MED)*, pages 707–712. IEEE, 2017.
- [22] Artemiy Oleinikov, Sanzhar Kusbayev, Almas Shintemirov, and Matteo Rubagotti. Safety-aware nonlinear model predictive control for physical human-robot interaction. *IEEE Robotics and Automation Letters*, 2021.
- [23] Johannes Pankert and Marco Hutter. Perceptive model predictive control for continuous mobile manipulation. *IEEE Robotics and Automation Letters*, 5(4):6177–6184, 2020.
- [24] Rien Quirynen, Karl Berntorp, Karthik Kambam, and Stefano Di Cairano. Integrated obstacle detection and avoidance in motion planning and predictive control of autonomous vehicles. In *2020 American Control Conference (ACC)*, pages 1203–1208. IEEE, 2020.
- [25] Roni Permana Saputra, Nemanja Rakicevic, Digby Chappell, Ke Wang, and Petar Kormushev. Hierarchical decomposed-objective model predictive control for autonomous casualty extraction. *IEEE Access*, 9:39656–39679, 2021.
- [26] Yuho Song and Kunsu Huh. Driving and steering collision avoidance system of autonomous vehicle with model predictive control based on non-convex optimization. *Advances in Mechanical Engineering*, 13(6):16878140211027669, 2021.
- [27] Anders Stenman. Model-free predictive control. In *Proceedings of the 38th IEEE Conference on Decision and Control (Cat. No. 99CH36304)*, volume 4, pages 3712–3717. IEEE, 1999.

- [28] Liyang Wang, Jinxin Zhao, and Liangjun Zhang. Navdog: robotic navigation guide dog via model predictive control and human-robot modeling. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, pages 815–818, 2021.
- [29] Shigeru Yamamoto. A new model-free predictive control method using input and output data. In *Advanced Materials Research*, volume 1042, pages 182–187. Trans Tech Publ, 2014.
- [30] Mario Zanon, Andrea Boccia, Vryan Gil S Palma, Sonja Parenti, and Ilaria Xausa. Direct optimal control and model predictive control. In *Optimal Control: Novel Directions and Applications*, pages 263–382. Springer, 2017.

Chapter 5

Continuous Kinematic-Model-Free Control

Conventional controllers require kinematic and dynamic (hereon referred to as kinodynamic) models of robots to perform position control, velocity control, gravity compensation, etc. [?]. Robots' kinematic structures and dynamic properties must be static for these models to remain relevant. In situations and environments where the robot is soft, morphable, malleable, transformable, or damaged, conventional controllers cannot control the robot directly [?]. In addition, should the robot's tilt with respect to the ground change, these controllers might not be able to compensate for gravity directly, causing the robot to behave unexpectedly [?]. Throughout the robotics research history, these model-based approaches have remained the prevalent control method for over half a century [?] due to their effectiveness in addressing very specific application domains where the aforementioned models remain static.

Over the past decade, several control approaches that are based on model learning and model approximation have emerged. Nevertheless, true model-free control remains a challenging problem. Previously, the encoderless controller [?] and kinematic-free controller [?] have successfully controlled a robot manipulator's end effector in task space without requiring any prior kinematic model of the robot. The controller also controlled a soft continuum robot's end effector position [?]. These controllers only managed to control the position of a robot's

end effector. On the other hand, the kinematic-model-free orientation controller extended the state-of-the-art and successfully performed simultaneous position and orientation control of a manipulator's end effector in task space [?]. This kinematic-model-free multi-point controller improved the controller's performance when applied to hyper-redundant robots by tracking and controlling multiple points along the robot's kinematic chain [?]. Despite the success of these controllers in controlling a robot without any prior model, they are discrete controllers and actuate the robot in discrete steps. They also do not take into account the effects of gravity. Whereas practical applications for robot manipulation require smoother and continuous motion.

Continuous control and gravity compensation are challenging to implement using model-free approaches. Up to our knowledge, no model-free approach exists in the literature that can perform gravity compensation with an adaptive model-learning controller. This chapter presents a velocity-based continuous kinematic-model-free controller that is capable of controlling a robot manipulator's end-effector position while performing gravity compensation without requiring any prior model of the robot.

The controller presented in this chapter extends the state-of-the-art discrete model-free controllers to perform continuous control which performs kinematic-model-free gravity compensation during control. Furthermore, the control is robust to kinematic or dynamic changes in the robot or environment which was validated through experiments.

5.1 Related Work

Since the emergence of the very first robots until the more recent collaborative robots, robots have been controlled using conventional methods that rely on prior knowledge of the robot's kinematic and dynamic models [?]. Such controllers impose strict limitations on the robot's structure and environment, assuming static robot configuration and dynamic parameters [?]. Recently, model-learning controllers have received increasing interest from researchers. Some researchers attempted to approximate a model for robots prior to manipulation [?, ?] while

others tried to identify the parameters of dynamically undefined robots prior to control [?]. Machine learning [?] and reinforcement learning, in particular, [?] were used by some researchers to control robots. However, all of these control methods require joint position feedback during control.

Kinematic-model-free controllers can control a robot without prior knowledge of the robot's kinematic or dynamic models. These controllers work by generating local linear models that capture the local kino dynamics of the robot, which are then used to actuate the robot towards the target configuration. Since these controllers require no joint angle information, no assumptions are made about the robot's kinematic structure or dynamic parameters; i.e. the robot is treated as a black box. In particular, the Encoderless controller [?] and the Kinematic-Free [?] controller were used to control the position of a 2-DOF planar robot's end effector, performing simple reaching tasks. The kinematic-model-free controller was also applied to a continuum soft robot to manipulate the end-effector position in [?] and [?]. The kinematic-model-free position controller was then extended to control the position and orientation of a 3-DOF robot in 3D space simultaneously using locally weighted dual quaternions [?]. All of these kinematic-model-free controllers are, however, discrete.

Robots have been controlled using both discrete [?] and continuous control [?] methods. Robot dynamics generally require continuous control for critical tasks such as gravity compensation [?]. However, high-level controllers that are less critical could be implemented in discrete time [?]. Researchers also used hybrid discrete-continuous control methods to control robots [?, ?]. Discrete controllers have also been converted to equivalent continuous controllers, and vice versa [?].

Applying precise actuator power to balance out the joint torques caused by the weights of the links can be challenging [?]. Robot gravity compensation can also be performed passively using counterweights and springs [?]. Robotic systems such as unmanned aerial vehicles (UAVs) and autonomous underwater vehicles (AUVs) cannot compensate for gravity mechanically and,

thus, require active compensation [?, ?]. Gravity compensation requires an accurate robot model for control [?] where some researchers used PD control to achieve gravity compensation [?, ?, ?]. Researchers in [?] developed a force-based control framework with virtual gravity compensation. Other researchers have looked into gravity compensation learning and set-point regulation using iterative learning [?] and model learning [?]. All these methods require joint angle information during control, limiting their application to rigid robots with accurate joint position feedback.

5.2 Continuous KMF Control

A rigid-body robotic system can be obtained using the following general dynamic equation:

$$M(q(t))\ddot{q} + c(q, \dot{q}) + g(q(t)) + \tau_{ext} = \tau(t) \quad (5.1)$$

where q are the joint angles, $\dot{q}$ are the joint velocities, $\ddot{q}$ are the joint accelerations, M is the inertia matrix, c is the Coriolis vector, g is the gravity vector, τ_{ext} is the external torque, and τ is the applied torque. The fundamental assumption made behind the kinematic-model-free control approach is that measured end-effector observation is directly proportional to the control signal sent to the robot actuators within the local region around the end effector. The equation above can be written more simply as:

$$\tau = M\ddot{q} + c\dot{q} + g + \tau_{ext} \quad (5.2)$$

Rearranging the terms and assuming negligible Coriolis and centrifugal torques and no external forces:

$$\tau - g = M\ddot{q} \quad (5.3)$$

Since $\dot{q} = J^{-1}\dot{x}$, we have:

$$\ddot{x} = JM^{-1}(\tau - g) \quad (5.4)$$

Let δ_p be one control step where $\tau = k_1\tau_p$ with a rectangular pulse and $g = k_2\tau'_p$ with $\tau'_p = \tau_p(t - \delta_p)$ and both k_1 and k_2 are constants:

$$\Delta v = JM^{-1}k\Delta\tau_p \quad (5.5)$$

where $k = k_1 = k_2$. Combining all the constant terms into C :

$$\Delta v = C\Delta\tau_p \quad (5.6)$$

It is shown that during a small quasi-static movement, the change in the robot's end-effector velocity is directly proportional to the change in the robot's joint torques:

$$\Delta v \propto \Delta\tau_p \quad (5.7)$$

A high-level flowchart of the continuous kinematic-model-free controller is shown in Figure 5.1. The controller performs both exploration and exploitation. At the core of the idea lies the assumption that it is possible to obtain the local kinodynamic information of the robot by generating pseudo-random actuation control signals and observing their corresponding effects on the end effector.

Exploration is triggered whenever the robot behaves differently from what is expected or when there is no sufficient data, which is the case at time $t = 0$. During exploration, pseudo-random actuation control signals are sent to robot actuators and their corresponding effects on the end effector are observed. The actuation primitives signal sent to each joint actuator in the robot is defined as follows:

$$\tau(t) = \tau(t - \delta_p) + \Delta\tau(t) \quad (5.8)$$

where $\tau(t - \delta_p)$ is the previous actuation sent, $\tau(t - \delta_p) = 0$ for $t = 0$. Considering previous actuation primitives, gravity compensation is achieved, unlike the previous work performed in

$$\Delta\tau(t) = \begin{cases} \tau_p & t \in [t_0, t_0 + \delta_p) \\ 0 & \text{otherwise} \end{cases}. \quad (5.9)$$

$$\Delta v(t) = v(t) - v(t - \delta_p) \quad (5.10)$$

$$\Delta\tau(t) = \tau(t) - \tau(t - \delta_p) \quad (5.11)$$

relative end-effector velocities are stored in matrices:

$$A_1 = \begin{bmatrix} \Delta\tau_p^1(p_1) & \Delta\tau_p^1(p_2) & \cdots & \Delta\tau_p^1(p_k) \\ \Delta\tau_p^2(p_1) & \Delta\tau_p^2(p_2) & \cdots & \Delta\tau_p^2(p_k) \end{bmatrix} \quad (5.12)$$

$$A_2 = \begin{bmatrix} \Delta v_x(p_1) & \Delta v_x(p_2) & \cdots & \Delta v_x(p_k) \\ \Delta v_y(p_1) & \Delta v_y(p_2) & \cdots & \Delta v_y(p_k) \end{bmatrix} \quad (5.13)$$

where $\Delta\tau_p^j(\hat{p})$ is the magnitude of the relative actuation primitive of actuator j , $\Delta v_x(p_1)$ is the relative end-effector velocity in the x direction, $\Delta v_y(p_1)$ is the relative end-effector velocity in the y direction, and k is the k -nearest neighbour (k -NN) primitives and observations recorded relative to the end effector's current position. At the start of each control cycle, the desired relative end-effector velocity is calculated as follows:

$$b_2 = \Delta v_d = v_t - v_c = \frac{p_t - p_c}{\delta_t} - v_c \quad (5.14)$$

where v_t is the target end-effector velocity, v_c is current end-effector velocity, p_t is the target end-effector position, p_c is the current end-effector position, and δ_t is the desired trajectory time. In vector form:

$$b_2 = \begin{bmatrix} \Delta v_{d_x}(\hat{p}) \\ \Delta v_{d_y}(\hat{p}) \end{bmatrix} \quad (5.15)$$

Alternatively, intermediate targets along the path can be selected. To actuate the end effector with the desired velocity, Δv_d , the corresponding actuation primitive must be calculated:

$$b_1 = \begin{bmatrix} \Delta\tau_p^1(\hat{p}) \\ \Delta\tau_p^2(\hat{p}) \end{bmatrix} \quad (5.16)$$

The fundamental idea is to represent $\hat{p}$ as a linear combination of k -NN actuation primitives. By using a set of unknown weights, $w = [w_1, w_2, \dots, w_k]^T$, a linear combination of the k -NN

primitives can be expressed as:

$$A_1 w = b_1 \tag{5.17}$$

The coefficients, w , can be obtained by solving:

$$A_2 w = b_2 \tag{5.18}$$

Since A_2 might not be full-rank, multiple solutions for w might exist. To resolve this issue, least squares regression is used to find the smallest vector w with a valid solution. The calculated value of w is then used to find the desired relative actuation primitives, b_1 , to cause the effect, b_2 , to the robot's end effector.

5.3 Results

The continuous kinematic-model-free controller was tested on a simulated and a physical robot manipulator. Both the simulated robot and physical robot were planar 2-DoF rigid manipulators.

5.3.1 Simulated Results

The kinematic-model-free controller was tested on a simulated 2-DoF robot manipulator using MATLAB's Robotics toolbox [?]. The robot's motion was purely planar in the xy-plane, with gravity acting in the negative y-direction. Two experiments were conducted: in the first experiment, the robot had to follow a circular path while compensating for gravity; in the second test, the robot had to reach a target point while compensating for sudden changes in the direction of gravity.

Tracing a Circle

In the first test, the robot followed a circular path discretized into waypoints and passed to the continuous kinematic-model-free controller, as depicted in Figure 5.2. The robot started with an exploratory phase and followed the path by calculating actuation primitives. Another

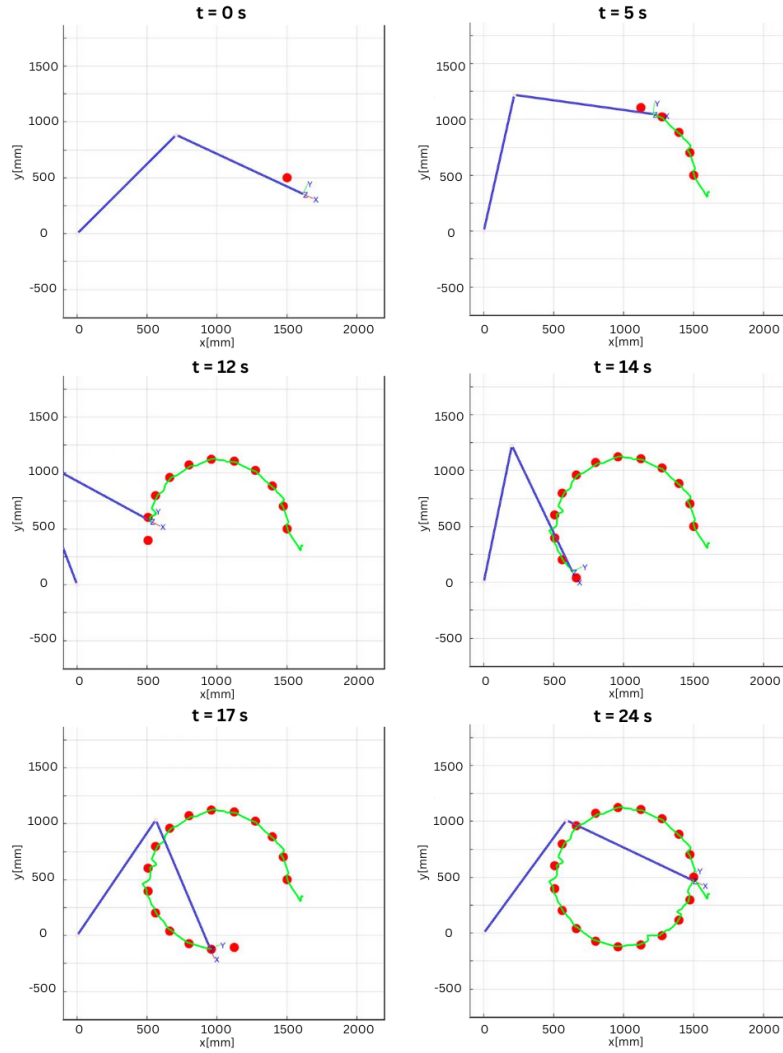


Figure 5.2: A planar 2-DoF robot manipulator following a circular path using continuous kinematic-model-free control.

exploration phase was triggered whenever a discrepancy was measured between the desired and actual end-effector velocities. It is important to note that the simulation was quasi-static and paused during controller computations that do not consider the full dynamics of a physical system (such as joint limits, joint frictions, etc.). The simulated robot managed to trace the circle and compensate for the effects of gravity during motion.

Robustness Against Environment Dynamics Change

The robustness of the controller was tested against changes in the direction of gravity. The changes tested were exaggerated; 90/180-degree changes during execution. The robot had to reach a target point in 2D space. See Figure 5.3. During execution, the direction of gravity was changed randomly for every set number of steps in a direction aligned with any of the axes, causing perpendicular shifts or complete 180-degree changes in the direction of gravity (marked by blue circles whenever a change in gravity direction occurs). The controller started with an exploration phase with gravity pointing in the negative y-direction. The controller then calculated the required actuation primitives to bring the robot's end-effector towards the target point. Whenever a discrepancy between the desired and actual end-effector velocities is measured, another exploration phase is triggered. A new exploration phase was usually triggered right after a change in gravity's direction occurred. The robot managed to reach the final target point. Once again, it is important to note that the simulation was quasi-static and paused during controller computations and did not consider physical limitations and phenomena (joint limits, joint frictions, etc.).

5.3.2 Real-world Experimentation Results

Real-world experimentation of the continuous kinematic-model-free controller was performed on a physical 2-DOF robot. The physical robot comprised of aluminium links and Hebi X-Series actuators, (X5-9) capable of delivering 9 Nm of continuous torque, which were controlled using torque signals [?]. The experimental setup is shown in Figure 5.4. Reflective markers on the robot's end effector were tracked using OptiTrack to measure the tip position in space [?].

Six experiments were conducted: 1) the robot manipulator had to reach a target point in task space while compensating for the effects of gravity; 2) the controller's adaptability was tested by altering the kinematics of the robot, which was achieved by extending the robot's end effector; 3) the controller's adaptability was tested again by altering the kinematics of the robot by offsetting the end effector; 4) the controller's robustness was tested by changing the order of

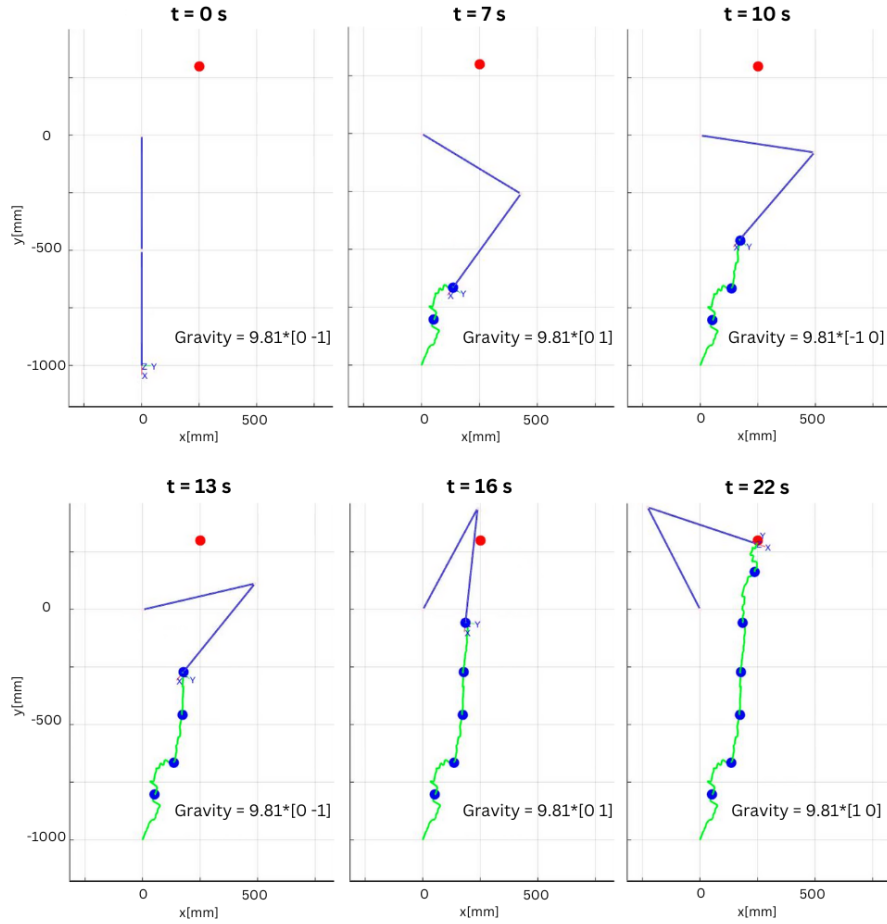


Figure 5.3: A planar 2-DoF robot manipulator that reaches a target point while compensating for gravity. The direction of gravity is changed as indicated on top of each plot every few steps. The beginning of the change is marked by a blue circle in the path taken by the robot. Despite the changes in the direction of the force of gravity, the controller manages to reach the target position marked by a red circle.

the actuators; 5) the robustness of the controller was tested by changing the robot's dynamics by adding weight to the robot; 6) the robustness of the controller was tested by changing the environment dynamics by altering the directing of gravity which was achieved by tilting the robot. The controller's performance was tested against a pre-tuned conventional PID controller that utilized the robot's kinematic model and dynamic model to compensate for gravity and robot dynamics in all the experiments. The control loop frequencies tested were 10 Hz, 20 Hz, 30 Hz, 40 Hz, and 50 Hz.

Target Reaching

In the first experiment, the robot manipulator started from an initial position of $[x_i, y_i] = [2530, 395]$ and had to reach a target position of $[x_t, y_t] = [1905, 820]$, which was obtained by moving the arm manually to the desired position and measuring the position using the motion capture system. The results of the continuous kinematic-model-free controller are shown in Figure 5.5. It was observed that the kinematic model-free controller managed to reach the target positions when the control loop frequency was 20 Hz or higher. Moreover, it can be seen that the robot reached the target position faster as the frequency was increased, shown in Figure 5.6. The same experiment was also performed using a conventional controller. The results of the conventional controller are shown in Figure 5.7. The controller brought the robot's end effector to the target position for all frequencies tested. The robot also reached the position faster for higher frequencies, shown in Figure 5.8.

Figure 5.9 compares both controllers at 40 Hz. Both controllers managed to reach the target position. The continuous kinematic-model-free controller took 1.7 seconds to reach the desired target, while the conventional controller took 5.125 seconds. The kinematic-model-free controller terminated once it reached the target within 50 mm, which was preset as the termination condition, to avoid unnecessary exploratory behaviour near the target.

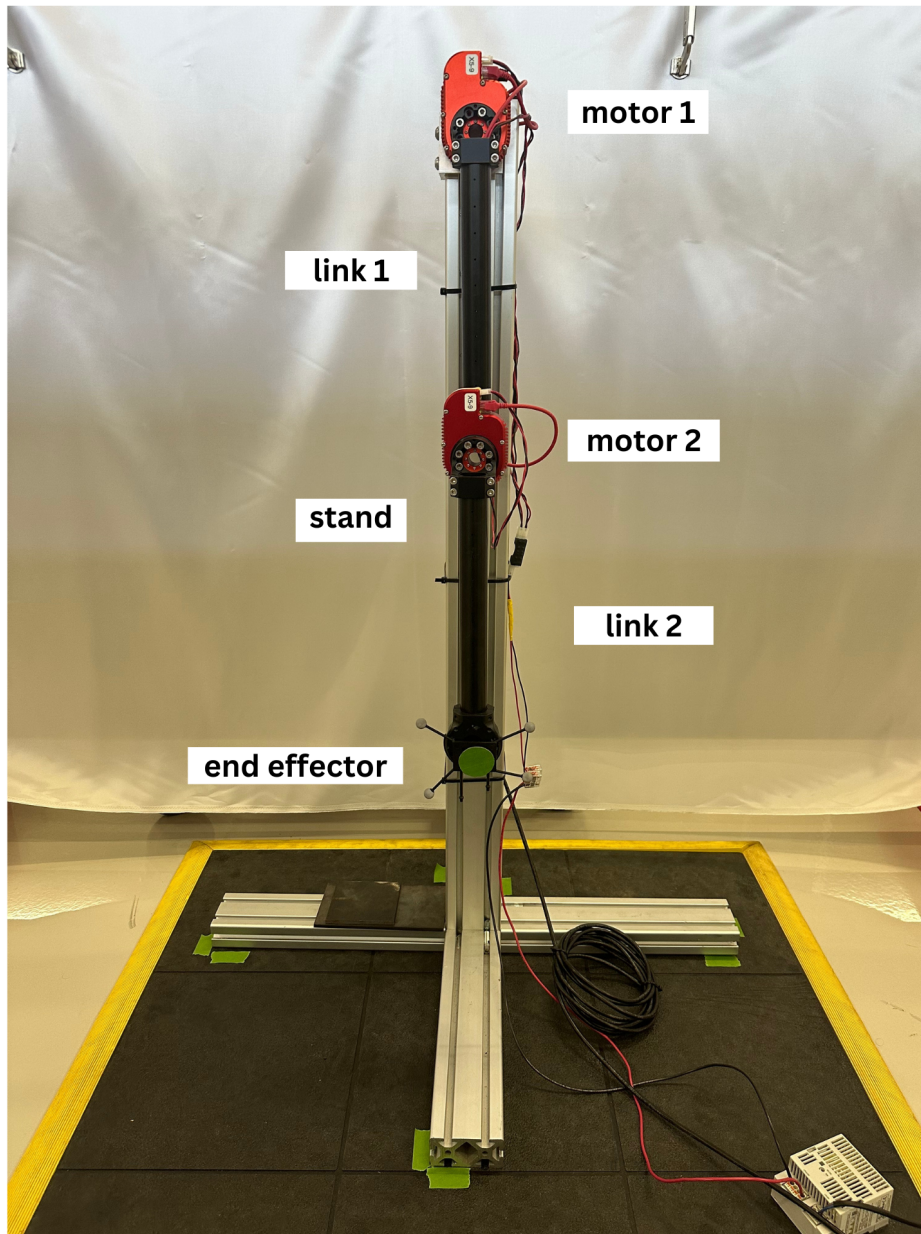


Figure 5.4: A planar 2-DoF physical robot manipulator using Hebi X5-9 actuators.

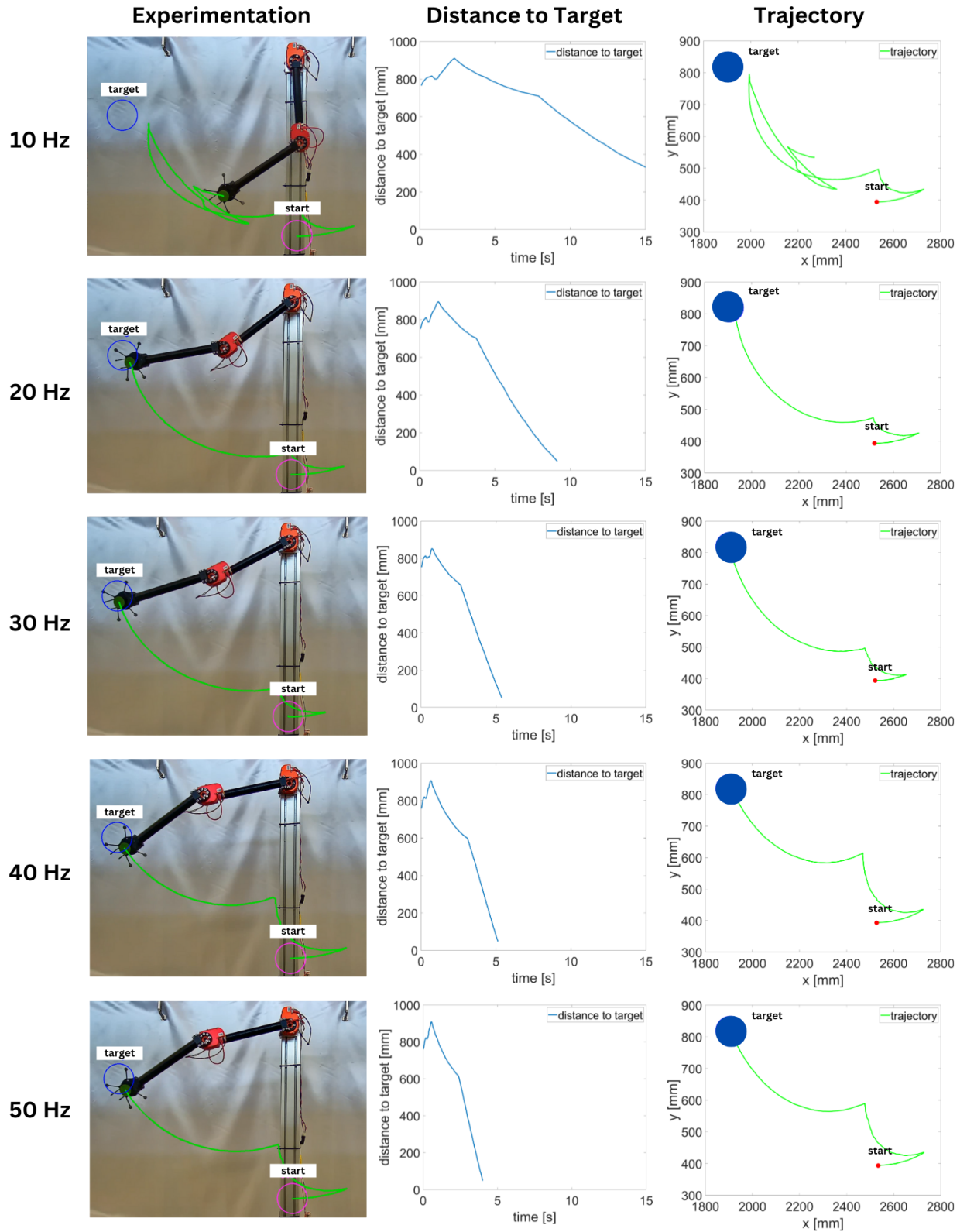


Figure 5.5: Target reaching using continuous kinematic-model-free control. The controller reached the target position with a 20 Hz or higher control loop frequency.

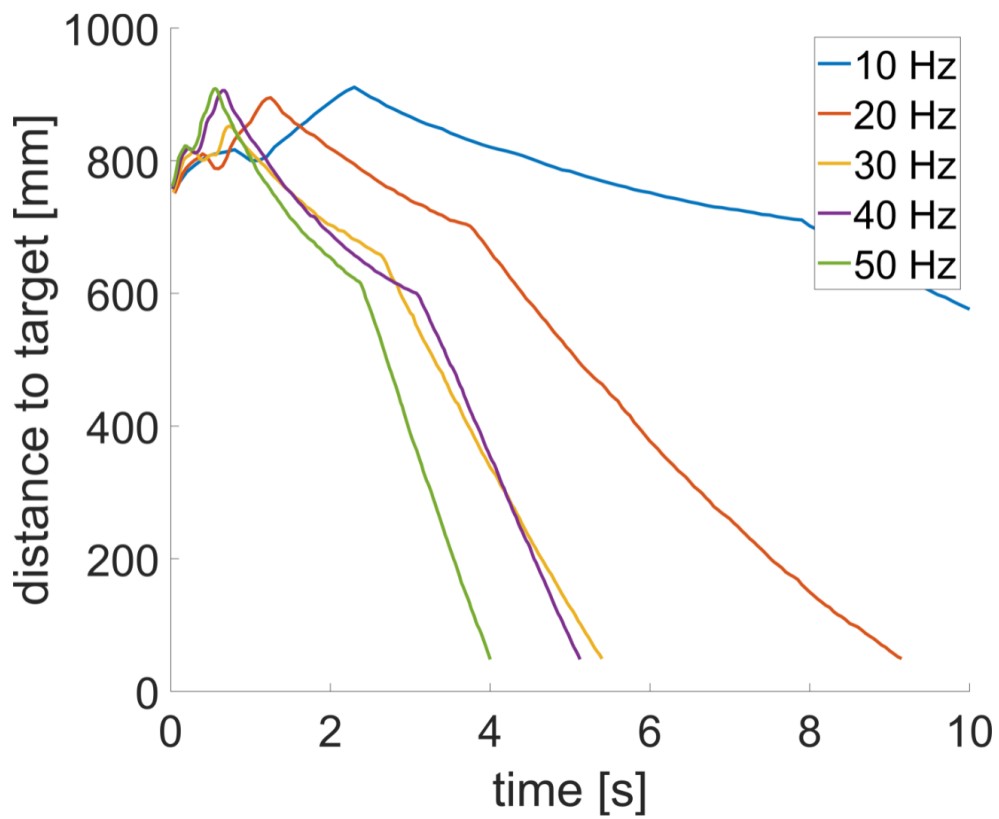


Figure 5.6: End-effector distance to target over time for various control loop frequencies of the continuous kinematic-model-free controller tested.

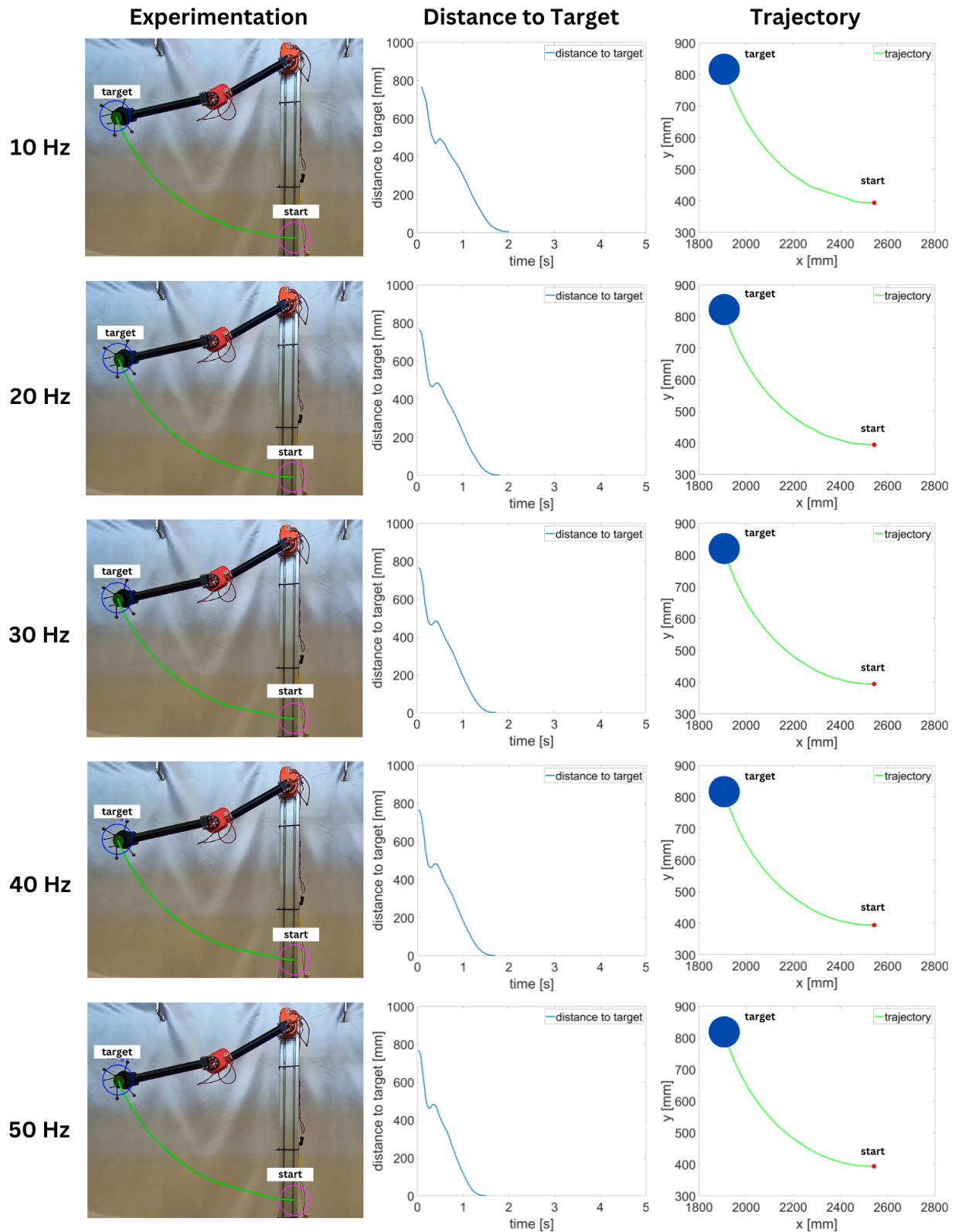


Figure 5.7: Target reaching using conventional control. The controller reached the target position with all control loop frequencies tested.

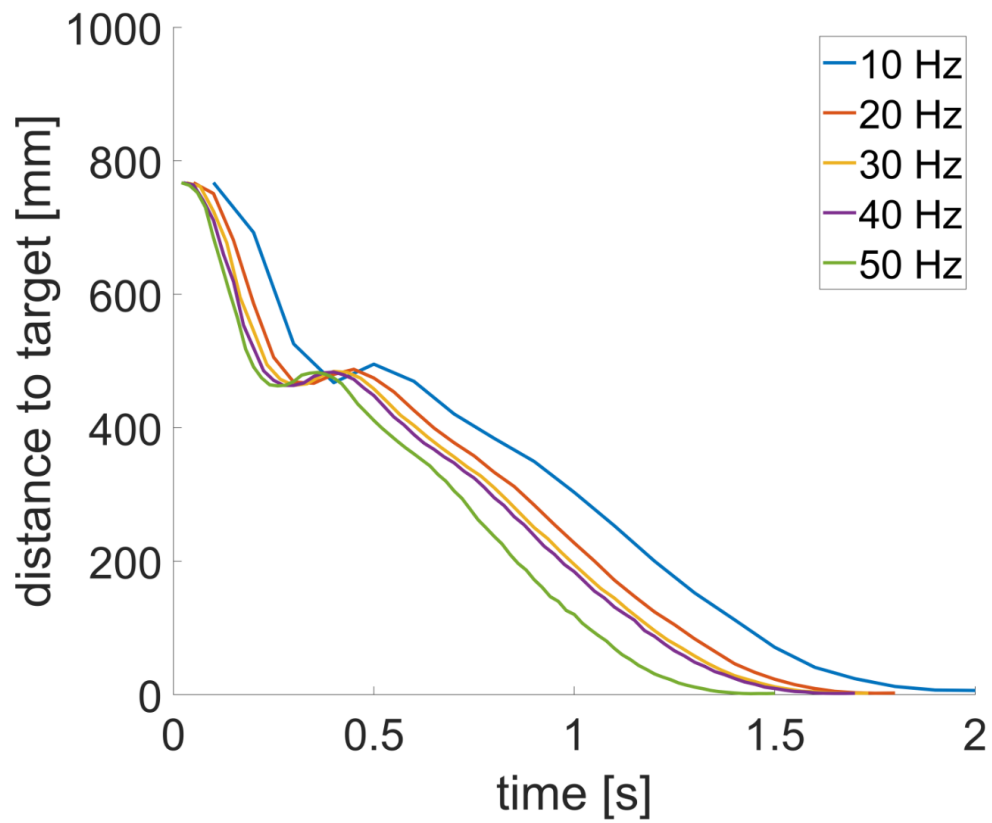


Figure 5.8: End-effector distance to target over time for various control loop frequencies of the conventional controller tested.

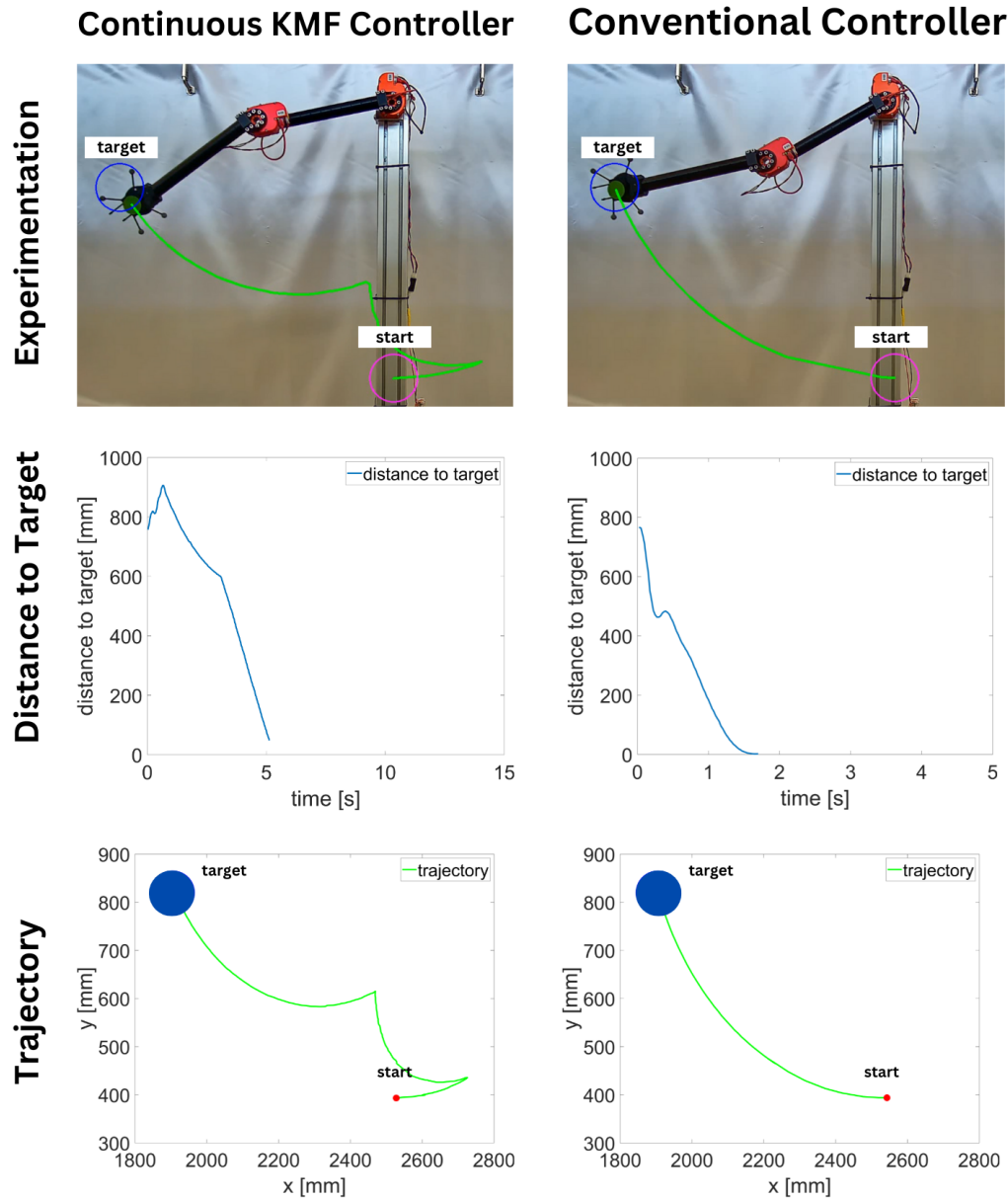


Figure 5.9: A comparison between continuous kinematic-model-free control and conventional control for target reaching with a control loop frequency of 40 Hz.

End-Effector Extension

In the second experiment, the adaptability of the continuous kinematic-model-free controller was tested against a change in the robot's kinematic structure. This was performed by extending the robot's end effector. See Figure 5.10. The end effector was extended by 250 mm in the negative y direction. The robot started from an initial position of $[x_i, y_i] = [2520, 257]$ and had to reach a target position of $[x_t, y_t] = [1905, 820]$. The results of the continuous kinematic-model-free controller are shown in Figure 5.11. The kinematic model-free controller was observed to reach the target positions when the control loop frequency was 20 Hz or higher. The same experiment was also performed using the conventional controller. The results of the conventional controller are shown in Figure 5.12. The controller failed to bring the robot's end effector to the target position for all frequencies tested since the extension was not taken into account by the kinematic model of the robot nor was the controller able to adapt to such changes. The robot's model must be updated manually for having this controller to work with the extension.

Figure 5.13 compares both controllers for 40 Hz. The continuous kinematic-model-free controller brought the robot's end effector to the target position. On the other hand, the conventional controller did not adapt to the end effector extension and was not able to reach the target position. The positional error for the continuous kinematic-model-free controller was calculated to be 5.82%, and for the conventional controller was calculated to be 8.31%. The continuous kinematic-model-free controller took 9.925 seconds to reach the desired target, while the conventional controller took 1.7 seconds. Since the kinematic-model-free controller does not depend on prior knowledge of the robot's kinematic or dynamic models, it successfully reached the target position despite the change in the robot's kinematics, demonstrating its adaptability to such changes.

End-Effector Offset

In the third experiment, once again, the adaptability of the continuous kinematic-model-free controller was tested against a change in the robot's kinematic structure. This was performed by offsetting the robot's end effector. See Figure 5.14. The end effector was offset by 225 mm

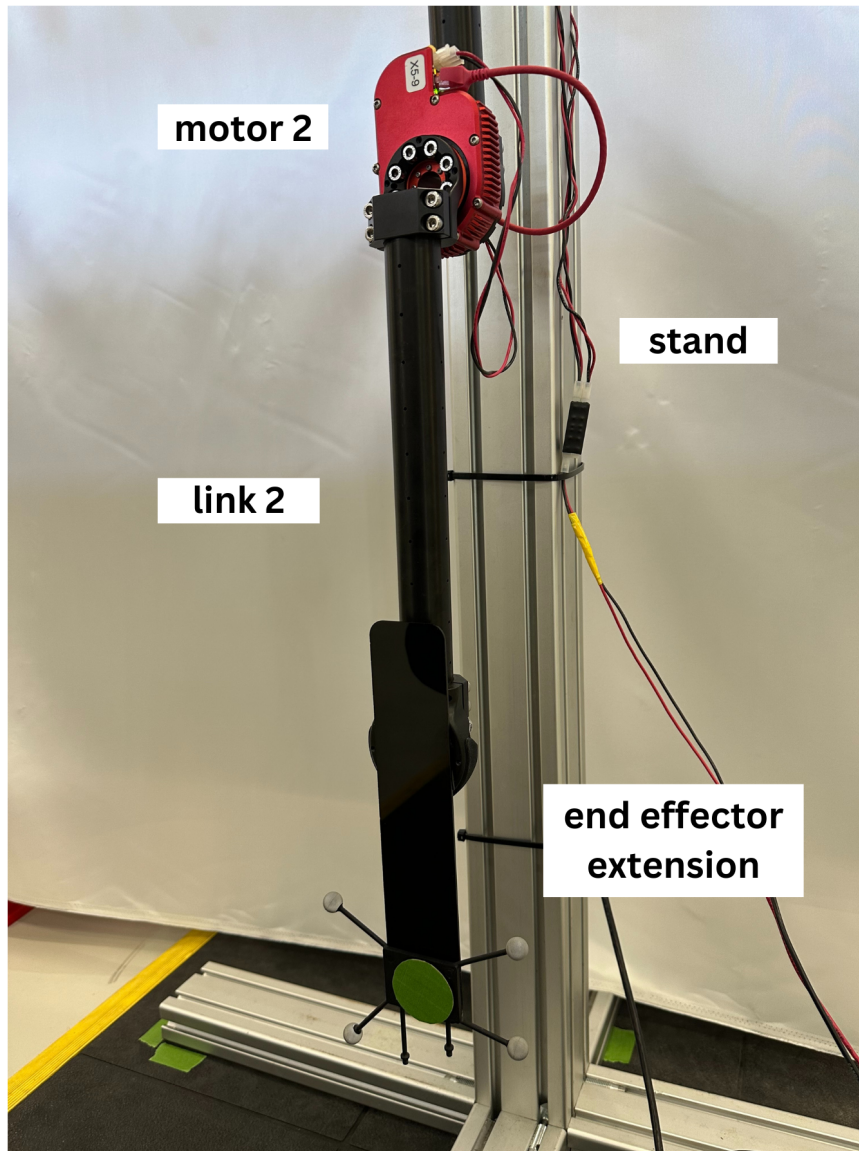


Figure 5.10: Robot manipulator with end-effector extension.

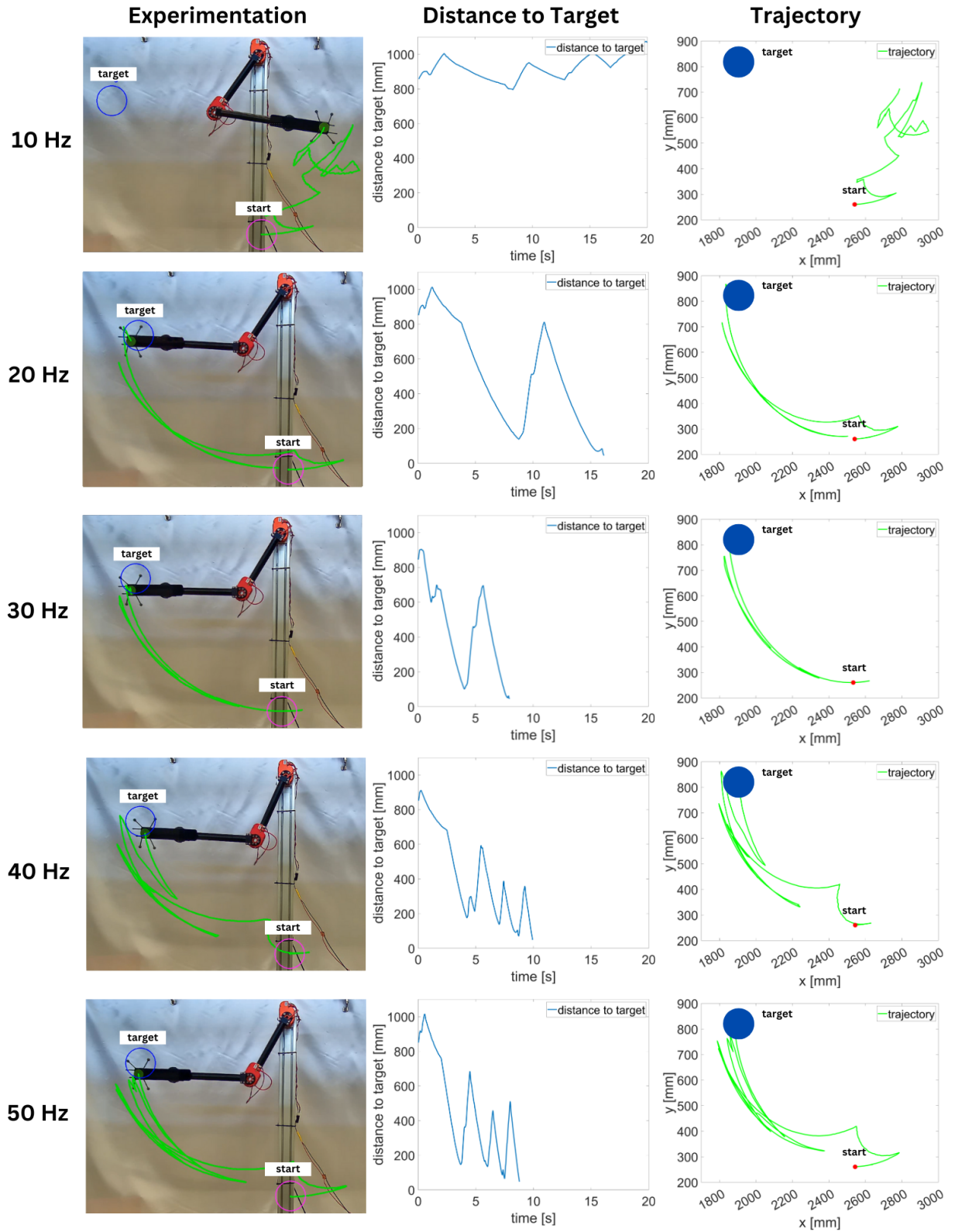


Figure 5.11: Target reaching with end-effector extension using continuous kinematic-model-free control. The controller reached the target position with a 20 Hz or higher control loop frequency.

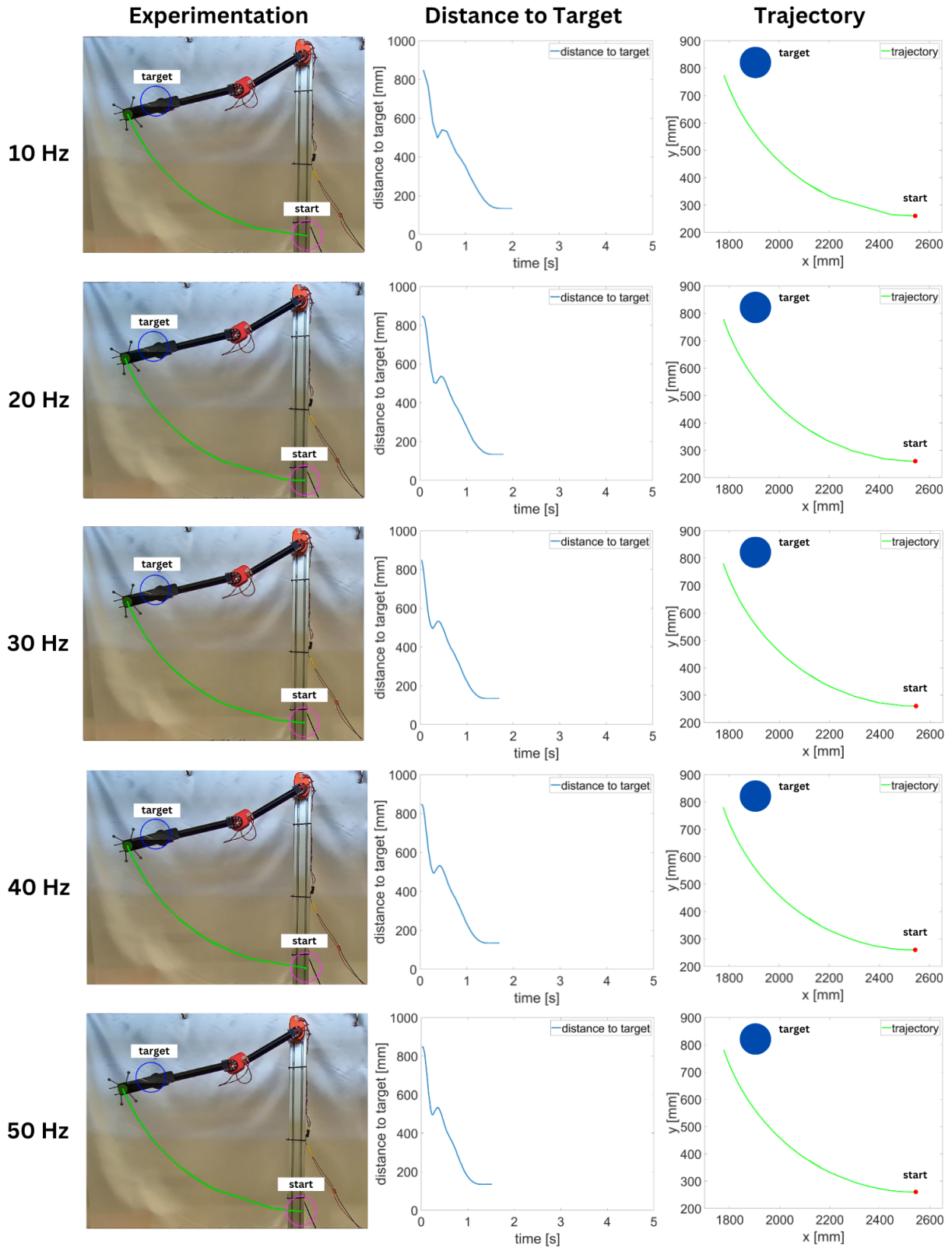


Figure 5.12: Target reaching with end-effector extension using conventional control. The controller did not reach the target position for all control loop frequencies tested.

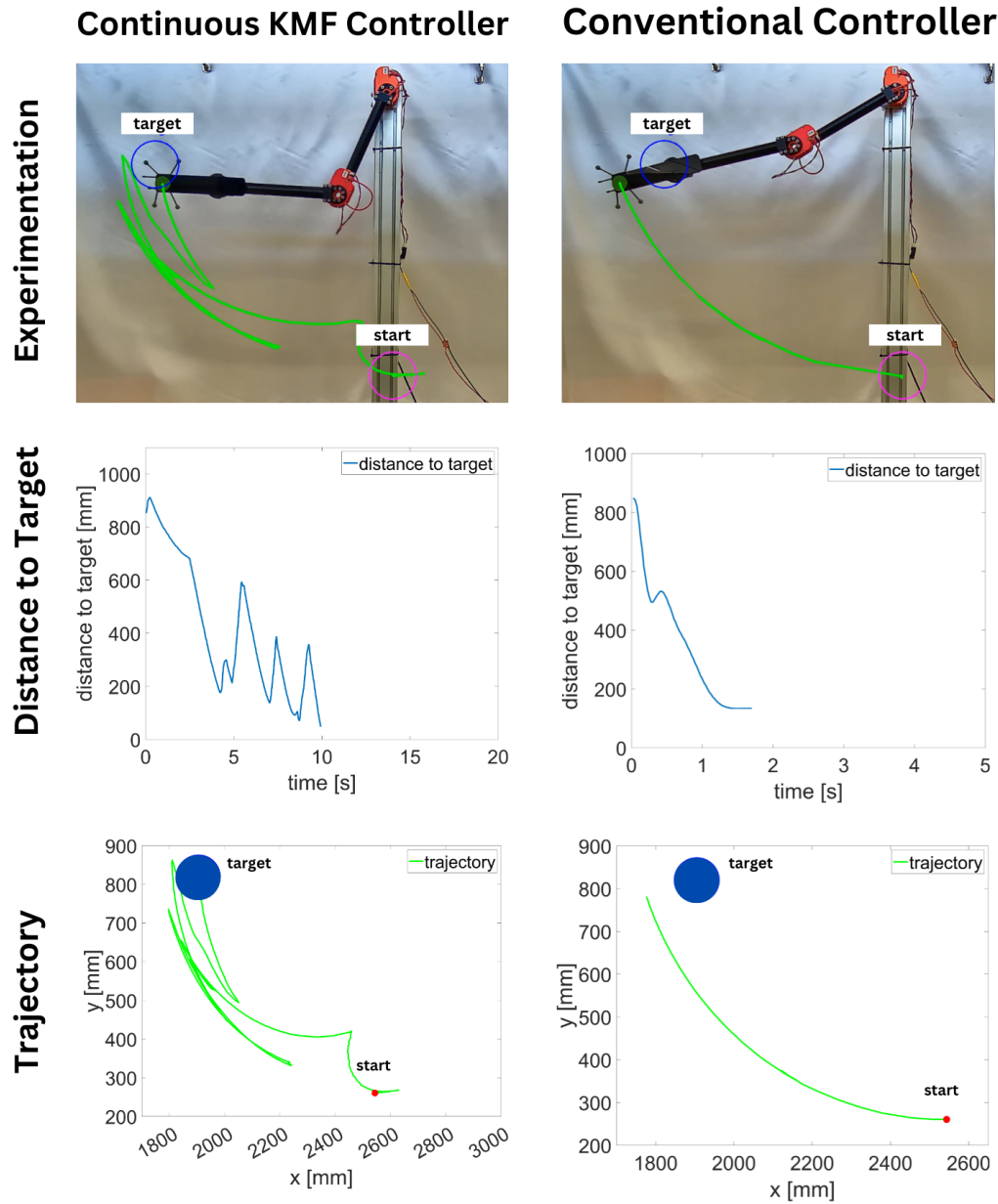


Figure 5.13: A comparison between continuous kinematic-model-free control and conventional control for target reaching with end-effector extension and a control loop frequency of 40 Hz.

and 160 mm in the x and y directions, respectively. The robot started from an initial position of $[x_i, y_i] = [2739, 537]$ and had to reach a target position of $[x_t, y_t] = [1905, 820]$. The results of the continuous kinematic-model-free controller are shown in Figure 5.15. The kinematic model-free controller was observed to reach the target positions when the control loop frequency was 30 Hz or higher. The same experiment was also performed using the conventional controller. The results of the conventional controller are shown in Figure 5.16. The controller failed to bring the robot's end effector to the target position for all frequencies tested since the offset was not taken into account by the kinematic model of the robot nor was the controller able to adapt to such changes.

Figure 5.17 compares both controllers for 40 Hz. The continuous kinematic-model-free controller brought the robot's end effector to the target position. On the other hand, the conventional controller did not adapt to the end effector offset and was not able to reach the target position. The positional error for the continuous kinematic-model-free controller was calculated to be 2.57%, and for the conventional controller was calculated to be 19.46%. The continuous kinematic-model-free controller took 6.85 seconds to reach the desired target, while the conventional controller took 1.7 seconds. The kinematic-model-free controller successfully reached the target position despite the change in the robot's kinematics, demonstrating its adaptability to such changes.

Switching Actuators

In the fourth experiment, the adaptability of the continuous kinematic-model-free controller was tested against a change in the actuators' order. This was performed by switching the order of the motors. See Figure 5.18. The robot started from an initial position of $[x_i, y_i] = [2530, 395]$ and had to reach a target position of $[x_t, y_t] = [1905, 820]$. The results of the continuous kinematic-model-free controller are shown in Figure 5.19. The kinematic model-free controller was observed to reach the target positions when the control loop frequency was 20 Hz or higher. The same experiment was also performed using the conventional controller. The results of the conventional controller are shown in Figure 5.20. The controller failed to bring the robot's end effector to the target position for all frequencies tested since the actuator order shift was not

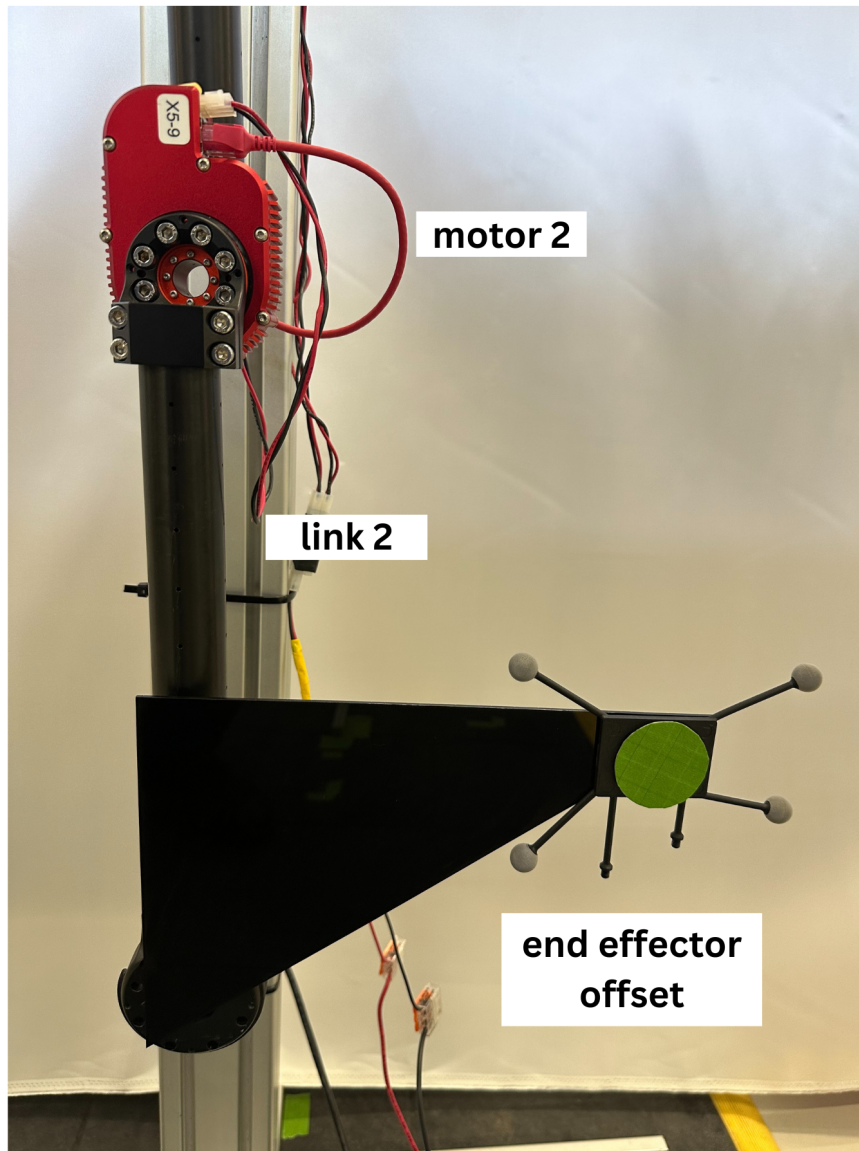


Figure 5.14: Robot manipulator with end-effector offset.

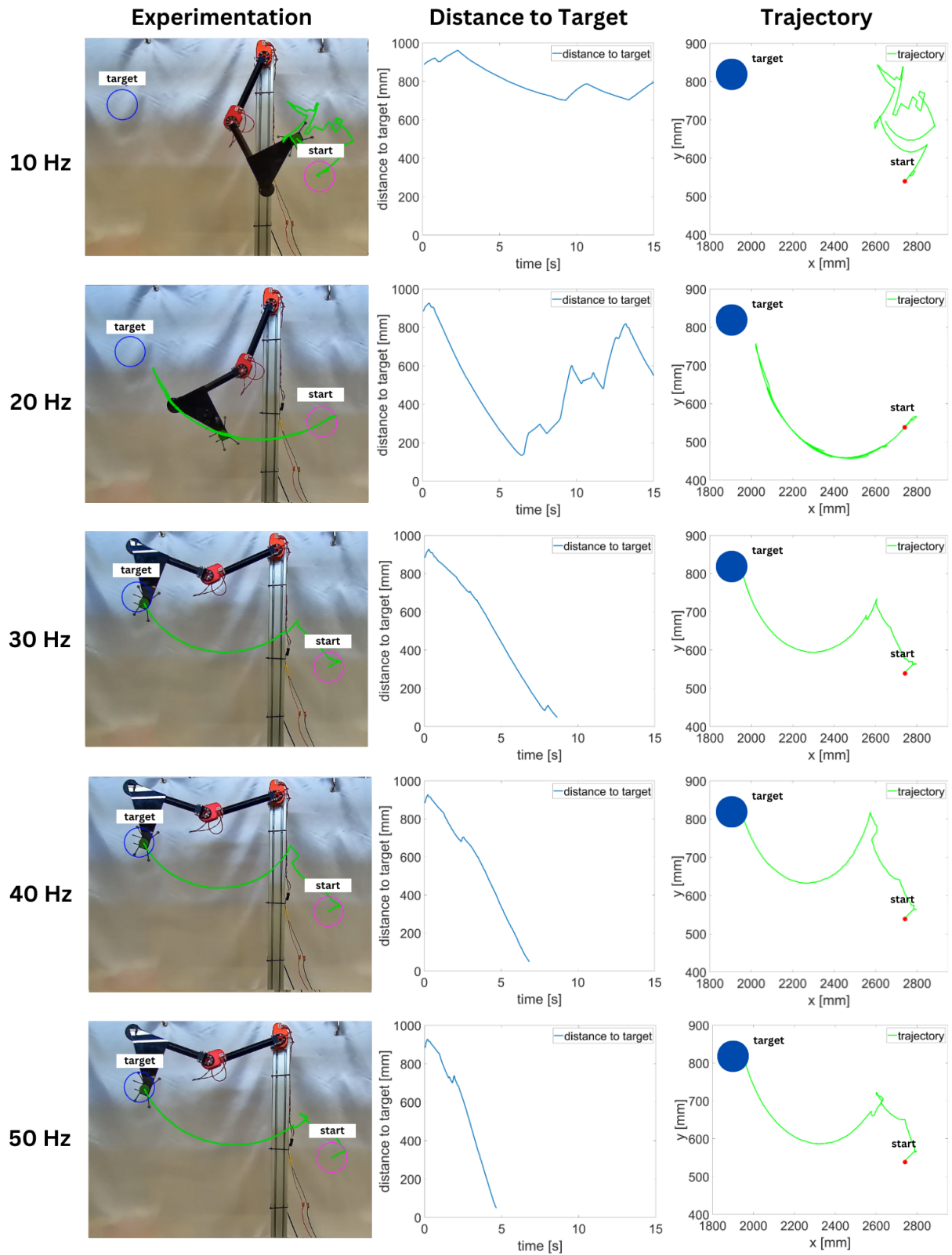


Figure 5.15: Target reaching with end-effector offset using continuous kinematic-model-free control. The controller reached the target position with a 30 Hz or higher control loop frequency.

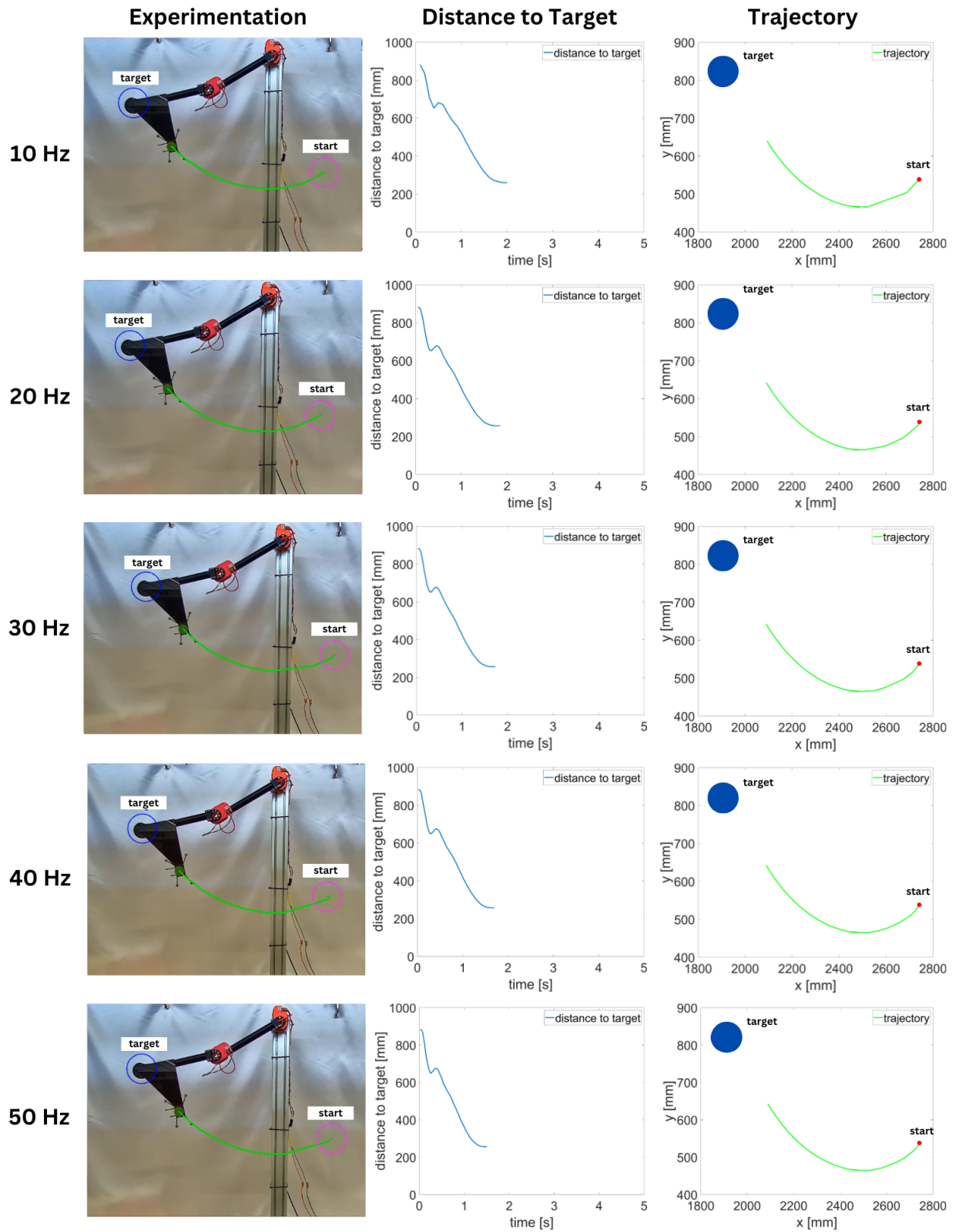


Figure 5.16: Target reaching with end-effector offset using conventional control. The controller did not reach the target position for all control loop frequencies tested.

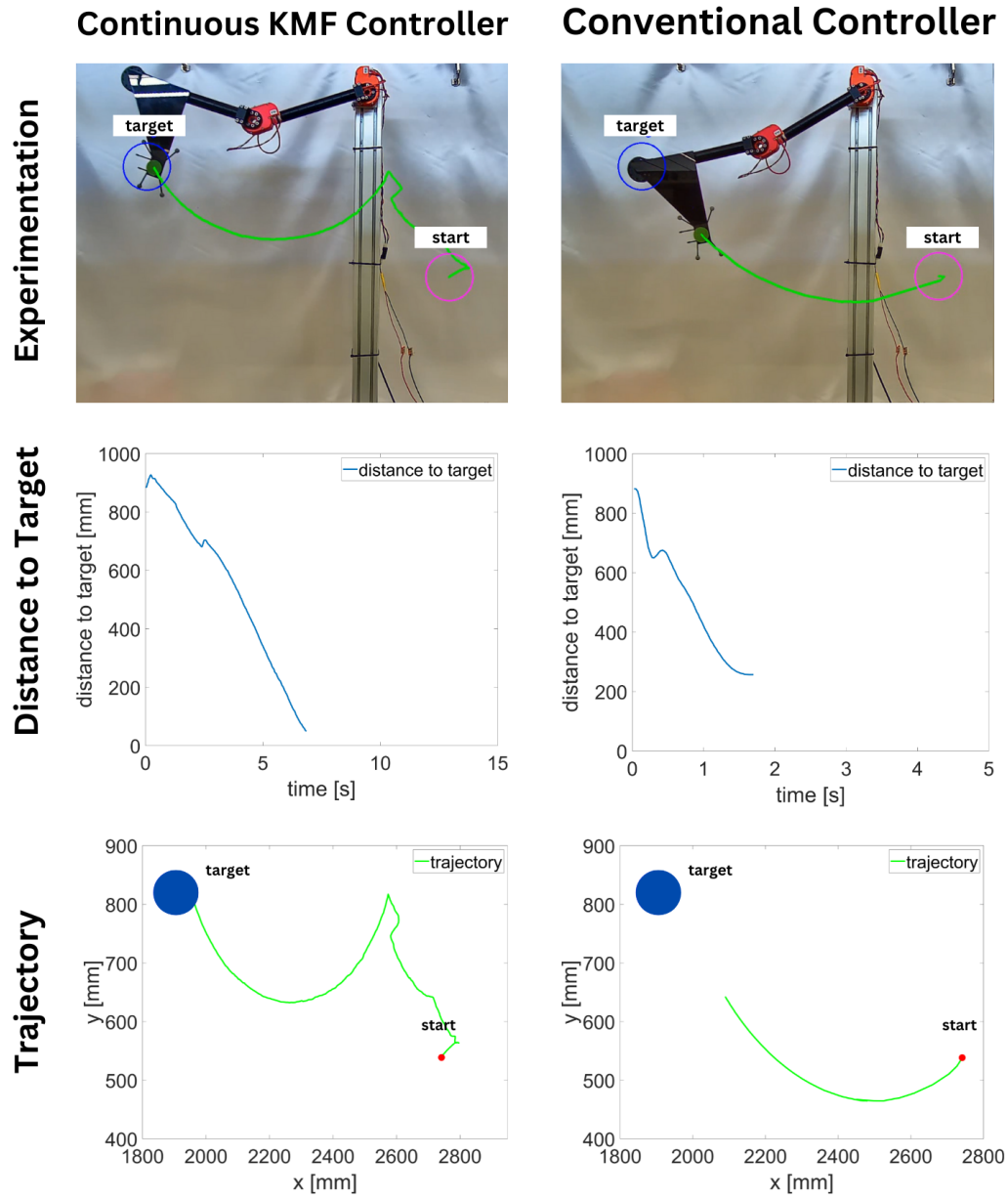


Figure 5.17: A comparison between continuous kinematic-model-free control and conventional control for target reaching with end-effector offset and a control loop frequency of 40 Hz.

taken into account by the robot's kinematic model, nor was the controller able to adapt to such a change. In all trials for the conventional controller, the controller was manually terminated to avoid a full rotation that would damage the wires.

Figure 5.21 compares both controllers for 40 Hz. The continuous kinematic-model-free controller brought the robot's end effector to the target position. On the other hand, the conventional controller did not adapt to the end effector offset and was not able to reach the target position. The positional error for the continuous kinematic-model-free controller was calculated to be 5.32%, and for the conventional controller was calculated to be 45.32%. The continuous kinematic-model-free controller took 3.7 seconds to reach the desired target, while the conventional controller was terminated after 1.5 seconds. The kinematic-model-free controller successfully reached the target position despite the change in the order of the robot's actuators, demonstrating its adaptability to such changes.

Adding Weight

In the fifth experiment, the robustness of the continuous kinematic-model-free controller was tested against changes in the robot dynamics. This was performed by adding weight to the robot manipulator. See Figure 5.22. The weight added was a rectangular metal bar that weighed 0.45 kg. The robot started from an initial position of $[x_i, y_i] = [2530, 395]$ and had to reach a target position of $[x_t, y_t] = [1905, 820]$. The results of the continuous kinematic-model-free controller are shown in Figure 5.23. The kinematic model-free controller was observed to reach the target positions when the control loop frequency was 40 Hz or higher. The results of the conventional controller are shown in Figure 5.24. The controller failed to bring the robot's end effector to the target position for all frequencies tested. However, it brought the robot's end effector close to the target position. This "sagging" was observed for all frequencies tested primarily due to the added weight not accounted for in the robot's dynamic model.

Figure 5.25 compares both controllers for 40 Hz. The continuous kinematic-model-free controller brought the robot's end effector to the target position. On the other hand, the

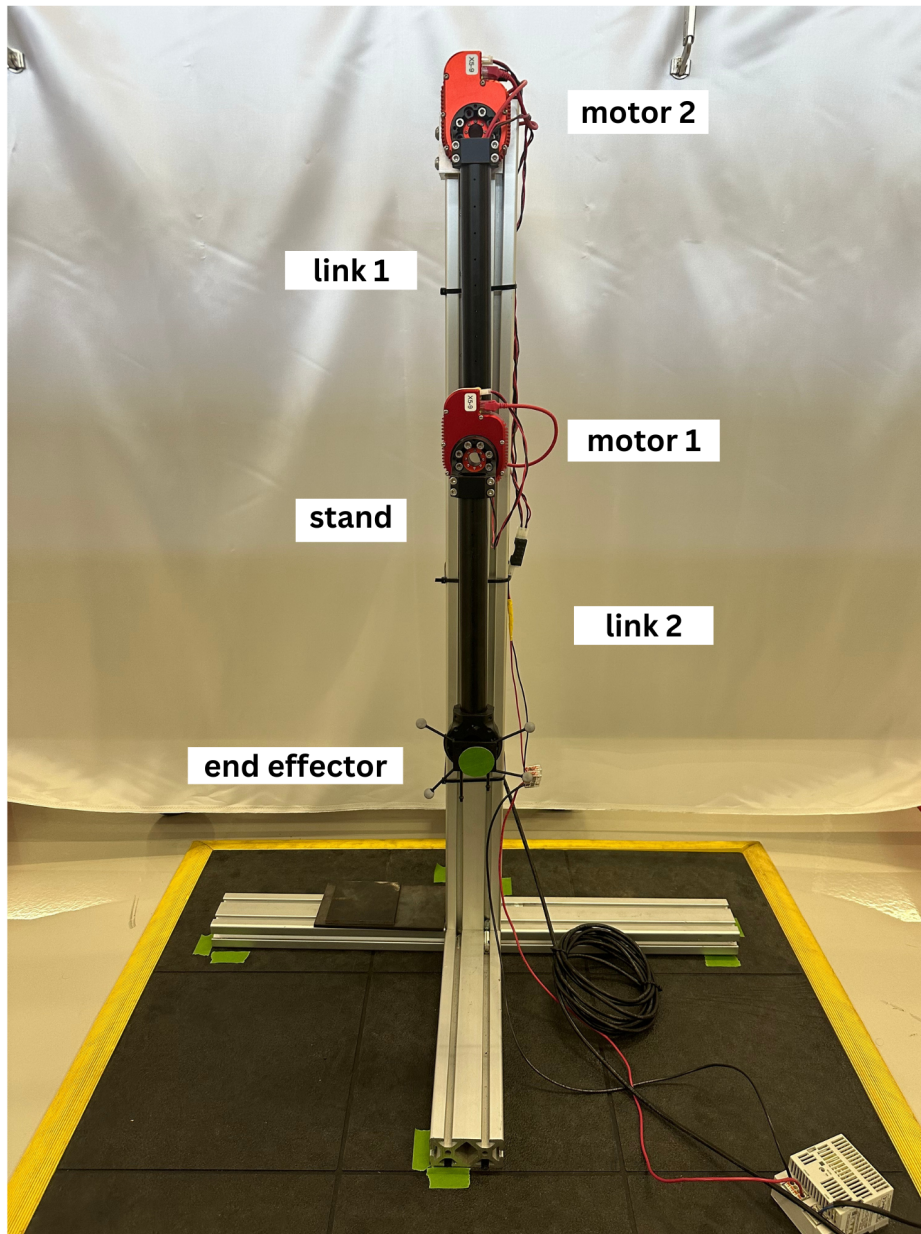


Figure 5.18: Robot manipulator with actuators order switched.

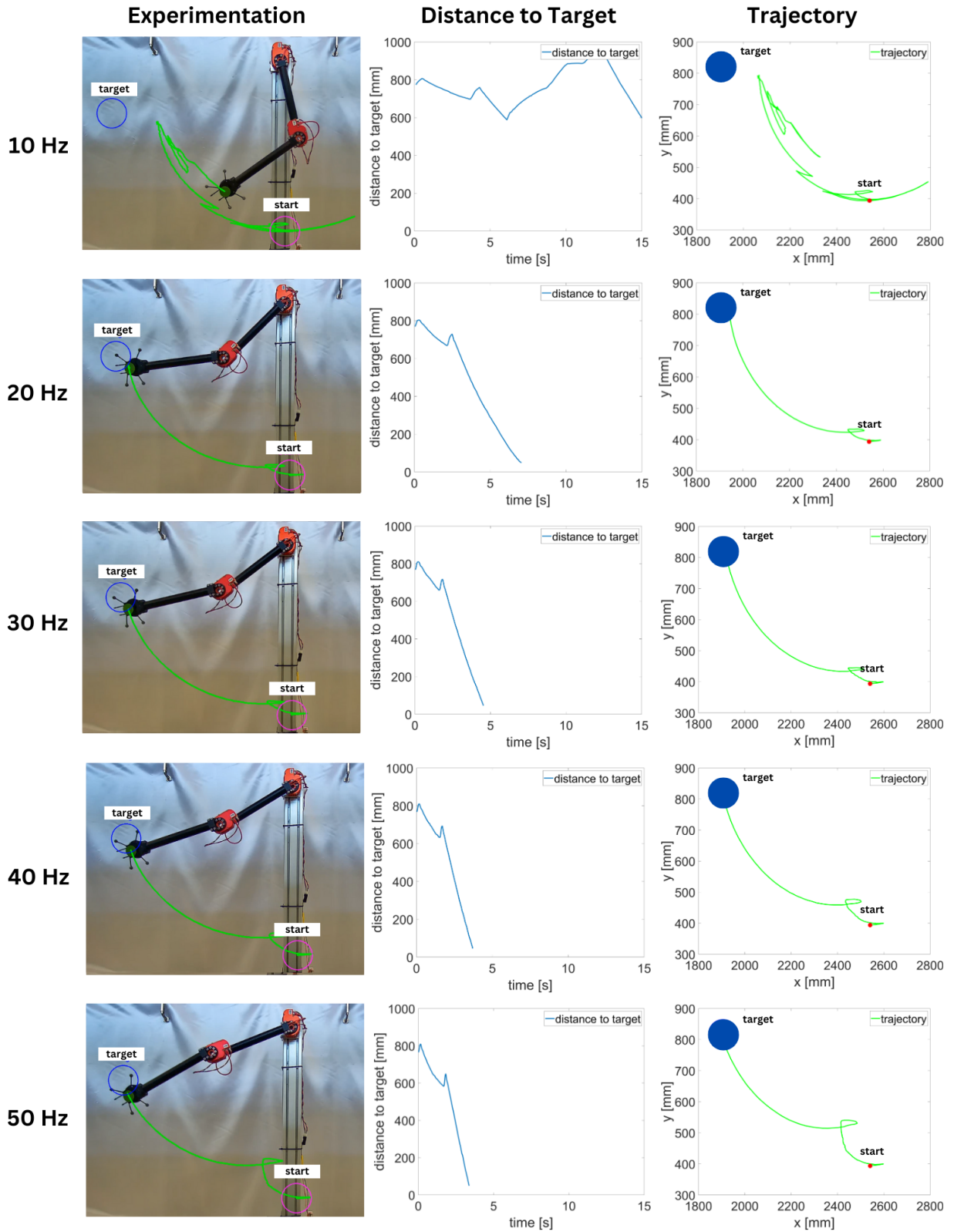


Figure 5.19: Target reaching with switched actuators using continuous kinematic-model-free control. The controller reached the target position with a 20 Hz or higher control loop frequency.

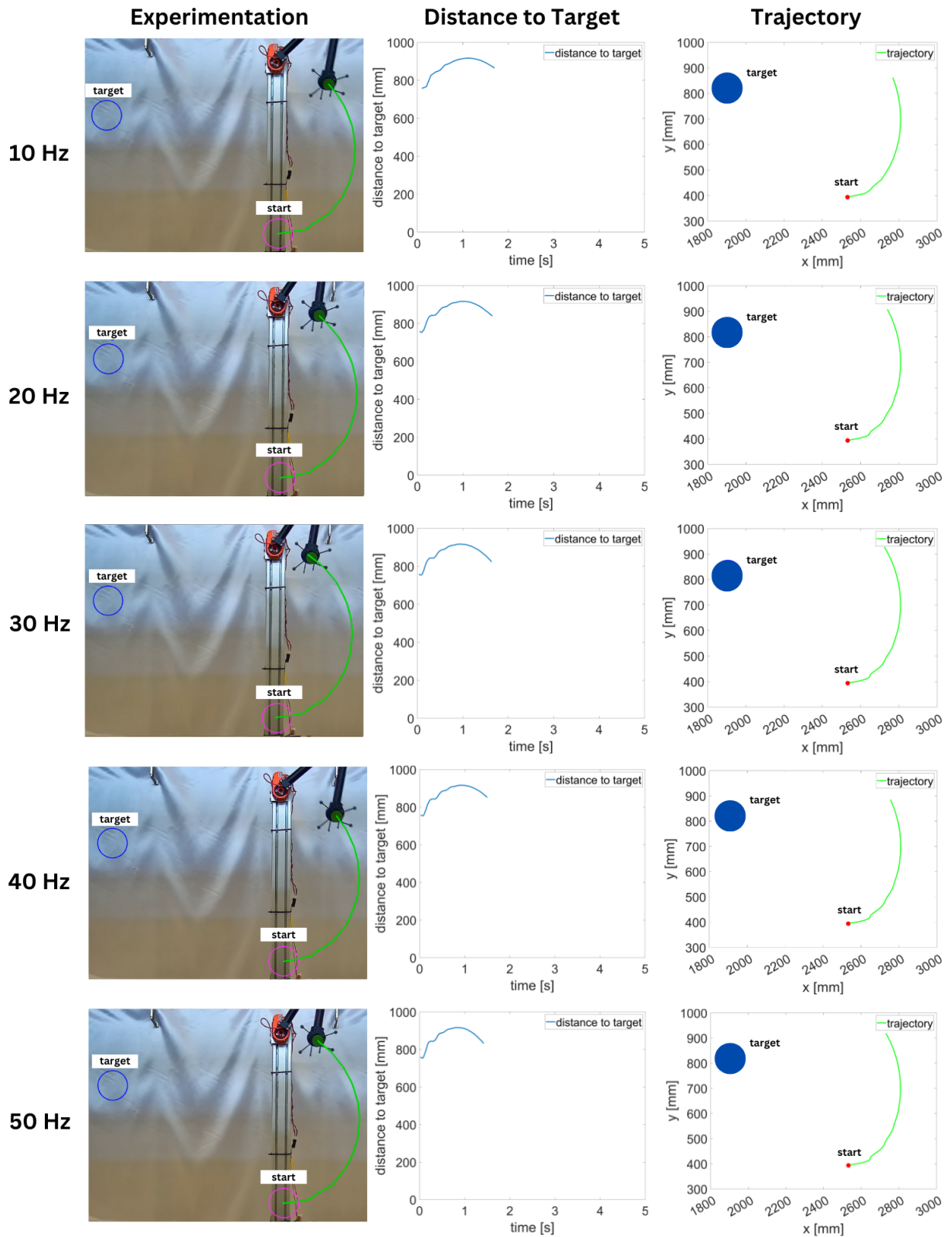


Figure 5.20: Target reaching with switched actuators using conventional control. The controller did not reach the target position for all control loop frequencies tested.

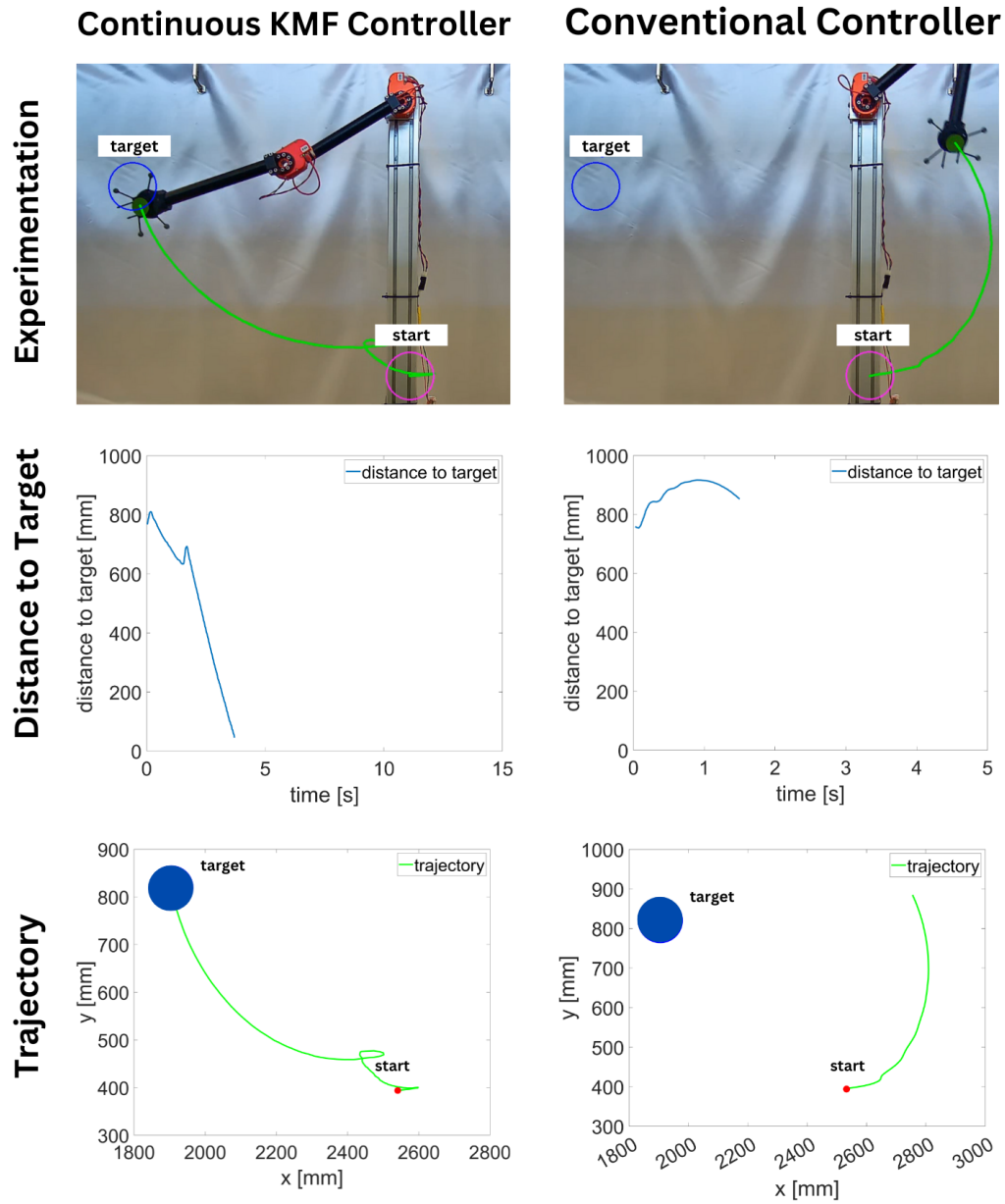


Figure 5.21: A comparison between continuous kinematic-model-free control and conventional control for target reaching with added weight and a control loop frequency of 40 Hz.

conventional controller sagged below the target position due to the added weight that was not accounted for in the robot's dynamic model. The positional error for the continuous kinematic-model-free controller was calculated to be 5.12%, and for the conventional controller was calculated to be 7.35%. The continuous kinematic-model-free controller took 6.7 seconds to reach the desired target, while the conventional controller took 1.7 seconds. The kinematic-model-free controller successfully reached the target position despite the change in the robot's dynamics, demonstrating its robustness against such changes.

Tilting Robot

In the sixth experiment, the robustness of the continuous kinematic-model-free controller was tested against changes in environmental dynamics. This was performed by changing the direction of gravity, which was achieved by tilting the robot 30 degrees. See Figure 5.26. The robot's end effector started from an initial position of $[2849, 437]$ and had to reach a target of $[2260, 970]$. The results of the continuous kinematic-model-free controller are shown in Figure 5.27. The kinematic model-free controller was observed to reach the target positions when the control loop frequency was 20 Hz or higher. The results of the conventional controller are shown in Figure 5.28. Once again, the controller failed to bring the robot's end effector to the target position for all frequencies tested, primarily due to the change in the direction of gravity that was not accounted for in the robot's dynamic model.

Figure 5.29 shows a comparison between both controllers for 40 Hz. The continuous kinematic-model-free controller brought the robot's end effector to the target position. On the other hand, the conventional controller did not manage to bring the robot's end effector to the target position due to the change in the direction of gravity that was not accounted for in the robot's dynamic model. The positional error for the continuous kinematic-model-free controller was calculated to be 5.05%, and for the conventional controller was calculated to be 19.68%. The continuous kinematic-model-free controller took 4.675 seconds to reach the desired target, while the conventional controller took 4.775 seconds. Since the kinematic-model-free controller does not depend on prior knowledge of the robot's kinematic or dynamic models, it

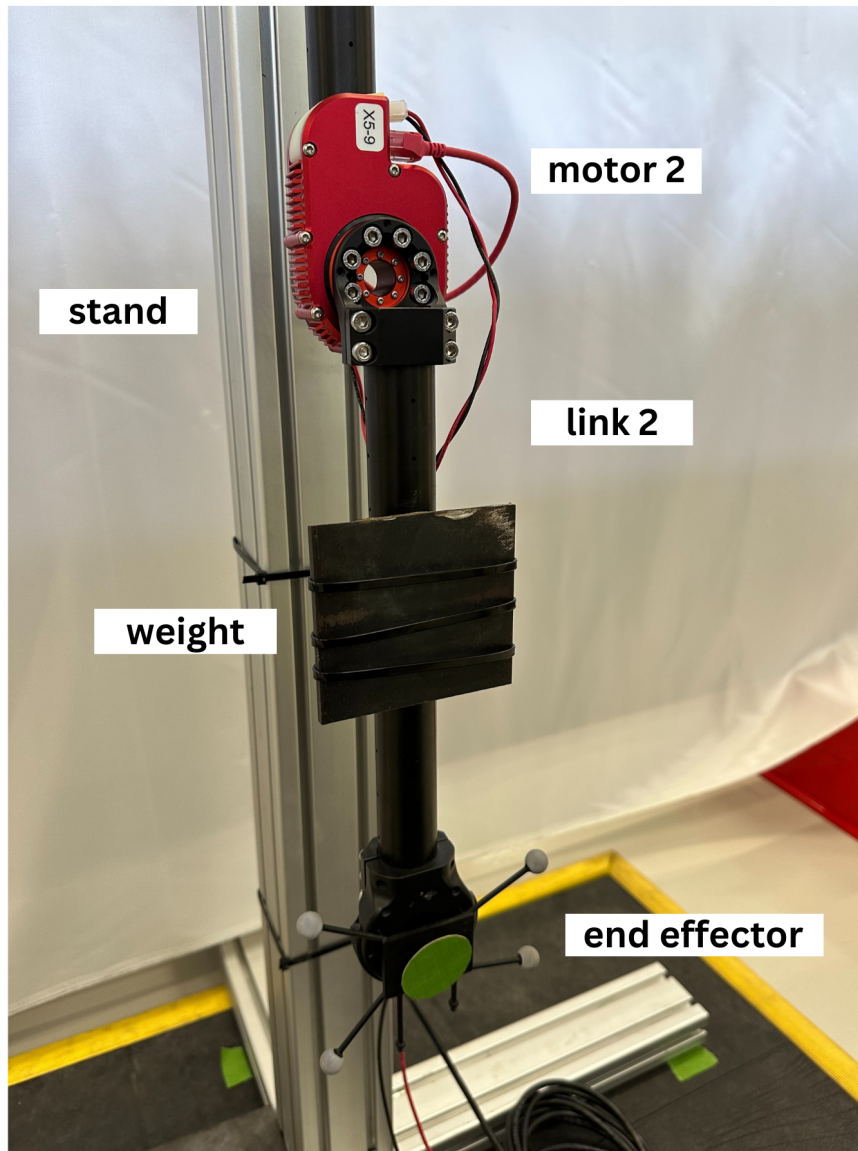


Figure 5.22: Robot manipulator with added weight.

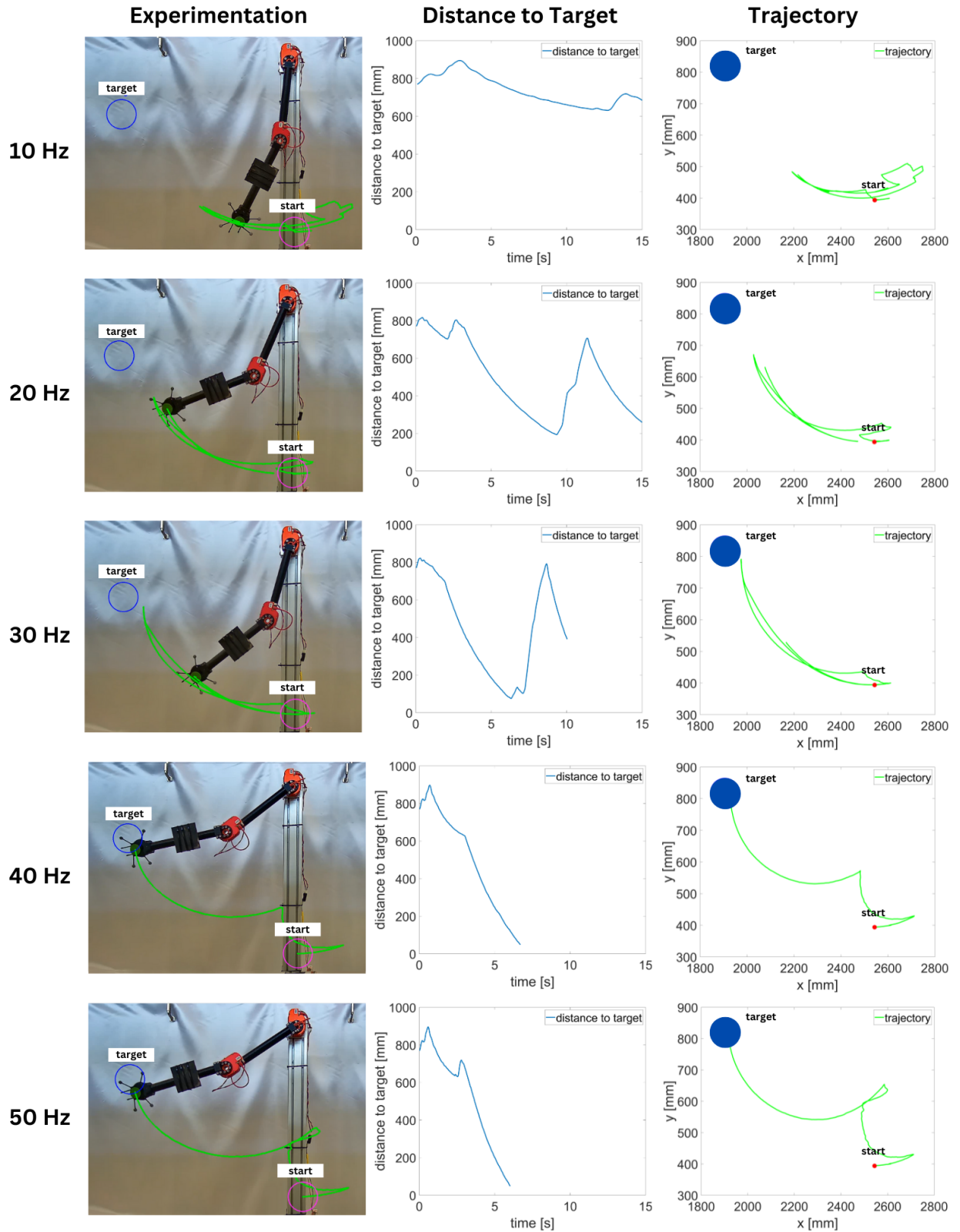


Figure 5.23: Target reaching with added weight using continuous kinematic-model-free control. The controller reached the target position with a 40 Hz or higher control loop frequency.

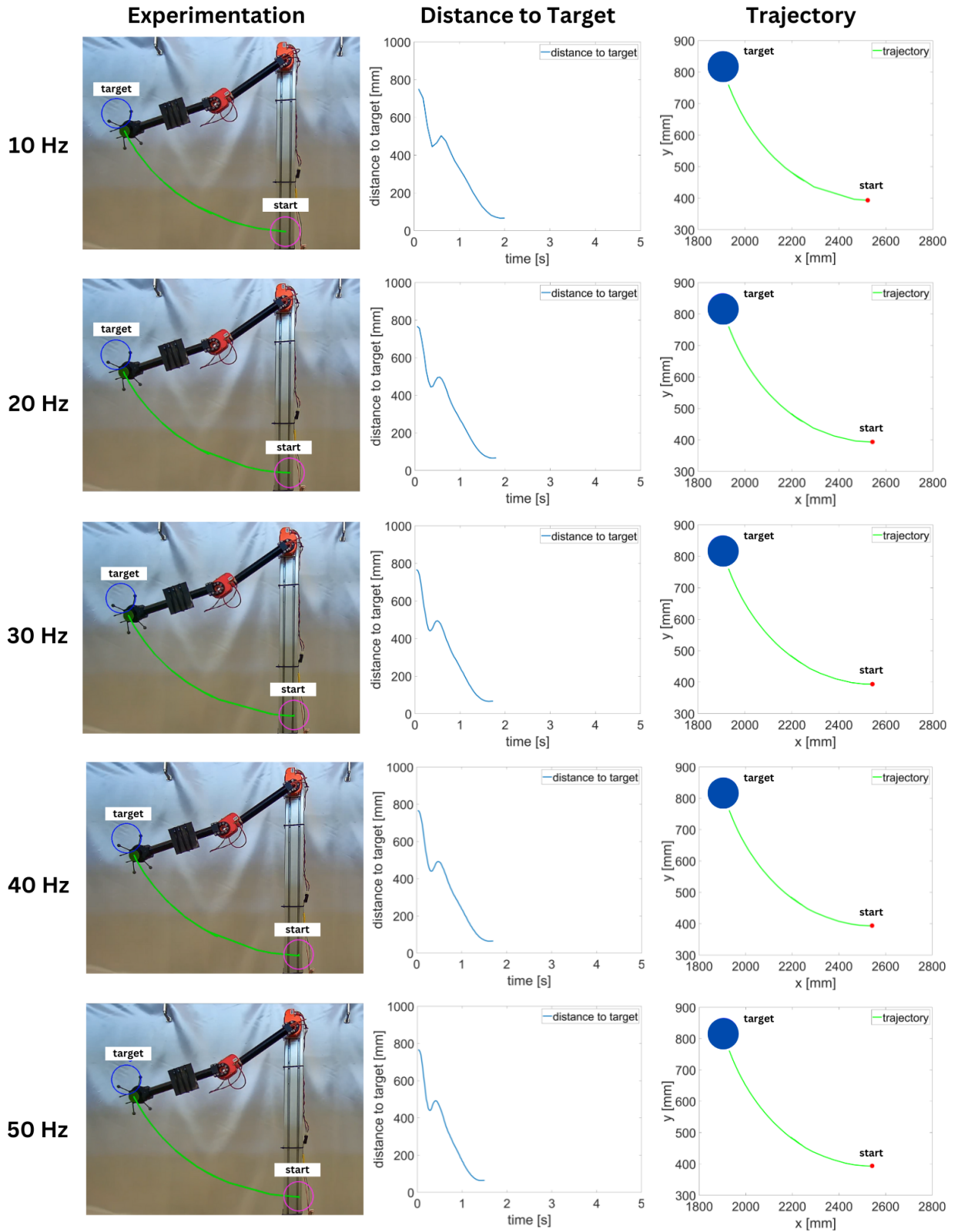


Figure 5.24: Target reaching with added weight using conventional control. The controller sagged below the target position for all control loop frequencies tested.

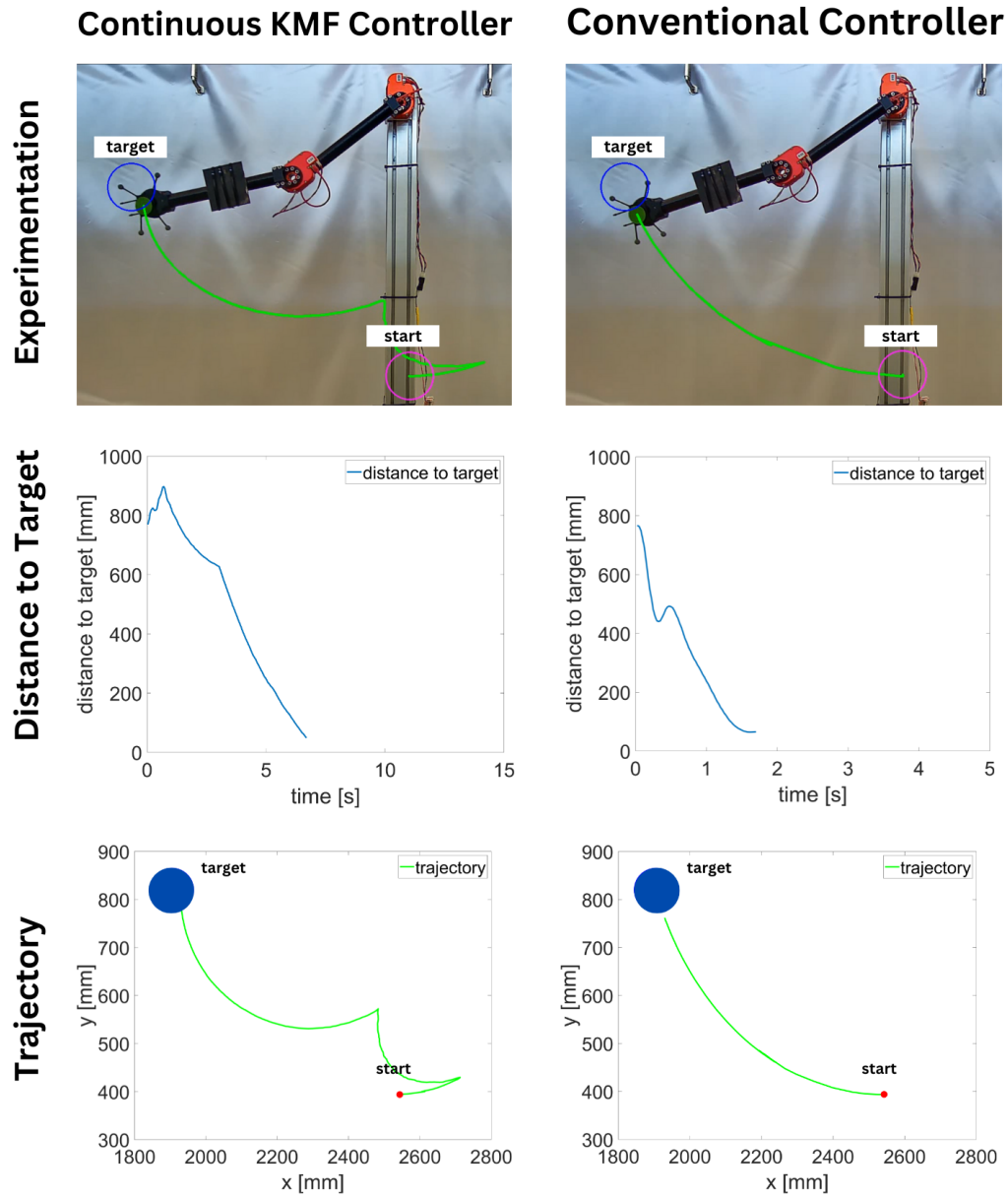


Figure 5.25: A comparison between continuous kinematic-model-free control and conventional control for target reaching with added weight and a control loop frequency of 40 Hz.

successfully reached the target position despite the change in the direction of gravity in the robot's environment, demonstrating its robustness against changes in the direction of gravity.

5.4 Discussion

The continuous kinematic-model-free controller was introduced in this chapter that can run continuously at higher speeds by sampling end-effector velocities and using that data to build volatile local linear models that are used for calculating actuation primitives that can bring the robot's end effector to a desired target position. The controller was tested on a simulated 2-DoF and a physical 2-DoF rigid robot manipulator. The simulated robot had to trace a circle while compensating for gravity. The robustness of the controller was also tested by varying the direction of gravity during execution. As for the physical experimentation, six experiments were performed with varying control loop frequencies. In the first experiment, the robot has to reach a target position. It was observed that the continuous kinematic-model-free controller managed to reach the target with a control loop frequency of 20 Hz or higher. Kinematic-model-free controllers are advantageous over traditional controllers for adapting to kinematic and dynamic changes in both the robot and the environment. The adaptability of the controller was demonstrated by altering the kinematics of the robot in the second and third experiments, where the robot's end effector was extended and offset, respectively. In the fourth experiment, the order of the actuators was swapped to further test the controller's adaptability to such changes. In the fifth and sixth experiments, the robustness of the controller against dynamic changes was tested by adding weight to the robot and changing the direction of gravity. It was observed that the controller managed to reach the target position with added weight at 40 Hz and with gravity direction change at 20 Hz. The continuous kinematic-model-free controller was compared to a conventional controller in all the experiments. In the presence of any kinematic or dynamic change, the continuous kinematic-model-free controller outperformed the conventional controller by reaching the target position. At the same time, its counterpart did not manage to reach the target. The random exploratory behaviour of the velocity-based continuous kinematic-model-free end-effector position controller does not allow for a smooth transition from one target point

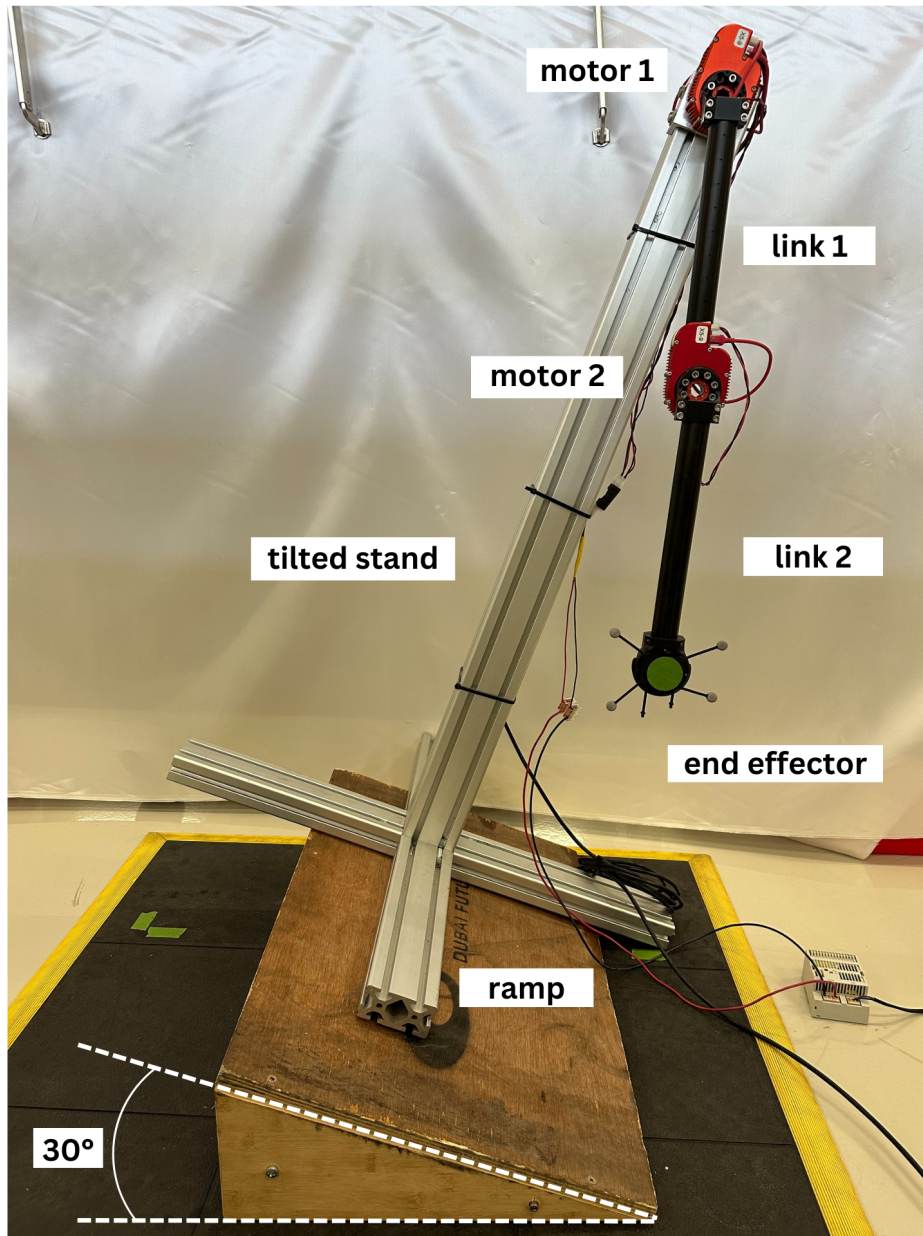


Figure 5.26: Robot manipulator with tilting ramp.

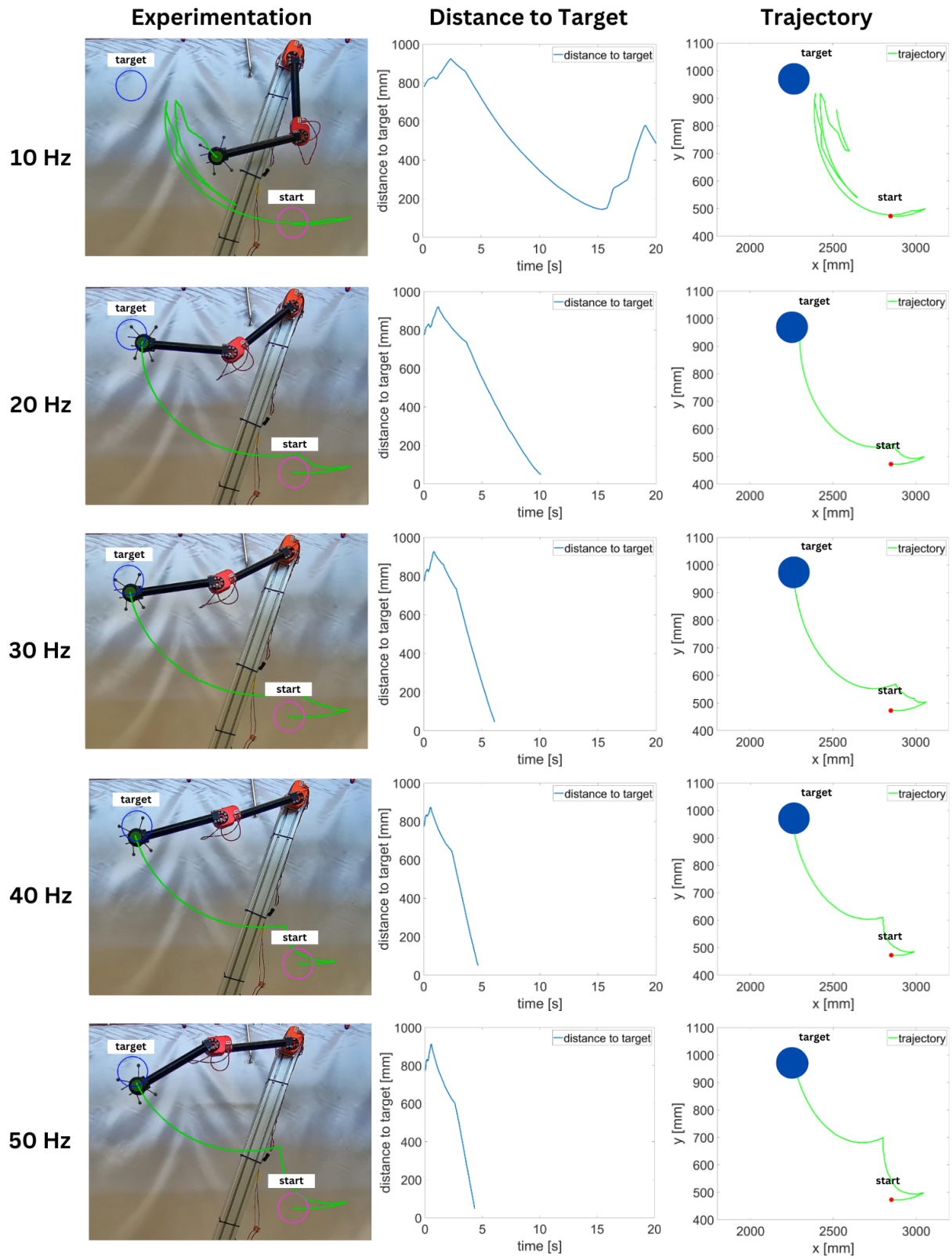


Figure 5.27: Target reaching with tilting ramp using continuous kinematic-model-free control. The controller reached the target position with a 20 Hz or higher control loop frequency.

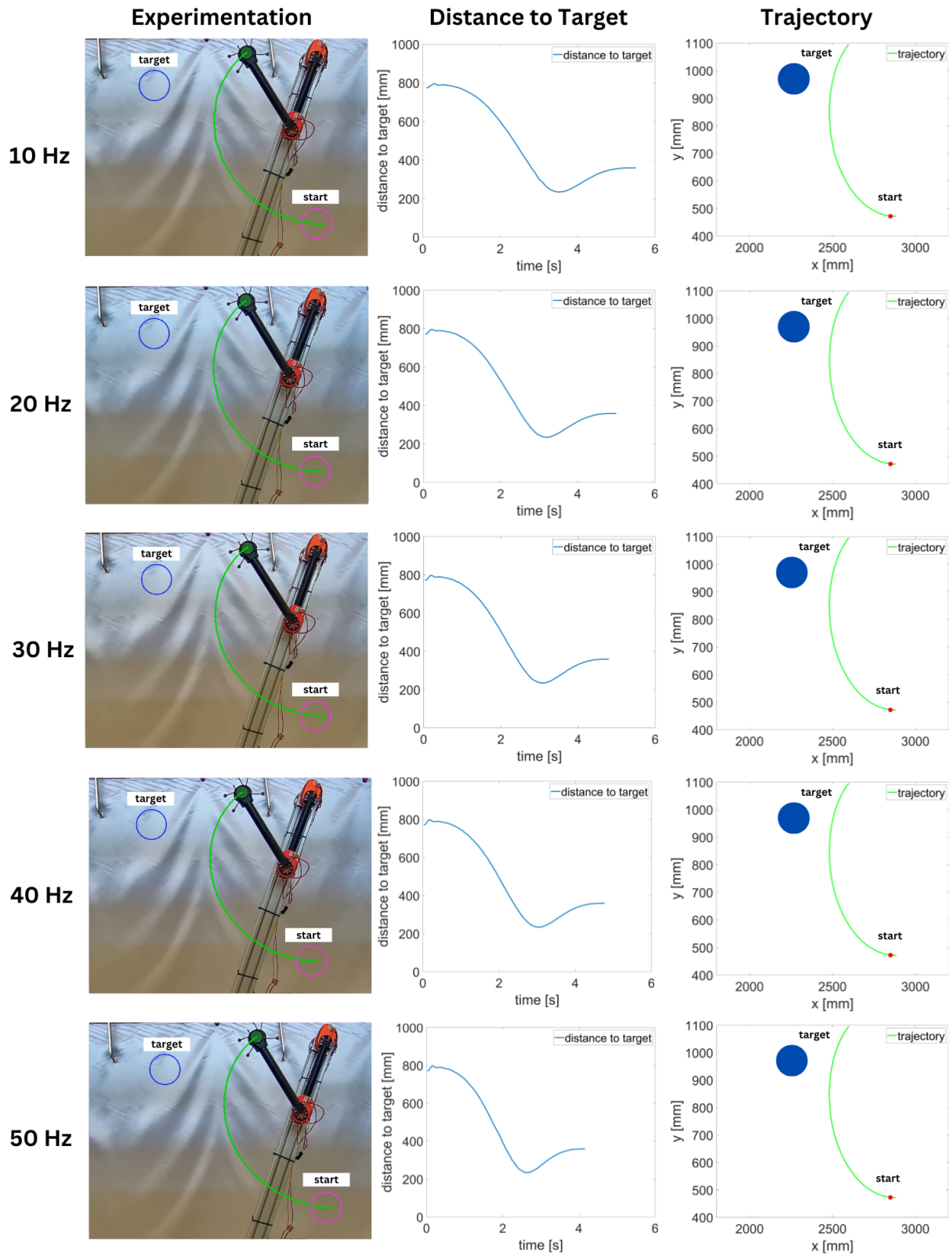


Figure 5.28: Target reaching with tilting ramp using conventional control. The controller failed to reach the target position for all control loop frequencies tested.

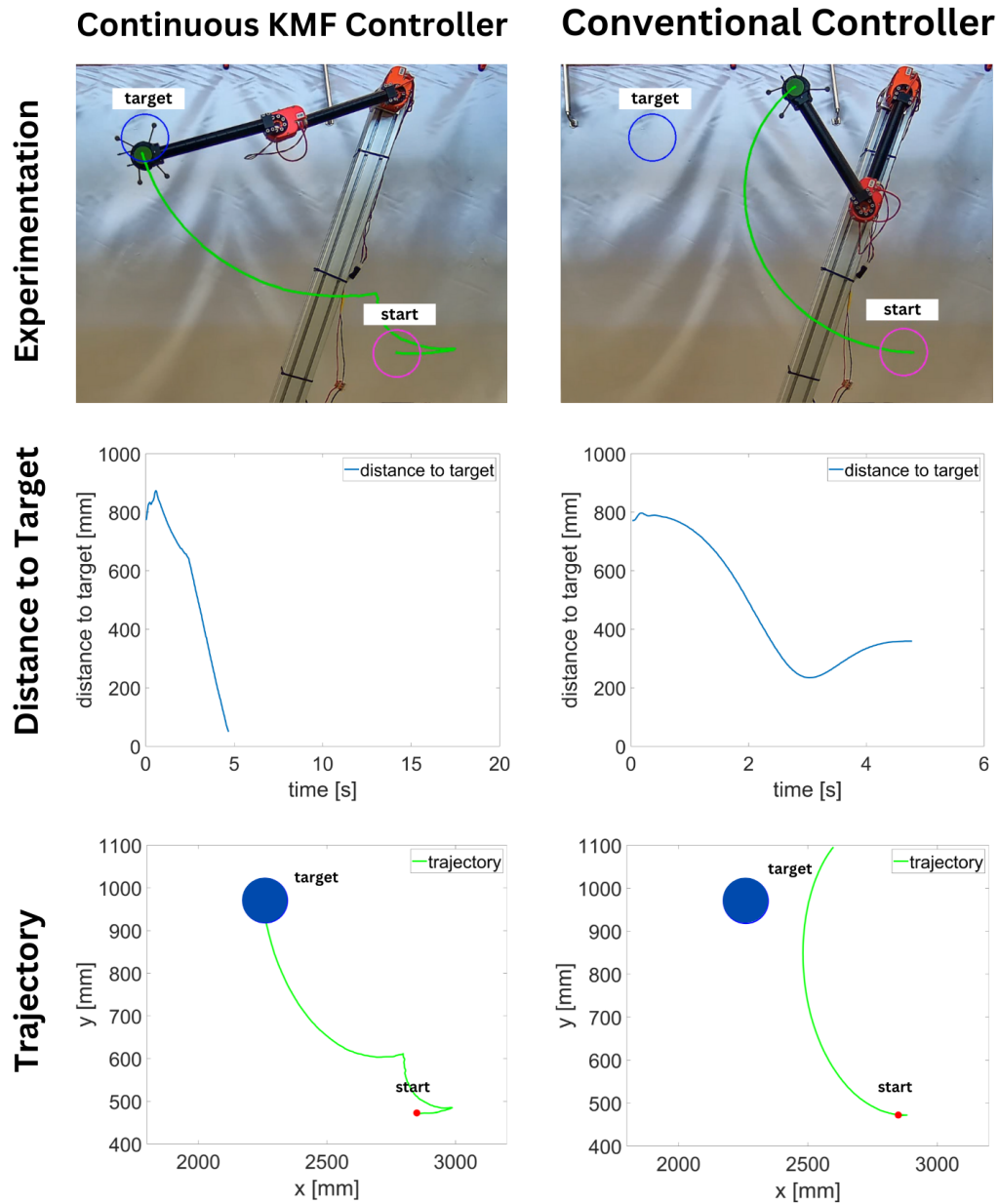


Figure 5.29: A comparison between continuous kinematic-model-free control and conventional control for target reaching with tilting ramp and a control loop frequency of 40 Hz.

to another in physical experimentation, causing the robot to behave unexpectedly. Future research will focus on the development of a more informed exploratory behaviour that will allow for continuous kinematic-model-free multi-point reaching, path following, or even developing a kinematic-model-free end-effector velocity controller that will pave the path to more useful practical applications of continuous kinematic-model-free control.

References

- [1] Ayush Agrawal and Koushil Sreenath. Discrete control barrier functions for safety-critical control of discrete systems with application to bipedal robot navigation. In *Robotics: Science and Systems*, 2017.
- [2] Ahmad AlAttar, Ikhlas Ben Hmida, Federico Renda, and Petar Kormushev. Kinematic-model-free tip position control of reconfigurable and growing soft continuum robots. In *2023 IEEE International Conference on Soft Robotics (RoboSoft)*. IEEE, 2023.
- [3] Ahmad AlAttar, Francesco Cursi, and Petar Kormushev. Kinematic-model-free redundancy resolution using multi-point tracking and control for robot manipulation. *Applied Sciences*, 11(11):4746, 2021.
- [4] Ahmad AlAttar and Petar Kormushev. Kinematic-model-free orientation control for robot manipulation using locally weighted dual quaternions. *Robotics*, 9(4):76, 2020.
- [5] Vigen Arakelian. Gravity compensation in robotics. *Advanced Robotics*, 30(2):79–96, 2016.
- [6] T Ashuri, A Armani, R Jalilzadeh Hamidi, T Reasnor, S Ahmadi, and K Iqbal. Biomedical soft robots: current status and perspective. *Biomedical Engineering Letters*, 10:369–385, 2020.
- [7] Martin Buss and Thomas Schlegl. A discrete-continuous control approach to dextrous manipulation. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 276–281. IEEE, 2000.
- [8] Chien-Chern Cheah, Chao Liu, and Jean-Jacques E Slotine. Adaptive jacobian tracking control of robots with uncertainties in kinematic, dynamic and actuator models. *IEEE transactions on automatic control*, 51(6):1024–1029, 2006.
- [9] Peter Corke. *Robotics, vision and control: fundamental algorithms in MATLAB® second, completely revised*, volume 118. Springer, 2017.

- [10] Francesco Cursi, George P Mylonas, and Petar Kormushev. Adaptive kinematic modelling for multiobjective control of a redundant surgical robotic tool. *Robotics*, 9(3):68, 2020.
- [11] Alessandro De Luca and Stefano Panzieri. An iterative scheme for learning gravity compensation in flexible robot arms. *Automatica*, 30(6):993–1002, 1994.
- [12] Alessandro De Luca, Bruno Siciliano, and Loredana Zollo. Pd control with on-line gravity compensation for robots with elastic joints: Theory and experiments. *automatica*, 41(10):1809–1819, 2005.
- [13] Gen Endo, Tetsuo Hagiwara, Yoshihide Nakamura, Hiroyuki Nabae, and Koichi Suzumori. A proposal of super long reach articulated manipulator with gravity compensation using thrusters. In *2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1414–1419. IEEE, 2018.
- [14] Chase Frazelle, Ian Walker, Ahmad AlAttar, and Petar Kormushev. Kinematic-model-free control for space operations with continuum manipulators. In *2021 IEEE Aerospace Conference (50100)*, pages 1–11. IEEE, 2021.
- [15] Ashitava Ghosal. *Robotics: fundamental concepts and analysis*. Oxford university press, 2006.
- [16] Hebi Robotics. Hebi robotics x-series actuator, 2023.
- [17] Rafael Kelly. Pd control with desired gravity compensation of robotic manipulators: a review. *The International Journal of Robotics Research*, 16(5):660–672, 1997.
- [18] Norihiro Koizumi, Shin’ichi Warisawa, Hiroyuki Hashizume, and Mamoru Mitsuishi. Continuous path controller of slave manipulator for remote ultrasound diagnostic system. *Journal of the Robotics Society of Japan*, 23(5):619–628, 2005.
- [19] Petar Kormushev, Sylvain Calinon, and Darwin Caldwell. Reinforcement learning in robotics: Applications and real-world challenges. *Robotics*, 2(3):122–148, 2013.

- [20] Petar Kormushev, Yiannis Demiris, and Darwin G Caldwell. Encoderless position control of a two-link robot manipulator. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 943–949. IEEE, 2015.
- [21] Petar Kormushev, Yiannis Demiris, and Darwin G Caldwell. Kinematic-free position control of a 2-dof planar robot arm. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5518–5525. IEEE, 2015.
- [22] Serdar Küçük. *Serial and Parallel Robot Manipulators: Kinematics, Dynamics, Control and Optimization*. BoD–Books on Demand, 2012.
- [23] Gaofeng Li, Dezhen Song, Shan Xu, Lei Sun, and Jingtai Liu. Kinematic-free orientation control for a deformable manipulator based on the geodesic in rotation group $so(3)$. *IEEE Robotics and Automation Letters*, 3(3):2432–2438, 2018.
- [24] Gaofeng Li, Lei Sun, Shan Xu, Dezhen Song, and Jingtai Liu. A hybrid model and kinematic-free control framework for a low-cost deformable manipulator using in home service. In *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 1002–1007. IEEE, 2016.
- [25] Giulio Milighetti and H-B Kuntze. On the discrete-continuous control of basic skills for humanoid robots. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3474–3479. IEEE, 2006.
- [26] Koena Mukherjee, Indra Narayan Kar, and Rajendra Kumar Purushottam Bhatt. Adaptive gravity compensation and region tracking control of an auv without velocity measurement. *International Journal of Modelling, Identification and Control*, 25(2):154–154, 2016.
- [27] NaturalPoint. Motion capture systems - optitrack, 2023.
- [28] Saeed Rafee Nekoo, José Ángel Acosta, and Anibal Ollero. Gravity compensation and optimal control of actuated multibody system dynamics. *IET Control Theory & Applications*, 16(1):79–93, 2022.

- [29] Jonathan R Nistler and Majura F Selekwa. Gravity compensation in accelerometer measurements for robot navigation on inclined surfaces. *Procedia Computer Science*, 6:413–418, 2011.
- [30] Jerry Pratt, Peter Dilworth, and Gill Pratt. Virtual model control of a bipedal walking robot. In *Proceedings of International Conference on Robotics and Automation*, volume 1, pages 193–198. IEEE, 1997.
- [31] Keehong Seo, Joohyung Kim, and Kyungshik Roh. Towards natural bipedal walking: Virtual gravity compensation and capture point control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4019–4026. IEEE, 2012.
- [32] LS Shieh, H Wang, and RE Yates. Discrete-continuous model conversion. *Applied Mathematical Modelling*, 4(6):449–455, 1980.
- [33] Mark W Spong and Masayuki Fujita. Control in robotics. *The Impact of Control Technology: Overview, Success Stories, and Research Challenges*, 2011.
- [34] Nathan Ulrich and Vijay Kumar. Passive mechanical gravity compensation for robot manipulators. In *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, pages 1536–1537. IEEE Computer Society, 1991.
- [35] Gregory C White and Yangsheng Xu. An active vertical-direction gravity compensation system. *IEEE transactions on instrumentation and measurement*, 43(6):786–792, 1994.
- [36] Jun Wu, Jinsong Wang, and Zheng You. An overview of dynamic parameter identification of robots. *Robotics and computer-integrated manufacturing*, 26(5):414–419, 2010.
- [37] Chenglong Yu, Zhiqi Li, and Hong Liu. Research on gravity compensation of robot arm based on model learning. In *2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 635–641. IEEE, 2019.
- [38] Arturo Zavala-Rio and Victor Santibanez. Simple extensions of the pd-with-gravity-compensation control law for robot manipulators with bounded inputs. *IEEE Transactions on Control Systems Technology*, 14(5):958–965, 2006.

Chapter 6

Conclusion

Robot control has been a thriving field of research since the emergence of robots. Robots have been predominantly controlled using conventional methods that rely on the kinematics and dynamics of rigid bodies for calculating joint-space to task-space transformations. These traditional controllers rely heavily on the robot’s kinematics and dynamics to remain static. However, soft robots, morphing robots, malleable robots, transforming robots, and evolving robots can be challenging to model accurately and control. Tremendous research has been done in the area controls for soft robots. Most of the controllers found in literature can be categorized into two main classes: Model-based controllers, which rely on models describing the system’s motion for controller derivation; and Model-free controllers, which use data-driven and empirical methods instead.

Kinematic-model-free control methods can overcome these challenges as these control methods do not require any prior knowledge of the robot’s kinematic or dynamic models. Kinematic-model-free controllers work by gathering information through short exploratory actuations, building volatile local linear models that approximate the robot’s local kinematic and dynamic behaviour, and estimating an actuation signal that moves the robot towards a given reference target. The research presented in this thesis details and extends the state-of-the-art kinematic-model-free controller in many fronts to tackle several open problems.

The kinematic-model-free controller was first extended to perform simultaneous position and orientation control of a planar rigid robot's end effector by incorporating end-effector orientation as part of the control parameters. The controller was tested on a simulated rigid robot to perform kinematic-model-free target position reaching and planar target pose reaching. The kinematic-model-free controller was also deployed on a physical manipulator constrained to move in the horizontal axis with 2-DoFs to perform end-effector target position reaching. The controller was then adapted to control soft continuum robots. Simulation tests were conducted using MATLAB's SoRoSim toolbox where the tip-position of a soft continuum robot with thread-like actuators was controlled. The controller was tested with growing/shrinking soft continuum robots too. The robustness of the controller was demonstrated by the coping of the controller in the presence of actuator failures. The controller was also validated by deploying it on a physical continuum soft robot that was pneumatically-actuated to perform tip-position control. The robot successfully traced a line and a lemniscate path.

Quaternion mathematics were used to represent the robot's end-effector position and orientation for kinematic-model-free position and orientation control. Dual-quaternion mathematics was then used to develop a kinematic-model-free robot end-effector pose controller. The controller was tested on a simulated rigid robot. Controlling a single point, the end-effector, on serial hyper-redundant manipulators with the dual-quaternionic kinematic-model-free pose controller underperformed. To resolve redundancy in hyper-redundant robot manipulators, the dual-quaternionic kinematic-model-free multi-point controller was introduced that can control the pose of several points along the kinematic chain. This controller was tested on a hyper-redundant simulated robot manipulator to achieve specific robot configurations.

The kinematic-model-free controller was combined with model-predictive-control theory to develop kinematic-model-free predictive-control that was used for path planning and obstacle avoidance. Both simulated and physical experimentation was conducted to verify the controller. The controller was experimented on a 2-DoF planar rigid robot that performed target reaching while avoiding virtual obstacles. In one experiment, one large virtual obstacle was present

and in another experiment, two smaller virtual obstacles were present. It was observed that when exploratory behaviour was triggered near obstacles, collisions may occur. More informed exploratory behaviour could help resolve this issue.

Finally, the kinematic-model-free controller was updated to run continuously at higher speeds by sampling end-effector velocities and using that data for building volatile local linear models for calculating actuation primitives to actuate the robot's end effector towards a desired target point in task space. The controller was tested on both a simulated and a physical 2-DoF rigid robot manipulator. The simulated robot had to trace a circle while compensating for the effects of gravity. The robustness of the controller was also tested by varying the direction of gravity during execution. As for the physical experimentation, the robot had to reach a target point in task space. The control loop frequency was varied to inspect how the performance of controller was improved. The robustness of the controller was tested by changing the direction of the gravity vector and by adding weight to the robot. The continuous kinematic-model-free controller was compared to a conventional controller in all the experiments. In the presence of any kinematic or dynamic change, the continuous kinematic-model-free controller outperformed the conventional controller.

The kinematic-model-free controller requires no prior knowledge of the robot's kinematics or dynamics for control. The controller works by sampling end-effector state during pseudo-random exploratory behaviour and building volatile local linear models that are used for calculating actuation primitives that will bring the robot to the target position, orientation, or configuration. The controller is capable of controlling end-effector position, orientation, or pose. The controller was demonstrated to work on different robots, both simulated and physical robots and both rigid and soft robots. The research covered in this thesis pushed the kinematic-model-free controller to new fronts and supported it with tests and experiments that verified the approach. Along the way, many other interesting research problems surfaced that could be further explored in future research: optimization of actuation signal waveform for motion, learning local kinodynamics using higher order models, dual-quaternion based optimization for local linear learning, obstacle

avoidance without prior path planning, environment-aware intelligent exploration, velocity control of robot end effector, continuous control of physical robot tip position for multi-point reaching or tracing, control of morphing or evolving robots, and kinematic-model-free control for robot grasping or locomotion.

Appendix A

Mathematics

The underlying assumption behind the kinematic-model-free controller is that the end-effector displacement is proportional to the robot's joint torques locally. A rigid-body robotic system can be obtained using the following general dynamic equation:

$$M(q(t)\ddot{q} + c(q, \dot{q}) + g(q(t)) + \tau_f(t) = \tau(t) \quad (\text{A.1})$$

where q are the joint angles, $\dot{q}$ are the joint velocities, $\ddot{q}$ are the joint accelerations, M is the inertia matrix, c is the Coriolis vector, g is the gravity vector, τ_f is the friction torque, and τ is the applied torque. More simply:

$$\tau = M\ddot{q} + c\dot{q} + g + \tau_{ext} \quad (\text{A.2})$$

Rearranging terms and assuming zero gravity and negligible Coriolis and no external forces:

$$\ddot{q} = M^{-1}\tau \quad (\text{A.3})$$

Since $\dot{q} = J^{-1}\dot{x}$, we have:

$$\ddot{x} = JM^{-1}\tau \quad (\text{A.4})$$

Integrating twice:

$$x = \int \int JM^{-1}\tau dt \quad (\text{A.5})$$

Over some duration d_p :

$$\Delta x = \int \int_{t_0}^{t_0+d_p} JM^{-1}\tau dt \quad (\text{A.6})$$

Assuming J and M are constant:

$$\Delta x = JM^{-1} \int \int_{t_0}^{t_0+d_p} \tau dt \quad (\text{A.7})$$

Let d_p be one control step where $\tau = k\tau_p(t)$ with a rectangular pulse:

$$\Delta x = JM^{-1} \int \int_{t_0}^{t_0+d_p} k\tau_p dt \quad (\text{A.8})$$

Since the torque is constant over the duration:

$$\Delta x = JM^{-1}k\tau_p \int \int_{t_0}^{t_0+d_p} dt \quad (\text{A.9})$$

$$\Delta x = JM^{-1}k\tau_p \left[\frac{t^2}{2} + bt + c \right]_{t_0}^{t_0+d_p} \quad (\text{A.10})$$

Combining all the constant terms into C :

$$\Delta x = C\tau_p \quad (\text{A.11})$$

As such, we note that the end-effector displacement is directly proportional to the joint torques within the local region:

$$\Delta x \propto \tau_p \quad (\text{A.12})$$

In the case of a planar 2-DoF manipulator, 4 exploratory torques are applied in series

$\{\tau_p(p_1), \tau_p(p_2), \tau_p(p_3), \tau_p(p_4)\}$ where $\tau_p(p_1) = -\tau_p(p_2)$, $\tau_p(p_3) = -\tau_p(p_4)$, and $\tau_p(p_1) \perp \tau_p(p_3)$. It follows from the direct proportionality of the end-effector displacement and the joint torques that $\Delta x(p_1) = -\Delta x(p_2)$, $\Delta x(p_3) = -\Delta x(p_4)$, and $\Delta x(p_1) \perp \Delta x(p_3)$ within the local region.

Since the matrix A_2 is a rectangular matrix of size $m \times k$ with rank r , where m is the spacial dimensions and k is the kNN samples used for building the local linear model, the matrix A_2 can be factored using QR factoring with column pivoting:

$$A_2 = QRP \quad (\text{A.13})$$

where Q is an $m \times r$ matrix of orthonormal columns, R is an $r \times r$ matrix, and P is an $n \times n$ permutation matrix. It is noted that:

$$A_2 P^T = [A_2^1 \ A_2^2] = QR = Q[R^1 \ R^2] \quad (\text{A.14})$$

where A_2^1 is an $m \times r$ matrix with linearly independent columns and QR factorization $A_2^1 = QR^1$ and A_2^2 is an $m \times (n - r)$ with columns that are linear combinations of A_2^1 . As such, the columns of A_2^1 , which is denoted as $\hat{A}_2$, are a basis selected from A_2 . Using this basis, the linear combination is expressed as follows:

$$\hat{A}_2 w = b_2 \quad (\text{A.15})$$

where $w = [w_1, w_2, \dots, w_k]^T$ is the vector of weights that are calculated as follows:

$$w = \hat{A}_2^{-1} b_2 \quad (\text{A.16})$$

where the Moore–Penrose inverse is used. The calculated weights are then used to calculate the actuation required to move the end effector towards the desired target:

$$b_1 = \hat{A}_1 w \quad (\text{A.17})$$

where $\hat{A}_1 = A_1^1$ from $A_1 P^T = [A_1^1 \ A_1^2]$.

As for the continuous kinematic-model-free controller, the end-effector velocity change is proportional to the robot's joint torques change locally. Once again, starting from the robot dynamics equation is as follows:

$$M(q(t)\ddot{q} + c(q, \dot{q}) + g(q(t)) + \tau_f(t) = \tau(t) \quad (\text{A.18})$$

More simply:

$$\tau = M\ddot{q} + c\dot{q} + g + \tau_{ext} \quad (\text{A.19})$$

Rearranging terms and assuming zero gravity and negligible Coriolis and centrifugal torques and no external forces:

$$\ddot{q} = M^{-1}\tau \quad (\text{A.20})$$

Since $\dot{q} = J^{-1}\dot{x}$, we have:

$$\ddot{x} = JM^{-1}\tau \quad (\text{A.21})$$

Let d_p be one control step where $\tau = k\tau_p(t)$ with a rectangular pulse::

$$\dot{x} = \int_{t_0}^{t_0+d_p} JM^{-1}k\tau_p dt \quad (\text{A.22})$$

Since the torque is constant over the duration:

$$k\dot{x} = JM^{-1}\tau_p [bt + c]_{t_0}^{t_0+d_p} \quad (\text{A.23})$$

Combining all the constant terms into C :

$$\dot{x} = C\tau_p \quad (\text{A.24})$$

which can be rewritten as:

$$v = C\tau_p \tag{A.25}$$

where v is the end-effector linear velocity. The equation can then be written as:

$$\Delta v = C\Delta\tau_p \tag{A.26}$$

The change in the robot's end-effector velocity is directly proportional to the change in robot joint torques within the local region.