Manipulation of Natural Deformables: Simulation, Inference, and Learning

Jayadeep Jacob

Supervisor: Prof. Fabio Ramos

A thesis submitted in fulfilment of the requirements for the degree of
Doctor of Philosophy

Faculty of Engineering
School of Computer Science
The University of Sydney

The research reported in this thesis was supported by the award of a Research Training Program scholarship to the Candidate.

May, 2026

Statement of Originality

I attest that this thesis is the product of my original work and intellectual effort. To the best of my knowledge, it does not contain any content previously published or written by another individual, nor has it been submitted in whole or substantial part for any other degree or diploma at this or any other institution, except where explicitly cited. All assistance and sources used in its preparation have been fully acknowledged.

. . . . . .

Jayadeep Jacob

Supervisor: Prof. Fabio Ramos PhD Candidate: Jayadeep Jacob

Abstract

Over the past decades, machine learning has transformed robotic manipulation, enabling autonomous systems to perform complex rigid body tasks, by learning from simulation datasets as well as real-world interactions. Yet, this progress has not translated to deformable objects, characterised by high-dimensional state space and non-linear dynamics. Handling natural variants of deformables, such as tree branches and plant stems, is an even greater challenge, exacerbated by their non-uniform geometry and asymmetric dynamics. Moreover, viable solutions must quantify uncertainties from imperfect sensors and inaccurate models to account for the probabilistic nature of the physical world. To address these challenges, we present an ensemble of learning frameworks to model the intricate plant topology, estimate dynamic parameters, and learn efficient control strategies for manipulating natural deformables.

First, we present a simulation-driven inverse inference approach to model the uncertain dynamics of plant stems under deformation, in a *real-to-sim* context. The ground truth trajectories are gathered by probing real branches, whereas the simulations are guided by coarse spring abstractions deployed on parallel physics simulators. Framing system identification as a Bayesian inference problem, we estimate the multi-modal, spring parameter posterior density. The proposed non-parametric method can incorporate biological assumptions, quantify the estimation uncertainty, and tolerate contact instabilities.

Next, we train a non-prehensile, contact-sensitive, reinforcement learning policy to interact with tree branches, with minimal ruptures, in a *sim-to-real* setting. The crude spring abstractions are integrated with a parametric L-system model to build a procedural forest generator, replete with realistic, self-similar branching patterns. Furthermore, we take a vision-free track to generate novel contact detection features exclusively from internal time-series inputs. The novel proprioceptive, contact-aware approach transfers zero-shot from simulation to real, manipulating plant stems with unseen geometry and dynamics, exhibiting unique contact minimisation strategies.

Finally, we expand the *sim-to-real* strategy, emphasising manipulation with the whole arm instead of just the end-effector, treating the deformables as a collection. To learn a computationally efficient policy, we leverage a distributional state representation, applying kernel operator theory on a multi-modal feature set, combining camera vision and proprioception. Our blank slate policy learning approach can autonomously discover creative de-occlusion strategies for clearing electrical power lines of overhanging foliage in simulation and real-world environments.

Acknowledgments

Doctoral candidature has been anything but a joy ride, at times an exhilarating exploration of the research rabbit holes, and at others, a descent into the inexplicably absurd. Thanks to the kindness of countless characters around me, the trial has passed, and I can wake up now. The chief among them is my supervisor and sensei, Prof. Fabio Ramos, whose intelligence is only surpassed by his humility and patience. Fabio, I am deeply grateful for your enduring support and guidance throughout this journey, thank you for showing me this wonderland of scientific enquiry. I was also fortunate to receive the support of equally exceptional co-supervisors and collaborators from CSIRO, Dr. Tirthankar Bandyopadhyay, Dr. Paulo Borges, and Dr. Jason Williams. Tirtha, thank you for your mentorship, guidance, and for the many seemingly crazy ideas that turned into valued publications. Paulo, I sincerely thank you for your kind words of encouragement at every meeting without fail. Jason, thanks for your support during the initial days, and crucially, for choosing me to pursue this.

I am grateful for the Australian Government's support via the RTP Fee Offset scholarship, and for the research funding provided by Data61 CSIRO, without which this work would not have been possible.

Thanks to the exceptionally bright and fun folk at Usyd: Wenzheng Zhang, Zeya Yin, Houston Warren, Paco Tsang, Lucas Barcelos, William Zhi, Tin Lai, Shizhe Cai, Kunal Ostwal, Ethan Hirschowitz, Nirmal Ghotekar, and Kim Bente. Thank you for the numerous coffees, lunches, collaborations, and discussions on research and life. I am fortunate to have crossed paths with you all.

Somewhere along the course of life, I came to realise that only two things are truly worth any pursuit: wisdom and love; everything else is just vanity. Wisdom arises from reason, and even in the most subjective of realms, the only reasoning tool we have is that of scientific enterprise. This work is a modest attempt toward that first pursuit. As for love, I have been blessed with a wonderful family, to whom I reserve my deepest gratitude: my better half, Jiny Jacob, and our gorgeous daughter, Meera. This work would not have been possible without your extraordinary patience, unwavering support, and the warmth you bring into my life. Thank you for keeping me grounded in testing times and for sharing in the joy of better days.

With the benefit of hindsight, I can confidently say that all of you have written this thesis with me, and this journey belongs to us all.

Contents

Nomenclature

x	a scalar
$\mathbb{R}$	a set of real numbers
$\mathbb{R}^{a \times b}$	a set of a by b real matrices
$f(\cdot)$	a function
$p(X \mid Y)$	the conditional Probability Density Function of a random variable X given Y
$\mathbb{E}[\cdot]$	expected value
$SE(3)$	a special Euclidean group representing 3D space
$k(x, x')$	a kernel function, quantifying the similarity between two data points x and x'
μ	a vector representing the means of multiple variables
Σ	a matrix representing the covariances between pairs of variables
$\mathcal{N}(\mu, \Sigma)$	normal (Gaussian) distribution with mean vector μ and covariance matrix Σ
τ	a trajectory, typically a sequence of states over time
$\mathcal{T}$	a set of trajectories, e.g., $\mathcal{T}_{\text{sim}} = \left\{\tau_{\text{sim}}^{j}\right\}_{j=1}^{P}$.
θ	parameters to be optimised in a model or algorithm
w	network weights, typically parameters of a neural network learned during training
π	a policy, typically a reinforcement learning policy mapping states to actions
π_w	a policy parameterised by neural network weights w

List of Figures

List of Tables

Authorship Attribution

The contributions presented in this thesis have been published in the following conferences and journals, which form the core chapters of this thesis. The author's attribution includes, but is not limited to, the motivation, conceptualisation, formalisation, derivation, experimentation, and communication of the following academic articles.

Chapter 3 appears as: Jacob, J., Bandyopadhyay, T., Williams, J., Borges, P., & Ramos, F. (2024). Learning to simulate tree-branch dynamics for manipulation. *IEEE Robotics and Automation Letters*, *9*(2), 1748–1755

Chapter 4 appears as: Jacob, J., Cai, S., Borges, P. V. K., Bandyopadhyay, T., & Ramos, F. (2025, June). Gentle manipulation of tree branches: A contact-aware policy learning approach. In P. Agrawal, O. Kroemer & W. Burgard (Eds.), *Proceedings of the 8th conference on robot learning* (pp. 631–648, Vol. 270). PMLR. https://proceedings.mlr.press/v270/jacob25a.html

Chapter 5 appears as: Jacob, J., Zhang, W., Warren, H., Borges, P., Bandyopadhyay, T., & Ramos, F. (2026). Deformable cluster manipulation via whole-arm policy learning. *IEEE Robotics and Automation Letters*, *11*(3), 2951–2958

. . . . . .
Jayadeep Jacob

As supervisor for the candidature upon which this thesis is based, I can confirm that the authorship attribution statements above are correct.

. . . . . .
Prof. Fabio Ramos

Generative AI Attribution

During the preparation of this thesis, the author used *Grammarly* for minor text corrections (e.g., spelling, grammar, and sentence clarity) and *Google Gemini* for limited boilerplate code generation supporting the research publications underlying this thesis. These tools were not used in the research design, experimental design, motivation, or development of ideas. All substantive content, research findings, and articulation of this thesis are the work of the author. The author confirms that where text or code was modified with the assistance of generative AI tools, the outputs were reviewed for accuracy and appropriateness. The author takes full responsibility for the submitted thesis and affirms that the work is original and independently produced, in accordance with University of Sydney guidelines on the responsible use of generative AI.

Nothing in biology makes sense except
in the light of evolution.
THEODOSIUS DOBZHANSKY

CHAPTER I

Introduction

Artificial Intelligence has made tremendous progress in recent years, primarily driven by Large Language Models (LLMs) trained on internet-scale textual data, fuelling widespread anticipation of the impending shift to Artifical General Intelligence (AGI). Yet, in comparison to humans and other animals, robots that can dexterously perform simple physical tasks, such as navigating a tree branch, planting a seed in soil, or even peeling a banana, involving gross and fine motor skills, are hard to find. Why is it that despite the abundance of agents that can write convincing computer code and produce tertiary level prose, autonomous machines operating in the physical domain, particularly out in the natural world, seem beyond reach?

In the evolutionary timescale, human language is a relatively new phenomenon, originating just about 135k years ago (Miyagawa et al., 2025). In contrast, organisms with complex limbs and digits, such as tetrapods, have been around much longer, at

(a) A Bossou chimpanzee infant using a hand crafted leaf-spoon to drink water from a tree cavity, Guinea, West Africa, from (MacKenzie et al., 2025).

(b) A spider monkey, native to tropical forests of South America, balancing by its tail, (Sharp, 2023). (c) Human reaching for an orange through foliage, (Jensen, 2025).

Figure 1.1: Complex manipulations in the natural world.

least since the Devonian period, some 400 million years ago (Coates & Clack, 1990), providing an ample headstart. This era is also characterised by the emergence of land-based flora, turning the surrounding environment from a semi-arid landmass to one teeming with diverse plants, ferns, and trees. The co-evolutionary innovations from the interplay among these early organisms, their environment, and each other, are the genetic predecessors of the dexterous manipulation skills we see in the natural world today, see Fig. 1.1. Then it is not the surprising that Bossou chimpanzees can fold leaves to form water scoops (Tonooka, 2001), spider monkeys can navigate tree branches using their tails, and human toddlers can push aside foliage to pick fruits, while the robots have a hard time catching up. Seamlessly operating in the messy, outdoor, natural world, with limited human supervision, may very well be the final frontier towards an autonomous human-level machine intelligence.

1.1 Natural Deformables

In the robotics parlance, the term *Deformable* refers to any entity that not only accelerates, but also changes shape, for instance, bends, twists, or folds, when an external wrench is applied (Arriola-Rios et al., 2020; H. Yin, Varava & Kragic, 2021). Examples of deformable objects include one-dimensional household items like ropes and cables, two-dimensional ones like fabrics and towels, and three-dimensional ones such as soft vegetables and sponges. When subjected to a distorting wrench, the final position of all points in a deformable body cannot be specified with a sequence of translations and rotations in $SE(3)$, as a time integral. What's more, the position of most, if not all, points in its configuration space must be known for a reasonable approximation of its final state. This behaviour is in contrast to the well-explored field of rigid bodies (Andrychowicz et al., 2020; Mahler et al., 2017), for instance, solid cubes or ceramic cups, where the overall arrangement in 3D space can be estimated with just one or two points, as long as the object geometry is known. In other words, deformable objects have very high, often infinite Degree of Freedoms (DOFs), rendering their state (i.e., the current configuration and shape at any given time) extremely high-dimensional. Consequently, precise model synthesis or state estimation for such a body is often intractable.

In addition to the object geometry, the nature of physical deformation is influenced by material properties, such as stiffness, elasticity, and friction. Therefore, estimation of the true physics parameters, or a close approximation, is a precursor to autonomous, intentional, deformable manipulation. However, despite robust physics knowledge, the system dynamics can be highly non-linear in that tiny changes in the wrench location or quantity may generate large, counterintuitive deformations, which are hard to predict accurately (Xu et al., 2022). To provide a vivid illustration, imagine poking a balloon (filled with air or water) at various locations and trying to predict the resultant surface topology.

Furthermore, a large proportion of feasible deformable configurations can include self-overlaps, where certain parts occlude the whole, and the global geometry may not be fully observable. As Jiang et al. (2024) has observed in the context of plasticine manipulation, humans offset this ambiguity by relying on multiple sensory inputs, vision to obtain a global configuration, and touch to ascertain the local deformation

patterns, suggesting a similar multi-modal pathway for robots as well.

In this thesis, we focus on a specific subclass of naturally occurring soft objects termed *Natural Deformables*, which include tree branches, plant stems, leaves, and vines. In contrast to household items, such as cables or clothes, natural deformables have additional pertinent characteristics. First, they exhibit non-uniform geometry, for instance, plant shoots are irregular with varying thickness and length at various levels. Similarly, uneven foliage distribution, asymmetrical crowns, and variable forking patterns all render shape modelling for plants problematic (Prusinkiewicz & Lindenmayer, 2012). Second, unlike cables or ropes, different branches within the same plant may have varying stiffness and elasticity, i.e., they have highly irregular dynamics. Third, they present axial disparities in deformation characteristics; for instance, stems may bend along certain axes but not others. Finally, unlike their synthetic counterparts, the material composition of natural objects, which in turn affects the deformation, changes with time and is harder to quantify. As an example, the water content in a branch at a given instant directly influences the maximum external force it can tolerate without rupture.

To summarise, a few challenges in manipulating deformable objects are as follows:

- *High-dimensional, continuous configuration space*;
- *Lack of precise physics knowledge* & *intractable model synthesis*;
- *Complex and non-linear dynamics*;
- *Non-trivial state representation*;
- *Partial observability*.

Beyond the above, natural deformables present further challenges:

- *Non-uniform geometry*;
- *Asymmetric dynamics with axial disparities*;
- *Time-varying material composition*.

Therefore, robotic manipulation of natural deformable materials is substantially more arduous. Nevertheless, irrespective of deformability, operating robots in the natural environment comes with its own pitfalls and possibilities.

1.2 Robotics in the Field

Much of modern robotics research focuses on laboratory settings or industrial automation tasks with strict interaction boundaries. However, the potential benefits offered by autonomous machines in outdoor environments appear substantial; some of the current field robotic applications can be seen on Fig. 1.2. These automated platforms integrate a diverse array of electrical sensors, such as RGB-D cameras, Global Positioning System (GPS), Light Detection and Ranging (LiDAR), and Inertial Measurement Unit (IMU)s, along with modern data analytics tools for mapping the surroundings and localising its own position within the environment.

In the case of agriculture, robotic automation has seen a sharp increase lately, predominantly to mitigate labour shortage, improve worker safety, and raise productivity. Mobile robots have been used for land preparation, sowing, and plowing under GPS guidance while Unmanned Aerial Vehicle (UAV)s, equipped with vision systems, have been used for aerial crop monitoring and spraying pesticides, fertilisers, and herbicides (Oliveira, Moreira & Silva, 2021a). Still other systems, with built-in computer-vision algorithms (e.g. CNNs) are used for weed identification and segregation (C. Wang et al., 2025). Crucially, a variety of automated grippers, pickers, and clippers are used for robotic harvesting of fruits and vegetables (e.g., Fig. 1.2 (a)-(b) shows apple and sweet pepper harvesting systems), aided by machine learning techniques (Zhou et al., 2022).

Likewise, automated robotic systems for forestry and woodland management help improve sustainability and human safety. They can monitor biodiversity, survey tree health, and can track illegal logging for habitat preservation (Pollayil et al., 2023). More advanced systems like drone seeders and ground sapling planters aid reforestation efforts (Oliveira, Moreira & Silva, 2021b). Notably, they also assist in wildfire prevention and firefighting in terrains hostile to humans. Ground robots, such as the one shown in Fig. 1.2 (c) from the SEMFIRE (Couceiro et al., 2019) project, can create fire breaks by removing organic flammable material. Other recent proposals include unmanned extinguishers, fire reconnaissance, and autonomous delivery and evacuation modules (Gromek & Lowe, 2025). Beyond forestland, robotic platforms have been used in other natural environments for cave exploration (Hudson et al., 2022), marine archaeology (Khatib et al., 2016), and extra-terrestrial exploration (T. Zhang et al., 2019) as well.

(a) A multi-arm sweet pepper harvesting framework (Lenz et al., 2024).

(b) An apple picking arm mounted on a drone, (Tevel, 2023).

(c) A wildfire prevention and forest landscaping robot (Ferreira et al., 2023).

(d) A UAV-UGV pair performing cave exploration (Hudson et al., 2022).

Figure 1.2: Field Robotic Applications.

1.3 Role of Uncertainty

A pivotal issue when operating robots, particularly outdoors, is the *uncertainty* or ambiguity inherent in nature. It is a term used to characterise the likelihood of actions taken by a robot and the consequent effects on the surrounding environment and on the robot itself. Physical sources of uncertainty (Senanayake, 2024), in the context of natural deformables, could include:

- *Environmental Variability*: The physical environment is dynamic with continuous spatio-temporal changes. For example, when trying to pick a fruit, the surrounding branches, foliage, and the fruit itself might move around in irregular, non-repeatable patterns.
- *Sensor limitations*: Typical sensors used for robotic manipulation, such as RGB-D cameras, LiDAR, and GPS suffer from a variety of limitations. As an example,

RGB-D cameras perform poorly when capturing reflective or dark surfaces, such as dense foliage, often producing noisy depth data resulting in poor 3D reconstruction. Similarly, the sensor performance may be adversely affected by outdoor lighting changes, rain, fog, or snow.

- *Partial observability*: The target object, for example, a fruit to be grasped, may be occluded by foliage, protruding limbs, or even other parts of the manipulator, limiting the signal input to the sensors. Notably, Zhou et al. (2022) have found that the success rates of modern fruit harvesting methods, in the presence of partial occlusions are about 50-75%, which falls to 5% in the case of complete occlusions.

- *Limited Generalisation*: In practice, machine learning models (described in Section 1.4) are trained on limited data and may not generalise well to novel scenarios during inference. For instance, a computer vision model that is trained to identify strawberries may not be able to detect other fruits, due to changes in colour, texture, or shape.

As an illustration, consider a robotic manipulator attempting to move a palm-sized rock fragment lying in the sand, with a fixed force (e.g., 5 Newtons), similar to the setting shown in Fig. 1.3. The resultant distance the rock moves would be affected by diverse variables: a) the attributes of the rock, such as the mass and surface geometry, b) attributes of the robot, such as the sensor noise, c) attributes of the terrain, such as the elevation and surface dampness, and d) the interaction attributes, such as the specific contact points, friction coefficients, among others. Performing this experiment would quickly reveal two parallel, yet distinct, influences on the sources of uncertainty. First, while the physics itself is

Figure 1.3: An experimental setup from Bandyopadhyay et al. (2024), depicting autonomous operations in a seismic activity zone.

deterministic, predictions of the target rock positions are guaranteed to be off, due to insufficient knowledge of the aforementioned parameters governing it (the unknown unknowns). Second, if the experiment is repeated multiple times, it is all but certain that the final state of the rock will differ each time due to the inherent randomness in nature (the known unknowns). To employ precise terminology, the two components contributing to uncertainty in robotic manipulation (Kroemer, Niekum & Konidaris, 2021; Senanayake, 2024) are:

- *Epistemic uncertainty*: Also called the systematic uncertainty, this component arises from an imperfect model of the world; for instance, from the lack of accurate parameters governing the physics (e.g., stiffness, friction). This component is reducible with additional training data or using interactive perception or with improved model synthesis.

- *Aleatoric uncertainty*: Also called statistical uncertainty, this component stems from the intrinsic stochasticity in the sensors, robot actuators, and the physical environment; therefore, it is not irreducible. Intuitively, it is often associated with a coin-flipping experiment, in which the outcome remains unpredictable irrespective of any additional information incorporated.

However, given the physics, the chances of the rock travelling 100 meters or not moving at all are both unlikely outcomes, whereas it moving a few centimetres is much more plausible. In other words, the outcomes may be expressed in probabilistic terms; for instance, as a normal distribution with a mean and a variance, which can act as a measure of the uncertainty. Such a probabilistic quantification of the uncertainty allows the robots to generate competing hypotheses, evaluate their likelihoods, and make optimal decisions, which is vital in building reliable and safe robotic systems. Crucially, robots that learn from the data, either in hindsight or interactively, in the hope of generalising to unseen situations, must account for such ambiguity, noise, and randomness implicit in the data.

1.4 Machine Learning for Control

The term Machine Learning (ML) encompasses a disparate array of methods and algorithms involved in learning rule sets from known data, that can generalise to unseen data, and accomplish tasks without explicit hand-coded instructions. At the broadest level, it includes three directions: a) Supervised Learning, b) Unsupervised Learning, and c) Reinforcement Learning (Russell & Norvig, 1995). Both supervised and unsupervised frameworks operate on large datasets, in hindsight. Given sufficient pairs of input features and output labels (x_i, y_i), termed a labelled dataset, the former learns a mapping function $f : X \rightarrow Y$ to accurately predict y_t given an unseen x_t (e.g., image classifiers or regression models). On the other hand, the unsupervised model aims to identify hidden patterns in unlabelled datasets, $X = \{x_1, x_2, \dots, x_n\}$ by grouping similar points, purely as a descriptive procedure (e.g., clustering). In contrast, Reinforcement Learning (RL) is prescriptive and feedback-driven, ideal for robotic tasks; therefore, it is a recurring concept throughout this thesis. It attempts to learn an optimal policy (a strategy) $\pi : S \rightarrow A$, by interacting with the environment iteratively through trial and error, searching for the best action $a \in A$ to take when the environment is in state $s \in S$, such that a reward signal from the environment is maximised over the long term.

The majority of recent advancements in ML stem from the Deep Learning (DL) paradigm. Essentially, it refers to multilayer neural networks, inspired by animal brains, that can generate multi-level abstractions of the training data (LeCun, Bengio & Hinton, 2015). To learn the rule set of the data and approximate complex functions, it iteratively optimises a network of weights (termed model parameters), through back propagation. Deep networks have been widely used for speech recognition (Hinton et al., 2012), image recognition (He et al., 2016; Krizhevsky, Sutskever & Hinton, 2012), and generative modelling (Goodfellow et al., 2020; Kingma, Welling et al., 2013). Deep Reinforcement Learning (DRL) is a related extension, which integrates RL principles with deep neural networks; specifically, the network acts as a function approximator, enabling the agent to generalise from observed environment states in the past, through induction, to unseen states in the present (Russell & Norvig, 1995).

1.4.1 Robotic Control

Robots are built to move around the environment, physically interact with objects, and manipulate them sufficiently to perform tasks, such as crop monitoring, vine pruning, and fruit picking. This necessitates controlling the robot through a variety of tools and techniques to regulate its behaviour towards the required objective, commonly referred to as *Robotic Control* (Siciliano et al., 2009), a cornerstone of robotic manipulation. Classical control techniques (for e.g., Proportional–integral–derivative (PID), Linear Quadratic Regulator (LQR), or Model Predictive Control (MPC)), in robotics or other dynamical systems, assumes the availability of a known (or approximated) model; therefore, provide formal stability guarantees (Brunke et al., 2022), but performs poorly under uncertain operating conditions.

In contrast to these model-driven approaches, Machine Learning offers a data-driven framework that is robust to uncertainties. In practice, however, a hierarchy of controllers operate simultaneously to execute a task. For instance, a high-level trained RL policy may output the desired trajectory, i.e., a sequence of positions or velocities, which a low-level controller like PID may track by regulating robot actuator torques or joint angles (Kroemer, Niekum & Konidaris, 2021) by sending signals to the onboard Digital Signal Processors (DSPs). In this thesis, we focus on high-level control tasks, such as trajectory learning and decision-making under uncertainty, utilising data-driven approaches, with a particular emphasis on Deep Reinforcement Learning. DRL has been extensively used for robotic manipulation for in-hand manipulation (Akkaya et al., 2019; Andrychowicz et al., 2020), fabric manipulation (Jangir, Alenya & Torras, 2020), and industrial assembly tasks (B. Tang et al., 2023), among others.

1.4.2 System Identification

A key distinction, pertinent to this work, is between *Model-Free* and *Model-Based* learning frameworks (De Oliveira, 2023). As the name implies, model-free approaches obtain a control policy directly through interaction without explicitly constructing a model of the dynamical system. In general, most DRL methods discussed above follow the model-free route, primarily because they are simpler to implement. A relevant example from agriculture could be a robotic arm forming a visuo-motor policy π to establish the actuator torques ($a \in A$), at each time-step, for inducing a specific degree

of curvature in a plant stem ($s \in S$), directly from camera feedback, through trial and error. In this case, the physical attributes of the plant, the robot, and the interaction are implicitly encoded within the control policy, while the symbolic physical laws governing the deformation itself are effectively ignored. However, model-free methods are notoriously sample inefficient (Memmel et al., 2024), requiring a large number of experimental trials or expert demonstrations to converge well.

In contrast, model-based approaches, such as *System Identification* methods, initially learn a model of the dynamics. This learned model can then be leveraged to generate experiences and evaluate decisions (Tedrake, 2022). In the preceding agriculture example, this might involve learning parameters governing the deformation at first, for instance, the modulus of elasticity (Young's modulus). As this parameter describes the plant material's resistance to elastic deformation, a dynamical system can be subsequently approximated to estimate the torque-curvature relationship. In fact, a key research question we consider in this thesis is whether a subset of similar parameters can be learned, with *Parameter Inference* techniques, to predict the response of a tree branch under physical stress. However, on the flip side, such system identification approaches may not scale well to high-dimensional complex models; therefore, the jury is still out on the comparative merits of model-free vs model-based learning.

1.4.3 Simulation-Driven Learning

Regardless of the learning objective (control policy or the dynamical model), real-world sample collection is expensive, time-consuming, and potentially unsafe. Consequently, physics simulators such as Isaac Gym (Makoviychuk et al., 2021), Gazebo (Koenig & Howard, 2004), and MuJoCo (Todorov, Erez & Tassa, 2012), have emerged as a feasible alternative, providing a controlled and repeatable learning environment. These high-fidelity simulators incorporate natural physical laws (e.g., gravity, friction, and sensor noise) for a realistic representation of the real-world conditions. Furthermore, in some simulators, particularly from the Isaac family (Makoviychuk et al., 2021; Mittal et al., 2023), both the simulation and the learning process can be scaled up on modern GPUs, through parallelisation, to meet the exorbitant sample demands of DRL. An illustrative example for such a parallel training is shown in Fig. 1.4.

However, deploying a simulation-trained policy in the real world is fraught with

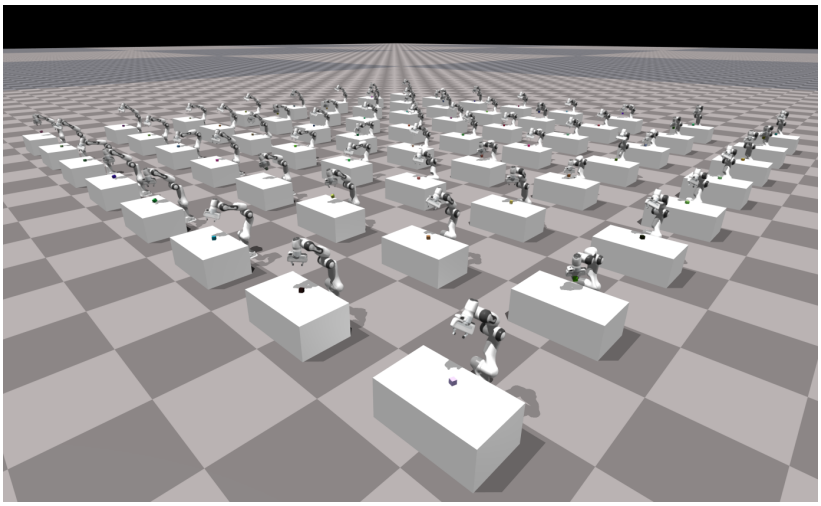


Figure 1.4: Parallel simulation of Franka Panda robots performing cube manipulation, demonstrating scalable training and control (Makoviychuk et al., 2021).

additional challenges, notably the so called *sim-to-real gap* (Peng et al., 2018), which stems from differences between an ideal simulation world and the uncertain reality. It is often contrasted to the *real-to-sim gap* (Antonova et al., 2022; Heiden et al., 2022), which refers to similar discrepancies when trying to replicate real-world behaviour in simulation, i.e., in the opposite direction. In either case, the discrepancies can originate from modelling (e.g., mass, inertia, material properties), sensing (e.g., lighting conditions, camera parameters), or actuation (e.g., robot joint friction, damping, servo noise), among others (Da et al., 2025). Nevertheless, simulation-driven learning has proved successful across a variety of real-world tasks, such as dexterous in-hand manipulation (Andrychowicz et al., 2020) and agile locomotion (Hwangbo et al., 2019). Therefore, throughout this thesis, simulation plays a critical role in designing, training, and evaluating control policies for manipulating natural deformable objects that are transferrable to the real world.

1.5 Research Questions

Considerable progress has been made in manipulating rigid body objects within constrained spaces such as laboratory environments and warehouses (Andrychowicz et al., 2020; Mahler et al., 2017); however, this success has not been fully translated to natural deformable settings, despite many potential benefits (Kurtser & Edan, 2020; Zhou et al., 2022). A significant gap persists in the literature pertaining to geometry simulation, dynamics modelling (Yandun, Silwal & Kantor, 2020), state representation (C. H. Kim et al., 2024), and uncertainty quantification, when natural deformable objects such as tree branches, plant stems, and vegetation are involved. Furthermore, current attempts at interactive learning are restricted due to the prohibitive cost of physical sampling (C. Tang et al., 2025), the sim-to-real (Peng et al., 2018) and real-to-sim (Antonova et al., 2022; Heiden et al., 2022) gaps, and the lack of high-quality multi-modal perception feeds (M. A. Lee et al., 2019). In summary, autonomous manipulation of natural deformable objects is still in its infancy, and this thesis aims to provide some foundational answers. Specifically, we consider the following research questions.

- *Simulation*: What is the best way to simulate realistic plant geometries and occlusion patterns, replete with deformable dynamics, and amenable to robotic manipulation?
- *Dynamics*: Can we learn a set of physics parameters from the real world to simulate the uncertain behaviour of tree branches under physical stress?
- *Learning*: How to learn control policies in simulation, that are transferrable to the real world and robust to uncertainties, to manipulate tree branches?
- *Perception*: What sensing modalities are appropriate for manipulating natural deformables? To what extent is vision limited, and what are the alternatives?
- *Representation*: How to transform and encode the state of natural deformables and other scene objects, to accelerate the learning process?
- *Cost*: For each question above, how can simulator capabilities and algorithmic improvements be best leveraged to drive down the overall operational expenditure? For instance, can we minimise equipment costs, reduce training duration, or mitigate plant damage?

1.6 Contributions

The key contributions of this thesis are organised as three core chapters, explaining our novel methodologies and observations, briefly summarised as follows:

1.6.1 Learning to Simulate Tree-Branch Dynamics

In Chapter 3, we build a model for the dynamics of tree branches under robotic manipulation, in a real-to-sim context. Here, we take a probabilistic inverse inference approach, aided by coarse-grained spring models, to accurately replicate real-world branch behaviour in simulation. Our novel non-parametric approach, termed *Neural Network guided Stein Variational Gradient Descent (NNSVGD)*, can additionally incorporate biological assumptions as learned prior distributions in a Bayesian setting. Through elaborate experiments and comparisons, we demonstrate that the inferred posterior and the predicted deformation trajectories are robust to real-world uncertainties, such as sensor noise, contact instabilities and gripper location errors.

1.6.2 Gentle Branch Manipulation with Contact-Awareness

In Chapter 4, we focus on simulating the plant geometry and learning a control policy, suitable for a sim-to-real transfer, with a particular emphasis on minimising plant ruptures in the real world. First, we develop a procedural forest simulator, with realistic morphology and dynamics, based on the L-system formalism from botanical literature. To address the limitations from vision, we build novel contact detection features from proprioceptive time-series inputs and learn a DRL policy for manipulation. The proposed *Proprioceptive Contact-Aware Policy (PCAP)* can transfer zero-shot from simulation to real, manipulating plant stems with novel geometry and dynamics unseen during training, while exhibiting unique contact minimisation strategies.

1.6.3 Deformable Cluster Manipulation with Whole-Arm

In Chapter 5, we continue to focus on learning a control policy suitable for a sim-to-real transfer, but emphasising manipulation with the whole arm instead of just the end-effector, while considering the natural deformables as a collection rather than

as individual elements. In this work, we first integrate camera point clouds with the proprioceptive contact detector from Chapter 4, to enable a multi-modal feedback. Next, we leverage a distributional state representation, aided by kernel operator theory, to formulate a computationally cheap DRL policy. In addition, we propose a novel occlusion heuristic for de-occlusion and target exposure tasks. Finally, with comprehensive simulation and real experiments, we demonstrate the practical applicability of this framework in a novel setting to clear electrical power lines of overhanging foliage.

1.7 Outline

Chapter	Research Question(s)	Key Technique
Chapter 3	*Simulation*, *Dynamics*	System Identification
Chapter 4	*Simulation*, *Learning*, *Perception*, *Cost*	Reinforcement Learning
Chapter 5	*Learning*, *Perception*, *Representation*, *Cost*	Reinforcement Learning

Table 1.1: **Chapter-wise mapping to research gaps.** A summary of how each chapter aligns with research questions from Section 1.5, and the key technique used.

First, a brief background of the relevant material is introduced in Chapter 2. It includes the notations, theoretical frameworks, and the conceptual background underpinning our research. Thereafter, the thesis is organised as three core chapters, explaining our novel methodologies and contributions, as described in the preceding section. Each contribution section, i.e., Chapter 3, Chapter 4, and Chapter 5, presents the problem setup, experimental findings, analysis, and detailed literature review specific to the problem. A summary of how each contribution chapter maps to the category of research questions from Section 1.5, along with the dominant technique used in the chapter, is shown in Table 1.1. Following the core sections, in Chapter 6, we summarise the contributions, acknowledge limitations, and suggest avenues for future research. Finally, additional materials, such as pointers to the supplement, videos, and websites, are provided in the Appendix.

Probability theory is nothing but
common sense reduced to calculation.
Pierre Simon Laplace

CHAPTER 2

Background

In this chapter, we briefly outline the fundamental notations, theoretical frameworks, and conceptual background that form the basis for the methods this thesis proposes.

2.1 Probability Theory

Probability theory is a central framework to describe, quantify, and analyse the notion of uncertainty (as described in Section 1.3), in robotics and other fields. Notions from this section is reused across all core chapters in our work. A few relevant fundamentals are summarised below, for a comprehensive review of probability from robotics and machine learning perspectives, see Thrun (2002) and Bishop (2006), respectively.

Probability Distribution: When an uncertain quantity, for instance, a robot sensor measurement riddled with noise, is modelled as a random variable X, the *Distribution* $p(x)$, quantifies the chances of probable outcomes (e.g., possible readings). For an experiment with discrete X, such as a coin flip, the *Probability Mass Function (PMF)* $p(x)$ is defined as $p(x) = \mathbb{P}(X = x)$, where $\mathbb{P}$ is the underlying probability measure function which assigns probabilities to events, and it satisfies the condition $\sum_x p(x) = 1$. For continuous X, such as a robot pose, $p(x)$ is a *Probability Density Function (PDF)* satisfying $\int p(x)\,dx = 1$.

Expectation: The *Expectation* of a continuous random variable X, which has a probability density of $p(x)$, is defined as the average value weighted according to the distribution, often serving as a point estimate (e.g., mean robot pose) and expressed as:

$$\mathbb{E}_{x\sim p}[X] = \int x p(x)\,dx. \tag{2.1}$$

Likelihood: A related notion is that of *Conditional Probability Distribution*, describing the chance of an outcome, given that another event has already occurred. To illustrate, given a true robot pose x, the probability of observing a noisy sensor reading of z could be quantified as $p(z|x)$. *Likelihood* specifies the reverse direction $p(x|z)$, asking the question: if a sensor reading of z is observed, how likely is that the actual robot pose is x?

Bayes' Rule: A seminal concept in probabilistic inference (Chapter 3), the *Bayes' theorem* connects *prior* and *posterior* probabilities given new evidence:

$$p(x|z) = \frac{p(z|x)p(x)}{p(z)} = \frac{p(z|x)p(x)}{\int p(z|x')p(x')\,dx'}. \tag{2.2}$$

Here, the prior distribution, often called prior *beliefs* p(x), represents the initial knowledge, for example, a belief about the robot's actual position x, before any sensor measurement. Once a reading z is observed, the belief about x is updated to the posterior probability density $p(x|z)$, based on the likelihood $p(z|x)$. The denominator $\int p(z|x')p(x')\,dx'$, termed the normalising *evidence*, is computed over all possible outcomes; therefore, it is intractable.

2.2 Kernel Methods

This section serves as a brief introduction to *Kernel Methods* and their associated terminology, which form the basis of our inference method in Chapter 3 and the representation technique used in Chapter 5. For a comprehensive survey of the topic, refer Ghojogh et al. (2021) and Schölkopf, Smola, Bach et al. (2002).

Kernel methods offer a rich framework to approximate complex non-linear functions by lifting the input data to a high-dimensional feature space, where the mathematical operations are more tractable. The key insight of the *feature map* $\varphi : \mathcal{X} \to \mathcal{F}$ is that data points in the higher-dimensional target space $\mathcal{F}$ may become linearly separable, making them amenable to linear classification, for example. The target space $\mathcal{F}$ is typically an infinite-dimensional *Hilbert Space*, equipped with inner products $\langle \_, \_ \rangle$. In this setting, a positive definite kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, can be used to evaluate the similarity between points, i.e., $k(x, x') = \langle \varphi(x), \varphi(x') \rangle_{\mathcal{F}},\ x, x' \in \mathcal{X}$ in the Hilbert space, without explicitly computing the mapped points, a technique often referred to as the *kernel trick* in machine learning literature.

2.2.1 Reproducing Kernel Hilbert Spaces

The aforementioned Hilbert space, for example, the common Euclidean space $\mathbb{R}^d$, is a vector space, which is considered complete under the inner product induced metric. A *Reproducing Kernel Hilbert Space (RKHS)* then is a subspace $\mathcal{H}$ designed for functions, for which point evaluation constitutes a continuous linear functional. In other words, an RKHS of functions $f : \mathcal{X} \to \mathbb{R}$ is uniquely determined by a positive definite kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, such that the evaluation of the function at a specific point x can be expressed as an inner product in this space with the given kernel function.

Reproducing Kernel: The notion of expressing the evaluation at a point x itself as an inner product with the *canonical feature map* $k(\cdot, x)$ is called the *Reproducing Property*, defined as:

$$\forall f \in \mathcal{H},\ \forall x \in \mathcal{X} :\ f(x) = \langle f(\cdot),\ k(\cdot, x) \rangle_{\mathcal{H}}, \tag{2.3}$$

Note that x is a parameter in $k(\cdot, x)$ and $\varphi(x) = k(\cdot, x)$ and $\varphi(x') = k(\cdot, x')$.

Finite Feature Representation: The applicability of RKHS projections in machine learning stems from the *Representer Theorem*, first discovered by (Kimeldorf & Wahba, 1971). Assume we are searching for an optimal $f^* \in \mathcal{H}$, for instance, a minimal regularised empirical loss over a set of data points: $\{(x_i, y_i)\}_{i=1}^n$. In a classical regression setting, with a loss $\mathcal{L}$ and a regularisation coefficient $\lambda > 0$, this could look like:

$$f^* = \min_{f \in \mathcal{H}} \sum_{i=1}^{n} \mathcal{L}(y_i,\, f(x_i)) \;+\; \lambda \left\| f \right\|_{\mathcal{H}}^2. \tag{2.4}$$

Intuitively, the representer theorem states that, this optimal (but intractable) solution can be alternatively constructed by taking a weighted sum of kernel evaluations between the input points, thereby reducing the infinite-dimensional search to finding a finite vector: $\alpha \in \mathbb{R}^n$.

$$f^* = \sum_{i=1}^{n} \alpha_n k(x, x_n). \tag{2.5}$$

2.2.2 Popular Kernel Functions

Well-known kernels that yield smooth similarity measures and follow the stationarity constraint $k(x, x') = k(x - x')$, include the Gaussian Radial Basis Function kernel (RBF):

$$k(x, x') = \exp\left(-\frac{||x - x'||^2}{2\sigma^2}\right),\ \sigma > 0 \tag{2.6}$$

and the Laplacian (exponential) kernel:

$$k(x, x') = \exp\left(-\frac{||x - x'||}{\ell}\right),\ \ell > 0. \tag{2.7}$$

The tunable bandwidth parameters, σ for the Gaussian RBF kernel, and ℓ for the Laplacian kernel, control the smoothness as a trade-off between capturing local details and global patterns. Throughout this work, we utilise the RBF kernel, which is considered highly effective for mapping complex, non-linear inputs into a more amenable feature space of infinite dimensions.

2.2.3 Kernel Mean Embedding

Akin to point mappings, the notion of RKHS projection can be extended to the space of probability distributions as well (Muandet et al., 2017). In this idea, the *Kernel Mean Embedding (KME)* or the feature map $\varphi(\mathbb{P})$ of a probability measure $\mathbb{P}$ can be computed as:

$$\varphi(\mathbb{P}) = \mu_{\mathbb{P}} := \mathbb{E}[k(\cdot, x)] = \int k(\cdot, x)\, d\mathbb{P} \in \mathcal{H}. \tag{2.8}$$

The kernel mean operator projects a distribution to a single mean function in $\mathcal{H}$ (see Fig. 2.1). For a subclass of kernels, called characteristic kernels, which includes the commonly used RBF, the statistical features of $\mathbb{P}$ are retained after embedding it into $\mathcal{H}$; more precisely, there is no information loss during the projection.

Injectivity Property: Given two probability measures, $\mathbb{P}$ and $\mathbb{Q}$, the feature map is *injective* for characteristic kernels, i.e., $\mu_{\mathbb{P}} = \mu_{\mathbb{Q}}$ if and only if $\mathbb{P} = \mathbb{Q}$. In other words, distributional differences will carry over to the RKHS; consequently, KME can be used to formulate distance metrics between distributions, such as the Maximum Mean Discrepancy (MMD) metric,

$$\mathrm{MMD}(\mathbb{P}, \mathbb{Q}) = \|\mu_{\mathbb{P}} - \mu_{\mathbb{Q}}\|_{\mathcal{H}}. \tag{2.9}$$

Empirical Approximation: The term $\int k(\cdot, x)\, d\mathbb{P}$ from Eq. (2.8) can be loosely interpreted as the expected kernel similarity between x and all points of the true measure $\mathbb{P}$ in RKHS. However, given that the actual distribution is not often available in practical settings, the mapping $\mu_{\mathbb{P}}$ cannot be computed; instead, it must be empirically estimated from finite i.i.d samples $x_i \sim \mathbb{P}$. If N samples are drawn, the empirical approximation $\overline{\mu_{\mathbb{P}}}$ can be written as:

$$\overline{\mu_{\mathbb{P}}} := \frac{1}{N}\sum_{i=1}^{N} \varphi(x_i) = \frac{1}{N}\sum_{i=1}^{N} k(\cdot, x_i). \tag{2.10}$$

Furthermore, one can formulate empirical metrics, such as $\left\|\overline{\mu}_{\mathbb{P}} - \overline{\mu}_{\mathbb{Q}}\right\|_{\mathcal{H}}$ for distributional divergence, similar to Eq. (2.9).

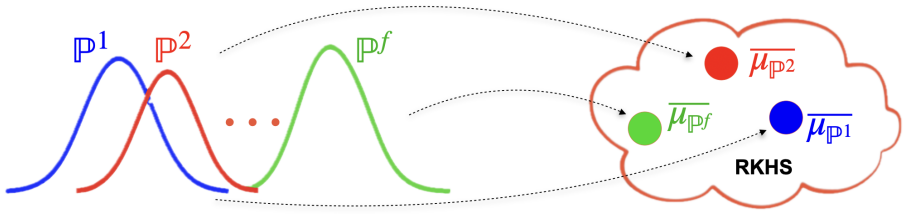


Figure 2.1: Mapping a set of distributions into an RKHS via an expectation operation

2.2.4 Random Fourier Features

A notable challenge of kernel representation is its scalability. The formulation in Eq. (2.10) can be computed with a finite dimensional $N \times N$ gram matrix. However, this approach is not scalable for large N, within the context of modern big-data based machine learning. By contrast, a scalable alternative is to represent the kernel function through its spectral representation, specifically, with a *Random Fourier Feature (RFF)* approximation (Rahimi & Recht, 2007).

Spectral Representation: Kernels satisfying a stationarity constraint $k(x, x') = k(x - x')$, such as the RBF or the Laplace kernel, can be defined as a Fourier transform of a non-negative measure, through *Bochner's Theorem*,

$$k(x, x') = \int p(\omega)\, e^{-i\omega^\top (x - x')}\, d\omega. \tag{2.11}$$

In this case, $p(\omega)$ is called the spectral density of the kernel and $\omega_r \sim p(\omega)$ denotes the frequencies sampled from the spectral density. Further, this expression can be approximated and simplified with Monte Carlo sampling (Warren & Ramos, 2024):

$$k(x, x') \approx \frac{1}{R} \sum_{r=1}^{R} \cos(\omega_r^\top (x - x')). \tag{2.12}$$

Expressing, $k(x, x') := \varphi(x)^\top \varphi(x')$, the following feature map $\varphi(x)$ would evaluate to the same estimate as above in Eq. (2.12).

$$\varphi(x) = \frac{1}{\sqrt{R}} \begin{bmatrix} \cos(\omega_1^\top x) \\ \sin(\omega_1^\top x) \\ \vdots \\ \cos(\omega_R^\top x) \\ \sin(\omega_R^\top x) \end{bmatrix}. \tag{2.13}$$

In other words, the inner product of the finite-dimensional RFF feature map, $\varphi(x)$ above, approximates the kernel, with the quality of approximation improving as R increases. Crucially, this finding allows many kernel algorithms (e.g., Gaussian Process regression) to be approximated in $\mathcal{O}(R^3)$ time instead of $\mathcal{O}(N^3)$, which is computationally efficient when $R << N$ (Lázaro-Gredilla et al., 2010). Finally, the empirical kernel mean from formulation Eq. (2.10) can be approximated with the RFF as:

$$\overline{\mu_{\mathbb{P}}} := \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\sqrt{R}}\Big[\cos(\omega_1^\top x_i),\ \sin(\omega_1^\top x_i),\ \dots\dots,\cos(\omega_R^\top x_i),\ \sin(\omega_R^\top x_i)\Big]. \quad (2.14)$$

On the application front, kernel methods have found widespread applicability in robotics. To name a few examples, Marinho et al. (2016) treat manipulator trajectories as functions in RKHS, to optimise paths through gradients, while Vien, Englert and Toussaint (2016) model RL policies and value functions in RKHS, yielding compact and flexible controllers. The distributional equivalent, KME, has been used for demonstration-based learning (K.-E. Kim & Park, 2018), whereas KME and MMD have been used for particle filtering and parameter inference for robot simulators (Adhikary & Boots, 2022). The scalability advantage of Fourier approximation with RFF has been leveraged for Simultaneous Localization and Mapping (SLAM) (Kapushev et al., 2021), real-to-sim parameter estimation (Antonova et al., 2022), and for high-dimensional control tasks (Watson & Peters, 2023).

In this thesis, our non-parametric inference technique NNSVGD to estimate the physics parameters of tree branches, described in Chapter 3, performs gradient descent in RKHS to minimise the trajectory deviations between simulation and the real world. Furthermore, in Chapter 5, we extensively utilise a distributional representation of the visual point cloud generated via KME and RFF as input observations to our DRL policy learning framework. We demonstrate the kernel representation to be sufficiently informative and highly efficient for both training and real-time inference.

2.3 Probabilistic Inference

Uncertainty is fundamental to nature, and its role in operating robots outdoors was briefly discussed in Section 1.3. Probabilistic inference approaches provide a formal framework for reasoning about this uncertainty, and they underpin many learning and perception frameworks in robotics. This section briefly summarises key concepts from Bayesian, Variational, and Stein-based inference methods, with a particular emphasis on their relevance to this thesis and robotics in general.

Statistical Inference deals with estimating unknown quantities, such as model weights, physics parameters, or latent variables, from known data. In a robotic setting, this could manifest as inferring the true orientation of an object from noisy camera sensor feedback or updating beliefs about the robot's own state and the surrounding environment as new uncertain data arrives. To illustrate, let x denote a latent variable representing the current robot state, z the noisy sensor measurement, and $p(z|x)$ the probability of observing the sensor reading given the true state. Here, the goal of inference is to compute either an approximated version of the true robot state $\hat{x}$ or the conditional probability $p(x|z)$, assuming x to be a PDF rather than a fixed value. In this context, a few fundamental distinctions in terminology are listed below:

Frequentist vs Bayesian: Frequentist and Bayesian approaches offer two alternate pathways to handle uncertainty in statistical inference. The former interprets the probability of an event as the long-run frequency when repeated trials are performed. In the preceding example, this might involve estimating the fixed, true value of the robot state $\hat{x}$, for instance, via Maximum Likelihood Estimation (MLE)(Thrun, 2002):

$$\hat{x}_{\mathrm{ML}} = \arg\max_{x} p(z \mid x). \tag{2.15}$$

On the other hand, Bayesian approaches consider probabilities as subjective; for example, it treats the robot state x as a PDF, incorporates prior beliefs about the state to estimate posterior belief $p(x|z)$, which in turn becomes the prior in the next time step, through the Bayes' rule from Eq. (2.2). A popular Bayesian approach, Maximum A Posteriori (MAP) estimation aims to approximate the true state by maximising the posterior probability $p(x \mid z)$.

$$\hat{x}_{\mathrm{MAP}} = \arg\max_{x} p(x \mid z) = \arg\max_{x} p(z \mid x)p(x). \tag{2.16}$$

Parametric vs Non-Parametric: Formally, the inference process assumes a set of assumptions or a statistical model that generates the observed data with latent variables x and observations z, with joint density $p(x, z) = p(x)p(z|x)$. The *parametric* approach assumes this joint density to conform to an established family of probability distributions (e.g., Gaussian) that can be described with a finite number of unknown parameters (e.g., the population mean μ or covariance Σ). In contrast, *non-parametric* approaches do not assume any specific functional form for the distributions, and the PDF shape is completely data-driven. While parametric models are simpler to implement, it is harder to make distributional assumptions about complex natural systems such as ours; consequently, we leverage the flexibility of non-parametric approaches.

2.3.1 Bayesian Inference

Bayesian Inference applies Bayes' rule to update posterior beliefs given new evidence, while accounting for prior knowledge. In a Bayesian model, a noteworthy distinction in notation is to use the parameters that define the shape of the distribution θ, rather than latent attributes z, as the unobserved random variable to be modelled.

$$p(\theta \mid x) = \frac{p(x \mid \theta)\, p(\theta)}{\int p(x \mid \theta)\, p(\theta)\, d\theta}. \tag{2.17}$$

The parameters θ could be the mean (μ) and the covariance (Σ) of a Gaussian distribution, or it could be the coefficients (slopes and intercept) in a linear regression setting. Here, $p(\theta)$ is the prior, $p(x \mid \theta)$ is the likelihood of the parameters, and $p(\theta \mid x)$ is the posterior to be estimated (MacKay, 2003).

However, in many cases, exact inference of the Bayesian posterior is either computationally expensive or plain intractable as it requires computing the normalisation term $\int p(x \mid \theta)\, p(\theta)\, d\theta$ over the entire domain of the distribution. Therefore, in practical settings, an alternate class of algorithms called Approximate Bayesian Computation (ABC) is used. Two well-known approaches in the approximate inference model are: *Markov Chain Monte Carlo (MCMC)* and *Variational Inference (VI)*.

2.3.2 Markov Chain Monte Carlo

The most classic of the inference techniques, MCMC is a sampling-based stochastic method. It aims to construct an ergodic Markov Chain whose equilibrium distribution is the posterior $p(\theta|x)$. Here, the term equilibrium (or stationary) implies that on repeated traversal through the chain, for example with a random walk, the distribution of states will converge to the target posterior.

A commonly used MCMC method is the three-step Metropolis-Hastings algorithm given below. At time step t, given a proposal distribution q and the current state $\theta^{(t)}$:

1. *Propose a candidate:*

$$\theta' \sim q(\theta' \mid \theta^{(t)}).$$

2. *Compute an acceptance probability:*

$$\alpha = \min\left(1, \frac{p(x \mid \theta')\, p(\theta')\, q(\theta^{(t)} \mid \theta')}{p(x \mid \theta^{(t)})\, p(\theta^{(t)})\, q(\theta' \mid \theta^{(t)})}\right).$$

3. *Accept or reject:*

$$\theta^{(t+1)} = \begin{cases} \theta' & \text{with probability } \alpha \\ \theta^{(t)} & \text{otherwise} \end{cases}.$$

Notice that in the second step above, to compute an acceptance probability, only the likelihood and prior terms are required, thereby overcoming the computational intractability problem associated with directly calculating the full posterior distribution. MCMC methods are asymptotically complete, i.e., under certain conditions and given sufficient iterations, they are guaranteed to converge to the target posterior. However, as they lack a sense of directionality between iterations (e.g., through gradients), they are inefficient in high-dimensional tasks, especially in robotics, where real-time inference is a pre-requisite. Multiple variants of MCMC, such as Stochastic gradient Hamiltonian Monte Carlo (SGHMC), have been proposed to overcome this limitation. For an authoritative review of MCMC and Monte Carlo methods in general, refer Shonkwiler and Mendivil (2009).

2.3.3 Variational Inference

In contrast to sampling-based methods, the VI approach approximates posterior densities through optimisation steps, without relying on stochasticity (Ganguly & Earp, 2021). It assumes a family of tractable distributions $\mathcal{Q}$, that can sufficiently approximate the target posterior $p(\theta \mid x)$. VI samples a variational distribution from this family, i.e., $q(\theta) \sim \mathcal{Q}$, and searches for an optimal q^* that minimises the Kullback-Leibler divergence (KL) to the true PDF:

$$q^*(\theta) = \arg\min_{q \in \mathcal{Q}} \mathrm{KL}(q(\theta) \parallel p(\theta \mid x)). \tag{2.18}$$

With a little mathematical gymnastics, it can be shown that minimising the above expression is equivalent to maximising an Evidence Lower Bound (ELBO) term $\mathcal{L}(q)$, defined as:

$$\mathcal{L}(q) = \log p(x) - \mathrm{KL}(q(\theta) \parallel p(\theta \mid x)), \tag{2.19}$$

or its expectation equivalent:

$$\mathcal{L}(q) = \mathbb{E}_{q(\theta)}[\log p(x, \theta) - \log q(\theta)]. \tag{2.20}$$

The key insight is that $\mathcal{L}(q)$ involves expectations with respect to $q(\theta)$ and doesn't require computing the normalisation term $p(x)$, removing the source of intractability. If $\mathcal{Q}$ is chosen as a parametric family (for e.g., Gaussian, Bernoulli), the expression Eq. (2.20) can be computed analytically or approximated efficiently, with gradients, for example. However, the chosen variational family $\mathcal{Q}$ directly determines the inference quality, and simpler parametric choices may not fully capture multi-modal target posteriors. In contrast to MCMC, VI loses the advantage of asymptotic completeness; however, it is computationally efficient, making it an attractive option for both large-scale datasets (Blei, Kucukelbir & McAuliffe, 2017) and time-sensitive robotics tasks, such as motion planning (Yu & Chen, 2023).

2.3.4 Stein Variational Gradient Descent

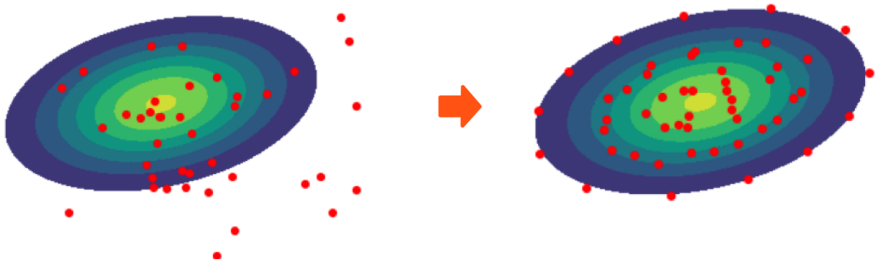


Figure 2.2: Initial vs final states of SVGD particles with the distribution contours.

A closely related inference technique that we rely on in Chapter 3 is *Stein Variational Gradient Descent (SVGD)*, originally proposed by (Q. Liu & Wang, 2016). It is a particle-based approach that takes inspiration from Variational Inference (Section 2.3.3), sampling-based methods (Section 2.3.2), and kernel theory (Section 2.2). The key idea is to start with a set of n random particles $\{\theta_i\}_{i=1}^n$, and iteratively transform them such that they form an equivalence to samples drawn from the target posterior, see Fig. 2.2. Given a small step size $\epsilon \in \mathbb{R}$, the update at time step t of the iteration is:

$$\theta_i^{t+1} \leftarrow \theta_i^t + \epsilon \phi^*(\theta_i^t) \;\; \forall i = 1, 2, ..., n. \tag{2.21}$$

The update direction ϕ is chosen to minimise the KL divergence between the empirical distribution (represented by the current state of the particles) and the target distribution. Leveraging Stein's identity, this update has a closed form involving the gradient of $\log p(\theta|x)$ and a kernel function k. In other words, if we assume $\phi \in \mathcal{H}$, where $\mathcal{H}$ is an RKHS defined by a kernel, such as an RBF: $k(\theta, \theta') = \exp(-\frac{||\theta-\theta'||^2}{2\sigma^2})$, the SVGD update can be viewed as a functional gradient ascent in that hyper space. The optimal ascent direction function ϕ^* can be written as:

$$\hat{\phi}^*(\theta) = \frac{1}{n}\sum_{j=1}^{n} k(\theta_j, \theta)\nabla_{\theta_j} \log\ p(\theta_j|x) + \nabla_{\theta_j} k(\theta_j, \theta). \tag{2.22}$$

This formulation may be viewed as an equilibrium between two conflicting forces: a) an attraction from the gradient ascent $\nabla_{\theta_j} \log\ p(\theta_j|x)$ to move the particles toward high density regions, and b) a repulsion from $\nabla_{\theta_j} k(\theta_j, \theta)$ to maintain diversity among particles. Mathematical sophistication aside, we focus on the following crucial insights:

- *Tractable*: The use of gradient in $\nabla_{\theta_j} \log\ p(\theta_j|x)$ eliminates the source of computational intractability, the normalisation constant $p(x)$ from Baye's theorem.
- *Non-parametric*: This approach does not require a family of variational distributions with known functional forms to be designated beforehand; consequently, it can approximate complex, multi-modal posteriors.
- *Parallel*: The update steps for individual particles are independent of each other; therefore, modern GPUs can be leveraged to parallelise and accelerate the inference process.

Bayesian methods demonstrate significant utility in many tasks since they are data-efficient. Bayesian inference underlies many robotics algorithms, such as Kalman Filters (Kalman, 1960) used for state estimation, Bayesian SLAM(Thrun, 2002) for localising the pose and mapping the environment, and for handling the uncertainty in simulation parameters (Ramos, Possas & Fox, 2019). Since variational methods tend to be scalable and faster, they have found utility in perception problems (Dehban et al., 2022), reinforcement learning (Furmston & Barber, 2010) and motion planning (Yu & Chen, 2023).

More recently, SVGD has gained traction because of its ability to sample from high-dimensional distributions efficiently, which is crucial for generating diverse plans under uncertainty. For instance, Pavlasek et al. (2023) re-frame planning as an inference problem operating with a diverse range of goal configurations, seeking to address intractable likelihoods via SVGD. Furthermore, recent work highlights its role in multi-modal trajectory optimisation problems, with Z. Yin et al. (2024) proposing Stein movement primitives by combining probabilistic motion primitives with SVGD, and Z. Yin et al. (2025) integrating SVGD with diffusion models and path signatures for diverse motion planning.

In Chapter 3, to estimate the physics parameters of tree branches and bridge the real-to-sim gap, we propose NNSVGD, a custom variant of SVGD. We show that this approach is significantly faster and can model high-dimensional, multi-modal distributions of deformation parameters. Furthermore, in the same chapter, we baseline against sampling-based approaches, such as MCMC and SGHMC, to demonstrate the strengths of Stein-based approaches.

2.4 Reinforcement Learning

In this section, we summarise a few concepts from Reinforcement Learning (RL), following standard definitions from the authoritative reference Barto (2021). Our policy learning frameworks in Chapter 4 and Chapter 5 rely heavily on RL to manipulate tree branches.

2.4.1 Overview & Terminology

RL provides a mathematical framework for controlling a dynamical system through sequential decision-making. An RL agent aims to compute the optimal sequence of actions, termed a policy, using observed state transitions and rewards generated by the system. Throughout this work, we assume the dynamical system, interchangeably termed an environment, is a fully observable, discrete *Markov Decision Process (MDP)*.

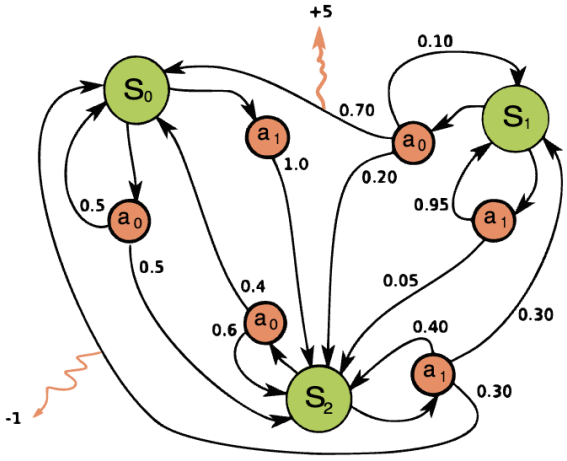


Figure 2.3: A basic MDP example depicting states (green), actions (orange circles), rewards (orange arrows), and transition probabilities, from (waldoalvarez, 2017).

MDP: Typically, an MDP (see Fig. 2.3) is modelled as a tuple of five elements $\langle S, A, P, R, \gamma \rangle$, where S represents the finite set of possible states and A is the collection of actions available to the agent. When the agent takes an action $a_t \in A$, at time step t, while the environment is in $s_t \in S$, the outcome is both stochastic and *Markovian*. In other words, there is a probability $P(s'_t|s_t, a_t)$ associated with the switch to the new state $s'_t \in S$; in addition, the transition depends only on the current s_t and not on the

past visited states. The transition also returns a reward $R(s_t, a_t, s'_t)$, while a discount factor $\gamma \in [0, 1]$ is designed to favour immediate rewards. If a policy $\pi : S \to A$ is defined as the state-to-action mapping, the aim of the RL agent is to find an optimal policy π^* that maximises the expected, cumulative, discounted reward, known as the return $J(\pi)$, by interacting with the environment iteratively through trial and error. Assuming an infinite-horizon problem, the return is summarised as:

$$J(\pi) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right]. \tag{2.23}$$

Value Functions: One way to solve an MDP is to estimate the state or state-action value functions. The state value function $V^\pi(s)$, often termed just *value function*, is the estimated cumulative return when starting in state s and following a policy π, essentially representing the goodness of being in that state. Along similar lines, the state-action value function, widely known as the *Q-value*, represents the value of a state-action pair when the agent follows π.

$$V^\pi(s) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right]. \tag{2.24}$$

$$Q^\pi(s, a) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s,\, A_t = a\right]. \tag{2.25}$$

Advantage Function: Subsequent to the above formulations, we can also define an advantage function that is a measure of the quality of a specific action a, compared to the average action in the current state s.

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s). \tag{2.26}$$

2.4.2 Policy Gradients

Traditional MDP solutions, such as dynamic programming, exploit the Bellman equations (Barto, 2021) to recursively improve the state or state-action values; nevertheless, they require explicitly storing these values for each observed transition, limiting the scalability. On the other hand, modern neural networks can learn a stochastic policy

$\pi_w(a|s)$, parameterised by the network weights w, to sample from, i.e., $a \sim \pi_w(a|s)$, that can generalise to high-dimensional, continuous state and action spaces, common in robotics tasks. The central premise of *Policy Gradient* methods is to optimise the policy directly based on the gradient of the return with respect to the network weights.

$$w_{t+1} = w_t + \alpha \nabla J(w_t). \tag{2.27}$$

A foundational result, called the Policy Gradient Theorem, provides an analytical expression for the gradient of the policy value, given below:

$$\nabla J(w_t) = \mathbb{E}_{\pi_w}\left[\sum_{t=0}^{T} \nabla \log \pi_w(a_t \mid s_t)\, Q^{\pi_w}(s_t, a_t)\right]. \tag{2.28}$$

Note that the above expression for the gradient does not involve the distribution of the states; therefore, parameter updates may be performed through sampling. This is crucial for a robotics setting where the dynamics of the model is usually unknown. A well-known algorithm, REINFORCE (Williams, 1992), follows this intuition and updates the weights iteratively using a Monte Carlo approach. Following the early success of REINFORCE, a variety of algorithms were developed such as the Deterministic Policy Gradients (Silver et al., 2014), Deep Deterministic Policy Gradient (Lillicrap et al., 2015), Asynchronous Advantage Actor-Critic (Mnih et al., 2016), and constrained methods like Trust Region Policy Optimisation (Schulman, Levine et al., 2015).

2.4.3 Proximal Policy Optimization

In this thesis, we use a related constrained variant called Proximal Policy Optimization (PPO), originally developed by (Schulman et al., 2017). The central goal of PPO is to maintain stable updates during policy learning, while preserving high sample efficiency. To further explain PPO, and its core concept, the *Clipped Surrogate Objective*, we first need to define two related concepts.

Advantage Estimate: In practice, the advantage function $A(s, a)$, from Eq. (2.26), is unknown, but an estimate $\hat{A}(s, a)$ can be obtained through sample roll-outs using techniques, such as *General Advantage Estimation* (Schulman, Moritz et al., 2015).

Probability Ratio: To measure the strength of the policy update between each time step, PPO defines a *Probability Ratio* $r_t(w)$ between the new policy $\pi_w(a_t|s_t)$ and the current policy $\pi_{w_{\text{old}}}$, both parameterised by the network weights w.

$$r_t(w) = \frac{\pi_w(a_t|s_t)}{\pi_{w_{\text{old}}}(a_t|s_t)}. \tag{2.29}$$

Clipped Surrogate Objective: PPO aims to reduce the deviation between w_{old} and w, to reduce large, destabilising parameter updates between policy updates. The proposed clipping mechanism constrains the probability ratio $r_t(w)$ to be close to 1, within a small interval of ϵ. A clip function $\mathbb{C}$ forces the probability ratio to satisfy $r_t(w) \in [1-\epsilon, 1+\epsilon]$. The overall maximisation objective can be expressed as:

$$J^{\text{CLIP}}(w) = \mathbb{E}_t\big[\min\big(r_t(w)\,\hat{A}_t,\ \mathbb{C}(\,r_t(w), 1-\epsilon, 1+\epsilon)\,\hat{A}_t\big)\big]. \tag{2.30}$$

Furthermore, during the optimisation phase, PPO performs multiple epochs of mini-batch Stochastic Gradient Descent (SGD) updates using the clipped objective above, leading to improved sample efficiency.

To conclude this section, it is worthwhile to discuss the advantages, disadvantages, and applications of RL, from a robotics perspective. In a model-free setting, an RL agent learns behaviours through trial and error to maximise the long-term reward without relying on prior distributions, demonstrations, or contextual knowledge. However, such a learning from the scratch approach, sometimes called the *tabula rasa* method, is not without pitfalls. RL often suffers from poor sample efficiency and exorbitant training times, while requiring complex manual reward shaping to sufficiently explore the task space. This is especially true in real-world robotics tasks where the state space is noisy, and physical interaction is costly and risky (Kober, Bagnell & Peters, 2013). However, popular alternatives, such as Imitation Learning (IL) or Learning from Demonstrations (LfD), commonly used for deformable manipulation tasks (A. X. Lee et al., 2015; Seita et al., 2023), lack the ability to go beyond sub-optimal human demonstrations (Zare et al., 2024). In fact, some of our experiments in Chapter 4 and Chapter 5 demonstrate that, given sufficient samples and well calibrated

features, Deep Reinforcement Learning can indeed come up with novel and creative strategies, well-beyond what humans can conjure up.

In the specific case of PPO, stable parameter updates ensured by clipping help prevent erratic behaviour in safety-critical real-world settings. In addition, the efficiency gains with mini-batch updates make it suitable for simulation-driven learning (Section 1.4.3). Consequently, it has become the de facto learning algorithm in many robotic simulators, specifically, the Isaac Gym simulator (Makoviychuk et al., 2021) that we leverage. Furthermore, PPO has shown promise in a wide range of robotic tasks, including humanoid locomotion (Kuo et al., 2023), navigation with obstacle avoidance (Taheri, Hosseini & Nekoui, 2024), and multi-agent coordination (W. Wang et al., 2024). On the manipulation front, PPO has been used for solving a Rubik's cube (Akkaya et al., 2019), for pushing or pick-and-place tasks involving household objects (Peng et al., 2018), and for dexterous in-hand block manipulation (Andrychowicz et al., 2020), all dealing with rigid bodies. In contrast, the literature on model-free RL for manipulation of deformable objects is relatively sparse, largely due to its poor sample efficiency, a significant gap that this thesis seeks to fill.

2.5 Solving the Reality Gap

While RL provides a powerful framework to learn control policies through active exploration, physical interaction with the real-world is expensive, time-consuming, and potentially unsafe. We briefly saw in Section 1.4.3 how physics simulators offer a controlled and repeatable alternative to represent the geometry, model the dynamics, and learn manipulation policies for robotics. However, simulation-driven learning is limited by a diverse array of model inaccuracies, simulator constraints, and limited knowledge of the real world, a challenge often summarised as the *reality gap*. In this section, we outline these challenges and two mitigation techniques that are central to the core chapters that follow.

2.5.1 The Physics of Simulation

Physics simulators approximate the continuous-time model of a dynamical system, which includes the robot, the interacting objects (e.g., plant stems, foliage), and the physical environment (e.g., friction, gravity). Let the continuous state of such a system be $s(t) \in \mathbb{R}^l$, the control input be $u(t) \in \mathbb{R}^m$, and $y(t)$ be the observable sensor measurement of the true state $s(t)$. Then, the true dynamics can be written as:

$$\dot{s}(t) = f_{\text{real}}(s(t), u(t)), \quad y(t) = h_{\text{real}}(s(t)). \tag{2.31}$$

It is possible to approximate such a continuous differential equation through numerical methods, such as Euler's method for integration. As a simple zeroth-order example, if the state and the time derivative, $s(t = t_0)$ and $\dot{s}(t = t_0)$ is known at time t_0, then for a small incremental time step Δt, a naive future state can be estimated as:

$$s(t_0 + \Delta t) \approx s(t_0) + \Delta t \dot{s}(t_0). \tag{2.32}$$

A simulator, in essence, is such a *discrete-time integrator*, that can roll out a sequence of states, termed trajectories τ_{sim}, over time, given the approximated physics models $\{f_{\text{sim}}, h_{\text{sim}}\}$, and a set of simulation parameters $\theta \in \mathbb{R}^n$.

$$s_{k+1} = s_k + \Delta t f_{\text{sim}}(s_k, u_k; \theta), \quad y_k = h_{\text{sim}}(s_k; \theta). \tag{2.33}$$

$$\tau_{sim}^{\theta} := \{s_0^{\theta}, s_1^{\theta}, ..., s_K^{\theta}\}. \tag{2.34}$$

A second crucial component of the simulator is a *solver* that resolves forces, constraints, and contacts between the objects and the environment. In other words, the simulator function f_{sim} involves all the physics computations, including applying control $u(t)$, calculating the forces, and maintaining the contact dynamics to return the accelerations needed to determine $\dot{s}(t)$. This may involve some variation of the Newton-Euler equations governing the multi-body dynamics (Tedrake, 2022).

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \boldsymbol{B}u. \tag{2.35}$$

Here, $M(q)$ is the mass/inertial matrix, $C(q, \dot{q})\dot{q}$ represents Coriolis and centrifugal forces and $G(q)$ is the vector of gravitational forces. Additionally, the control input u is converted into generalised forces by the matrix $\boldsymbol{B}$, and the generalised coordinates q and their time derivatives $\dot{q}$ constitute the state $s(t)$, from Eq. (2.33). A variety of solvers, such as direct solvers, conjugate gradient/conjugate residual solvers, and Gauss-Seidel solvers, are used by popular physics simulators (Narang et al., 2022). For a comprehensive guide on contact resolution for simulation, refer Andrews, Erleben and Ferguson (2022). The Isaac Gym (Makoviychuk et al., 2021) simulator used in this thesis, and its underlying PhysX (NVIDIA, 2025) physics engine use a *Semi-implicit Euler* method for numerical integration and a *Temporal Gauss-Seidel* constraint solver for contact resolution.

In this context, the simulation vs reality gap can be defined as the deviation between the models, f_{real} and f_{sim}:

$$\varepsilon_k = f_{real}(s_k, u_k) - f_{sim}(s_k, u_k; \theta). \tag{2.36}$$

It may arise from a variety of factors, including but not limited to:

- *Incomplete knowledge or uncertainty about θ.*
- *The integrator time discretisation process.*
- *Numerical errors and the contact solver tolerances.*
- *Aleatoric uncertainty, i.e., the inherent stochasticity in the environment.*

To provide some specific examples, there may be mismatches between the idealised sensor data in simulation when compared to the real-world sensors riddled with noise, distortion, and occlusions. Similarly, the wear-and-tear in real-world robot actuators and the signal latencies may not be captured in simulation. Crucially, simulations involving multiple bodies are high-dimensional constraint satisfaction problems, and their solvers require complex matrix inversions/pivots, which slow down their convergence. Consequently, they are often traded off against accuracy, by limiting the solver iterations for example, escalating the reality gap.

2.5.2 Domain Randomisation

One technique to mitigate the reality gap is introducing variability in θ during the training phase in simulation. This class of *forward* sim-to-real adaptation technique, termed *Domain Randomisation (DR)*, treats the simulation as a data generation process and the simulation parameters as samples from a PDF, i.e., $\theta \sim p(\theta)$. The key idea is to expose the policy to a variety of likely physical states that help it generalise to the real world. Then the maximisation objective of an RL policy is:

$$J(\pi_w) = \mathbb{E}_{\theta \sim p(\theta)}\left[\sum_{k=0}^{\infty} \gamma^k R(s_k, u_k)\right], \quad s_{k+1} \sim f_{sim}(\cdot; \theta),\ u_k \sim \pi_w(\cdot|s_k). \tag{2.37}$$

Then, the optimal network weight set is:

$$w^* = \arg\max_w \mathbb{E}_{\theta \sim p(\theta)}[J(\pi_w; \theta)]. \tag{2.38}$$

Earlier versions of DR randomised parameters from the visual domain (Sadeghi & Levine, 2016; Tobin et al., 2017), including object colour, background lighting, surface texture, and the camera field of view. Later works (Peng et al., 2018) showed that randomising dynamics parameters, such as object mass, joint damping, and the observation noise, can also lead to a robust policy. In this thesis, we domain randomise the geometry (e.g., branch angles, elongation rate, trunk thickness) and dynamics (e.g., stiffness, friction) of tree branches, for policy learning in Chapter 4 and Chapter 5.

A key consideration for DR is what shape and form the distribution $p(\theta)$ should take? In the simplest case, $p(\theta)$ can be a uniform distribution that leverages human domain expertise to set the boundaries (Tobin et al., 2017), i.e.:

$$\theta_i \in [\theta_i^{\text{low}}, \theta_i^{\text{high}}], \quad i = 1, \dots, N. \tag{2.39}$$

An alternative option (Vuong et al., 2019) is to consider $p(\theta)$ itself to be parameterised by ϕ, i.e., $\theta \sim p_\phi(\theta)$, by formulating Domain Randomisation as a bi-level optimisation problem. In formal terms:

$$\phi^* = \arg\min_\phi \mathcal{L}(\pi_{w^*(\phi)}; e_{\text{real}}). \tag{2.40}$$

$$\text{where } w^*(\phi) = \arg\max_w \mathbb{E}_{\theta \sim p_\phi(\theta)}[J(\pi_w; \theta)]. \tag{2.41}$$

To simplify, the first objective is to optimise ϕ to reduce the deviation $\mathcal{L}(.)$ between the simulation-trained policy π_w^* and a real evaluation environment e_{real}. In contrast, the second objective is to utilise samples drawn from the distribution $\theta \sim p_\phi(\theta)$ to search for the optimal network weights $w^*(\phi)$ to maximise the expected return. This intuition to guide the DR with real-world data, rather than employing vanilla uniform distributions, forms the basis of the next section on *Simulation Parameter Inference*.

2.5.3 Simulation Parameter Inference

As an alternative to manually crafting the parameter distribution $p(\theta)$, the simulation parameter inference approach iteratively refines this distribution to match real observations (Cranmer, Brehmer & Louppe, 2020). This *inverse* real-to-sim adaptation technique is closely related to the system identification approach examined in Section 1.4.2. The key difference is that system identification aims to tune true physics parameters, such as mass and friction, to output a dynamics model that predicts the behaviour of the real system. In contrast, simulation parameter inference refines the parameters θ to make simulators more realistic, where the θ may not have any physical meaning at all (Heiden et al., 2022). In other words, the aim is to predict a realistic outcome, rather than model the physics. Nevertheless, in this thesis, we interchangeably use both terms to mean learning the deformation parameters from the physical environment to generate realistic simulation behaviour. In practice, the real-to-sim parameter inference step is a precursor to learning a control policy in simulation, which can then be transferred sim-to-real; therefore, the overall pipeline is often referred to as the *real-to-sim-to-real* loop (Lim et al., 2022).

To formalise the simulation parameter inference approach, let the dataset D consisting of simulation (size P) and real (size Q) trajectories be defined as:

$$\begin{aligned} D &= \mathcal{T}_{\text{real}} \cup \mathcal{T}_{\text{sim}}, \quad \text{where} \\ \mathcal{T}_{\text{sim}} &= \left\{\tau_{\text{sim}}^{j}\right\}_{j=1}^{P}, \quad \mathcal{T}_{\text{real}} = \left\{\tau_{\text{real}}^{i}\right\}_{i=1}^{Q}. \end{aligned} \tag{2.42}$$

Then an optimal point estimate for the simulation parameters may be obtained based on some notion of a loss $\mathcal{L}$:

$$\theta^* = \arg\min_{\theta} \ \mathcal{L}(\mathcal{T}_{\text{real}}, \mathcal{T}_{\text{sim}}). \tag{2.43}$$

Alternatively, to quantify the uncertainty of these parameters, a likelihood term may be defined as:

$$p(D \mid \theta) = \exp\left(-\frac{\mathcal{L}(\theta)}{\alpha}\right). \tag{2.44}$$

where α is a constant. Then a Bayesian model (Section 2.3.1), such as MCMC or VI, may be used to infer the posterior parameter distribution $p(\theta \mid D)$. This idea serves as the foundation for the next chapter, where we attempt to replicate real-world tree-branch deformation behaviour in simulation.

Computer simulation makes it practical to make mistakes on purpose.

Unknown

CHAPTER 3

Learning to Simulate Tree-Branch Dynamics

3.1 Introduction

Simulation driven learning approaches have been gaining popularity to address the challenging task of manipulating deformables. In this chapter, we study the deformation characteristics of a tree branch under forces exerted by a robotic arm via simulation-based parameter inference. Learning the branch dynamics through simulation has significant practical ramifications, particularly for the robotics community. First, it adds the missing physics link to visual tree reconstruction works; for example Lowe and Pinskier (2022) and Quigley et al. (2021), to create a complete digital twin with both perceptual features and inferred dynamical properties. Second, such a digital

Figure 3.1: Autonomous navigation task: Our field robot obstructed by vegetation.

model reduces the reality gap, often denoted as the real-to-sim gap in the robotics context, and facilitates massively parallel policy learning, inexpensive data collection, safe exploration, and most importantly, real-time control estimation. On the application front, the presence of occlusions from branches, other fruit clusters, stems and foliage pose a significant challenge to robotic fruit harvesting tasks (Zhou et al., 2022), which are typically addressed through costly environment modifications to improve visibility and reachability (van Herck et al., 2020). The harvest success rates of state-of-the-art methods, in the presence of partial and complete occlusions are 50-75% and 5% respectively (Zhou et al., 2022). Along the same lines, for a mobile robot to perform autonomous navigation in natural environments, such as forests and grasslands; where plant foliage, non-compliant branches and dense vegetation pose obstacles; the conventional approach has been to avoid them altogether, quite unlike human subjects who seamlessly couple locomotion and manipulation to interact with objects, clear the path, and navigate. Having a simulated twin with inferred dynamics can overcome these barriers to improve fruit detection rate and grasp quality in harvesting tasks, and path planning with contact around obstacles for locomotion. Finally, modelling the deformation behaviour across all branch grasp locations allows the arm to effectively trade-off the low force and high reach required at the outer edge vs the high force and low reach at the inner fork, thereby, amplifying the autonomous decision making capacity of the manipulator. It is worth noting that many biological organisms, including humans, possess an instinctive understanding of physical dynamics, which allows them to operate seamlessly in the natural world; see Appendix A.2 for examples.

In general, the goal of an inverse inference approach is to estimate the latent para-

meters of a physical phenomenon based on its observed effects and subsequently to forecast future system behaviour. A subclass of this model, commonly known as simulation-based inference (refer Section 2.5.3), leverages high fidelity simulations to implicitly define a statistical model and perform the inference through either Frequentist or Bayesian paradigms. The parameters themselves may not necessarily have any physical meaning (Cranmer, Brehmer & Louppe, 2020); moreover, the objective of the estimation is to predict a useful outcome rather than to accurately estimate the true parameters (Heiden et al., 2022).

A broad outline of our workflow is as follows: to begin with, we capture the branch geometry with crude spring abstractions using purpose-built simulations executed on massively parallel, GPU based, black-box simulators. Next, we actively probe the real branches to measure the observed effects of forces. While individual twigs are rigid by nature, the additional degrees of freedom incurred due to the dynamics of coupled branches makes the problem non-linear and high dimensional. We consider the measured deformation trajectories from the real world as i.i.d samples from a ground truth distribution. Finally, guided by a Bayesian model, we use a probabilistic inference algorithm to infer the posterior density of spring parameters and to predict the branch deformation behaviour. A visual representation of the steps involved is provided in Fig. 3.2.

Overall, this chapter lies within the broader objective of translating the success of manipulation schemes within structured environments, guided by simulation driven inverse inference (Antonova et al., 2022; Chebotar et al., 2019), out to the natural world. However, simulation of outdoor environment is beset by both epistemic (e.g., unaccounted factors, such as wind; model simplicity) and aleatoric (e.g., blurred vision due to limited lighting, noisy sensors) uncertainties, as described in Section 1.3. Therefore, optimisation models that capture point estimates of simulation parameters, from the Frequentist paradigm, are not sufficient in place of an uncertainty aware Bayesian model, that we use. Although not the focus of this chapter, we postulate that, on a temporal scale, each inferred posterior can feed into the next iteration as a prior, leading to faster convergence over time and that the posteriors can generalise to different branches, trees and perhaps even across species.

Our experiments demonstrate that the learnt model can estimate deformation trajectories given the end-effector forces, quantify the model uncertainty, display ro-

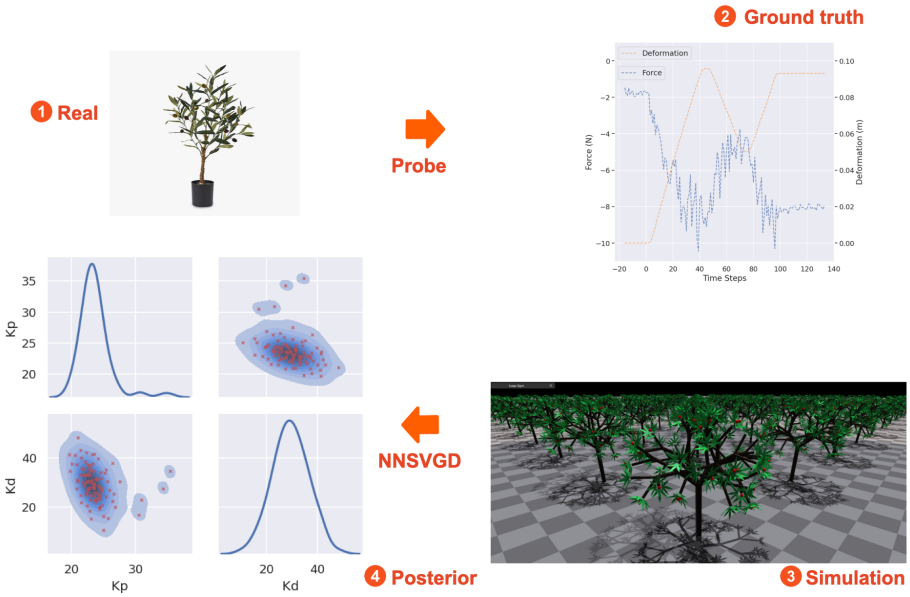


Figure 3.2: An overview of the proposed system identification approach

bustness in presence of perturbations and can perform the inference in near real time. Overall, our contributions are summarised as below:

1. We propose to model the dynamics of deformable branches as mass-spring abstractions on a parallel non-differentiable simulator, obtain reference trajectories by actively probing real trees, and infer the spring parameters via a probabilistic inverse inference algorithm.

2. We incorporate biologically motivated assumptions into the Bayesian inference workflow by formulating inequality constraints as differentiable and learnt joint prior distributions.

3. We demonstrate the real-to-sim transfer with two different real-world robots, operating on distinct physical trees, thereby additionally validating model independence from both arm kinematics and branch topology.

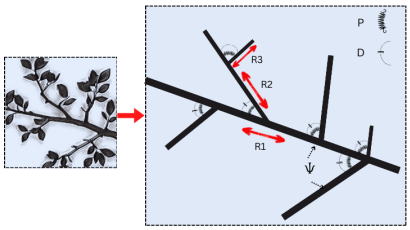
4. Finally, we show that the learnt model and the predicted deformation trajectories are robust to noise perturbations from the sensors and variations in branch grasp locations where the manipulation forces are applied.

3.2 Related Work

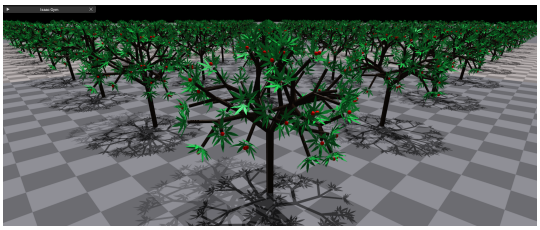
Manipulation of deformable objects, such as clothes, elastic bands, and ropes has gained much traction over the years, owing to the broad range of ensuing applications. The comprehensive surveys from H. Yin, Varava and Kragic (2021) and Arriola-Rios et al. (2020) detail established techniques to represent shapes, mange the dynamics, and apply learning strategies, all in order to perform control and planning with deformables. While mass-spring models have been used to build coarse-grained, but computationally efficient models for suturing (Schulman et al., 2013) and fabric management (Makris, Kampourakis & Andronas, 2022), its more accurate, but slower counterpart, the Finite Element Method (FEM) models have been used for object tracking (Bozic et al., 2020) and manipulating rubber tubes (Rambow et al., 2012). Perhaps more in line with this work, both mass-spring and FEM models are used in conjunction, the latter as a reference, to acquire deformation trajectories for motion planning around soft toys (Frank et al., 2014).

In contrast, dynamic behaviour modelling of tree branches for active manipulation is rather rare in literature. Spring based models have been used to aid visual reconstruction of trees (Yandun, Silwal & Kantor, 2020) or to animate the interaction between rain drops and branches (Yang et al., 2010); however, the spring parameters are specified in advance. On the other hand, applying spring models to estimate branch behaviour under wind loading has been extensively studied in forestry (De Langre, 2008); for example, to measure the overall sway of tree (James et al., 2014). Such works depend on accurate modelling of the wind forces, can only compute parameters along the wind direction, and therefore they are unsuitable for active robotic manipulation. To the best of our knowledge, no works exist in the literature that actively manipulate tree branches to estimate the spring parameters through simulation.

On the inference front, probabilistic methods have been proposed to estimate parameters to bridge the gap between simulation and reality, the real-to-sim problem, for both rigid bodies (Heiden et al., 2022) and deformables (Antonova et al., 2022). While these approaches estimate parameter posteriors through the Bayesian framework, they almost always start with an uninformative prior, a uniform prior to enforce limits, or other well known distributions to represent the subjective belief about the parameters. Weighing the inference technique alone, our approach appears closest to



(a) Mass-Spring-Damper representation of a branch system.



(b) Parallel instances of tree simulations on NVIDIA Isaac Gym.

Figure 3.3: Coarse-grain simulation design and training environments.

the work of Heiden et al. (2022), which introduces a constrained SVGD algorithm inspired by the Lagrange multiplier optimisation strategy. In comparison, we treat the simulator as a black-box, do not rely on accurate gradients for the simulation roll-outs, and our inference formulation can embody inequality assumptions as prior beliefs.

3.3 Methodology

3.3.1 Mass-Spring-Damper Model

Our work utilises the well established *Mass-Spring-Damper System (MSS)* abstraction to simplify the intricate tree geometry. MSS models are computationally efficient and simple to implement; however, the spring constants are hard to obtain from material properties without reference models. In this chapter, we describe the branch structures as cylindrical links coupled with a torsional spring-damper system (Fig. 3.3 a).

The dynamics of a single branch, and consequently that of the entire branch system, can be derived from Newton's second law of motion. Given an external torque T_{ext}, branch stiffness P, damping factor D, moment of inertia J, and angular displacement ψ, this relationship can be obtained as:

$$T_{ext} = J\ddot{\psi} + D\dot{\psi} + P\psi. \tag{3.1}$$

We propose to model the relation $f : T_{ext} \mapsto (\psi, \dot{\psi})$, by comparing the model displacement trajectories to reference trajectories from the real world. We use physics simulators to model this MSS abstraction, which additionally helps to account for the latent attributes that affect the branch motion, such as gravity and the branch fork angle.

3.3.2 Tree Simulation

Generating a digital twin for a physical entity enables cost effective data generation and robust control policy learning; moreover, GPU-accelerated simulators provide the advantage of faster training and efficient run time inference. This section describes our approach to generating crude digital tree replicas that run in parallel to perform simulator driven learning.

To begin with, we purpose-built a tool for generating tree geometry. The geometric structures can be generated from scratch, or alternatively, it can be integrated with a perception system to create replicas of real world trees. The focus of this chapter is the former, where the geometric attributes are specified in advance. The resulting coarse tree model (Fig. 3.3b) takes cues from the natural world to replicate traits of real trees, such as having a fractal pattern and adhering to the cross sectional area-preservation principle (Minamino & Tateno, 2014), also known as the Leonardo da Vinci's rule. Furthermore, our approach is not bound by branch topology, and it can accommodate larger plant models built from Lindenmayer systems (L-system).

Subsequently, the MSS model is implemented as a multi-link structure with branches approximated as cylindrical links and branch forks represented by actuated revolute joints. The branch motion is simulated via a *Proportional–Derivative (PD)* controller governed by tunable stiffness and damping gain parameters, $\theta = \{K_p, K_d\}$. Consequently, the branch motion estimated by the PD controller is:

$$T_{ctrl} = K_p\psi + K_d\dot{\psi}, \tag{3.2}$$

where T_{ctrl} is the controlling torque exerted by the simulated branch to offset the positional ψ and velocity $\dot{\psi}$ errors. The final structure is represented with a simulator agnostic *Unified Robotics Description Format (URDF)*. While the simplified mass-spring model is an imprecise approximation of the ground truth, particularly when compared to other more realistic *FEM* models, we show that it is sufficient to capture the multi-branch dynamics.

Finally, we choose NVIDIA Isaac Gym (Makoviychuk et al., 2021) to simulate tree structures; specifically, its joint space PD controller to apply deformation torques and measure the resulting motion characteristics. Isaac Gym supports tensor based distributed environments, and it permits parallel execution of the inference without CPU bottlenecks. However, Isaac Gym simulations are non-differentiable; therefore,

unlike Heiden et al. (2022), throughout this chapter we use finite difference methods to approximate gradients.

3.3.3 Probabilistic Parameter Inference

The key idea of our approach is to model the relationship between the force required to counteract the simulated branch resistance T_{ctrl} and the resulting branch deformation trajectory, through inverse inference of stiffness K_p and damping K_d gain parameters. The moment of inertia is computed by the simulator directly from the branch topology, and therefore is known a priori. The cost of deformation is established by comparing the position and velocity trajectories between the ground truth and the simulation through a loss function. The ground truth trajectories can be generated in two ways; first, as yet another independent articulated body simulation as described in section Section 3.4.1, and second using real robotic arms on physical trees, as described in section Section 3.4.2.

We define force profile F as a sequence of force vectors applied on the target branch, at a location C, over g time steps, i.e., $F := \{f_0, f_1, ..., f_g \mid f \in \mathbb{R}^3\}$. The resulting deformation at each time step t is defined as the change in the position of the branch, at the force location, w.r.t its initial position, i.e., the deviation of the point C_t from its rest position C_0. The position trajectory τ is then the sequence of such deformations, over g time steps, while its derivative w.r.t time gives $\dot{\tau}$, the velocity trajectory. τ_{gt} and τ_{sim} represent the branch displacement for ground truth and simulation respectively. However, the force profile is constrained to the gravity axis (downward) as a simplification measure during experiments.

A general parameter inference problem is characterised by three terms: a parameterised simulator, a loss, and an inference framework. First, we define the simulator as a function S that steps forward in time, given simulation parameters θ and force profile, to roll out a trajectory τ_{sim} over time.

$$S : (\theta, s_i, f_i) \mapsto s_{i+1},\ \theta \in \mathbb{R}^N \tag{3.3}$$

$$\therefore \tau_{sim}^{\theta} := \{s_0^{\theta}, s_1^{\theta}, ..., s_g^{\theta}\}, \tag{3.4}$$

where N is the simulation parameter count. For each branch, we take $N = 2$ to represent its PD coefficients, K_p and K_d. The relationship to be modelled then is

$f : F \mapsto (\tau, \dot{\tau})$. Second, we define the loss L over simulation parameters as deviations of the branch position and velocity trajectories from a ground truth.

$$L(\theta) := \left\| \tau_{sim}^{\theta} - \tau_{gt} \right\|_2 + \left\| \dot{\tau}_{sim}^{\theta} - \dot{\tau_{gt}} \right\|_2. \tag{3.5}$$

Finally, similar to Heiden et al. (2022) and Antonova et al. (2022), we use Bayesian inference (refer Section 2.3.1) to quantify the uncertainty associated with the simulation parameters. The canonical Bayes' rule relates the parameter posterior probability density $p(\theta|\tau_{sim})$ to the corresponding likelihood $p(\tau_{sim}|\theta)$ and the prior $p_0(\theta)$,

$$p(\theta|\tau_{sim}) \propto p(\tau_{sim}|\theta)\, p_0(\theta). \tag{3.6}$$

While point estimates of optimal parameters can be obtained as $\theta^* = \underset{\theta \in \mathbb{R}^N}{\operatorname{argmin}} L(\theta)$, i.e., by minimising the objective function, we instead attempt to successively sample promising regions of the posterior density $p(\theta|\tau_{sim})$ in order to capture its multiple modalities. Furthermore, we compute the likelihood term through an exponential transformation of the loss, along the lines of simulated annealing architectures (Kirkpatrick, Gelatt Jr & Vecchi, 1983),

$$p(\tau_{sim}|\theta) := \frac{e^{-L(\theta)/kT}}{Z_\theta} \because L(\theta) \geq 0. \tag{3.7}$$

where Z_θ is the constant set of all possible energy states, k the Boltzmann constant, and T the fixed annealing temperature,

$$\therefore p(\theta|\tau_{sim}) = \frac{e^{-L(\theta)/kT}\, p_0(\theta)}{Z_\theta\, Z_\tau}. \tag{3.8}$$

where Z_τ is the normalisation constant. We do away with the intractable $Z_\theta Z_\tau$ term by choosing Stein Variational Gradient Descent (see Section 2.3.4) as our inference algorithm.

While our approach can be generalised to a full 3D system, in this chapter, we only consider deformations in the vertical plane, implemented by restricting the force profile to the gravity-axis, i.e., perpendicular to the ground. Deformation along other axes, such as horizontal twists can be estimated by using higher-dimensional hinges at each joint in the sequence, i.e., by increasing the θ dimensions. However, we restrict the forces to tolerable limits and leave the discontinuous dynamics of branch rupture for

future work. Finally, we treat each τ_{sim} trajectory as an i.i.d sample from the underlying distribution, compute the likelihood term in Eq. (3.7) over multiple trajectories, and use the log form to avoid vanishing floating points,

$$\log\ p(\tau_{sim}|\theta) := \sum_{\tau} \frac{-L(\theta)}{kT} - \log\ Z_\theta. \tag{3.9}$$

3.3.4 Stein Variational Gradient Descent

We leverage the SVGD (Q. Liu & Wang, 2016) inference approach, presented in Section 2.3.4, to estimate the branch simulation parameters. SVGD has been shown to converge faster than alternatives such as MCMC, predominantly due to its deterministic and iterative progression, akin to the traditional Gradient Descent optimisation.

The ϕ^* term from Eq. (2.22), which represents the optimal function that minimises the KL divergence between the distribution represented by the current particle state and the expected distribution $p(\theta|\tau_{sim})$, can now be formulated as:

$$\hat{\phi}^*(\theta) = \frac{1}{n}\sum_{j=1}^{n} k(\theta_j, \theta)\nabla_{\theta_j} \log\ p(\theta_j|\tau_{sim}) + \nabla_{\theta_j} k(\theta_j, \theta). \tag{3.10}$$

While SVGD allows for placing the particles on a GPU and performing the operations independently, the traditional performance bottleneck has been in executing the simulations to derive the gradients. In our case, the term $\nabla_{\theta_j} \log\ p(\theta_j|\tau_{sim})$ is approximated via finite differences instead. Our simulator choice, Isaac Gym, is non-differentiable but supports efficient gradient approximations through GPU based simulations; therefore, the entire inference process runs in parallel. On substituting the prior and likelihood terms from Eq. (3.8), we obtain:

$$\hat{\phi}^*(\theta) = \frac{1}{n}\sum_{j=1}^{n} -k(\theta_j, \theta)\nabla_{\theta_j}\frac{L(\theta)}{kT} + k(\theta_j, \theta)\nabla_{\theta_j} \log\ p_0(\theta_j) + \nabla_{\theta_j} k(\theta_j, \theta). \tag{3.11}$$

Notice that the intractable constant terms from Boltzmann distribution and from the Bayes' rule, Z_θ and Z_τ vanish due to the gradient, which is a quintessential feature of SVGD.

3.3.5 Joint Parameter Inference

A multi-branch joint parameter estimation problem is an extension of Eq. (3.6) to R coupled branches that actively take part in the collective dynamics.

$$p(\theta_1, ..., \theta_R|\tau_{sim}) \propto p(\tau_{sim}|\theta_1, ..., \theta_R)\, p_0(\theta_1, ..., \theta_R). \tag{3.12}$$

This includes the child branch which is actively probed and all its main parent branches up to the fork where any deformation is observed due to the probing. While we limit our experiments with $R = 3$, the model itself has no such limitation, except for the marginal increase in computational cost per increment due to the increased dimensionality of the search space.

3.3.6 Smooth Box Prior

We use the prior $p_0(\theta)$ from Eq. (3.8) to define a search range for the inference algorithm in order to ensure convergence within a reasonable time. Using a uniform prior results in non-differentiable regions around the prior limits. For that reason, we use the Smoothed Box prior from GPyTorch library (Gardner et al., 2018) to generate a smoothed version of the uniform prior to ensure differentiability at all points. Given a box defined by the lower (θ_l) and upper (θ_u) bound of simulation parameters $B := \{\theta \,|\, \theta \in [\theta_l,\ \theta_u],\ \theta \in \mathbb{R}^N\}$, and a variance σ^2, the fully differentiable smoothed box prior is approximated as:

$$p_0(\theta) \sim \exp\big(-\frac{d(\theta, B)^2}{\sqrt{2\sigma^2}}\big), \tag{3.13}$$

where the distance d is defined as $d(\theta, B) = \min_{\theta' \in B} |\theta - \theta'|$.

3.3.7 Neural Network based Inequality Prior

Further to constraining the search range with a smooth box prior, we propose to incorporate biological assumptions on the relationship between the branch parameters into the prior $p_0(\theta)$ when more than one branch is involved.

For the simplest multi-branch case, where $R = 2$, i.e., one parent child pair each with a single parameter $\theta_1 \in \mathbb{R}$ and $\theta_2 \in \mathbb{R}$ respectively, an inequality assumption

that defines the joint parameter relationship could be:

$$y(\theta_1, \theta_2) := \begin{cases} 0 & \theta_1 \geq \theta_2 \\ -\eta & \theta_1 < \theta_2 \end{cases}, \tag{3.14}$$

where η is a scaling factor and $\eta \gg 0$. We propose to train a simple one-hidden layer, feed forward, neural network regressor to model y, and then use the predicted y' as an approximation for log of prior, i.e., $\log\ p_0(\theta_1, \theta_2) \approx y(\theta_1, \theta_2)$. The corresponding prior can then be formulated as:

$$p_0(\theta_1, \theta_2) \approx \begin{cases} 1 & \theta_1 \geq \theta_2 \\ e^{-\eta} & \theta_1 < \theta_2 \end{cases}. \tag{3.15}$$

This implies that the child branch parameter θ_2 (K_p or K_d) is assumed to be lower than that of the parent θ_1. Intuitively, from the area-preservation principle (Minamino & Tateno, 2014), the child branch cross-sectional area is less than that of the parent, while elasticity theory dictates that the bending stiffness is proportional to the area for a cylinder. The neural network is modelled as:

$$y'(\theta_1, \theta_2) = w_0 + \sum_k w_k \sigma(w_{k0} + w_{k1}\theta_1 + w_{k2}\theta_2), \tag{3.16}$$

where σ is any activation function, k the number of hidden units, and w the network weights.

We justify our design choices with the following arguments: First, the regressor ensures that the predicted y' is continuous and fully differentiable at all points in the domain, subject to the network activation function choice. Second, Eq. (3.15) guarantees that $p_0(\theta)$ is non-negative. Third, the use of SVGD as the inference algorithm renders irrelevant the need to normalise $p_0(\theta)$. Therefore, y' can indeed be considered as an un-normalised probability density that satisfies the differentiability requirements of the term $\nabla_{\theta_j} \log\ p_0(\theta_j)$ from Eq. (3.11). Fourth, training the model on a log scale prevents prior values from getting extremely small to machine limits, keeps it comparable to the likelihood, and therefore adequately constrains the SVGD convergence to the region of interest. Finally, using a trained model to predict joint prior opens up the possibility of learning directly from the tree geometry instead of hand coding an assumption *a priori*. Most importantly, we assert that this approach is consistent

with the Bayesian paradigm, where assumed knowledge is expressed with priors, unlike other methods that penalise constraint violations at the likelihood - CSVGD (Heiden et al., 2022), for example.

However, parameter estimation with Eq. (3.14) and Eq. (3.16) generates non-trivial flat regions in the target density, thereby slowing down the inference process. A better alternative is to penalise the difference in parameters as per the prior belief, as given by:

$$y(\theta_1, \theta_2) := \begin{cases} 0 & \theta_1 \geq \theta_2 \\ -\eta(\theta_2 - \theta_1) & \theta_1 < \theta_2 \end{cases}. \tag{3.17}$$

Then the prior can be formulated as:

$$p_0(\theta_1, \theta_2) \approx \begin{cases} 1 & \theta_1 \geq \theta_2 \\ e^{-\eta(\theta_2 - \theta_1)} & \theta_1 < \theta_2 \end{cases}, \tag{3.18}$$

where η is a scaling factor that determines the sharpness of the gradient and $\eta > 0$. Note that we do not make any assumptions about the relationship between the parameters when the relationship is valid, and the penalty applies only when the constraint is violated. For the simple inequality case we implemented using Eq. (3.18), the reduction in search space is at least exponential in the number of priors, indicating that the benefits of adding the prior grows exponentially for every increment in R.

3.4 Experimental Setup

The following subsections describe our experimental strategies and the evaluation metrics used.

3.4.1 Simulated Ground Truth

In the most basic set up, we capture both ground truth and the replica measurements from the Isaac Gym simulator. While the geometric representation of the branch structure is same in both (as described in Section 3.3.2), the execution process remains independent, i.e., no information is leaked between ground truth and inference workflows except for the force-deformation profiles. As opposed to the hardware experiments, in the sim-to-sim case, we use the simulator API to apply the deformation forces rather than using simulated arms.

Figure 3.4: Experimental setup for robotic tree interaction.(a) Kinova Jaco-2 arm on synthetic plant. (b) Franka-Emika Panda arm on a real farm tree.

3.4.2 Real-World Ground Truth

This section describes the scenario where the ground truth is captured from real world using a physical robot arm. To show that our framework is agnostic to arm kinematics, we use two independent hardware platforms (Fig. 3.4) for the real measurements: 1) a Kinova Jaco-2 6 DoF curved wrist manipulator, and 2) a Franka-Emika Panda 7 DoF manipulator. In the first setup, we use the Kinova manipulator to probe and deform an artificial olive tree. In contrast, we operate the mobile base mounted Franka arm on a larger, real farm tree.

A few general principles applicable to both executions are examined below. The end-effector grasp position is taken as synonymous to the branch force location C; therefore, the Cartesian trajectory of the end-effector acts as a proxy for the branch deformation trajectory τ, at the grasp location. We command the arm using position control to deform the branch downward, in the gravity axis, while ignoring force and deformation in all other axes. To infer the joint parameters of coupled branches, we only measure the trajectories of the child branch to which the arm is attached, obviating the need for any visual tracking. We utilise the end-effector force readings exposed by the arm, which are typically inferred using forward kinematics, instead of the more accurate external sensor attachments at the end-effector itself, and consequently the readings are rather noisy. Additionally, the robot model could have position-sensitive bias in force readings, due to gravity compensation issues, sensor misalignment or

improper arm calibration. However, we show that our framework is robust to these distortions. Both force and deformation measurements are independently retrieved from the arms using their respective Robot Operating System (ROS) apis. Additionally, we neither use a perception mechanism to capture the tree characteristics nor do we attempt automated grasping.

Finally, we measure the branch attributes essential to create the digital replica, which are: the radius, length and density of the branch; and the angle at which the branch forks from its parent. While it is not necessary to simulate the entire tree, it is imperative that all upper level branches that participate in the collective dynamics of the target branch are accounted for. The density of wood material is assumed to be constant for a tree species (or the synthetic material), which can be derived indirectly from the mass and volumetric measurements of a single cut out fragment.

3.4.3 Evaluation Metrics

To assess the estimation reliability, first we compare the inferred parameter distribution with the pre-specified, parameter values from the ground truth, using the sim-to-sim setup. In both sim-to-sim and real-to-sim experiments, we compare the unseen *Ground Truth (GT)* deformation trajectory to the predicted samples drawn from the estimated trajectory distribution $p(\tau_{sim}|\theta^*)$ corresponding to the converged particle set θ^*. Additionally, in the latter case, we compute the 95% *Confidence Interval (CI)* over the estimated trajectories and assess the fraction of true trajectory points that fall outside the interval. The distribution itself is constructed from the samples of trajectories corresponding to θ^*, leveraging Kernel Density Estimation with Gaussian kernels and Scott's rule of thumb for bandwidth selection. In the sim-to-sim experiments, we use a hold-out dataset for evaluation, while in real-to-sim, we use k-fold cross validation due to the smaller sample size. In either case, the evaluation trajectories are unseen during the parameter inference workflow. Samples of GT force-deformation measurements, along with all experiment details, such as the dataset volume and the hyper-parameters used are available in the supplementary material Appendix A.3.

3.5 Experiments and Results

We assess the quality of our approach using the following experimental scenarios.

3.5.1 Baseline Comparison

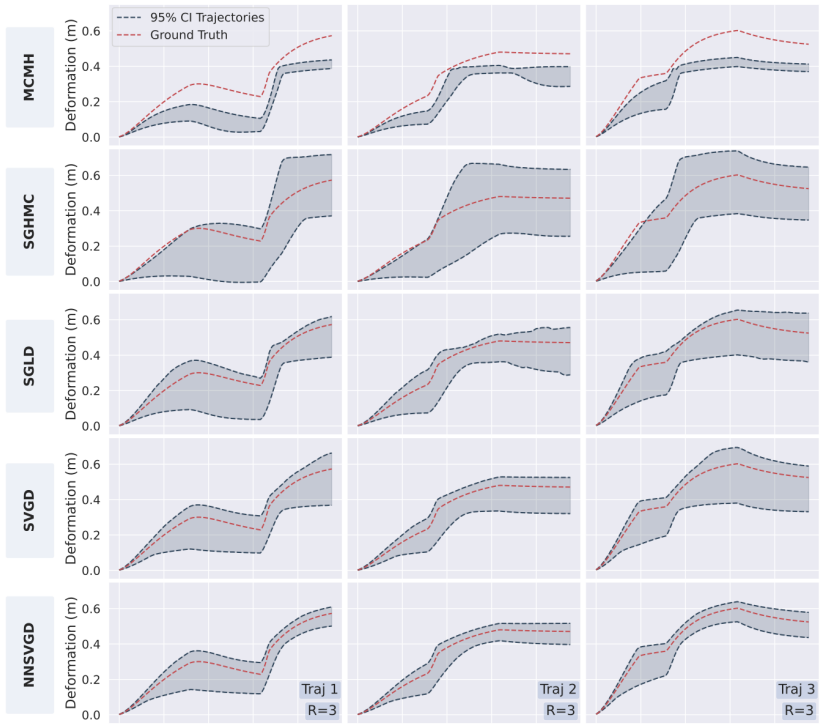


Figure 3.5: Comparison of NNSVGD (ours) against baseline inference algorithms. 95% CI of the predicted distribution of test trajectories $p(\tau_{sim}|\theta^*)$, computed with $R = 3$, is shown in grey against the preset ground truth τ_{gt} in red.

In the first scenario, illustrated in Fig. 3.5, we baseline our inference approach, SVGD with a neural network prior (NNSVGD), against four alternatives, which are: a) the classical Metropolis–Hastings (MCMH) algorithm b) Stochastic gradient Hamiltonian Monte Carlo (SGHMC) method c) Stochastic Gradient Langevin Dynamics (SGLD) and a basic d) Stein Variational Gradient Descent (SVGD). In each of the four baseline algorithms, we use the smooth box prior described in section Section 3.3.6 to set the search limits, and wherever a gradient is required, we use the finite difference based approximation. This scenario is executed using the sim-to-sim setup with a

multi-branch simulated tree geometry ($R = 3$), imitating a joint grandparent-parent-child motion. We keep all hyper-parameters, such as the number of iterations and the particle count, similar across the algorithms. In the gradient based methods, we use an Adam optimiser with a small learning rate to guide the descent.

The results in Fig. 3.5 indicate all gradient based methods outperform the random-walk based MCMH. Furthermore, samples from the Stein methods, i.e., SVGD and NNSVGD, are better distributed around the ground truth τ_{gt} trajectory, while additionally displaying sharper convergence and lower prediction uncertainty, indicated by the CI width. For identical iteration counts, Stein methods outperform chain based SGHMC and SGLD chiefly because the particles are updated in parallel, a feature exploited by our parallel simulator.

3.5.2 Impact of Neural Network Prior

With the same setup, the benefit of having a neural network prior, described in section Section 3.3.7, is examined both visually and quantitatively. First, the intuitive visual result in Fig. 3.6, shows that the search space restriction from the inequality prior forces the spring parameters to have a sharper convergence, for all three branches. Consequently, the sample trajectories predicted by NNSVGD show lower CI widths in comparison to SVGD, from Fig. 3.5. Furthermore, the multi-modal nature of the branch1 posterior indicates the presence of multiple solutions, a quintessential feature of inverse inference.

In the first quantitative test, we compute the Negative Log-Likelihood (NLL) of the estimated posterior from 5 random training trajectories. Secondly, we compute the 95% CI of the predicted trajectories averaged over 25 unseen test trajectories and assess the fraction of GT points that fall outside the interval, as detailed in Section 3.4.3. Table 3.1 demonstrate that the addition of the neural network prior results in a lower NLL and a lower CI for the same % GT enclosure, suggesting its superiority.

	95% CI width	% GT outside CI	NLL
NNSVGD	0.11	<1%	2268.1
SVGD	0.19	<1%	3050.7

Table 3.1: NN prior benefits: Quantitative metrics.

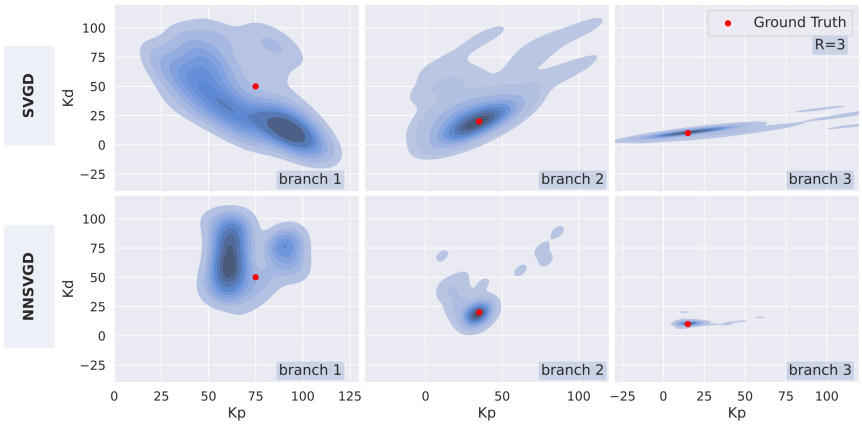


Figure 3.6: Joint posteriors with and without the neural network prior, computed with $R = 3$. The red dots indicate the preset GT for stiffness K_p and damping K_d.

3.5.3 Tolerance to Perturbations

In the second scenario, we perform two sets of experiments to measure the robustness of the model against erratic sensors and variations in branch grasp locations. In both sets, the metrics are computed as an average over 25 unseen test force-deformation trajectories, 3 of which are shown along with the results. In the first set, we inject Gaussian noise of standard deviation σ into the ground truth deformation and force trajectories, leveraging the sim-to-sim setup. The results in Fig. 3.7, show that for moderate levels of noise, the predicted distribution is well-aligned to the ground truth; furthermore, the 95% CI attributes indicate that the model is able to capture the injected noise within the estimated uncertainty. Secondly, we vary the grasp locations, selected randomly along the branch, where the force is applied. This experiment mimics the situation where the arm needs to ascertain whether to grasp towards the outer edge of a branch to reduce the applied forces, or move towards the inner fork due to the arm reach constraints. The results, in Fig. 3.8, confirm that our model can infer the deformation on one location using trajectories obtained by grasping another.

3.5.4 Hardware Experiments

Here, we perform real-world experiments and capture multiple pairs of force-deformation trajectories using the two setups described in Section 3.4.2. The converged NNSVGD particles and the corresponding deformation trajectory samples are used to construct a

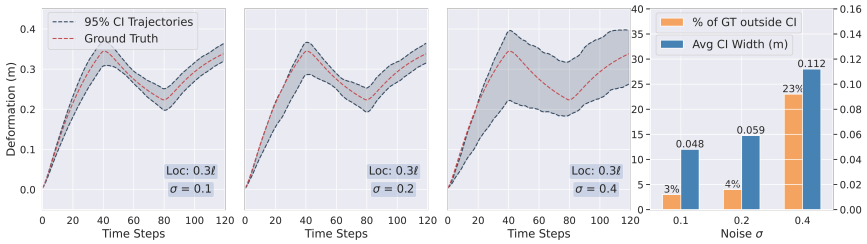


Figure 3.7: Robustness against noise perturbations: 95% CI of predicted distribution against GT for varying noise σ, measured relative to the force, keeping the grasp location fixed. The bar chart shows the CI width along with %GT points outside the interval (lesser the better for both).

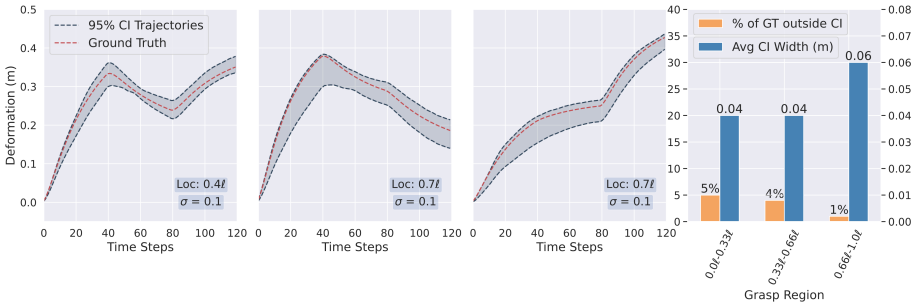


Figure 3.8: Robustness against grasp variations: 95% CI of predicted distribution against GT for varying grasp locations, keeping noise σ fixed. The grasp location is represented as a fraction of the branch length, from the parent joint; for e.g., Loc: 0.75ℓ is the mid point of the outer half.

trajectory distribution $p(\tau_{sim}|\theta^*)$ and the 95% CI for each time-step, shown in Fig. 3.9. The CI attributes are tabulated in Table 3.2.

	Kinova ($R = 1$)	Franka ($R = 1$)	Kinova ($R = 2$)
Avg 95% CI width	0.052	0.066	0.039
Avg %GT outside CI	5.0%	7.0%	23.9%

Table 3.2: Real-to-sim CI attributes.

The results show that despite the heteroschedastic noise in the ground truth, our model is able to predict the deformation quite well. For instance, across the three farm tree trajectories, on average, 93.0% of true points lie within the 95% Confidence Interval and the average interval width is 6.6cm. We deem such metrics as acceptable for a majority of practical applications discussed earlier. Remarkably, these high quality

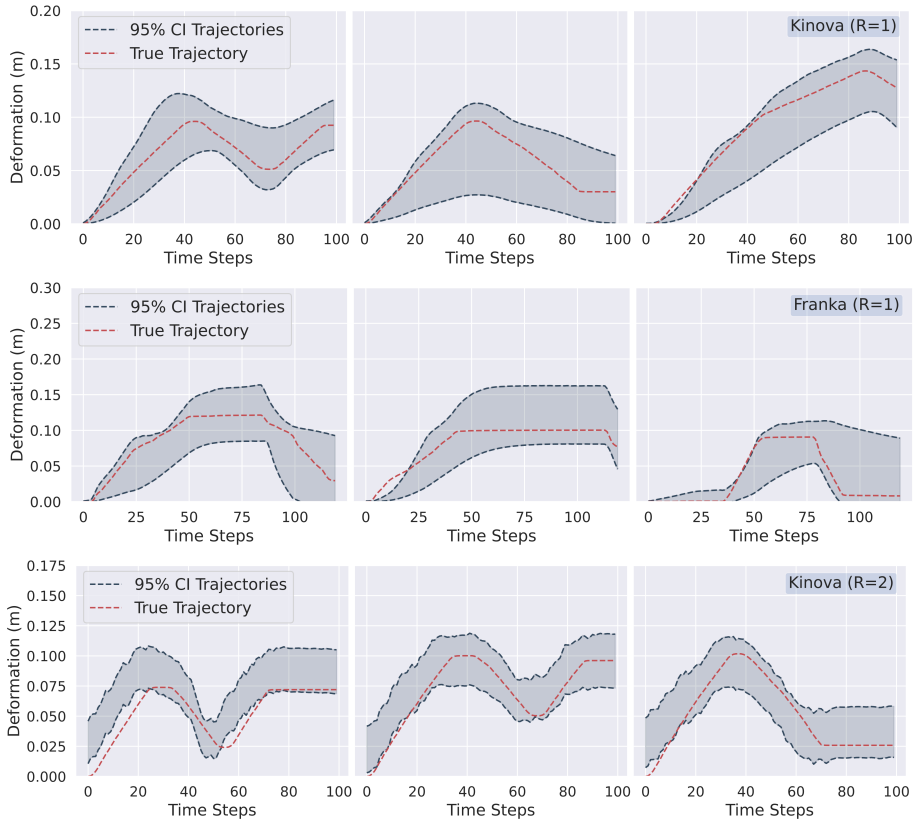


Figure 3.9: Real-to-sim system identification results. Top: Kinova ($R = 1$), Middle: Franka ($R = 1$), Bottom: Kinova ($R = 2$). Shaded regions represent the 95% CI of the predicted distribution $p(\tau_{sim}|\theta^*)$.

real-world predictions are obtained by using as little as three training trajectories. However, from the real-world branch behaviour, we observed that our approach works the best for deformation in regions that are closer to the horizontal ($+45°$ to $-45°$ from the ground axis). Given the force is applied along the gravity axis, this corresponds to a region where its normal component significantly outweighs the sheer. Therefore, the proposed method could be improved by using the branch angles captured through perception to apply normal forces to the branch instead, or alternatively, by representing a single real branch with multiple links to increase pliability, both of which we leave aside for future consideration.

3.6 Summary

In this chapter, we have demonstrated a method to learn the dynamic behaviour of tree branches by utilising the simulation-based parameter inference approach. In addition, we described a method for injecting biological assumptions into the Bayesian model via a neural network prior. Previous works on tree dynamics are either focused on perception or on the biological aspects of the tree; therefore, they are not designed for active branch manipulation, a key requirement for robotic harvesting and locomotion tasks. The comprehensive results indicate a robust model that captures the parameter uncertainty quite well even under noise and grasp location perturbations. However, notice that the overarching motivation for such a system identification method is to lead to robust manipulation policies. Therefore, in the following two chapters, we explore novel pathways for policy learning, building on the insights from this chapter.

All models are wrong, but some are useful.

George Box

CHAPTER 4

Gentle Branch Manipulation with Contact-Awareness

4.1 Introduction

Active interaction with natural deformables, such as tree branches, plants, and grass, has been a persistent challenge in robotics; however, the potential benefits are immense for diverse fields, including agriculture, autonomous navigation, and de-mining. The agriculture industry, in particular, benefits from automation as limited labour resources and excessive wastage threaten sustainable farming practices.

Robotic interaction with the natural environment is complex due to various factors. First, image-based perception, a cornerstone of deformable manipulation in labs, has limited success outdoors (Nguyen, Kuhnert & Kuhnert, 2012), as moving branches,

occluding foliage, and poor illumination disrupt camera vision. Second, the complex geometry and non-linear dynamics of natural entities limit model-based methods. Additionally, the deformable behaviour of tree branches necessitates reactive control policies that generalise to unseen scenarios. On the other hand, training model-free learners for long trajectories is computationally expensive, particularly with vision-derived states, while collecting real-world samples is complex. Vision techniques that work well with rigid objects are inadequate for deformable objects (Antonova et al., 2022) like vines and stems. Vision systems also fail to assess rigidity where stress-induced deformation is non-uniform. Consequently, state-of-the-art crop harvesting systems, predominantly vision-based planners, have success rates from 50% to 75% with partial occlusions, dropping to 5% when targets are fully occluded (Zhou et al., 2022).

While collision avoidance is conventional wisdom, contact is inevitable in messy environments; thus, minimising impact cost, both in plant damage and arm torques, is more prudent. Deep Reinforcement Learning (DRL) on powerful GPU-based simulators is a safer, sample-efficient, and cost-effective option for contact-rich tasks (Handa et al., 2023; B. Tang et al., 2023). However, transferring policies from simulation to reality, known as the sim-to-real problem (W. Zhao, Queralta & Westerlund, 2020), is notoriously hard due to discrepancies between the two. Domain Randomisation, referring to introducing perturbations in simulation parameters, helps create robust models generalising well to reality. Effective sim-to-real transfer and path planning with deformable contacts require realistic simulation of intricate geometry and deformation parameters to represent real-world occlusion patterns and interaction behaviour.

Given these challenges, this chapter proposes the following approach: First, we draw inspiration from plant morphology using the parametric L-system (Lindenmayer system) (Prusinkiewicz & Lindenmayer, 2012) to simulate realistic branching structures coupled with dynamics from crude spring actuation from Chapter 3. We domain randomise over both the L-system and dynamic spring parameters to generate a procedural forest for training our *Proprioceptive Contact-Aware Policy (PCAP)* for a reaching task, targeting, for instance, pruning locations, ripe fruits, or diseased regions. The PCAP states include only observations from proprioceptive sensors, not including vision or external torque sensors in favour of a blind reach policy. An independent classifier predicts contact from internal sensor measurements, which aids reward engineering. Finally, we perform zero-shot sim-to-real transfer by extending techniques

from B. Tang et al. (2023) to operate on real branches with varying dynamics. With just thirty minutes of simulation training on an RTX 4090, novel strategies emerge, allowing the arm to explore the task space while minimising contacts and contact impact. These strategies and the reactive behaviour translate well to the real world, where branch topology, obstruction patterns, dynamics parameters, and contact forces are all unknown. To summarise, our contributions in this chapter are as follows:

1. We devise a procedural forest generator based on the L-system formalism and Domain Randomisation, where the individual trees are faithful to the real world in geometric complexity, dimensional accuracy, and occlusion patterns.
2. We design an RL-based framework trained in simulation, which derives observations and rewards from an independent collision detection classifier and proprioceptive sensor measurements only.
3. We demonstrate that the trained reactive policy and the learned contact minimisation strategies can adapt zero-shot from simulation to real to navigate unseen branch topology and contact patterns with the help of a few engineering optimisations.

4.2 Related Work

This section reviews advancements in sim-to-real transfer, deformable manipulation and robot-plant interaction. Sim-to-real transfer is essential for applying simulation-trained policies to real-world tasks. Many studies focus on deformable object manipulation and contact-rich tasks. For instance, the IndustReal framework (B. Tang et al., 2023) used domain randomisation to bridge the sim-to-real gap. This approach varied simulation parameters, improving policy adaptability and performance in transferring complex assembly tasks. Antonova et al. (2022) used Bayesian methods and parameter inference to manipulate deformable objects while Fu et al. (2018) used sophisticated steering needles to operate inside a deformable environment, human lungs. W. Zhao, Queralta and Westerlund (2020) provided a survey on sim-to-real transfer, categorising methods into domain randomisation, domain adaptation, and imitation learning, highlighting their combined effectiveness. Dextreme (Handa et al., 2023) and Shadow

Hand (Andrychowicz et al., 2020) frameworks focus on agile in-hand manipulation, using high-fidelity simulations and comprehensive RL regimes for successful sim-to-real transfer.

Photo-realistic tree representations have been proposed with Lindenmayer-System (Prusinkiewicz, 1986; Prusinkiewicz & Lindenmayer, 2012; Prusinkiewicz et al., 1996), using polygons (Lintermann & Deussen, 1999; Weber & Penn, 1995), or with space colonisation algorithm (Runions, Lane & Prusinkiewicz, 2007). Furthermore, plant simulations for physical interactions are implemented with expensive FEM models (Y. Zhao & Barbič, 2013) or using cheaper mass-spring systems (Yandun, Silwal & Kantor, 2020), similar to Chapter 3. From a learning perspective, Yandun et al. (2021) explored an RL-based system aided by vision for known plant topologies, emphasising the need for detailed prior knowledge, often impractical in dynamic environments. C. H. Kim et al. (2023) developed graph representations to predict interactions between robots and plant structures but restricted to simulation where the node positions are known apriori.

Policy learning for contact-rich tasks typically relies on external force/torque (F/T) sensors (Kozlovsky, Newman & Zacksenhouse, 2022), tactile sensing (Andrychowicz et al., 2020), or vision (B. Tang et al., 2023). These add-ons work well for contacts localised to the end-effector, but with plant manipulation, contact can occur throughout the arm. Contact-aware reaching in plants using whole-arm tactile sensors is presented in Jain et al. (2013), while internal sensors are used in Pang and Tedrake (2022), but in a model-based indoor setting. In contrast, we use proprioceptive measurements and a model-free context. Accurate models of tree branch dynamics have been proposed by Jacob et al. (2024) (Chapter 3), using internal torque sensors to estimate parameters like stiffness and damping. More broadly, Zhou et al. (2022) surveyed intelligent robots for fruit harvesting, identifying trends and challenges, including the need for advanced perception systems and multi-modal sensors to improve accuracy and reliability in agricultural robotics.

4.3 Methodology

This section presents the architectural design and the methodology developed for this chapter. The methodology is visually outlined in Fig. 4.1.

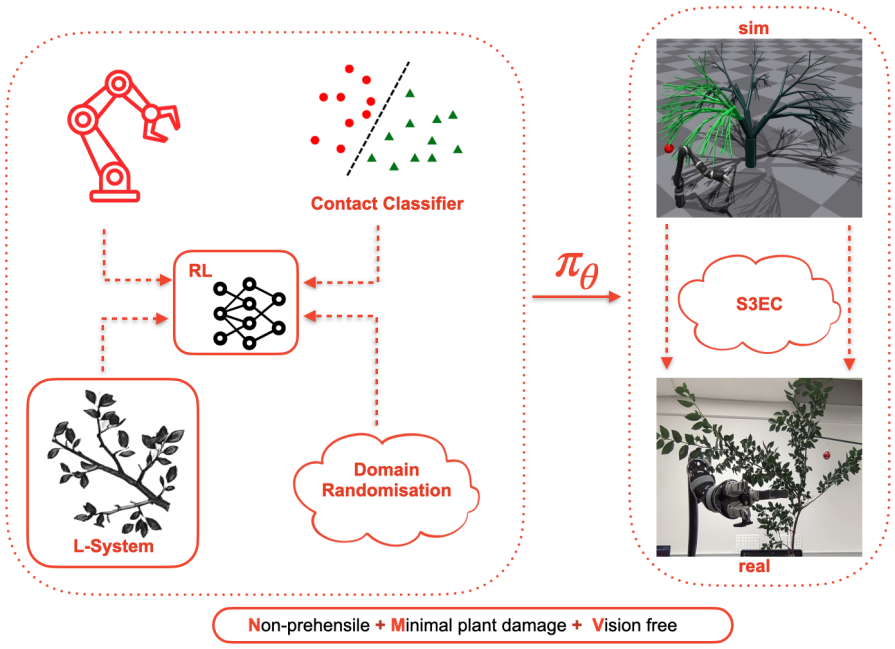
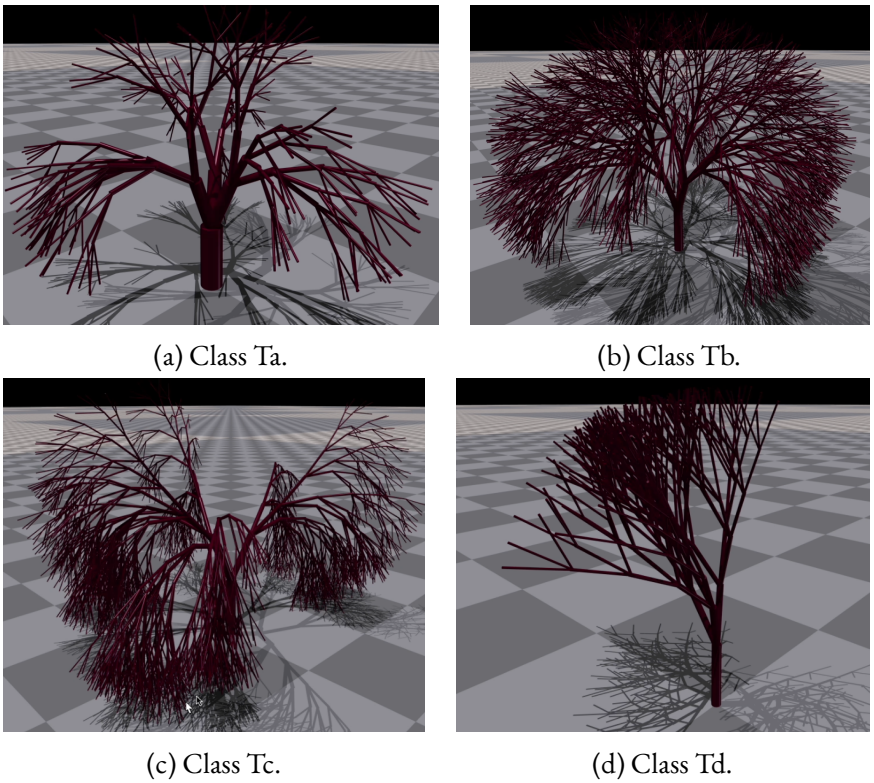


Figure 4.1: An overview of the proposed vision-free policy learning approach.

4.3.1 Shape Representation: L-System

The L-system formalism (Honda, 1971; Prusinkiewicz & Lindenmayer, 2012) uses formal grammar theory and fractal geometry to model the morphology of plants and other organic structures like algae. A parallel rewriting process updates morphological attributes by applying production rules, starting from an axiom and operating on a fixed alphabet of symbols. The resulting sequence of strings can be interpreted using turtle graphics (Prusinkiewicz, 1986), which moves in 2D or 3D space by attributing specific meanings to each symbol, such as move-forward or push-pop from a queue, to generate graphical structures akin to real-world organic growth patterns. The parametric L-system associates parameters with symbols to diversify and control limb generation. Formally, a parametric oL-system is defined by $G = \langle V, \Sigma, \omega, P \rangle$, where V is the alphabet, Σ the parameter set, ω the initial axiom, and P the production rules $\{p_1, p_2, p_3, ...\}$. We extend the *turtle* interpretation to physics simulators by generating independent cylindrical links that can be connected and actuated. We favour ternary branching structures over monopodial and sympodial variations, as real-world manipulation problems are more challenging with the former. We experiment with



(a) Class Ta. (b) Class Tb.

(c) Class Tc. (d) Class Td.

Figure 4.2: L-system derived intricate branching structures for ternary classes Ta, Tb, Tc, and Td, simulated on Isaac Gym.

four ternary structures: classes Ta, Tb, Tc, & Td, from (Prusinkiewicz & Lindenmayer, 2012). The ternary L-system we used and the parameters for Tb class Σ_b is given by the formulation:

$$\begin{aligned}
&\Sigma_b : \{d_1 \mapsto 137.5, d_2 \mapsto 137.5,\\
&\quad a \mapsto 18.95, l_r \mapsto 1.109,\\
&\quad v_r \mapsto 0.14, n \mapsto 8\};\\
&\omega : \frac{!(1)F(200)}{45A};
\end{aligned}
\qquad
\begin{aligned}
&p1 : A : * \rightarrow\\
&\quad !(v_r)F(50)[\&(a)F(50)A]/(d_1)\\
&\quad [\&(a)F(50)A]/(d_2)[\&(a)F(50)A] \qquad (4.1)\\
&p2 : F(l) : * \rightarrow F(l \cdot l_r);\\
&p3 : !(w) : * \rightarrow !(w \cdot v_r).
\end{aligned}$$

The simulated trees corresponding to the above formulation are shown in Fig. 4.2(a)-(d). For detailed rules, parameters, and growth attributes, see Prusinkiewicz and Lindenmayer (2012); extensive simulations are available in our website, refer Appendix B for more details.

4.3.2 Branch Dynamics

We extend the coarse-grained abstractions from Chapter 3 to model the deformable plant behaviour by approximating branch segments with L-system derived rigid cylindrical links, connected via revolute joints and actuated with Proportional–Derivative (PD) controllers. However, unlike Chapter 3, where the distribution of stiffness and damping parameters $\phi = \{K_p, K_d\}$ of the torsional mass-spring-damper system (Fig. 4.3b) is estimated from real deformations, we do not perform system identification, instead experiment with two models to populate them in advance, namely:

1. Rudimentary model: where ϕ is fixed for a given branch level (e.g., $R2$ in Fig. 4.3b represents the second level) and it decreases exponentially for each level away from the trunk, starting from an upper bound ϕ_u tuned manually based on a real tree branch, and subject to a lower bound ϕ_l to avoid simulator instabilities.

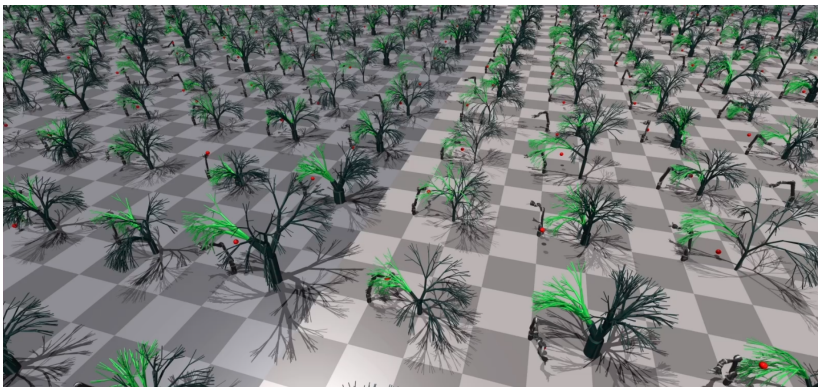
2. Beam deflection model (C. H. Kim et al., 2023):

$$K_p = \frac{E\pi r^4}{2l}, \quad K_d = K_p/10, \tag{4.2}$$

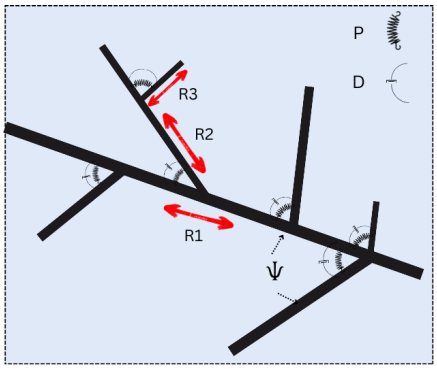
   where E is the Young's modulus of the plant material, and (r, l) are the radius and length of the links.

4.3.3 Domain Randomisation

We learn a robust policy despite the lack of strong vision priors by randomising over both the shape representation and the dynamics parameters. In the former case, we inject Gaussian perturbations ($\sigma = 0.1$) to the L-system parameters Σ, which represents structural traits such as divergence angle (d_1, d_2), elongation rate (l_r), width increase rate (v_r), etc., to create a procedural forest (Fig.Fig. 4.3b) for each ternary class, where individual instances have unique, diverse, and realistic forms. This paradigm fits neatly with our distributed physics simulator choice Isaac Gym (Makoviychuk et al., 2021), where individual tree instances can be placed into each parallel environment. As for the dynamics, while the deformation parameters ϕ of the rudimentary model are randomised (Peng et al., 2018) with a Gaussian ($\sigma = 1.0$), we do not perturb the Beam



(a) Training with our domain-randomised forest generator (ternary class Ta).



(b) Mass-Spring-Damper branch system, similar to Chapter 3.

(c) Real setup: Kinova arm reaching for a previously unseen target (in red), obstructed by a real tree branch.

Figure 4.3: Experimental setup: simulation and real-world environments.

deflection model as the L-system shape randomisation already influences the downstream dynamics. Furthermore, during the policy learning phase in the simulation, we randomise the reach target and the part of the tree the robot has access to.

4.3.4 Contact Detection

We use proprioceptive measurements to develop a reactive policy without vision guidance, specifically the noisy joint torques, following a supervised learning approach similar to Popov, Klimchik and Mavridis (2017) and Z. Zhang et al. (2020). Unlike prior works, however, where local collision detection is the end goal, we incorporate online predictions from a pre-trained model into an RL framework for a reach policy

with *gentle* contact.

The proposed online collision detector workflow is as follows: First, we execute arbitrary trajectories with a Kinova 6-DOF Jaco 2 arm, obstructing the path at regular intervals with a soft touch or real branch contact, recording obstruction times. Second, the resulting time series dataset, i.e., sequences of joint velocity $\dot{\theta}_t$ and joint torque τ_t, is annotated with collision labels (1-contact, 0-no contact) for each time step t. Next, we train a binary classifier to detect *bumps* in the signals, indicating contact. Finally, real-time predictions from the classifier are used for policy learning (see Section 4.3.5) as part of the state observation o_t and reward computation r_t. This classifier is deployed only during RL inference for real-world executions. In contrast, simulation training and testing use a proxy classifier to check if the net rigid body contact force measurements $F_t \in \mathbb{R}^{l\times 3}$, available from Isaac Gym for all l robot links, breach a force threshold, $\mathbb{I}(\|F_t\|_2 > f_u)$. The contact impact threshold f_u is manually tuned based on real branch compliance during training.

Feature extraction for the classifier focuses on amplitude variations, disregarding the frequency spectrum. We create sliding window features, $[\tau_t, \tau_{t-1}, \tau_{t-2}, ..., \tau_{t-(m-1)}]$, over the last m time steps. Apart from min and max, we compute statistical moments, including mean, variance, skewness, and kurtosis for each of the six dofs separately. We also use a smoothed version of τ_t by averaging the last k neighbours to account for high sensor noise levels. We add joint velocity values $\dot{\theta}_t$ (commanded and executed) and the raw torque measurements τ_t for each joint to the feature set. Overall, we capture 28 trajectories with soft contacts on each of the 7 links of the arm in 4 directions, plus 12 trajectories without obstructions. The final dataset has 90 features, with $m = 10$ and $k = 3$ performing best. We tune hyper-parameters manually and use two lightweight classifiers: a 3-layer Neural Network and a Random Decision Forest, settling on the latter for its high specificity.

4.3.5 Policy Learning

We formulate the reaching task as a discrete-time sequential decision problem where the agent attempts to maximise the expected discounted rewards accumulated by interacting with an environment over T time steps. Traditionally, such problems are modelled as an MDP framework (see Section 2.4), defined by the quintuple

$\langle S, A, P, R, \gamma \rangle$, where S is a finite set of possible states, A the set of actions taken by the agent, $P(s'|s, a)$ the state transition model, $R(s, a, s')$ the reward received on transitioning from s to s' upon action a, and $\gamma \in [0, 1]$ the discount ratio, subject to $s, s' \in S$ and $a \in A$. Given the agent has access only to a noisy representation of the state, $o \in O$, actions can be sampled from a learned stochastic policy $\pi_w(a|o)$ parameterised by the network weights w, i.e., $a \sim \pi_w$, where the learning objective was to maximise $\mathbb{E}_\pi\left[\sum_{t=0}^{T-1} \gamma^t R(s_t, a_t)\right]$. Specifically, we use Proximal Policy Optimization (PPO) (Schulman et al., 2017) variation from the rl-games library (Makoviichuk & Makoviychuk, 2022). The PPO method, detailed in Section 2.4.3, is designed to avoid extensive destabilising updates w.r.t the policy at a previous time step, by maximising a clipped surrogate objective.

Action & Observation Space: In our discrete-time setting, each action a_t corresponds to a 6-dof velocity target, clamped to the arm limits. In velocity control mode, the arm is essentially non-compliant to external disturbances, generating any amount of force to clear obstacles, subject to joint torque limits. We justify our use of velocity control over the more compliant torque/impedance control with similar reasoning as mentioned earlier, viz. deploying torque/impedance control requires accurate F/T sensor measurements, plant manipulation requires whole body contact detection, and Kinova torque readings suffer from high noise perturbations.

On the other hand, our observation o_t consists of the robot joint pose, joint velocity, deviation of the end-effector pose from the reach target, end-effector orientation, and most crucially, the predicted contact flag from the aforementioned classifier. Notably, our observation space does not include any real-time inputs from the tree, such as branch images or measurements from visual tags. Besides the cost implications, adding any of these inputs can cause an explosion in training time in the context of a distributed simulator, not to mention the corresponding sim-to-real hurdles.

Reward Formulation: Our extensive reward-engineering nudges the arm to evade plant contact, reduce impact upon contact, and minimise ruptures, thereby decreasing overall plant damage while reaching the target. We favour a dense reward function, described in Eq. (4.3), over episodic rewards for faster learning and easier credit assignment during each exploration step.

We formally define the reward structure as below:

$$r_d = \left[\frac{1}{1+\|d_t\|_2^2}\right]^2, \quad r_b = \begin{cases} 7r_d & \text{if } \|d_t\|_2 < 0.0125, \\ 3r_d & \text{if } \|d_t\|_2 < 0.025, \\ r_d & \text{if } \|d_t\|_2 < 0.05, \end{cases} \quad r_s = -\sum_{j=1}^{6} \dot{\theta}_{tj}^2, \tag{4.3}$$
$$r_c = \begin{cases} 1.2 & \text{if } \|F_t\|_2 \leq f_u, \\ 0 & \text{otherwise} \end{cases}, \quad r_p = \begin{cases} 0 & \text{if } \|F_t\|_2 \leq 2f_u, \\ -1.2 & \text{otherwise} \end{cases}.$$

The reward r_t at each training time step t consists of five distinct components along with their corresponding scaling factors, which are:

1. a distance reward r_d that progressively increases as the end-effector reaches closer to the target, where the distance $d_t \in \mathbb{R}^3$ is computed as the difference of the end-effector position from the target;

2. a bonus distance reward r_b for close target proximity required for grasps;

3. a smoothness reward r_s on the six joint velocities;

4. a collision reward r_c to encourage contact evasion or prefer low impact collisions;

5. a rupture avoidance reward to penalise extreme contact forces. For the proposed PCAP algorithm, we define $r_t = \{g_d \cdot (r_d + r_b) + g_s \cdot r_s\} \cdot r_c + r_p$, where g_d, g_s are scaling factors. The product term with r_c ensures that the agent receives distance rewards only for the steps where the contact forces are none or negligible; for instance, contact with leaves or thin limbs. Intuitively, the additional rupture penalty r_p is to account for cases where the agent learns to apply quick bouts of extreme forces, in an otherwise smooth path, to rupture branches as a strategic sacrifice to acquire the long-term reward of moving closer to the target through the breach, a behaviour that we experimentally observe.

4.4 Experimental Setup

4.4.1 Hardware Setup

Our policies are trained on a server equipped with NVIDIA RTX 4090 GPU, 40-core CPU, and 64GB of RAM, while the Kinova ROS api runs on a second workstation with a basic 8-core CPU, and 32GB of RAM, connected to the robot. They interact with each other via a custom REST service that controls the latency specifications of the arm.

The input joint velocity frequency requirement for Kinova ROS api is 60Hz-100Hz due to an on-board DSP that loops close to 10ms. On the other hand, the publish rate for sensor metrics ($\theta_t, \dot{\theta}_t, q_t$, e.t.c,) is 10Hz on the ROS topics. Therefore, to prevent oversampling at fetch, we nudge the policies during simulation training to generate actions at 10Hz; moreover, at the time of real deployment, we slice the output action a_t into n equal segments to feed the api at a higher frequency (e.g., 60Hz for $n = 6$), i.e, $\Delta\dot{\theta} = a_t/n$, given $\Delta\dot{\theta} \in \mathbb{R}^6$ for the six joints.

4.4.2 Segmented Steady State Error Control

Despite manual calibration, discrepancies persist in robot parameters, such as joint damping, friction, and gravity compensation values between simulation and real, and steady-state error is inevitable (B. Tang et al., 2023). Moreover, accurately replicating the complex contact dynamics between the branches and arm is challenging (Ferguson et al., 2021; Peng et al., 2018), especially when the robot operates in a velocity control mode. As an alternative to parameter estimation, (B. Tang et al., 2023) proposes to use *Policy Level Action Integrator (PLAI)* by integrating policy actions over time, effectively applying them to the desired state s_t^d in contrast to the current state s_t, i.e.:

$$s_{t+1}^d = s_t^d \oplus a_t = s_t^d \oplus \pi(o_t), \tag{4.4}$$

where $\oplus$ is the state update operation following a sampled action a_t from the policy π. Despite the absence of direct feedback control, except at policy evaluation with the current observation o_t, the authors show that PLAI acts as a continuous correction model robust to disturbances, compared to the conventional model, $s_{t+1}^d = s_t \oplus \pi(o_t)$, that transitions from the current state. However, when the control action regulates

joint velocity instead, we observe that the position error accumulates over time due to the integral effect, which is harder to control without direct feedback. Therefore, we follow a middle ground between PLAI and the nominal approach. We update from the last desired state for each segment within a time step but use the current state when updating across the trajectory time steps. We call our approach *Segmented Steady State Error Control (S3EC)*, as given by:

$$s^d_{i+1} = s^d_i \oplus \frac{\pi(o_t)}{n}, \quad s^d_i = \begin{cases} s_t & \text{if } i\%n = 0 \\ s^d_i & \text{otherwise} \end{cases}, \quad \text{for } i = 0, 1, 2, \dots, n \times t. \tag{4.5}$$

Intuitively, Eq. (4.5) could be viewed as local disturbance rejection over the short term, with periodic syncing to the current state to offset accumulated deviations.

4.4.3 Robot Parameters

The DR in Section 4.3.1 detailed how we introduce variability in both branch shapes and the tree dynamics. On the other hand, we do not randomise the robot dynamics. Instead, we tune the robot damping and friction parameters heuristically by comparing real torques and velocities to simulation-generated values, whereas the inertial priors are gleaned directly from Kinova specification. Furthermore, we perturb the measured contact forces in simulation with a small Gaussian noise ($\sigma = 1.$) to account for classifier generalisation error in real.

4.5 Experiments and Results

We evaluate the effectiveness of our proposed approach through the following three experiments.

4.5.1 Simulation Tests

Here, we compare four distinct policies by ablating over the observations and rewards: First, a baseline PPO that does not include a penalty for high contact forces and can, therefore, apply maximum arm torques (subject to arm torque limits) to push

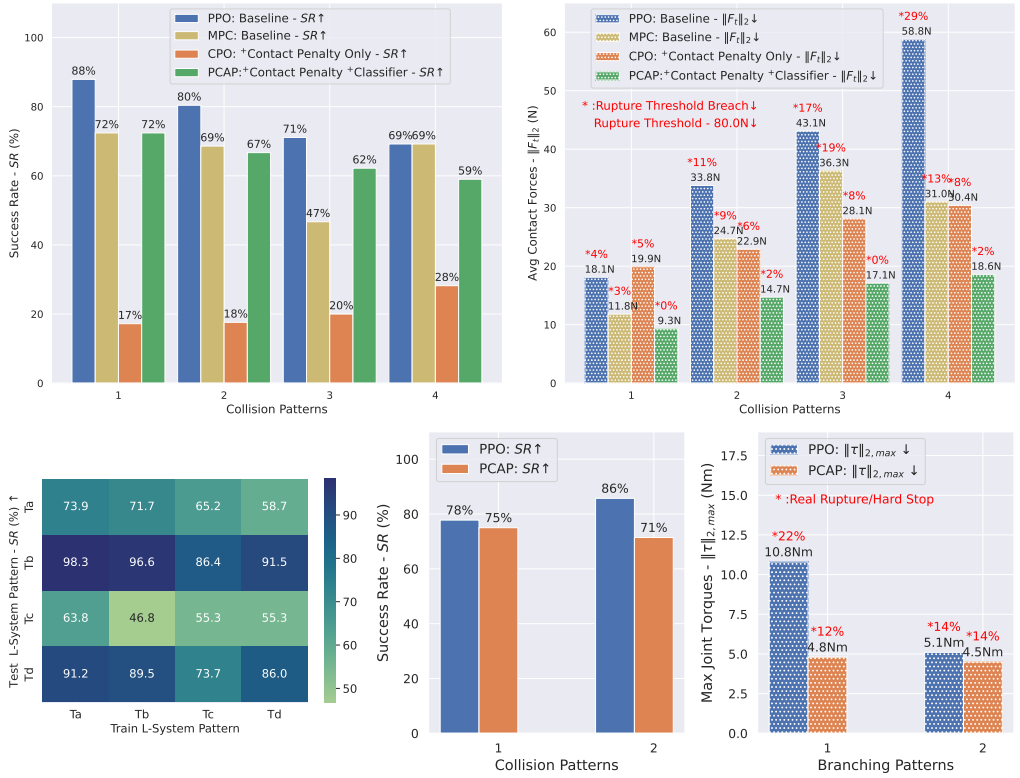


Figure 4.4: **(Top) Exp 1:** Comparison of Success Rate (SR) and Average Contact Forces across four policies: PPO, MPC, CPO, and PCAP (ours). For SR, higher values are better, for $\|F_t\|_2$ & Rupture values (red%), the lower the better.
(Bottom Left) Exp 2: SR for various L-System ternary classes. E.g., the cell (c:1,r:2,v:98.3) indicates a policy trained on a Ta forest but tested on a Tb instance.
(Bottom Right) Exp 3: Comparison of SR and Max Joint Torque averaged over trajectories, between PPO & PCAP. Note: $\|\tau\|_{2,max}$ (lower the better) is taken as a proxy for contact impact in real. Rupture values in red show %cases real branches broke or the arm had to be hard-stopped due to high forces.

through branch occlusions, disregarding any plant damage. Second, a Contact Penalty Only (CPO) policy uses our reward Eq. (4.3), but that does not take in the classifier predictions towards o_t. Third, a Model Predictive Control (MPC) baseline from Bhardwaj et al. (2022), however, without a prior for the tree geometry to ensure a fair comparison to our blind model. The fourth is our proposed PCAP approach, which combines both classifier prediction and reward Eq. (4.3). Specific policy details are presented in Table B.1, where $\theta_t, \dot{\theta}_t, q_t, \& \, d_t$ represent the joint angle (6D), joint velocity (6D), the gripper orientation (4D), and the distance of gripper from target

(3D). An important exception is that during real deployment, we replace $\mathbb{I}(\|F_t\|_2 > f_u)$ observation with the classifier prediction and update the reward terms accordingly. The results in Fig. 4.4: top, demonstrate that our PCAP framework is the best suited to avoid plant damage, with a comparable success rate to a baseline that ignores contact impact. To illustrate, the fourth block in both charts, corresponding to the hardest obstruction pattern, indicates that for a moderate drop in Success Rate (69 to 59%), the net forces acting on the plant reduce by $\approx$ 3x, while expected plant ruptures reduce by $\approx$ 15x. Additional experimental details, time limits, and success metric definitions can be found in Appendix B.3.

4.5.2 L-System Variability

We generate four distinct sets of ternary forests, each corresponding to classes Ta, Tb, Tc, & Td (Fig. 4.2). Then, we train PCAP policies on each forest and test its performance on all classes. The result heat map, presented in Fig. 4.4: bottom left, leads us to the following conclusions. Policies trained on class Ta outperform other classes (col 1), whereas the success rate is the lowest with class Tc (col 3). While numerous factors may impact the performance of a class, like the stochasticity during the forest generation and the specific combination of occlusions and target; primarily, this variation is a result of the hanging arch-like geometry (most evident in class Tc, arising from plant tropism) that varies between classes. In contrast, Tb class tasks are the easiest to solve (row 2), for all policies, presumably due to the thinner, pliant outer branches.

4.5.3 Sim-to-real

Here, we use real tree branches, with varying topology, extracted from two distinct species to obstruct the path between the Kinova arm and a reach target. However, in real, we compare only two policies: the baseline PPO and our contact-aware policy (PCAP). The policies trained beforehand in the Isaac Gym simulator are deployed in real as checkpoints and are used in conjunction with the pre-trained classifier described in Section 4.3.4. Unlike in simulation, where the contact forces between rigid bodies are available from Isaac Gym, in real, we use the joint torques as a proxy metric to compare PPO and PCAP. Specifically, we compute $\|\tau_t\|_2$ at each time step and choose the maximum within each test trajectory to indicate plant damage. Overall,

the real-world experiment results in Fig.(Fig. 4.4: bottom right) echo the simulation tests, proving that PCAP mitigates branch impact with solid success rates and successful sim-to-real transfer. **Additional experiments** are included in the Appendix for time efficiency (Appendix B.4), geometry ablation with and without L-system (Appendix B.5), dynamics ablation (Appendix B.7), and the classifier performance (Appendix B.8). Videos can be found in the website, refer (Appendix B.1).

4.6 Discussion

Throughout our experiments, in simulation and real, we observe the agent deploying novel and unexpected strategies to evade contacts and reduce contact forces. For lack of a better definition, we consider a behaviour novel if an ethologist would likely consider it *intelligent* if displayed by a biological organism. We encourage readers to view the extensive videos on our website and the supplement to see these behaviours in action.

Some strategies are highly beneficial but unsurprising, such as the arm slowing down (i.e., reducing robot joint acceleration) to mitigate the impulse from collision or pushing through regions with lower resistance, like pliant green foliage. In contrast, other demonstrated characteristics are unforeseen. First, the agent learns to use the continuous joints of the Kinova arm at the wrists to perform a rolling motion along obstructions. This rolling motion is observed both along and against the gravity axis (i.e., rolling up and down upright limbs) and in lateral directions. Second, from training, the agent forms an abstract awareness of the arch-like canopy shape typical of real-world plants. We deduce this from its attempts to repeatedly favour trajectories through the outer periphery of the arch, where the stiffness is lower, to get under the arch where there is no resistance, even when a shorter path exists through the top. Third, the agent pulls the arm back towards itself, often multiple times, when entangled in runners or facing stiff resistance. Furthermore, we notice the arm sliding on thicker branches, wiggling through occlusions, and shaking off entanglements. Arguably, the most vital result of our framework is the ability to perform reactive exploration of the task space; for instance, the arm pulls back from a heavy impact, changes trajectory direction, and advances again, often in repeated cycles, in search of a low-cost path. A few striking similarities between these emergent behaviours and evolved animal traits in nature are listed in the next section.

4.6.1 Resemblances to Biological Traits

Remarkably, some of the demonstrated PCAP behaviours have evolved in nature as well. For example, positive thigmokinesis refers to a pattern in which an organism, such as woodlice (Friedlander, 1964), slows down in response to external contact. Similarly, many organisms, such as rodents and ground beetles, exhibit both negative and positive thigmotaxis (Treit & Fundytus, 1988), orienting themselves away or towards impact as a movement response to tactile stimuli. Likewise, rolling behaviour is exhibited by many species for defensive locomotion, such as caterpillars (termed ballistic rolling) (H.-T. Lin, Leisk & Trimmer, 2011) and chitons (termed volvation). In some instances of our experiments, particularly when rolling and sliding strategies are exhibited, we observe that the agent exploits the momentum from the branch resistance to advance the arm in the direction of the opposing force, not unlike squirrels and monkeys which utilise branch momentum to traverse among trees. Nevertheless, we emphasise that given the enormity of evolutionary pressures and intents that shape biological complexity, our mention of similarity to animal behaviour should only be taken as a curious resemblance rather than as a scientific claim.

4.6.2 Failure Modes

There are multiple scenarios where PCAP fails in both simulation and real (refer to supplement video). In simulation, these are primarily due to the arm failing to reach the target within the specified step limit (sim: 800). Two frequent causes are (i) the arm getting entangled in branches from which it cannot retract and (ii) the arm exploring various paths without success. The agent fails the max limit (real:1000) from exploration in real as well in addition to ruptures from shoving motions and entanglements; although we hard-stop in advance in at least some cases.

4.7 Summary

We introduced a Proprioceptive Contact-Aware Policy (PCAP) for the delicate manipulation of tree branches, using a simulated forest with realistic branching structures derived from a parametric L-system model. Our approach avoids reliance on vision or external torque sensors, leveraging proprioceptive measurements and a contact

detection classifier to train reinforcement learning policies. We demonstrated the effectiveness of our method in both simulated and real-world environments, highlighting its robustness and adaptability to varying dynamics and unseen branch topologies. In this chapter, we acknowledge that the deliberate absence of vision-based feedback may limit performance in highly unpredictable scenarios. However, the objective here was to push the limits of a purely proprioceptive approach. In the following chapter, we therefore investigate efficient strategies for integrating this proprioceptive detector with vision-based feedback to enable robust policy learning.

The key to artificial intelligence has always been the representation.

Jeff Hawkins

CHAPTER 5

Deformable Cluster Manipulation with Whole-Arm

5.1 Introduction

Learning to manipulate deformable objects poses a formidable challenge in robotics (H. Yin, Varava & Kragic, 2021; Zhu et al., 2022); however, operating with clusters of deformables, such as cable bundles, multi-branch tree canopies, and cloth piles, is significantly harder. Although imprecise physics knowledge (Arriola-Rios et al., 2020) and sensor uncertainties are contributing factors, the principal challenge lies in inferring a compact, low-dimensional state abstraction and the temporal state evolution, based on noisy perception from a high-dimensional ground truth (Antonova et al., 2022). Furthermore, with deformable clusters, tiny differences in contact locations, material

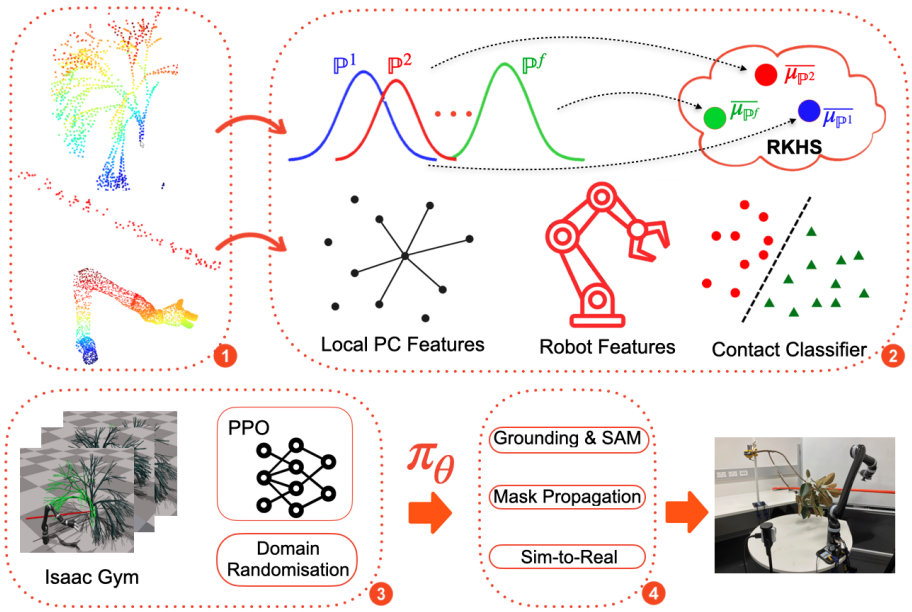


Figure 5.1: An overview of the proposed deformable cluster manipulation approach: (1) Segmented point clouds corresponding to the clustered deformable branches, clearance region, and the robot are captured from the scene. (2) Distribution embeddings representing global scene are generated via kernel mean operator. Additional features include neighbourhood point features, robot sensor metrics, and proprioceptive contact indicators. (3) RL training with domain randomised geometry and dynamics on Isaac Gym parallel simulator. (4) Inference & zero-shot transfer to the real world aided by Grounding DINO, SAM-HQ, and Cutie frameworks.

properties, and initial configuration among the individual cluster constituents (e.g. between thicker and thinner plant stems) give rise to non-linear resistance forces and divergent motion trajectories upon interaction. The resulting chaotic dynamics render model-based learning infeasible, while model-free solutions have high sample complexity, and therefore, the compactness and generation efficiency of the state representation gather significance (Maillard, Ryabko & Munos, 2011).

Novel deformable works manipulating cables (Grannen et al., 2020), dough (X. Lin et al., 2022), and clothes (Deng & Hsu, 2024) are all planning approaches; furthermore, they exclusively focus on end-effector-based control. In a cluster context, such grasp-based solutions tailored to individual deformable members become prohibitively long. On the contrary, a multi-link manipulation strategy substantially escalates the contact complexity, necessitating additional perception modalities apart from vision. However,

typical auxiliary sensors for whole-arm contact detection are either expensive (Jain et al., 2013) or too slow (Huang et al., 2024) for low-latency applications.

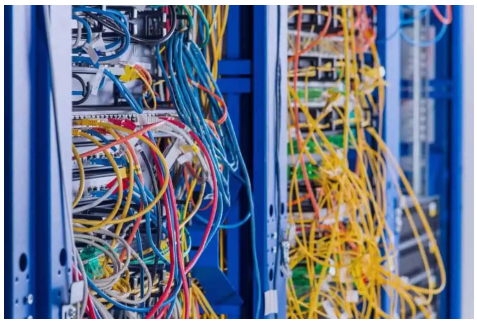
Addressing these challenges, we propose a model-free, non-prehensile, multi-modal, reinforcement learning approach (Fig. 5.1) operating on the cluster as a whole with full-body contact, but without external tactile sensing. We use a distributional representation (Antonova et al., 2022) of the cluster state generated by embedding point cloud samples into a Reproducing Kernel Hilbert Space (RKHS), without key-point intermediaries. Compared to traditional point cloud feature extractors, a distribution embedding is notably lightweight, enabling faster training and real-time Reinforcement Learning (RL) feedback. It eliminates the need for points to track identical cluster locations, is invariant to permutations, and can handle variable input sizes resulting from cluster members moving out of camera view between time steps. Further, the observations include a whole-arm contact classifier from Chapter 4 output from proprioceptive measurements to flag contact. Finally, we address the high sample requirements of on-policy RL by leveraging the parallelism of physics simulators for policy training and transferring it zero-shot to a real-world setting with completely unseen geometry and dynamics.

This work focuses on a specific subclass of problems, aiming to de-occlude local regions of the deformable cluster; for instance, to clear a cylinder-shaped region among tree branches, by manipulating a few of them together. Illustrative applications of such a skill abstraction include:

- *Clearing Power Lines*: Hazardous tree branches encroaching on overhead power infrastructure is a leading cause of forest fires and outages (Guggenmoos, 2003; Lowe et al., 2022). Pruning foliage near overhead lines (Fig. 5.2a). is dangerous for humans due to risks like falls, shocks, and tool accidents (Siebert et al., 2014); therefore, robotic assistive systems that can clear the branches away from the lines before pruning can enhance safety while reducing outages. In this instance, the cylindrical clearance region can be thought of as concentric to the physical power line.

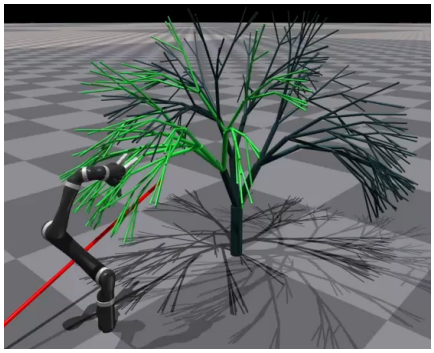
- *Autonomous Inspection*: Modern electrical and networking infrastructure, such as a data centre, is cluttered with deformable electrical and network cables obstructing inspection cameras (Fig. 5.2b). Autonomous monitoring robots

(a) Manual pruning of overhanging branches impeding power lines, from Siebert et al. (2014).



(b) Network cables occluding autonomous server inspection.

(c) Our laboratory setup mimicking the power line clearance scenario.



(d) Simulation in Isaac Gym with a parametric L-system ternary structure.

Figure 5.2: Deformable Clusters: Problem scenarios and experimental environments.

must flexibly manipulate cable bunches to restore camera line-of-sight for back panel inspection and anomaly detection. Here, the clearance region may be a virtual cylinder with the inspection camera on one face and the target panel on the other.

- *Agricultural Exposure*: Modern harvesting systems suffer from significantly lower success rates with partially (50%-75%) and fully occluded (5%) fruits (Zhou et al., 2022). Manipulating plant stems for exposing fruits or a diseased region to an external sensor requires a line-of-sight restoration through multi-branch clearance.

Among these, the first problem presents the most significant challenges due to the additional constraints imposed by the rigid power line on the robot's motion, and

thus serves as the primary focus of this chapter in both simulation and real settings, illustrated in Fig. 5.2 (c) & (d). While our aim is not to provide an industry-grade system, it is noteworthy that no fully autonomous solutions currently exist for the power line problem. Additionally, we provide simulation videos for the agricultural exposure task to emphasise the generality of our approach. Formally, our contributions in this chapter are as follows:

1. A multi-modal, whole-body contact policy framework to manipulate deformable clusters, learned from both point clouds and proprioceptive touch detection inputs.

2. An efficient distributional state representation of the complex deformable cluster geometry, with low computational overheads, in an RL policy learning context.

3. A novel context-agnostic occlusion heuristic and reward strategy generalizable to various exposure applications to clear deformables from a target region.

4. A specific implementation to clear power lines of overhanging tree branches and zero-shot sim-to-real transfer that handles unseen branch topology and displays novel clearance patterns in simulation and real.

Subsequent sections specifically focus on the power line clearance problem; however, the methodology is readily adaptable to others. We encourage readers to view the extensive supplementary videos to see our approach in action.

5.2 Related Work

For deformable manipulation with high-dimensional object geometries, point clouds offer a rich sensor-agnostic scene mapping. Notable 1D implementations for tracking (T. Tang & Tomizuka, 2022) and learning dense depth object descriptor features (Sundaresan et al., 2020) leverage point representations. Point clouds have been used to learn policies with 2D and 3D deformables as well for cloth manipulation (Deng & Hsu, 2024), liquid scooping (Seita et al., 2023), and dough cutting (X. Lin et al., 2022). On the policy front, these deformable works use either Motion Planning with primitives (Deng & Hsu, 2024), Task and Motion Planning (X. Lin et al., 2022), or

Behavioural Cloning (Seita et al., 2023; Sundaresan et al., 2020). In contrast, model-free reinforcement learning works on deformables like ours are scarce, largely due to poor sample efficiency. With rigid body point clouds, model-free policies have been proposed (M. Liu et al., 2022; Qin et al., 2023) for interactive tasks, including sim-to-real transfer. Relevant to our setup, a key study with rigid bodies from Ling et al. (2023) found that RL with 3D point clouds outperforms its 2D image counterpart when object-object relationship encoding is crucial for the task.

Post-scene capture, common deformable representation strategies include keypoints (T. Tang & Tomizuka, 2022), graphs (Ma, Hsu & Lee, 2021), and dense-descriptors (Sundaresan et al., 2020). Particularly relevant to this chapter, the deformable state can be represented as as a distributional embedding with a kernel mean operator (Antonova et al., 2022). They tackle a real-to-sim inference problem with an RKHS-net layer using a small number of keypoints (≈ 10) extracted from images. In contrast, we project entire point clouds (≈ 512) of multiple objects to RKHS and use the embeddings directly as RL observations in a sim-to-real context.

Our policy enables the arm to contort itself into shapes, such as hooks and props, leveraging the whole arm to push away the occlusion. While such formations are less prevalent, contact-aware reaching has been achieved with external full-body tactile sensors, for rigid clutter (Jain et al., 2013), or with internal sensors for tree branches, as described in Chapter 3. However, we go beyond simple reach to sophisticated clearance tasks and implement a multi-modal policy integrating touch and vision feedback.

Among deformables, interacting with plant structures is substantially harder owing to the non-uniformity in geometry, dynamics, and visual features. Yao et al. (2024) proposes a method to gently grasp and move a leaf aside to estimate the pose of hidden fruits. On the other hand, X. Zhang and Gupta (2023) pushes away multiple leaves simultaneously with the end-effector to reveal the space behind them using large amounts of real-world data. Both works assume soft leafy resistance, which is typically addressed in farms with simpler tricks, such as blowing compressed air (Yamamoto et al., 2014) to separate leaves and fruits. From a learning perspective, Yandun et al. (2021) and Subedi et al. (2025) employ vision-based RL for reaching vine pruning locations and for exposing fruits, respectively; however, both assume known plant topology while focusing on the end-effector motion. In comparison, our approach is designed to accommodate stiff resistance from multiple branches, uses the whole arm

to maximise applied torques, and is trained in simulation without any digital twins, plant mesh similarity, or real-world data. Finally, none of the aforementioned works are designed for the hybrid rigid-deformable problem of power line clearance.

5.3 Methodology

5.3.1 Deformable Simulation

Realistic simulation of the intricate tree-branch geometry is challenging; fortunately, this is well-studied in botanical literature under the L-system paradigm (Prusinkiewicz & Lindenmayer, 2012), predominantly for visualisation. An L-system borrows from formal grammar theory to model plant morphology by applying a recursive rule collection, starting from an axiom, mimicking the branching patterns. In Chapter 4, we have seen that this paradigm can be exported to 3D physics simulators, replete with realistic dynamics, by representing branch segments as rigid cylindrical links connected by revolute joints, amenable to robotic interaction. In this setup, branch deformations are modelled with a mass-spring-damper, from Chapter 3, actuated with proportional-derivative controllers. We borrow the recursion rules, growth attributes, and L-system parameters from Chapter 4; however, we extend it with spherical joints to enable multi-axis deformation and enhance compliance for clearance, refer Fig. 5.2(d). We run thousands of such tree models in parallel on the distributed physics simulator Isaac Gym (Makoviychuk et al., 2021).

5.3.2 Policy Learning

We formulate the de-occlusion task as a discrete-time MDP (see Section 2.4), where an agent learns a stochastic policy $\pi_w(a|o)$; parameterised by weights w. An MDP is defined by $\langle S, A, P_a, R_a, \beta \rangle$, where S is states, A is actions, P_a and R_a are the state transition model and the reward from action $a \in A$, and β is a discount factor. The agent aims to maximise $\mathbb{E}_\pi\left[\sum_{t=0}^{T-1} \beta^t R_a(s_t, a_t)\right]$, the expected discounted rewards over T steps. Crucially, the agent only has access to noisy observations $o \in O$; for instance, point clouds from the camera or robot sensor measurements rather than the true state. Our algorithm choice is Proximal Policy Optimisation (Schulman et al., 2017) from

RL Games (Makoviichuk & Makoviychuk, 2022). See Section 2.4.3 for more details on PPO.

5.3.3 Kernel Mean Embedding

In this section, we briefly re-examine relevant concepts from distribution embedding (Section 2.2.3) and its spectral approximation (Section 2.2.4) before applying it to deformables.

Akin to point mappings, a kernel mean operator lifts a probability measure $\mathbb{P}$ to a single mean function in $\mathcal{H}$, however, without any loss of information for a characteristic kernel such as RBF:

$$k(x, x') = \exp(-\frac{||x - x'||^2}{2\gamma^2}). \tag{5.1}$$

When the underlying distribution is not explicitly known, which often is the case, an empirical kernel mean $\overline{\mu_{\mathbb{P}}}$ can approximate the true kernel mean $\mu_{\mathbb{P}}$ from just the i.i.d samples $\{x_1, \dots, x_N\}$ drawn from the distribution, as given by:

$$\overline{\mu_{\mathbb{P}}} := \frac{1}{N}\sum_{i=1}^{N}\varphi(x_i) = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\langle\varphi(x_i), \varphi(x_j)\rangle_{\mathcal{H}}. \tag{5.2}$$

Furthermore, representing kernel functions through the Random Fourier Features (RFFs) (Rahimi & Recht, 2007), significantly reduces the computational overhead, leading to the formulation:

$$\overline{\mu_{\mathbb{P}}} := \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\sqrt{R}}\Big[\cos(\omega_1^\top x_i), \sin(\omega_1^\top x_i), \dots, \sin(\omega_R^\top x_i)\Big]. \tag{5.3}$$

For the deformable cluster manipulation problem at hand, we imagine x to be points from the 3D space of point clouds, to represent the camera vision feedback. In addition, we choose $R = 16$ and use the RBF kernel, which results from the Fourier transform of a Gaussian $p(\omega) \sim \mathcal{N}(0, 1)$.

5.3.4 Distribution Embedding for Deformables

Given a segmented point cloud $P_t \in \mathbb{R}^{N\times 3}$ at time-step t, our spatial abstraction assumes it as samples from an underlying distribution, regardless of the deformability.

The points in 3D coordinate space are elevated to the feature space $\mathcal{H}$ with an RFF approximated kernel mean, as described in Section 5.3.3, to form the key constituent of the observation space O. We capture four independent point clouds of varying sizes: P^{rob}, P^{clr}, P^{wbr}, P^{zbr}, representing the robot, clearance region (i.e. a slice of the power line), the whole tree, and a zoomed-in version of the branches near the occlusion. The robot points are sampled from the URDF mesh based on link positions at each time step. Points representing the cylindrical clearance region are sampled from the cylinder surface, given its fixed position and orientation. In contrast, camera sensors provide the deformable and time-varying points of the occluding branches.

There are several advantages to such a representation. Firstly, although the explicitly mapped feature locations are unknown, the agent can exploit the relative similarity of embedded points in RKHS, for instance, to minimise the foliage-power line entanglement or maximise the gripper-branch proximity, resembling the 'kernel trick' common in machine learning literature. Secondly, a distributional interpretation is invariant to point permutations and robust to the high noise content characteristic of streaming point clouds. Third, unlike conventional point cloud feature extractors like PointNet++ (Qi et al., 2017), kernel embeddings are resilient to variations in the size and structure of input point clouds, amplified by the arm, branches and the power line frequently occluding each other. Finally, and most crucially, this approach is significantly faster, reducing training time and enabling real-time inference, as our experiments demonstrate. This speed advantage stems from formulation in Section 5.3.3, where the feature map $\hat{\varphi}(x)$ can be derived with a single matrix product of the input points x and the pre-computed Fourier coefficients $\omega_r^\top$ of complexity $\mathcal{O}(N \times R)$, followed by a sin/cos transformation, and an additional row-average to obtain the embedding $\overline{\mu_{\mathbb{P}}}$, a highly efficient pipeline scalable to thousands of environments, amenable to GPU parallelisation.

5.3.5 Occlusion Heuristic

A key consideration in the de-occlusion problem is determining an effective yet simple measure to capture the temporal progression in occlusion level from manipulation. Common metrics include visible surface ratio, pixel counts (Bai et al., 2025) (e.g., red fruit pixels in a green foliage background), or labelled occlusion levels. However, these

metrics depend either on prior knowledge of the target geometry or partial observability at start, and they may not be smooth enough for an RL agent. In contrast, our occlusion heuristic $h_t \in [0, 1]$ is independent of target characteristics or even its presence.

Given segmented point clouds $P^{(1)} \in \mathbb{R}^{N_1 \times 3}$, $P^{(2)} \in \mathbb{R}^{N_2 \times 3}$ of two entangled entities, the obstructing branches P^{zbr} and a virtual clearance region P^{clr} in our case, the h_t is computed as follows. First, we define a pairwise distance matrix $D \in \mathbb{R}^{N_1 \times N_2}$:

$$D_{i,j} = \|p_i^{(1)} - p_j^{(2)}\|_2 \text{ for } i = 1, \dots, N_1;\ j = 1, \dots, N_2. \tag{5.4}$$

Let $\mathcal{S} \subset \{1, \dots, N_1\} \times \{1, \dots, N_2\}$ be the indices set corresponding to the k smallest distances in D, i.e., the (i, j) indices of the k nearest-neighbor point pairs, and d_{th} be a distance threshold, then:

$$\mathcal{D}_k := \{D_{i,j} \mid (i, j) \in \mathcal{S}\}, \tag{5.5}$$

$$h(P^{(1)}, P^{(2)}) := \frac{1}{k} \sum_{d \in \mathcal{D}_k} \mathbb{I}(d < d_{\text{th}}). \tag{5.6}$$

Intuitively, it is the fraction of point pairs (from different groups) that breach a proximity threshold to the total number of pairs in the neighbourhood, where both the threshold d_{th} and the neighbourhood size k are tunable. In general, the larger the number of branch segments covering the clearance region, the higher the h_t. We set $k = 200$ and the threshold to be the clearance region cylinder diameter $d_{th} = 10$cm. We emphasise that the clearance region has no shape constraints due to its point representation; for instance, it can be defined to de-occlude an expected curved end-effector trajectory of a second manipulator in a dual-arm setting.

5.3.6 Observation & Reward Space

Our observation space O consists of five feature sets, listed in Table 5.1. First, we construct two distinct feature sets from point cloud information: the KME features described in Section 5.3.4, which characterise global structure, and a local feature group, which focuses on nearby point distances. The latter includes the Euclidean distance of the end-effector to the closest $k = 5$ branch points and a safety breach

Table 5.1: Observation Feature Groups

Group	Feature	Shape
Proprioceptive	joint pose: q	6
	joint velocity: $\dot{q}$	6
	ee quaternion: r_{EE}	4
KME	whole branches: $\overline{\mu_{\mathbb{P}}}[P^{wbr}]$	16
	zoomed-in branches: $\overline{\mu_{\mathbb{P}}}[P^{zbr}]$	16
	clearance region: $\overline{\mu_{\mathbb{P}}}[P^{clr}]$	16
	robot: $\overline{\mu_{\mathbb{P}}}[P^{rob}]$	16
Touch	contact indicator y/n: $\mathbb{I}(\|F_t\|_2 > f_u)$	1
Local PC	ee-branch dist: $\mathcal{D}_{k=5}(P^{ee}, P^{wbr})$	5
	safety breach indicator y/n	1
Occ Heuristic	$h_t(P^{wbr}, P^{clr})$	1
All Features		**88**

indicator to reduce the robot-line contacts, computed as:

$$\delta_{sm} = \mathbb{I}\left\{\frac{1}{|\mathcal{D}_5|}\sum_{d\in\mathcal{D}_5} d < d_{\text{sm}}\right\};\ \mathcal{D}_5 = \mathcal{D}_{k=5}(P^{rob}, P^{clr}). \tag{5.7}$$

As the branches overhanging the power line already act as conductors (Guggenmoos, 2003), the robot touching the power line in itself does not compromise the validity of our approach; nevertheless, a safety margin $d_{sm} = 4$cm is introduced to reduce the wear and tear to the line through reward shaping.

A novel aspect of our method is its multi-modality, attributable to a whole-arm touch detector. We have shown in Chapter 4 that in the absence of expensive tactile sensors, time-series sliding window torque measurements, such as:

$$[\tau_t, \tau_{t-1}, \tau_{t-2}, ..., \tau_{t-(m-1)}], \tag{5.8}$$

over the last m time steps along with other proprioceptive features, can be used to build a classifier trained to detect contact *bumps* at the current time-step t, from labelled touch data. While in Chapter 4, it is trained with touch features for a simple reach task, we extend that approach and demonstrate that the touch classifier is useful even

in the presence of vision and can aid complex manipulation tasks. During simulation, a proxy classifier $\mathbb{I}(\|F_t\|_2 > f_u)$ computes the force threshold f_u breaches by the robot links l, leveraging the net contact force $F_t \in \mathbb{R}^{l\times 3}$ exposed by Isaac Gym. The full body contact awareness from this touch detector is crucial for the arm to leverage its multiple links (e.g., forearm and wrist) for manipulation, to supplement the partial, noisy, and time-varying point clouds. Our simple reward structure listed below has three components:

1. a clearance reward r_h to encourage arm towards lower occlusion score h_t;
2. a smoothness reward r_q on the joint velocities $\dot{q}$;
3. a bonus for the time-steps when the arm links are clear of the power line safety margin.

The three components can be formalised as:

$$r_h = \left[\frac{1}{1+h_t^2}\right]^2,\ r_q = -\sum_{j=1}^{6}\frac{\dot{q}_{tj}^2}{100},\ r_{sm} = 0.4\cdot\mathbb{I}_{\delta_{sm}=0}. \tag{5.9}$$

5.3.7 Domain Randomisation

This work aims to generalise a learned manipulation policy to deformable scenarios with unseen geometry, undetermined dynamics, and devoid of chromatic affinity, by not relying on digital twins (Subedi et al., 2025), parameter inference (Antonova et al., 2022), and RGB images respectively during the training phase. Instead, we perturb simulation parameters, for example, L-system structural traits such as branch divergence angle and segment elongation rate, and dynamics parameters such as spring stiffness, damping and friction, among others. In particular, we sub-sample and permute all training point clouds while adding Gaussian noise during each time-step.

5.4 Experimental Setup

5.4.1 Simulation Setting

For the power line clearance scenario illustrated in Fig. 5.2(d), we run 6144 Gym environments in parallel, each of which contains a tree with deformable branches (in

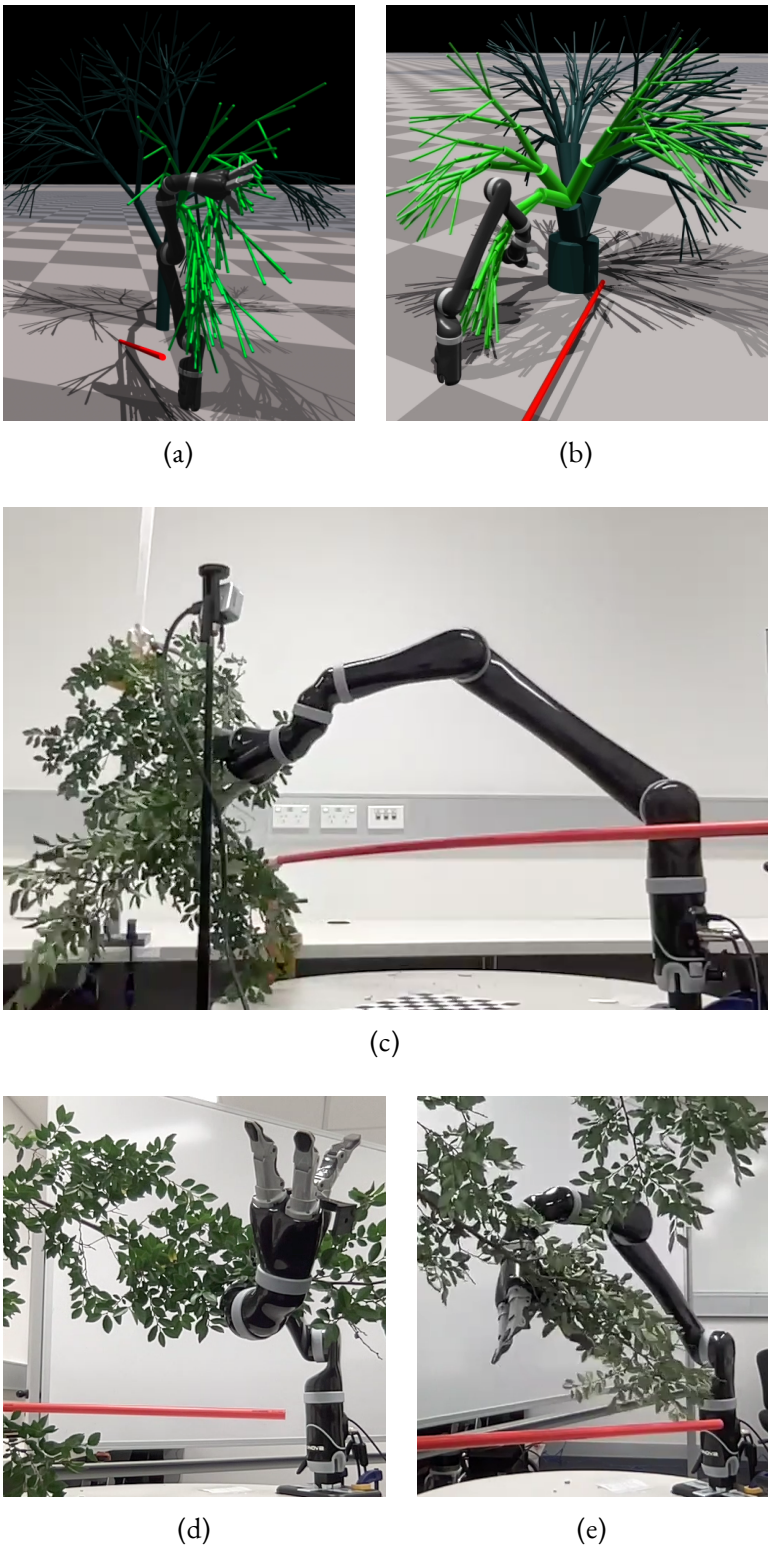


Figure 5.3: Clearance strategies and terminal state poses: (a)(b) Simulation trajectory states showing the whole arm being utilised to shield the power line. (c)(d) & (e) Similar real strategies using various arm links for clearance.

green), a rigid power line (in red), and a Kinova arm actuated via velocity control of its six link joints, omitting the end-effector. Each tree is formed by drawing from a Gaussian distribution ($\sigma = 0.1$) representing the L-system morphology parameters to ensure high diversity in topology but sufficient physical feasibility, in addition to arbitrary trunk rotation to expose the agent to varying branch occlusion patterns. Furthermore, the power line location is randomised, but constrained to the reach region of the gripper and semi-parallel (within $\pm 15°$) to the axis connecting the trunk and the robot base. Finally, we perform all experiments with an average $h_t \in [0, 1]$ of at least 0.7 across all environments, noting that occlusion score can increase as well from the arm pushing more branches closer to the power line.

A critical challenge within this framework is the contact explosion resulting from the combinatorial interactions among the multi-link arm, multi-branch tree, and the power line, exacerbated by a large number of training environments. This issue, common in contact-rich parallel simulations, can quickly overwhelm the simulation constraint solvers, triggering object inter-penetrations and CUDA memory overflows. It has been extensively studied by B. Tang et al. (2023) and Narang et al. (2022), which suggest complex CUDA-level penetration checkers and mesh-level contact binning, respectively. In contrast, we take a shortcut to contact reduction at the cost of a marginal drop in test performance. Specifically, during training, we allow the branches (but not the robot) to penetrate the power line through collision masking, thereby removing one key set of interactions. On the other hand, during simulation tests, all contacts are in place to accurately reflect real-world, but the environment count is set to a low value, 512 per test dataset. The technique mentioned above to allow for slight differences in training and test environments, along with careful tuning of the Temporal Gauss-Seidel (TGS) contact solver parameters, enables us to run large-scale training required for RL without compromising representational accuracy.

5.4.2 Hardware Design

In this chapter, we aimed to train a policy in simulation, running on an NVIDIA RTX 4090, and transfer it to real without a policy adaptation phase. During inference, the agent interacts with our real Kinova 6-DOF Jaco 2 through the Kinova ROS api and a custom REST interface, running on a separate low-grade workstation at

a 60Hz control rate. Unlike prior works focused on artificial plants with soft leafy resistance, our test branches are randomly chosen from real trees, ensuring sufficient stiffness to resist clearance. A single fixed Intel RealSense D405 camera, operating at 30 fps, provides the RGB-D (480×640) observation of the workspace, illustrated in Fig. 5.2(c). The segmented real branch point clouds are constructed from the RGB-D camera images; in contrast, simulation tree points are randomly sampled from the coarse-grained branch segment surfaces. The real robot point cloud is built once from meshes, just as in the simulation, and updated at each step with the 3D coordinate transformation tree from ROS.

Furthermore, to address the significant sim-to-real discrepancies in robot parameters, contact dynamics, and gravity compensation, exacerbated by velocity control, we leverage Segmented Steady State Error Control (S3EC) from Chapter 4. This scheme alternates between applying an action $a_t \sim \pi_w(\cdot|s_t)$ to the desired state s_t^d, instead of the current, but periodically synchronising s_t^d to s_t, in effect, rejecting short-term steady-state errors and clearing long-term accumulated offsets.

5.4.3 Real Vision Pipeline

This section outlines the key transformation steps required to convert the real-time streaming images into RL features. First, we use Grounding DINO (S. Liu et al., 2023) to locate the objects in the scene with simple text prompts. The resulting bounding boxes are passed to the Segment Anything Model (SAM-HQ) (Ke et al., 2023) model to generate object masks zero-shot. Segmented masks are then de-projected to the robot base frame, leveraging the depth channel input. A key concern at this stage is that manipulators operating in velocity control mode have a control rate lower bound, 60Hz in our case, required to maintain velocity. Current state-of-the-art grounding and segmentation tools are unable to achieve this without significantly compromising the mask quality. Therefore, we use a video object segmentation framework Cutie (Cheng et al., 2024), to propagate the mask across time-steps, occasionally replacing it with updated masks to prevent quality loss while tracking. Moreover, we use Python Shared Memory modules and asynchronous queues to go beyond the camera-constrained fps and meet the control frequency requirement. Notably, such a pipeline is problem agnostic; specifically, the grounding text prompt alone needs to be modified to switch

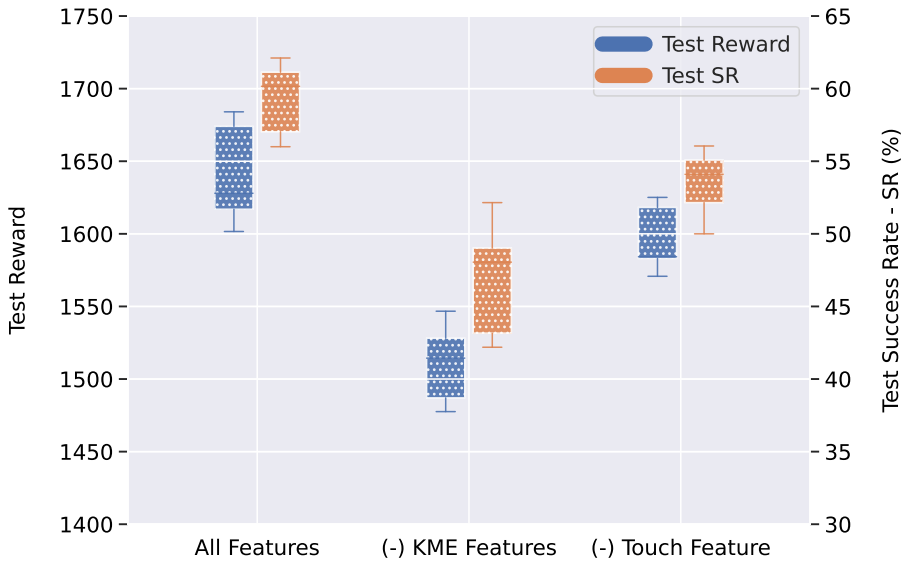


Figure 5.4: Ablation showing the relevance of key feature groups in our multi-modal policy. (-) indicates the removal of the single specified group. Each data point is a trained policy with a varying noise level. Boxes indicate median and IQR.

from the power line context to a cable inspection scenario, a promising direction for future work.

5.5 Experiments and Results

5.5.1 Ablation Study

First, we perform a simulation ablation study to justify our feature choices with the setup from Section 5.4.1. For each combination, we train independent policies by injecting varying levels of Gaussian Noise $\mathcal{N}(0, \sigma^2)$ into the point clouds with the 3σ rule, i.e., $\sigma = d_{\text{max}}/3$, d_{max} being the maximum deviation representative of the camera depth inaccuracy. The ablation results are shown in Fig. 5.4 and Table 5.2. The box plot data points are policies constructed with d_{max} levels ranging from 0.005 to 0.04, displaying the distribution of test rewards and success rates on a single unseen test set. On the other hand, each row in Table 5.2 is an individual policy tested on multiple test sets, listing a variety of additional evaluation metrics, with rows (a)-(e) showing the results applying a median noise $d_{\text{max}} = 0.02$, representative of the ablation study.

By contrast, row (f) shows our best model with a lower $d_{\max} = 0.005$, the depth inaccuracy of the D405 camera used for real experiments.

The tabulated evaluation metrics are described as follows: Training Reward (Train Rew) and Test Reward (Test Rew) assess the agent's performance during training and on unseen test scenarios, respectively. Test Success Rate (Test SR %) quantifies the percentage of test environments (Trials) where all occlusions were removed from the power line. Given that the branch collection will continue to resist clearance even after success is reached, we choose a conservative Test SR, considering a task as a success only if the line is occlusion-free for at least 10 consecutive steps. Occlusion reduction (Occ Drop %) shows the percentage drop in heuristic (h_t) from the start to end of test episodes averaged across all environments, while the (Touch %) reflects the frequency of undesirable instances where the arm breaches the pre-set safety margin δ_{sm} around the power line. The final metric (Steps in Succ) measures the efficiency of successful trajectories; as an example, the final row (f) implies that, on average, in a 1000 step trajectory, the arm starting from the home position, traversed to the occlusion locations and moved the occluding branches away from the line in $1000 - 429.8 = 570.2$ steps, while the remaining 429.8 steps were completely occlusion free for potential pruning, despite continued cluster resistance. All metrics except the power line touch (Touch %) are higher, the better.

The plot in Fig. 5.4 demonstrates that both the KME feature group capturing the global point cloud attributes and the contact detection classifier form essential constituents of our multi-modal policy, and removal of either would hurt performance. Along the same lines, Table 5.2 shows that despite the stochasticity of individual policies and high noise injection, the progressive addition of feature groups induces improvement across all evaluation metrics. To highlight, we report all test success rates without distinguishing between feasible and infeasible tasks, i.e., all occlusion patterns are randomly generated and may contain scenarios that cannot be solved by a single arm. In some unsolvable instances that contribute to the failures, the arm cannot generate sufficient torques (due to joint limits) to overcome the collective resistance of obstructing branches; in others, multiple disconnected branches can occlude the line at locations not closely spaced, only one of which can be removed at a time.

Table 5.2: Ablation study results: Detailed Metrics

	Description	d_{max}	**Train Rew**	**Trials**	**Test Rew**	**Test SR%**	**Occ Drop%**	**Touch%**	**Steps in Succ**
(a)	Proprioceptive + h_t	-	1495.6	1536	1413.6 ± 28.7	36.5 ± 2.3	18.0 ± 1.3	2.8 ± 0.3	237.7 ± 19.2
(b)	(a) + Local PC	0.02	1531.7	1536	1486.5 ± 49.1	43.8 ± 4.5	25.2 ± 3.6	2.1 ± 0.4	258.3 ± 32.1
(c)	(b) + Touch Feature	0.02	1606.6	1536	1506.1 ± 28.5	46.1 ± 2.6	27.9 ± 4.8	2.6 ± 0.8	268.9 ± 17.7
(d)	(b) + KME Features	0.02	1739.1	1536	1630.2 ± 31.6	55.7 ± 1.1	40.1 ± 0.4	1.9 ± 0.1	348.2 ± 22.7
(e)	All Features	0.02	1704.3	1536	1655.9 ± 26.4	63.4 ± 2.8	46.3 ± 3.3	3.8 ± 0.3	363.4 ± 20.6
(f)	**Final Model**	0.005	**1825.1**	**1536**	**1763.6 ± 21.5**	**68.4 ± 2.1**	**53.7 ± 2.2**	**3.3 ± 0.4**	**429.8 ± 25.6**

5.5.2 Representational Efficiency

Second, we evaluate the representational efficiency of our approach to other point cloud feature extractors, namely, a Pytorch variant (Yan, 2019) of the well-known PointNet++ (Qi et al., 2017) and the Point2Vec (Zeid et al., 2023). Specifically, we replace the four global point features from Table 5.1, i.e., $\overline{\mu_{\mathbb{P}}}[P^{wbr}]$, $\overline{\mu_{\mathbb{P}}}[P^{zbr}]$, $\overline{\mu_{\mathbb{P}}}[P^{clr}]$ and $\overline{\mu_{\mathbb{P}}}[P^{rob}]$ with embeddings built from PointNet++ and Point2Vec. In each case, we use pre-trained checkpoints, down-sample the input points, and employ PCA post-feature extraction wherever necessary for consistency with our approach. We train each set for 250 epochs with 1024 environments, and the corresponding time taken (*Train Time*) and the accumulated training reward (*Train Rew*) at the end are tabulated. In addition, we compute $GF_t^{(E)}$ denoting the average time taken to build the 4x global point features only for a single RL time-step for E environments. $GF_t^{(1024)}$ is indicative of the representation efficiency while training on 1024 environments and $GF_t^{(1)}$ represents the inference performance for one environment. The results are tabulated in Table 5.3, where lower is better for the time metrics, such as, $GF_t^{(1024)}$, $GF_t^{(1)}$, and *Train Time* while higher values are better for others.

Table 5.3: Representational Efficiency

	$GF_t^{(1024)}$ (secs)	$GF_t^{(1)}$ (secs)	Train Time (mins)	Train Rew	Max Envs
PointNet++	21.29	0.7706	1487.2	1345.6	1024
Point2Vec	0.204	0.0454	75.2	1379.6	1024
KME	**0.0012**	**0.0012**	**27.5**	**1449.9**	**6144**

Comparing the time taken for a single time-step and for the overall training, from Table 5.3, our kernel-based approach shows orders of magnitude improvement, with similar training rewards. For the PointNet++ and Point2Vec cases, the maximum possible environments (*Max Envs*) we could spin up in our hardware was 1024, an indication of the GPU memory constraint due to the heavy pre-trained checkpoints. Notice that our final KME-based model training, row (f) from Table 5.2, which uses 6x gym environments and 5x iterations compared to this experiment, takes just 4.5 hours to complete on an RTX 4090. Furthermore, the tabulated $GF_t^{(1)}$ values indic-

ate that both PointNet++ and Point2Vec cannot meet the 60Hz real-time inference frequency requirement, justifying our approach. However, we acknowledge that improved efficiency may be at the cost of expressivity and that advanced baseline versions or quantisation techniques can reduce this time difference.

5.5.3 Sim-to-real

Finally, we go beyond simulation validation to evaluate our approach in a real-world setting under laboratory conditions. In real, we test two branches from different species, modifying either the branch orientation or the power line position in each experiment. Furthermore, in each test, we start with an h_t of at least 0.8, and consider both partial occlusion clearance and any touch to the power line as failure instances. The results are tabulated in Table 5.4.

Table 5.4: Real Results

Branches	**Attempts**	**Test SR**
Branch 1	9	44.4%
Branch 2	26	42.3%

Corresponding real videos (see supplement and Fig. 5.3) demonstrate that the simulation-trained policy can adapt zero-shot to arbitrary branch structures with multiple forking patterns despite the presence of point cloud noise and leaf-induced clutter unseen in training. The trajectories exhibit novel characteristics, effectively using the forearm, upper and lower wrists, gripper, and the un-actuated end-effector. In some instances, we observe self-reconfiguring strategies to prop up the occluding branches, while in others, the arm inserts itself between the branches and the line to separate them. Further, these results validate our RL-based approach in discovering efficient strategies and adapting to complex scenarios, well beyond what humans can demonstrate.

5.6 Summary

To the best of our knowledge, this is the first work to propose a general framework for manipulating clusters of deformable objects using the whole arm. We introduced a highly efficient distributional state representation that integrates both point clouds and proprioceptive contact flags. Through comprehensive experiments, we demonstrated that this KME-based formulation is extremely fast to train and enables real-time inference. Additionally, using the proposed context-agnostic occlusion heuristic, we showcased the practical applicability of our approach, specifically in the task of clearing overhanging foliage from power lines. Crucially, similar to Chapter 4, our findings reaffirm that a DRL policy learning approach can autonomously discover novel de-occlusion strategies, and generalise to unseen branch geometry and dynamics, in both simulated and real-world settings.

Whatever has the nature of arising,
has the nature of ceasing.
SIDDHARTHA GAUTAMA

CHAPTER 6

Conclusion

This thesis has presented a brief work at the intersection of reinforcement learning, robotic control, and probabilistic inference towards autonomous manipulation of tree branches and plant stems. We introduced a set of novel solutions to simulate complex branching structures, infer physics parameters, and learn robust control strategies, firmly rooted in machine learning principles. While the remarkable diversity of instincts and capabilities displayed by biological organisms in manipulating natural deformables has provided us with the inspiration, the proposed methodologies hold significant practical relevance in both natural and industrial settings. We hope that the proposed ideas in this thesis will stimulate further inquiry in natural deformable manipulation within the broader robotics community.

6.1 Contribution Summary

A brief summary of the research questions and contributions of this thesis is as follows:

Firstly, in Chapter 3, we posed the question of whether a set of physics parameters can be learned from the real world to simulate the uncertain deformation behaviour of plant stems under physical stress. To answer, we mimicked the plant deformation behaviour with branching structures actuated with spring models. Subsequently, we took a simulation-driven inverse inference approach to predict deformation trajectories that are robust to physical uncertainties. Crucially, our coarse-grained simulation model effectively replicates real branch behaviour, with significantly lower computational cost, bypassing the need for more fine-grained simulation models (e.g., FEM) typically used in the industry.

Secondly, in Chapter 4, we examined the task of simulating realistic plant geometries and acquiring alternative sensing modalities for learning a policy that can be transferred to the real world. We found that a proprioceptive-only DRL policy, trained on L-system based branching structures, can manipulate plant stems while exhibiting strong robustness properties. We were also pleasantly surprised by the creative manipulation and self-correction strategies displayed by the RL agent, despite the lack of vision and the absence of expensive torque sensors. Regardless of the novelty of this proprioceptive-only model, we identified that a multi-modal approach combining both vision and touch may be essential for more complex manipulation tasks, leading to our next chapter.

Thirdly, in Chapter 5, we explored strategies for representation; specifically, the ways to transform and encode the state of natural deformables to accelerate the training process without losing the expressivity. We proposed a multi-modal (combining vision and proprioceptive contact features), non-prehensile, model-free DRL framework that operates on deformables as a collection with whole-arm contact. We demonstrated the practical viability of our method in a novel context to clear power lines of overhanging foliage. Furthermore, we discovered that a simple, context-agnostic, occlusion heuristic, based on nearest-neighbour point distances between branches and a virtual cylinder, enables us to extend the proposed approach to other applications, for instance, to clear line-of-sight for an inspection camera in an agricultural setting. Finally, our findings once again demonstrated that a tabula rasa policy learning approach can autonomously

discover novel de-occlusion strategies, generalising to unseen branch geometry and dynamics.

For each question examined above, we proposed additional pathways to improve the learning efficiency and reduce overall operational cost. Finally, throughout this work, we presented detailed empirical analysis, ablation results, and real-world experiments to demonstrate the efficacy of the proposed methodologies.

6.2 Limitations & Future Work

In this section, we acknowledge the key limitations of each core chapter and outline directions for future research.

6.2.1 Learning to Simulate Tree-Branch Dynamics

The axial disparities in the deformation characteristics implies that the inferred dynamical properties are local to the grasp region and the orientation of the applied force (the gravity axis); therefore, additional modelling may be necessary to generalise the deformation behaviour to other regions or across branches. However, given the relative symmetry in dynamics, i.e, branches of the same species with similar visual appearance are likely to exhibit similar deformation, refining the methodology from Chapter 3 in this direction is not infeasible. Another direction of future work is to build the spring model directly from a perception acquired branch topology, for instances using Lowe and Pinskier (2022). Furthermore, severe discrepancies between model estimates and true deformation behaviour could be used as an indicator of poor tree health or branch rupture. Crucially, a generalised version of the model could be used in tandem with branch manipulation policies, in an online setting. Overall, we foresee works that entail combining our uncertainty aware model with visual tree reconstruction works, generalising it to different branches, and eventually generating robust control policies for manipulation.

6.2.2 Gentle Branch Manipulation with Contact-Awareness

Limitations to this method chiefly include the reliance on internal sensors for contact-awareness. We acknowledge that the lack of camera vision feedback could restrict

performance in uncertain environments, but our key objective in Chapter 4 was to push the boundaries of a proprioceptive-only approach. Future work should explore integrating accurate external whole body tactile sensors (Huang et al., 2024), that are poised to become cost-effective in the near future. Furthermore, future work must examine improving the fidelity of simulated environments; for instance, with multi-axis deformable branches and simulated force sensors, to enhance the transferability of contact-aware policies to diverse real applications.

6.2.3 Deformable Cluster Manipulation with Whole-Arm

Further refinements to the framework presented in Chapter 5 are necessary to reduce the sim-to-real gap and improve real-world success rates. First, the perception quality may be enhanced by combining multiple camera images or, alternatively, using partial-to-complete visual reconstruction models. Second, real-world complexities may be introduced in simulation for an improved sim-to-real transfer; for instance, by combining visual tree reconstruction works (Lowe & Pinskier, 2022) that identify the stem structure beneath the leaf clutter, or the parameter inference methods proposed in Chapter 3. Third, instead of the proprioceptive contact-awareness model, external whole-arm tactile sensors may be leveraged. Finally, this approach can be readily adapted to industrial applications involving synthetic deformable objects, such as cables, ropes, or sutures, with minimal modifications.

APPENDIX A

Supplementary material for Chapter 3

A.1 Supplementary Website

Further visual results (including images and demonstration videos) can be accessed at the following URL:

- https://sites.google.com/view/lstbdm/home


Figure A.1: **(Left):** Monkey on a coconut tree, about to leap (Bhaskaranaidu, 2007). **(Right):** Indigenous people picking vegetables (Nhoch, 2022).

A.2 Inferred Dynamics in Biology

The forces of natural selection have provided biological organisms with an intuitive grasp of the real-world dynamics to seamlessly predict and interact with the physical objects. For instance, primates instinctively utilise branch momentum to traverse among trees; an illustrative image is provided in Fig. A.1: Left, captured just before such a leap. A successful leap, and possibly the survival of the individual, depends on its precise positioning on the branch. If the primate positions itself too far from the trunk (towards the right of the image), it can slip off or the branch can snap from the body weight; conversely, if the position is too close to the trunk (towards the left of the image), there is not sufficient momentum to pull off a jump. In other words, through real-time sensory feedback, and possibly encoded genetic information, the organism has an innate awareness of the optimal position, the branch deformation characteristics at that location, and the specific amount of force required for a successful leap. Similarly, we humans instinctively apply this innate understanding of physical dynamics, of grasp positioning and force exertion, when performing delicate tasks, such as harvesting vegetables without damaging the plant, see Fig. A.1: Right.

A.3 Additional Experiment Details

The hyper-parameters, experimental settings, and the performance metrics used in the sim-to-sim and real-to-sim experiments from Chapter 3 are noted in Table A.1 and Table A.2 respectively.

Type	Baseline (R=3)	svgd vs nnsvgd (R=3)	Robustness (R=2)
Data Attributes			
Training trajectory count	5	5	5
Test trajectory count	25	25	25
Data points per trajectory	120	120	120
θ Dimensions	6	6	4
K_p, K_d Search range	[5–200]	[5–200]	[5–200]
Hyper Parameters			
kT	1/100	1/100	1/40
SVGD Step size ϵ	3.0	3.0	3.0
Finite Difference Spacing	0.5	0.5	0.5
RBF σ	σ_m	σ_m	σ_m
Performance			
SVGD Iterations to convergence	70	70	90
Time to convergence (min)	3.5	3.5	3.1

Table A.1: Sim-to-Sim experimental settings and performance metrics.

Note the following points in regards to the tables.

1. **Convergence Time**: For the baseline set, the convergence time is provided for NNSVGD only. In the robustness set, where multiple NNSVGD experiments are conducted with varying noise and grasp locations, the convergence time is the overall average.

2. **Cross-Validation**: The metrics provided are per cross-validation cycle.

3. **Performance Metrics**: All performance metrics should be considered indicative only. This is because they depend on all listed attributes, in addition to trajectory noise, branch type, simulation environment, GPU state, etc.

4. **Hyper-parameters**: All hyper-parameters except kT are the same in every scenario. The kT hyper-parameter scales the sharpness of the likelihood in

Type	Kinova (R=2)	Kinova (R=1)	Franka (R=1)
Data Attributes			
Training trajectory count	3 (cv)	3 (cv)	3 (cv)
Test trajectory count	3 (cv)	3 (cv)	3 (cv)
Data points per trajectory	100	100	120
θ Dimensions	4	2	2
K_p, K_d Search range	[5–200]	[5–200]	[5–200]
Hyper Parameters			
kT	1/40	1/20	1/20
SVGD Step size ϵ	3.0	3.0	3.0
Finite Difference Spacing	0.5	0.5	0.5
RBF σ	σ_m	σ_m	σ_m
Performance			
SVGD Iterations to convergence	60	60	200
Time to convergence (min)	1.4	0.5	1.7

Table A.2: Real-to-Sim experimental settings and performance metrics.

proportion to the loss $L(\theta)$. Therefore, kT is tuned per number of dimensions, d, to reduce long convergence times.

5. **Simulation Configuration**: All unlisted simulation and optimisation configurations are set to be the same in every scenario.

6. **Kernel Bandwidth**: The kernel bandwidth, σ, is computed using the median heuristic, denoted by σ_m.

APPENDIX B

Supplementary material for Chapter 4

B.1 Supplementary Website

Further visual results (including images and demonstration videos) can be accessed at the following URL:

- https://sites.google.com/view/pcap/home

B.2 Observation/Reward Table

The observation and reward combinations for each policy type are presented in Table B.1, where $\theta_t, \dot{\theta}_t, q_t, \& \, d_t$ represent the joint angle (6D), joint velocity (6D), the gripper orientation (4D), and the distance of gripper from target (3D). The reward parameters r_d, r_s, r_b, r_c, r_p and the scaling factors g_d, g_s are described in section (Section 4.3.5)

Policy	$\boldsymbol{o_t}$	$\boldsymbol{r_t}$
PPO	$\theta_t, \dot{\theta}_t, q_t, d_t$	$g_d \cdot (r_d + r_b) + g_s \cdot r_s$
CPO	$\theta_t, \dot{\theta}_t, q_t, d_t$	$\{g_d \cdot (r_d + r_b) + g_s \cdot r_s\} \cdot r_c + r_p$
PCAP	$\theta_t, \dot{\theta}_t, q_t, d_t, \mathbb{I}(\lVert F_t \rVert_2 > f_u)$	$\{g_d \cdot (r_d + r_b) + g_s \cdot r_s\} \cdot r_c + r_p$

Table B.1: Training observations & rewards provided for different policies.

B.3 Experimental Setup

For all simulation experiments, we generate 60 random targets within the reach radius of the arm but constrained to the direction of the tree. Collision patterns, numbered 1 to 4 in Fig. 4.4: top, are constructed either by reorienting a test tree or changing the tree itself to expose the agent to varying occlusion patterns. In any case, the test trees are unseen during the training phase. We define success as the arm's end-effector achieving a position within 5cms of the desired target, regardless of the orientation, but subject to a maximum step limit. The SR is calculated as follows: First, we create an ensemble oracle with 2x PPO, 1x CPO, 1x PCAP, 1x MPC to find the feasible tasks within the arm torque, velocity limits based on the intuition that a task is solvable if any algorithm finds a feasible reach trajectory. Then, we baseline the depicted methods against the oracle with the ideal 100% success. However, for the Contact Force metrics $\lVert F_t \rVert_2$, we average over all the 60 tasks because the branches can rupture in the real world regardless of the reach success. We assume reach targets are known, as fruits and diseased regions generally have distinct and consistent visual features (colour, texture, shape) that differentiate them from the green of branches and foliage. Even when partially occluded, the conspicuous parts may still provide enough contrast for vision algorithms to predict the rest based on learned shapes, which is not the case with branches.

The real experimental setup and the comparison metrics are similar to simulation experiments, with a few notable differences. First, we only capture 30 test trajectories in real. Second, we hard stop the arm if the branches are likely to rupture to keep the geometry similar between PPO and PCAP. The rupture % values (in red) include both cases where the physical branches broke apart and where the arm was stopped in advance. On the other hand, in simulation, rupture % is computed as the proportion of steps over the trajectory length when the contact forces on the branch exceed a threshold (e.g., 80N) as a proxy metric. Here, we allowed the arm to time out regardless of reach failure or extreme forces. Third, for real experiments, due to the limited size of our tests and the sparsity of real branches, we disregard collision-free trajectories that avoid contact between the arm and the tree. In other words, we use only the targets where both PPO and PCAP has at least some interaction with the hindering tree. Finally, we do not use an oracle for real; the SR is simply the proportion of successful to attempted targets.

B.4 Time Efficiency

In every experiment throughout this work, we limit the number of steps (simulation: 800, real: 1000) the agent can take. The simulation step limit (800) was chosen by manually tuning to find the optimal trade-off between the success rate and training time, while a larger step limit (1000) is necessary in real to account for real-world uncertainties and to provide the robot additional time to recover from minor errors. The experiment in this section compares the normalised count of minimum steps the agent takes to reach the target, which we take as a proxy for the clock-time efficiency across all methods. The worst-case normalised reach step count of 1.0 indicates a timeout failure while the ideal 0.0 represents a success at the first time step. To exemplify, the final block from the results in Fig. B.1 indicates that for the fifth Collision Pattern, on average across 60 tasks, PCAP takes 670 steps to reach the target, while baseline PPO takes 550 steps if the allocated step budget is 1000.

The results demonstrate that, on average, PCAP takes 25% more time steps across collision patterns to reach the target compared to the baseline PPO. We emphasise that this slight loss of efficiency in PCAP is the trade-off for increased exploration of the task space and additional learnt manoeuvre such as rolling and retractions (rather

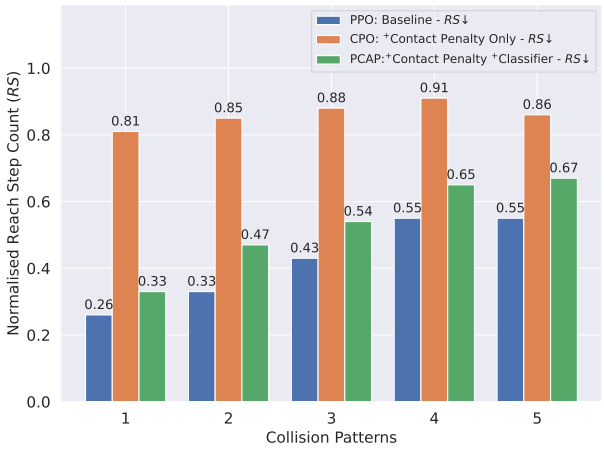


Figure B.1: Comparison of the number of RL steps taken to reach the target (or timeout) across three policies: PPO, CPO, & PCAP (ours). For RS, lower values are better, indicating lower time taken. The slight uptick in time consumption for PCAP reflects a cautious, gentle, and safer branch exploration.

than brute forcing through), resulting in gentler branch contacts. We also note that the intermediate CPO method is the worst performer primarily due to the significant number of failed tasks from timeouts.

B.5 Geometry Ablation

This section demonstrates the benefits of the novel L-System-based forest generator compared to the simpler branching structure proposed in Chapter 3. While it is well known that L-system generated topology can generate realistic branching patterns compared to naive models (sample images presented in Fig. B.2)., we show empirical evidence that the richer occlusion patterns are beneficial to robotic applications as well.

In this experiment, we train policies on two independent forests, one generated with the basic design in Chapter 3 and the other with our L-system models. In the former, we randomise branch lengths, radius, and divergence angles while ensuring a similar span for the part of the tree the robot has access to, whereas for the latter, we vary the L-system parameters as described in Section 4.3.1. The results in Fig. B.3 show that, on average, the success rate, contact forces, and branch rupture metrics are all superior for the L-system geometry. We hypothesise that this is because the

recursive fractal-based models allow for variations in geometric parameters, resulting in organic, asymmetrical forms that better capture the irregularities found in nature, whereas simple contrived structures, even with noise perturbations, lack this diversity limiting its policy robustness.

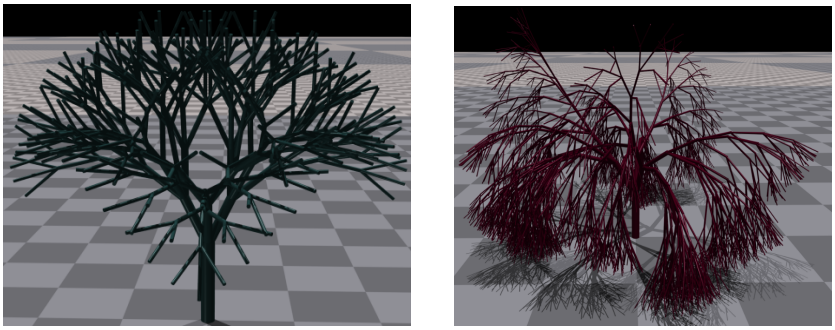


Figure B.2: **(Left):** A simplistic tree procedurally generated from Chapter 3 **(Right):** An L-System generated tree from the ternary classes Tc, based on our method. Both simulations are presented without any randomisation, to highlight the intrinsic advantage of L-system to generate diverse occlusion patterns, asymmetric arching angles and non-uniform parent-child splits across the topology. These differences in geometry explain why policies trained on L-system trees perform better (higher success rate and fewer ruptures) in Fig. B.3.

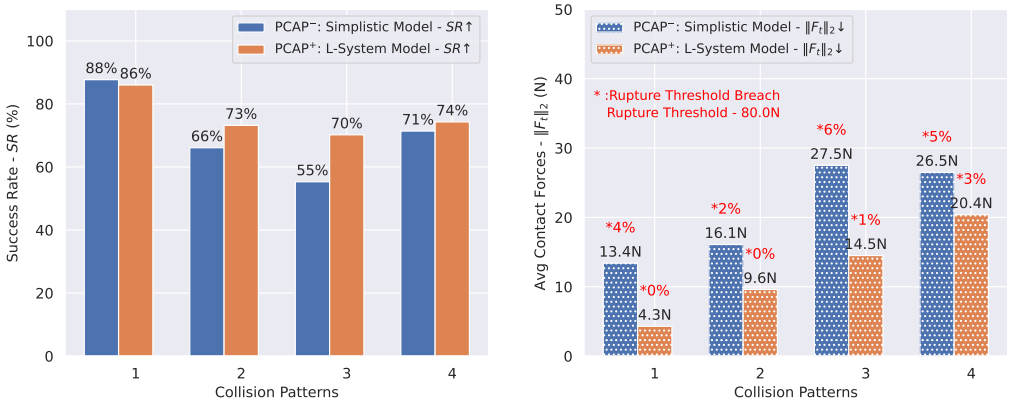


Figure B.3: Ablation study of the geometric models comparing the simplistic branching model from Chapter 3 and the L-system model (ours), keeping the Dynamics fixed. The comparison is performed by domain randomising both base models. For SR, higher values are better, for $\|F_t\|_2$, the lower the better.

B.6 Explicit Classifier vs Raw Joint Torques

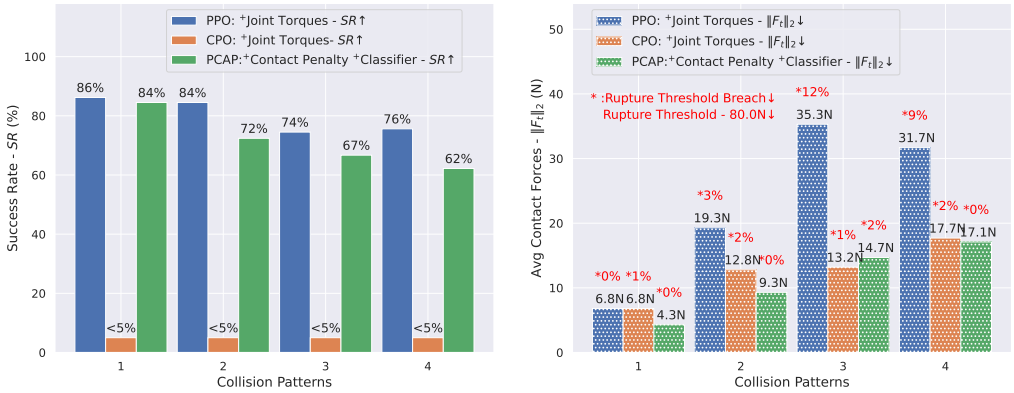


Figure B.4: Ablation results comparing PCAP with the explicit classifier to PPO$^+$ and CPO$^+$ with τ_t included as part of o_t.

As illustrated in Table B.1, PCAP uses joint position and velocity measurements along with the classifier output as part of the observations o_t. We justify excluding raw joint torques τ_t with the following arguments: First, real Kinova joint torque measurements and simulation values differ considerably owing to heteroscedastic noise (Fig. B.5) and inertial parameter discrepancies, posing substantial challenges for zero-shot sim-to-real transfer. The noise is primarily due to indirect torque inference rather than dedicated external sensors mentioned in Chapter 3. Second, we perform an additional ablation study (Fig. B.4), to demonstrate that even with perfect knowledge of the raw torques (in simulation), PCAP has a clear edge over PPO for contact forces ($\approx$ 2x) and ruptures ($\approx$ 5x) while the success rates are comparable. The results also show that CPO gains ground on the contact force/rupture metrics on adding τ_t to o_t, but sacrificing SR. We posit that the temporal dependency features of the contact detection classifier are the primary determinant of PCAP performance, which the RL agent cannot access in other instances. However, we acknowledge that swapping the MLP network of PPO for a component like Recurrent Neural Network (RNN) may result in performance on par with PCAP. Furthermore, the classifier also uses important non-torque features, such as commanded (due to velocity control mode) and executed joint velocities. It is conceivable that, even without any torque signals, the controller's inability to execute the commanded joint velocities could be an indication to the classifier of an obstacle on the way. Third, using an explicit classifier allows the

operators to incorporate expert knowledge apriori into the policy through the dataset collection and the feature selection process; for instance, ensuring contact with all links and in all directions. Finally, this modular architecture provides the flexibility to train and test the contact detector robustness independent of policy learning.

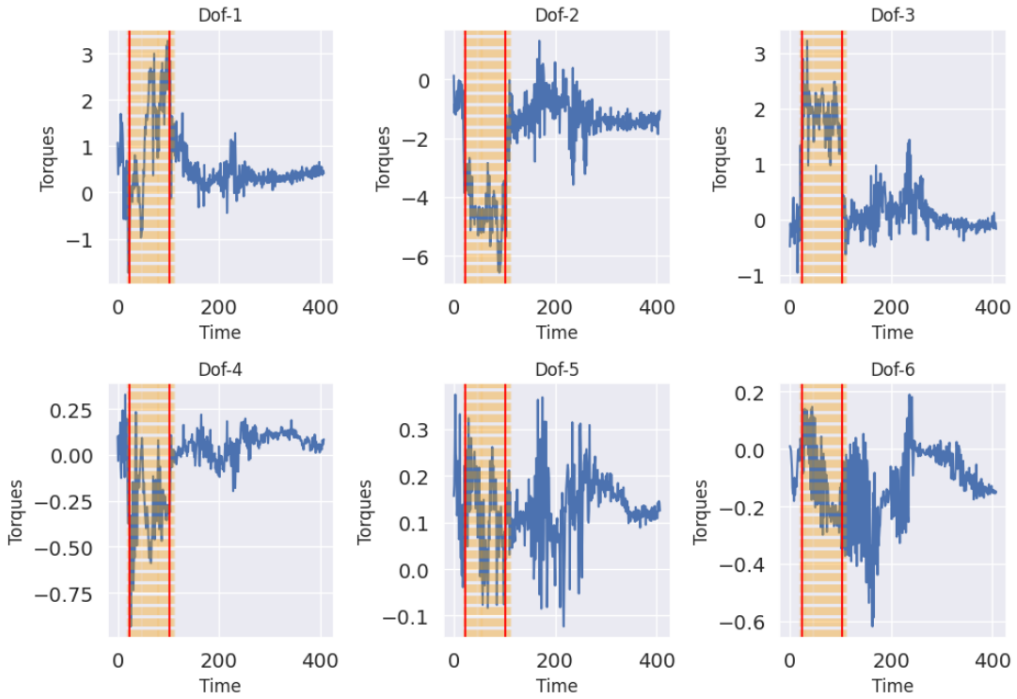


Figure B.5: A sample prediction given by the classifier. The input in blue is the time series noisy joint torques of the six dofs. Red solid lines represent the ground truth, i.e., the start and end of the obstruction. Yellow dashed lines represent the predicted contact at each time-step.

B.7 Dynamics Ablation

In our final ablation study, we keep the L-system geometry, generated with class Ta, fixed and vary the dynamics between the two models described in section Section 4.3.1. From Fig. B.6, the superior performance of the Rudimentary model compared to Beam deflection isn't surprising, given that in the former, ϕ is approximately the same for a specific branch level between training and test, i.e., except for the additional parameter

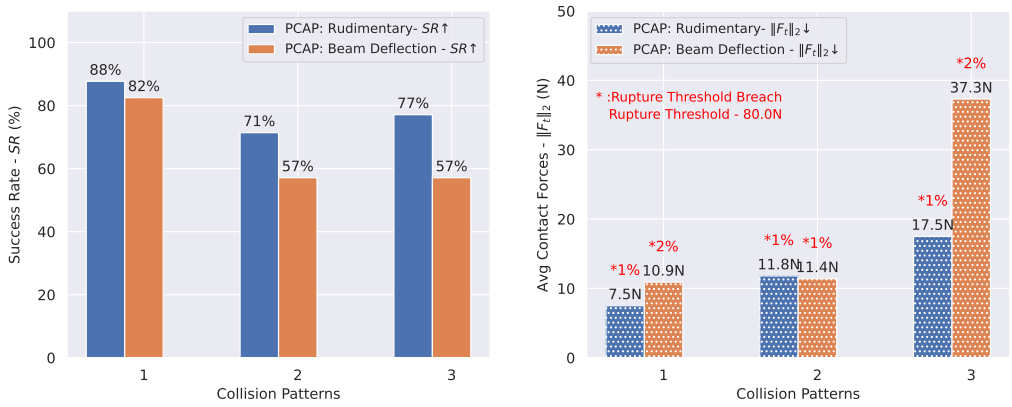


Figure B.6: Ablation on the tree dynamics models, keeping the L-system geometry fixed. Comparison of Success Rates (SR) and Average Contact Forces across two dynamics models: Rudimentary and Beam Deflection. For SR, higher values are better, for $\|F_t\|_2$, the lower the better.

perturbations while training. This result suggests that integrating system identification for dynamic parameters from Chapter 3 can significantly boost policy learning.

B.8 Classifier Metrics

Metrics for our contact detection classifier with two off-the-shelf classifiers are given in Fig. B.7: a 3-Layer Neural Network and a Random Decision Forest. For the real PCAP experiments, we use the latter mainly due to the low false positives.

B.9 L-System Perturbation

We domain randomise L-system morphological parameters Σ by injecting Gaussian Noise while recursing through the production rules. From Fig. B.8, we observe that low-level perturbations ($\sigma = 0.01$) create structures with limited diversity in topology, identical branches, and similar spans. By contrast, high noise levels ($\sigma = 0.25$) generate infeasible spans with few prospects for robot engagement. We settle for a Goldilocks level ($\sigma = 0.1$) with sufficient feasibility but high diversity. Notice the branch radius variations for the corresponding middle row, starting from the trunk. Additionally, we create thousands of tree structures to weed out intractable ones with low contact,

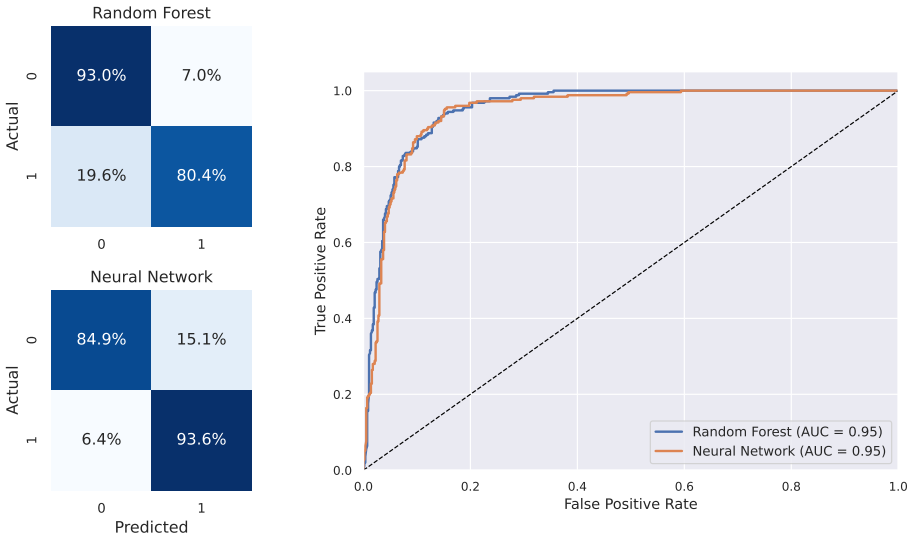


Figure B.7: Confusion matrix & ROC for the two contact detection classifiers.

move the trees along the gravity axis, and rotate them to maximise occlusions and arm interactions.

B.10 Hyper Parameters

Param	Parameter Description	Value
E	Young's modulus	3e9
g_d	Distance reward scaling factor	2.0
g_s	Smoothness reward scaling factor	0.01
f_u	Simulation force threshold	40N
	Simulation rupture threshold	80N
	Robot friction	[0.01,0.01,0.01,0.01,0.01,0.01]
	Robot damping	[80.,80.,80.,80.,80.,80.]

Table B.2: Additional hyper parameter and configuration settings.

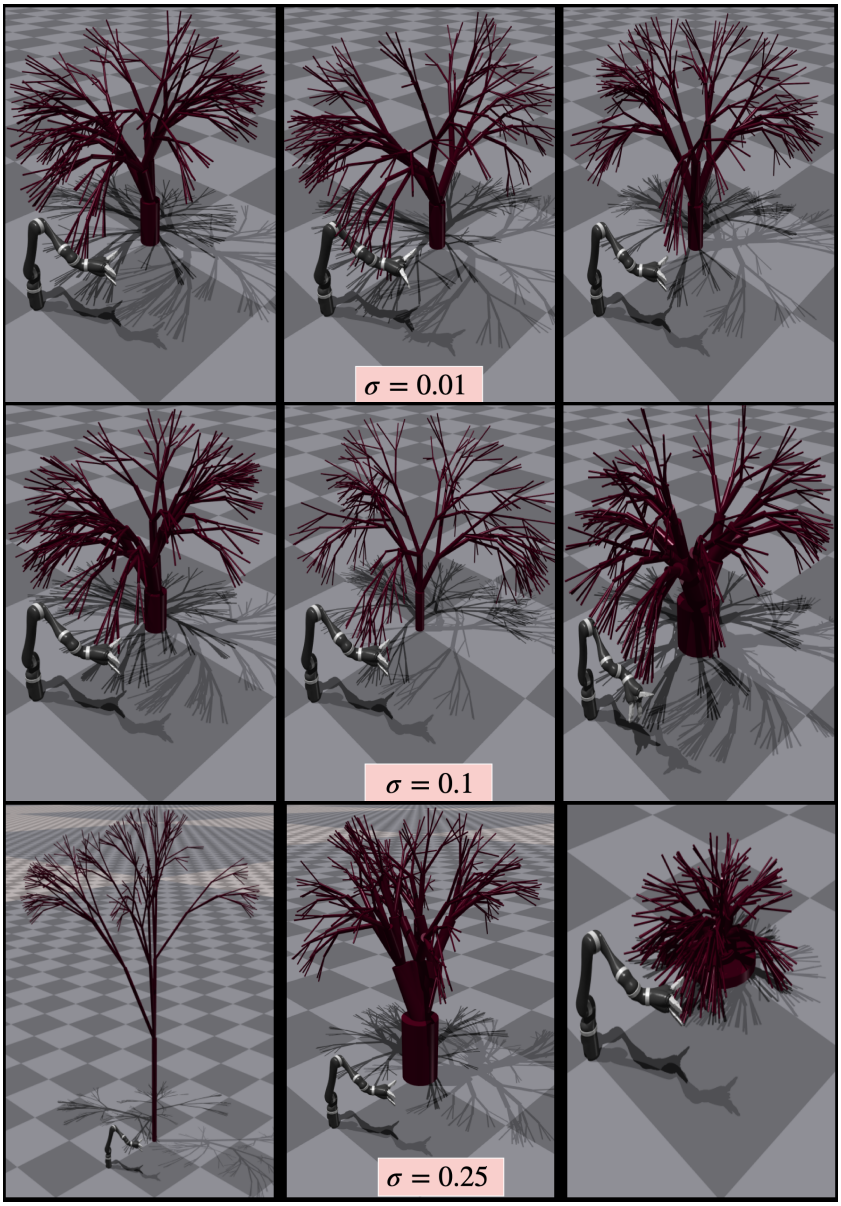


Figure B.8: Effects of perturbations on topology: Samples generated by varying noise levels injected in L-System parameters. **(Top Row)**: too low ($\sigma = 0.01$) , **(Middle Row)**: our choice ($\sigma = 0.1$), **(Bottom Row)**: too high ($\sigma = 0.25$)

APPENDIX C

Supplementary material for Chapter 5

C.1 Supplementary Website

Further visual results (including images and demonstration videos) can be accessed at the following URL:

- https://sites.google.com/view/dcmwap/home

Bibliography

Adhikary, S., & Boots, B. (2022). Sampling over riemannian manifolds using kernel herding. *2022 International Conference on Robotics and Automation (ICRA)*, 3646–3653.

Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., et al. (2019). Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*.

Andrews, S., Erleben, K., & Ferguson, Z. (2022). Contact and friction simulation for computer graphics. In *Acm siggraph 2022 courses* (pp. 1–172).

Andrychowicz, O. M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., et al. (2020). Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, *39*(1), 3–20.

Antonova, R., Yang, J., Sundaresan, P., Fox, D., Ramos, F., & Bohg, J. (2022). A bayesian treatment of real-to-sim for deformable object manipulation. *IEEE Robotics and Automation Letters*, *7*(3), 5819–5826.

Arriola-Rios, V. E., Guler, P., Ficuciello, F., Kragic, D., Siciliano, B., & Wyatt, J. L. (2020). Modeling of deformable objects for robotic manipulation: A tutorial and review. *Frontiers in Robotics and AI*, *7*, 82.

Bai, F., Li, Y., Chu, J., Chou, T., Zhu, R., Wen, Y., Yang, Y., & Chen, Y. (2025). Retrieval dexterity: Efficient object retrieval in clutters with dexterous hand. *arXiv preprint arXiv:2502.18423*.

Bandyopadhyay, T., Talbot, F., Bennie, C., Senaratne, H., Li, X., Tidd, B., Xi, M., Stiefel, J., Dedeoglu, V., Taylor, R., et al. (2024). Demonstrating event-triggered investigation and sample collection for human scientists using field robots and large foundation models. *Robotics: Science and Systems*.

Barto, A. G. (2021). Reinforcement learning: An introduction. by richard's sutton. *SIAM Rev*, *6*(2), 423.

Bhardwaj, M., Sundaralingam, B., Mousavian, A., Ratliff, N. D., Fox, D., Ramos, F., & Boots, B. (2022). Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation. *Conference on Robot Learning*, 750–759.

Bhaskaranaidu. (2007). *Monkey on the coconut tree* [CC BY-SA 3.0, via Wikimedia Commons]. Retrieved August 12, 2025, from https://commons.wikimedia.org/wiki/File:Monkey_on_the_coconut_tree.JPG

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Blei, D. M., Kucukelbir, A., & McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American statistical Association*, *112*(518), 859–877.

Bozic, A., Zollhofer, M., Theobalt, C., & Nießner, M. (2020). Deepdeform: Learning non-rigid rgb-d reconstruction with semi-supervised data. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7002–7012.

Brunke, L., Greeff, M., Hall, A. W., Yuan, Z., Zhou, S., Panerati, J., & Schoellig, A. P. (2022). Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, *5*(1), 411–444.

Chebotar, Y., Handa, A., Makoviychuk, V., Macklin, M., Issac, J., Ratliff, N., & Fox, D. (2019). Closing the sim-to-real loop: Adapting simulation randomization with real world experience. *2019 International Conference on Robotics and Automation (ICRA)*, 8973–8979.

Cheng, H. K., Oh, S. W., Price, B., Lee, J.-Y., & Schwing, A. (2024). Putting the object back into video object segmentation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3151–3161.

Coates, M., & Clack, J. A. (1990). Polydactyly in the earliest known tetrapod limbs. *Nature*, *347*(6288), 66–69.

Couceiro, M. S., Portugal, D., Ferreira, J. F., & Rocha, R. P. (2019). Semfire: Towards a new generation of forestry maintenance multi-robot systems. *2019 IEEE/SICE International Symposium on System Integration (SII)*, 270–276.

Cranmer, K., Brehmer, J., & Louppe, G. (2020). The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, *117*(48), 30055–30062.

Da, L., Turnau, J., Kutralingam, T. P., Velasquez, A., Shakarian, P., & Wei, H. (2025). A survey of sim-to-real methods in rl: Progress, prospects and challenges with foundation models. *arXiv preprint arXiv:2502.13187*.

De Langre, E. (2008). Effects of wind on plants. *Annu. Rev. Fluid Mech.*, *40*, 141–168.

De Oliveira, L. B. (2023). *Probabilistic inference for model based control* [Doctoral dissertation].

Dehban, A., Zhang, S., Cauli, N., Jamone, L., & Santos-Victor, J. (2022). Learning deep features for robotic inference from physical interactions. *IEEE Transactions on Cognitive and Developmental Systems*, *15*(3), 985–999.

Deng, Y., & Hsu, D. (2024). General-purpose clothes manipulation with semantic keypoints. *arXiv preprint arXiv:2408.08160*.

Ferguson, Z., Li, M., Schneider, T., Gil-Ureta, F., Langlois, T., Jiang, C., Zorin, D., Kaufman, D. M., & Panozzo, D. (2021). Intersection-free rigid body dynamics. *ACM Transactions on Graphics*, *40*(4).

Ferreira, J. F., Portugal, D., Andrada, M. E., Machado, P., Rocha, R. P., & Peixoto, P. (2023). Sensing and artificial perception for robots in precision forestry: A survey. *Robotics*, *12*(5), 139.

Frank, B., Stachniss, C., Schmedding, R., Teschner, M., & Burgard, W. (2014). Learning object deformation models for robot motion planning. *Robotics and Autonomous Systems*, *62*(8), 1153–1174.

Friedlander, C. (1964). Thigmokinesis in woodlice. *Animal Behaviour*, *12*(1), 164–174.

Fu, M., Kuntz, A., Webster, R. J., & Alterovitz, R. (2018). Safe motion planning for steerable needles using cost maps automatically extracted from pulmonary images. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4942–4949.

Furmston, T., & Barber, D. (2010). Variational methods for reinforcement learning. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 241–248.

Ganguly, A., & Earp, S. W. (2021). An introduction to variational inference. *arXiv preprint arXiv:2108.13083*.

Gardner, J., Pleiss, G., Weinberger, K. Q., Bindel, D., & Wilson, A. G. (2018). Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. *Advances in neural information processing systems*, *31*, 7587–7597.

Ghojogh, B., Ghodsi, A., Karray, F., & Crowley, M. (2021). Reproducing kernel hilbert space, mercer's theorem, eigenfunctions, nystr\" om method, and use of kernels in machine learning: Tutorial and survey. *arXiv preprint arXiv:2106.08443*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, *63*(11), 139–144.

Grannen, J., Sundaresan, P., Thananjeyan, B., Ichnowski, J., Balakrishna, A., Hwang, M., Viswanath, V., Laskey, M., Gonzalez, J. E., & Goldberg, K. (2020). Untangling dense knots by learning task-relevant keypoints. *arXiv preprint arXiv:2011.04999*.

Gromek, P., & Lowe, T. (2025). Ground robot technologies in wildfire risk reduction. the viewpoint of the fire service. *Progress in Disaster Science*, 100435.

Guggenmoos, S. (2003). Effects of tree mortality on power line security. *Arboriculture & Urban Forestry (AUF)*, *29*(4), 181–196.

Handa, A., Allshire, A., Makoviychuk, V., Petrenko, A., Singh, R., Liu, J., Makoviichuk, D., Van Wyk, K., Zhurkevich, A., Sundaralingam, B., et al. (2023). Dextreme: Transfer of agile in-hand manipulation from simulation to reality. *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 5977–5984.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Heiden, E., Denniston, C. E., Millard, D., Ramos, F., & Sukhatme, G. S. (2022). Probabilistic inference of simulation parameters via parallel differentiable simulation. *2022 International Conference on Robotics and Automation (ICRA)*, 3638–3645.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. (2012). Deep neural networks

for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, *29*(6), 82–97.

Honda, H. (1971). Description of the form of trees by the parameters of the tree-like body: Effects of the branching angle and the branch length on the shape of the tree-like body. *Journal of theoretical biology*, *31*(2), 331–338.

Huang, B., Wang, Y., Yang, X., Luo, Y., & Li, Y. (2024). 3d-vitac: Learning fine-grained manipulation with visuo-tactile sensing. *arXiv preprint arXiv:2410.24091*.

Hudson, N., Talbot, F., Cox, M., Williams, J., Hines, T., Pitt, A., Wood, B., Frousheger, D., Surdo, K. L., Molnar, T., et al. (2022). Heterogeneous ground and air platforms, homogeneous sensing: Team csiro data61's approach to the darpa subterranean challenge. *Field Robotics*, *2*, 595–636.

Hwangbo, J., Lee, J., Dosovitskiy, A., Bellicoso, D., Tsounis, V., Koltun, V., & Hutter, M. (2019). Learning agile and dynamic motor skills for legged robots. *Science Robotics*, *4*(26), eaau5872.

Jacob, J., Cai, S., Borges, P. V. K., Bandyopadhyay, T., & Ramos, F. (2025, June). Gentle manipulation of tree branches: A contact-aware policy learning approach. In P. Agrawal, O. Kroemer & W. Burgard (Eds.), *Proceedings of the 8th conference on robot learning* (pp. 631–648, Vol. 270). PMLR. https://proceedings.mlr.press/v270/jacob25a.html

Jacob, J., Bandyopadhyay, T., Williams, J., Borges, P., & Ramos, F. (2024). Learning to simulate tree-branch dynamics for manipulation. *IEEE Robotics and Automation Letters*, *9*(2), 1748–1755.

Jacob, J., Zhang, W., Warren, H., Borges, P., Bandyopadhyay, T., & Ramos, F. (2026). Deformable cluster manipulation via whole-arm policy learning. *IEEE Robotics and Automation Letters*, *11*(3), 2951–2958.

Jain, A., Killpack, M. D., Edsinger, A., & Kemp, C. C. (2013). Reaching in clutter with whole-arm tactile sensing. *The International Journal of Robotics Research*, *32*(4), 458–482.

James, K. R., et al. (2014). A study of branch dynamics on an open-grown tree. *Arboriculture & Urban Forestry*, *40*(3), 125–34.

Jangir, R., Alenya, G., & Torras, C. (2020). Dynamic cloth manipulation with deep reinforcement learning. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 4630–4636.

Jensen, D. (2025). *Nsa souda bay gives back to the local community* [U.S. Navy photo by Delaney Jensen. Public domain, via Wikimedia Commons]. Retrieved July 8, 2025, from https://commons.wikimedia.org/wiki/File:NSA_Souda_Bay_gives_back_to_the_local_community_(8865256).jpg

Jiang, C., Xu, W., Li, Y., Yu, Z., Wang, L., Hu, X., Xie, Z., Liu, Q., Yang, B., Wang, X., et al. (2024). Capturing forceful interaction with deformable objects using a deep learning-powered stretchable tactile array. *Nature Communications*, *15*(1), 9513.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems.

Kapushev, Y., Kishkun, A., Ferrer, G., & Burnaev, E. (2021). Random fourier features based slam. *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 6597–6602.

Ke, L., Ye, M., Danelljan, M., Tai, Y.-W., Tang, C.-K., Yu, F., et al. (2023). Segment anything in high quality. *Advances in Neural Information Processing Systems*, *36*, 29914–29934.

Khatib, O., Yeh, X., Brantner, G., Soe, B., Kim, B., Ganguly, S., Stuart, H., Wang, S., Cutkosky, M., Edsinger, A., et al. (2016). Ocean one: A robotic avatar for oceanic discovery. *IEEE Robotics & Automation Magazine*, *23*(4), 20–29.

Kim, C. H., Lee, M., Kroemer, O., & Kantor, G. (2023). Towards robotic tree manipulation: Leveraging graph representations. *arXiv preprint arXiv:2311.07479*.

Kim, C. H., Lee, M., Kroemer, O., & Kantor, G. (2024). Towards robotic tree manipulation: Leveraging graph representations. *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 11884–11890.

Kim, K.-E., & Park, H. S. (2018). Imitation learning via kernel mean embedding. *Proceedings of the AAAI conference on artificial intelligence*, *32*(1).

Kimeldorf, G., & Wahba, G. (1971). Some results on tchebycheffian spline functions. *Journal of mathematical analysis and applications*, *33*(1), 82–95.

Kingma, D. P., Welling, M., et al. (2013). Auto-encoding variational bayes.

Kirkpatrick, S., Gelatt Jr, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, *220*(4598), 671–680.

Kober, J., Bagnell, J. A., & Peters, J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, *32*(11), 1238–1274.

Koenig, N., & Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566)*, *3*, 2149–2154.

Kozlovsky, S., Newman, E., & Zacksenhouse, M. (2022). Reinforcement learning of impedance policies for peg-in-hole tasks: Role of asymmetric matrices. *IEEE Robotics and Automation Letters*, *7*(4), 10898–10905.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*.

Kroemer, O., Niekum, S., & Konidaris, G. (2021). A review of robot learning for manipulation: Challenges, representations, and algorithms. *Journal of machine learning research*, *22*(30), 1–82.

Kuo, P.-H., Yang, W.-C., Hsu, P.-W., & Chen, K.-L. (2023). Intelligent proximal-policy-optimization-based decision-making system for humanoid robots. *Advanced Engineering Informatics*, *56*, 102009.

Kurtser, P., & Edan, Y. (2020). Planning the sequence of tasks for harvesting robots. *Robotics and Autonomous Systems*, *131*, 103591.

Lázaro-Gredilla, M., Quinonero-Candela, J., Rasmussen, C. E., & Figueiras-Vidal, A. R. (2010). Sparse spectrum gaussian process regression. *The Journal of Machine Learning Research*, *11*, 1865–1881.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436–444.

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

Lee, M. A., Zhu, Y., Srinivasan, K., Shah, P., Savarese, S., Fei-Fei, L., Garg, A., & Bohg, J. (2019). Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks. *2019 International conference on robotics and automation (ICRA)*, 8943–8950.

Lenz, C., Menon, R., Schreiber, M., Jacob, M. P., Behnke, S., & Bennewitz, M. (2024). Hortibot: An adaptive multi-arm system for robotic horticulture of sweet

peppers. *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 31–38.

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

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

Lin, H.-T., Leisk, G. G., & Trimmer, B. (2011). Goqbot: A caterpillar-inspired soft-bodied rolling robot. *Bioinspiration & biomimetics*, *6*(2), 026007.

Lin, X., Qi, C., Zhang, Y., Huang, Z., Fragkiadaki, K., Li, Y., Gan, C., & Held, D. (2022). Planning with spatial-temporal abstraction from point clouds for deformable object manipulation. *arXiv preprint arXiv:2210.15751*.

Ling, Z., Yao, Y., Li, X., & Su, H. (2023). On the efficacy of 3d point cloud reinforcement learning. *arXiv preprint arXiv:2306.06799*.

Lintermann, B., & Deussen, O. (1999). Interactive modeling of plants. *IEEE Computer Graphics and Applications*, *19*(1), 56–65.

Liu, M., Li, X., Ling, Z., Li, Y., & Su, H. (2022). Frame mining: A free lunch for learning robotic manipulation from 3d point clouds. *arXiv preprint arXiv:2210.07442*.

Liu, Q., & Wang, D. (2016). Stein variational gradient descent: A general purpose bayesian inference algorithm. *Advances in neural information processing systems*, *29*.

Liu, S., Zeng, Z., Ren, T., Li, F., Zhang, H., Yang, J., Li, C., Yang, J., Su, H., Zhu, J., et al. (2023). Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*.

Lowe, T., Lichman, S., Pinskier, J., Sun, C., & Dunstall, S. (2022). The identification and management of hazard trees to mitigate bushfire risk.

Lowe, T., & Pinskier, J. (2022). Tree reconstruction using topology optimisation. *Remote Sensing*, *15*(1), 172.

Ma, X., Hsu, D., & Lee, W. S. (2021). Learning latent graph dynamics for deformable object manipulation. *arXiv preprint arXiv:2104.12149*, *2*.

MacKay, D. J. (2003). *Information theory, inference and learning algorithms*. Cambridge university press.

MacKenzie, C., Brodnan, S., Felsche, E., Sabbi, K., Otali, E., Wrangham, R., Rosati, A. G., & Machanda, Z. P. (2025). Wild chimpanzees (pan troglodytes schweinfurthii) use tools to access out of reach water. *American Journal of Primatology*, *87*(4), e70036.

Mahler, J., Liang, J., Niyaz, S., Laskey, M., Doan, R., Liu, X., Ojea, J. A., & Goldberg, K. (2017). Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*.

Maillard, O.-A., Ryabko, D., & Munos, R. (2011). Selecting the state-representation in reinforcement learning. *Advances in Neural Information Processing Systems*, *24*.

Makoviichuk, D., & Makoviychuk, V. (2022). Rl-games: A high-performance framework for reinforcement learning. *Denys88/rl_games*.

Makoviychuk, V., Wawrzyniak, L., Guo, Y., Lu, M., Storey, K., Macklin, M., Hoeller, D., Rudin, N., Allshire, A., Handa, A., et al. (2021). Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*.

Makris, S., Kampourakis, E., & Andronas, D. (2022). On deformable object handling: Model-based motion planning for human-robot co-manipulation. *CIRP Annals*.

Marinho, Z., Dragan, A., Byravan, A., Boots, B., Srinivasa, S., & Gordon, G. (2016). Functional gradient motion planning in reproducing kernel hilbert spaces. *arXiv preprint arXiv:1601.03648*.

Memmel, M., Wagenmaker, A., Zhu, C., Yin, P., Fox, D., & Gupta, A. (2024). Asid: Active exploration for system identification in robotic manipulation. *arXiv preprint arXiv:2404.12308*.

Minamino, R., & Tateno, M. (2014). Tree branching: Leonardo da vinci's rule versus biomechanical models. *PloS one*, *9*(4), e93535.

Mittal, M., Yu, C., Yu, Q., Liu, J., Rudin, N., Hoeller, D., Yuan, J. L., Singh, R., Guo, Y., Mazhar, H., Mandlekar, A., Babich, B., State, G., Hutter, M., & Garg, A. (2023). Orbit: A unified simulation framework for interactive robot

learning environments. *IEEE Robotics and Automation Letters*, *8*(6), 3740–3747. https://doi.org/10.1109/LRA.2023.3270034

Miyagawa, S., DeSalle, R., Nóbrega, V. A., Nitschke, R., Okumura, M., & Tattersall, I. (2025). Linguistic capacity was present in the homo sapiens population 135 thousand years ago. *Frontiers in Psychology*, *16*, 1503900.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. *International conference on machine learning*, 1928–1937.

Muandet, K., Fukumizu, K., Sriperumbudur, B., Schölkopf, B., et al. (2017). Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends® in Machine Learning*, *10*(1-2), 1–141.

Narang, Y., Storey, K., Akinola, I., Macklin, M., Reist, P., Wawrzyniak, L., Guo, Y., Moravanszky, A., State, G., Lu, M., et al. (2022). Factory: Fast contact for robotic assembly. *arXiv preprint arXiv:2205.03532*.

Nguyen, D.-V., Kuhnert, L., & Kuhnert, K. D. (2012). Spreading algorithm for efficient vegetation detection in cluttered outdoor environments. *Robotics and Autonomous Systems*, *60*(12), 1498–1507.

Nhoch, S. (2022). *Indigenous people's living, such as picking vegetables in the forest and foothills as food* [CC BY-SA 4.0, via Wikimedia Commons]. Retrieved August 12, 2025, from https://commons.wikimedia.org/wiki/File:Indigenous_people%27s_living_%2C_such_as_picking_vegetables_in_the_forest_and_foothills_as_food.jpg

NVIDIA. (2025). *Nvidia physx sdk* [Accessed: 2025-08-04]. https://developer.nvidia.com/physx-sdk

Oliveira, L. F., Moreira, A. P., & Silva, M. F. (2021a). Advances in agriculture robotics: A state-of-the-art review and challenges ahead. *Robotics*, *10*(2), 52.

Oliveira, L. F., Moreira, A. P., & Silva, M. F. (2021b). Advances in forest robotics: A state-of-the-art survey. *Robotics*, *10*(2), 53.

Pang, T., & Tedrake, R. (2022). Easing reliance on collision-free planning with contact-aware control. *2022 International Conference on Robotics and Automation (ICRA)*, 8375–8381.

Pavlasek, J., Lewis, S. R., Sundaralingam, B., Ramos, F., & Hermans, T. (2023). Ready, set, plan! planning to goal sets using generalized bayesian inference. *7th Annual Conference on Robot Learning*.

Peng, X. B., Andrychowicz, M., Zaremba, W., & Abbeel, P. (2018). Sim-to-real transfer of robotic control with dynamics randomization. *2018 IEEE international conference on robotics and automation (ICRA)*, 3803–3810.

Pollayil, M. J., Angelini, F., de Simone, L., Fanfarillo, E., Fiaschi, T., Maccherini, S., Angiolini, C., & Garabini, M. (2023). Robotic monitoring of forests: A dataset from the eu habitat 9210* in the tuscan apennines (central italy). *Scientific Data*, *10*(1), 845.

Popov, D., Klimchik, A., & Mavridis, N. (2017). Collision detection, localization & classification for industrial robots with joint torque sensors. *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 838–843.

Prusinkiewicz, P. (1986). Graphical applications of l-systems. *Proceedings of graphics interface*, *86*(86), 247–253.

Prusinkiewicz, P., Hammel, M., Hanan, J., & Mech, R. (1996). L-systems: From the theory to visual models of plants. *Proceedings of the 2nd CSIRO Symposium on Computational Challenges in Life Sciences*, *3*, 1–32.

Prusinkiewicz, P., & Lindenmayer, A. (2012). *The algorithmic beauty of plants*. Springer Science & Business Media.

Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, *30*.

Qin, Y., Huang, B., Yin, Z.-H., Su, H., & Wang, X. (2023). Dexpoint: Generalizable point cloud reinforcement learning for sim-to-real dexterous manipulation. *Conference on Robot Learning*, 594–605.

Quigley, E., Lin, W., Zhu, Y., & Fedkiw, R. (2021). Three dimensional reconstruction of botanical trees with simulatable geometry. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, *4*(3), 1–16.

Rahimi, A., & Recht, B. (2007). Random features for large-scale kernel machines. *Advances in neural information processing systems*, *20*.

Rambow, M., Schauß, T., Buss, M., & Hirche, S. (2012). Autonomous manipulation of deformable objects based on teleoperated demonstrations. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2809–2814.

Ramos, F., Possas, R. C., & Fox, D. (2019). Bayessim: Adaptive domain randomization via probabilistic inference for robotics simulators. *arXiv preprint arXiv:1906.01728*.

Runions, A., Lane, B., & Prusinkiewicz, P. (2007). Modeling trees with a space colonization algorithm. *Nph*, *7* (63-70), 6.

Russell, S., & Norvig, P. (1995). *Artificial intelligence: A modern approach*. Prentice Hall.

Sadeghi, F., & Levine, S. (2016). Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*.

Schölkopf, B., Smola, A. J., Bach, F., et al. (2002). Learning with kernels: Support vector machines, regularization. *Optimization, and Beyond. MIT press*, *1*(2).

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

Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). Trust region policy optimization. *International conference on machine learning*, 1889–1897.

Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2015). High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Seita, D., Wang, Y., Shetty, S. J., Li, E. Y., Erickson, Z., & Held, D. (2023). Toolflownet: Robotic manipulation with tools via predicting tool flow from point clouds. *Conference on Robot Learning*, 1038–1049.

Senanayake, R. (2024). The role of predictive uncertainty and diversity in embodied ai and robot learning. *arXiv preprint arXiv:2405.03164*.

Sharp, C. J. (2023). *Geoffroy's spider monkey (ateles geoffroyi yucatanensis), peten* [CC BY-SA 4.0, via Wikimedia Commons]. Retrieved July 8, 2025, from https://c

ommons.wikimedia.org/wiki/File:Geoffroy%27s_spider_monkey_(Ateles_geoffroyi_yucatanensis)_Peten.jpg

Shonkwiler, R. W., & Mendivil, F. (2009). *Explorations in monte carlo methods*. Springer.

Siciliano, B., Sciavicco, L., Villani, L., & Oriolo, G. (2009). *Robotics: Modelling, planning and control*. Springer.

Siebert, L. C., Toledo, L. F., Block, P. A., Bahlke, D. B., Roncolatto, R. A., & Cerqueira, D. P. (2014). A survey of applied robotics for tree pruning near overhead power lines. *Proceedings of the 2014 3rd International Conference on Applied Robotics for the Power Industry*, 1–5.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014). Deterministic policy gradient algorithms. *International conference on machine learning*, 387–395.

Subedi, N., Yang, H.-J., Jha, D. K., & Sarkar, S. (2025). Find the fruit: Designing a zero-shot sim2real deep rl planner for occlusion aware plant manipulation. *arXiv preprint arXiv:2505.16547*.

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

Taheri, H., Hosseini, S. R., & Nekoui, M. A. (2024). Deep reinforcement learning with enhanced ppo for safe mobile robot navigation. *arXiv preprint arXiv:2405.16266*.

Tang, B., Lin, M. A., Akinola, I., Handa, A., Sukhatme, G. S., Ramos, F., Fox, D., & Narang, Y. (2023). Industreal: Transferring contact-rich assembly tasks from simulation to reality. *arXiv preprint arXiv:2305.17110*.

Tang, C., Abbatematteo, B., Hu, J., Chandra, R., Martín-Martín, R., & Stone, P. (2025). Deep reinforcement learning for robotics: A survey of real-world successes. *Proceedings of the AAAI Conference on Artificial Intelligence*, *39*(27), 28694–28698.

Tang, T., & Tomizuka, M. (2022). Track deformable objects from point clouds with structure preserved registration. *The International Journal of Robotics Research*, *41*(6), 599–614.

Tedrake, R. (2022). Underactuated robotics: Algorithms for walking, running, swimming, flying, and manipulation. 2018. *URL http://underactuated. mit. edu*.

Tevel. (2023). *Flying Autonomous Robots*. Retrieved June 26, 2023, from https://www.tevel-tech.com/

Thrun, S. (2002). Probabilistic robotics. *Communications of the ACM*, *45*(3), 52–57.

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

Todorov, E., Erez, T., & Tassa, Y. (2012). Mujoco: A physics engine for model-based control. *2012 IEEE/RSJ international conference on intelligent robots and systems*, 5026–5033.

Tonooka, R. (2001). Leaf-folding behavior for drinking water by wild chimpanzees (pan troglodytes verus) at bossou, guinea. *Animal Cognition*, *4*, 325–334.

Treit, D., & Fundytus, M. (1988). Thigmotaxis as a test for anxiolytic activity in rats. *Pharmacology Biochemistry and Behavior*, *31*(4), 959–962.

van Herck, L., Kurtser, P., Wittemans, L., & Edan, Y. (2020). Crop design for improved robotic harvesting: A case study of sweet pepper harvesting. *Biosystems Engineering*, *192*, 294–308.

Vien, N. A., Englert, P., & Toussaint, M. (2016). Policy search in reproducing kernel hilbert space. *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 2089–2096.

Vuong, Q., Vikram, S., Su, H., Gao, S., & Christensen, H. I. (2019). How to pick the domain randomization parameters for sim-to-real transfer of reinforcement learning policies? *arXiv preprint arXiv:1903.11774*.

waldoalvarez. (2017). *Markov decision process* [CC BY-SA 4.0, via Wikimedia Commons]. Retrieved July 31, 2025, from https://commons.wikimedia.org/wiki/File:Markov_Decision_Process.svg

Wang, C., Song, C., Xu, T., & Jiang, R. (2025). Precision weeding in agriculture: A comprehensive review of intelligent laser robots leveraging deep learning techniques. *Agriculture*, *15*(11), 1213.

Wang, W., Mao, L., Wang, R., & Min, B.-C. (2024). Multi-robot cooperative socially-aware navigation using multi-agent reinforcement learning. *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 12353–12360.

Warren, H., & Ramos, F. (2024). Fast fourier bayesian quadrature. *International Conference on Artificial Intelligence and Statistics*, 4555–4563.

Watson, J., & Peters, J. (2023). Inferring smooth control: Monte carlo posterior policy iteration with gaussian processes. *Conference on Robot Learning*, 67–79.

Weber, J., & Penn, J. (1995). Creation and rendering of realistic trees. *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 119–128.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, *8*(3), 229–256.

Xu, Z., Chi, C., Burchfiel, B., Cousineau, E., Feng, S., & Song, S. (2022). Dextairity: Deformable manipulation can be a breeze. *arXiv preprint arXiv:2203.01197*.

Yamamoto, S., Hayashi, S., Yoshida, H., & Kobayashi, K. (2014). Development of a stationary robotic strawberry harvester with a picking mechanism that approaches the target fruit from below. *Japan Agricultural Research Quarterly: JARQ*, *48*(3), 261–269.

Yan, X. (2019). Pointnet/pointnet++ pytorch [Accessed: 2025-06-04].

Yandun, F., Parhar, T., Silwal, A., Clifford, D., Yuan, Z., Levine, G., Yaroshenko, S., & Kantor, G. (2021). Reaching pruning locations in a vine using a deep reinforcement learning policy. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2400–2406.

Yandun, F., Silwal, A., & Kantor, G. (2020). Visual 3d reconstruction and dynamic simulation of fruit trees for robotic manipulation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 54–55.

Yang, M., Huang, M.-C., Yang, G., & Wu, E.-H. (2010). Physically-based animation for realistic interactions between tree branches and raindrops. *Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology*, 83–86.

Yao, S., Pan, S., Bennewitz, M., & Hauser, K. (2024). Safe leaf manipulation for accurate shape and pose estimation of occluded fruits. *arXiv preprint arXiv:2409.17389*.

Yin, H., Varava, A., & Kragic, D. (2021). Modeling, learning, perception, and control methods for deformable object manipulation. *Science Robotics*, *6*(54), eabd8803.

Yin, Z., Lai, T., Barcelos, L., Jacob, J., Li, Y., & Ramos, F. (2025). Diverse motion planning with stein diffusion trajectory inference. *IEEE International Conference on Robotics and Automation (ICRA)*.

Yin, Z., Lai, T., Khan, S., Jacob, J., Li, Y., & Ramos, F. (2024). Stein movement primitives for adaptive multi-modal trajectory generation. *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 11901–11908.

Yu, H., & Chen, Y. (2023). Stochastic motion planning as gaussian variational inference: Theory and algorithms. *arXiv preprint arXiv:2308.14985*.

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

Zeid, K. A., Schult, J., Hermans, A., & Leibe, B. (2023). Point2vec for self-supervised representation learning on point clouds. *DAGM German Conference on Pattern Recognition*, 131–146.

Zhang, T., Xu, K., Yao, Z., Ding, X., Zhao, Z., Hou, X., Pang, Y., Lai, X., Zhang, W., Liu, S., et al. (2019). The progress of extraterrestrial regolith-sampling robots. *Nature Astronomy*, *3*(6), 487–497.

Zhang, X., & Gupta, S. (2023). Push past green: Learning to look behind plant foliage by moving it. *arXiv preprint arXiv:2307.03175*.

Zhang, Z., Qian, K., Schuller, B. W., & Wollherr, D. (2020). An online robot collision detection and identification scheme by supervised learning and bayesian decision theory. *IEEE Transactions on Automation Science and Engineering*, *18*(3), 1144–1156.

Zhao, W., Queralta, J. P., & Westerlund, T. (2020). Sim-to-real transfer in deep reinforcement learning for robotics: A survey. *2020 IEEE symposium series on computational intelligence (SSCI)*, 737–744.

Zhao, Y., & Barbič, J. (2013). Interactive authoring of simulation-ready plants. *ACM Transactions on Graphics (TOG)*, *32*(4), 1–12.

Zhou, H., Wang, X., Au, W., Kang, H., & Chen, C. (2022). Intelligent robots for fruit harvesting: Recent developments and future challenges. *Precision Agriculture*, *23*(5), 1856–1907.

Zhu, J., Cherubini, A., Dune, C., Navarro-Alarcon, D., Alambeigi, F., Berenson, D., Ficuciello, F., Harada, K., Kober, J., Li, X., et al. (2022). Challenges and outlook in robotic manipulation of deformable objects. *IEEE Robotics & Automation Magazine*, *29*(3), 67–77.

Acronyms

ABC: Approximate Bayesian Computation
AGI: Artifical General Intelligence
CI: Confidence Interval
CNN: Convolutional Neural Network
CPO: Contact Penalty Only
DL: Deep Learning
DOF: Degree of Freedom
DR: Domain Randomisation
DRL: Deep Reinforcement Learning
DSP: Digital Signal Processor
ELBO: Evidence Lower Bound
FEM: Finite Element Method
GPS: Global Positioning System
GT: Ground Truth
IMU: Inertial Measurement Unit
KME: Kernel Mean Embedding
KL: Kullback-Leibler divergence
LiDAR: Light Detection and Ranging
LLM: Large Language Model
LQR: Linear Quadratic Regulator
MDP: Markov Decision Process
MCMC: Markov Chain Monte Carlo
MCMH: Metropolis–Hastings
MLE: Maximum Likelihood Estimation
ML: Machine Learning
MMD: Maximum Mean Discrepancy
MPC: Model Predictive Control
MSS: Mass-Spring-Damper System
NLL: Negative Log-Likelihood
NNSVGD: Neural Network guided Stein Variational Gradient Descent
PCAP: Proprioceptive Contact-Aware Policy
PCA: Principal Component Analysis
PDF: Probability Density Function
PD: Proportional–Derivative
PID: Proportional–integral–derivative
PLAI: Policy Level Action Integrator
PMF: Probability Mass Function
PPO: Proximal Policy Optimization
RBF: Radial Basis Function kernel
REST: Representational State Transfer
RFF: Random Fourier Feature
RKHS: Reproducing Kernel Hilbert Space
RL: Reinforcement Learning
RNN: Recurrent Neural Network
ROS: Robot Operating System
RGB-D: Red Green Blue-Depth
S3EC: Segmented Steady State Error Control
SAM-HQ: Segment Anything Model
SGD: Stochastic Gradient Descent
SGHMC: Stochastic gradient Hamiltonian Monte Carlo
SLAM: Simultaneous Localization and Mapping
SR: Success Rate
SVGD: Stein Variational Gradient Descent
UAV: Unmanned Aerial Vehicle
UGV: Unmanned Ground Vehicle
URDF: Unified Robotics Description Format
VI: Variational Inference
i.e.: *id est*, that is
e.g.: *exempli gratia*, for example