Gentle Manipulation of Tree Branches: A Contact-Aware Policy Learning Approach

Jayadeep Jacob[1,2], **Shizhe Cai**[1] , **Paulo Borges**[2], **Tirthankar Bandyopadhyay**[2] , **Fabio Ramos**[1,3]

[1]University of Sydney, [2]Data61, CSIRO, [3]NVIDIA

Abstract: Learning to interact with deformable tree branches with minimal damage is challenging due to their intricate geometry and inscrutable dynamics. Furthermore, traditional vision-based modeling systems suffer from implicit occlusions in dense foliage, severely changing lighting conditions, and limited field of view, in addition to having a significant computational burden preventing real-time deployment. In this work, we simulate a procedural forest with realistic, self-similar branching structures derived from a parametric L-system model, actuated with crude spring abstractions, mirroring real-world variations with domain randomisation over the morphological and dynamic attributes. We then train a novel Proprioceptive Contact-Aware Policy (PCAP) for a reach task using reinforcement learning, aided by a whole-arm contact detection classifier and reward engineering, without external vision, tactile, or torque sensing. The agent deploys novel strategies to evade and mitigate contact impact, favouring a reactive exploration of the task space. Finally, we demonstrate that the learned behavioural patterns can be transferred zero-shot from simulation to real, allowing the arm to navigate around real branches with unseen topology and variable occlusions while minimising the contact forces and expected ruptures.
Website: https://sites.google.com/view/pcap/home

Keywords: Reinforcement Learning, Sim-to-Real, Deformable Manipulation, Robot-vegetation interaction

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 [1], 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 [2] 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 [3].

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 (RL) on powerful GPU-based simulators is a safer, sample-efficient, and cost-effective option for contact-rich tasks [4][5]. However, transferring policies from simulation to reality, known as the sim-to-real problem [6], 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 work proposes the following approach: First, we draw inspiration from plant morphology using the parametric L-system (Lindenmayer system) [7] to simulate realistic branching structures coupled with dynamics from crude spring actuation [8]. 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 [4] 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 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.

2 Related Works

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 [4] 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. [2] used Bayesian methods and parameter inference to manipulate deformable objects while [9] used sophisticated steering needles to operate inside a deformable environment, human lungs. [6] provided a survey on sim-to-real transfer, categorising methods into domain randomisation, domain adaptation, and imitation learning, highlighting their combined effectiveness. Dextreme [5] and Shadow Hand [10] 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 L-System [7][11][12], using polygons [13][14], or with space colonisation algorithm [15]. Furthermore, plant simulations for physical interactions are implemented with expensive FEM models [16] or using cheaper mass-spring systems [17][8]. From a learning perspective, [18] explored an RL-based system aided by vision for known plant topologies, emphasizing the need for detailed prior knowledge, often impractical in dynamic environments. [19] 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 [20], tactile sensing [10], or vision [4]. 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 [21], while internal sensors are used in [22], 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 [8], using internal torque sensors to estimate parameters like stiffness and damping. More broadly, [3] 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.

3 Approach

3.1 Shape Representation & Dynamics

L-system: The L-system formalism [7, 23] 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 [11], 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 0L-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 four ternary structures: classes Ta, Tb, Tc, & Td, from [7]. The ternary L-system we used and the parameters for Tb class Σ_b is provided in formulation (1), while the corresponding simulations are shown in Fig.1(a)-(d). For detailed rules, parameters, and growth attributes, see [7]; extensive simulations are available in our website.

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

Branch Dynamics: We extend the coarse-grained abstractions from [8] 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 [8], where the distribution of stiffness and damping parameters $\phi = \{K_p, K_d\}$ of the torsional mass-spring-damper system (Fig.2b) is estimated from real deformations, we do not perform system identification, instead experiment with two models to populate them in advance, namely: a) Rudimentary model: where ϕ is fixed for a given branch level (e.g., $R2$ in Fig.2a 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. b) Beam deflection model [19]: where $K_p = \frac{E\pi r^4}{2l}$ and $K_d = K_p/10$, where E is the Young's modulus of the plant material, and (r, l) are the radius and length of the links.

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.2b) 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 [24], where individual tree instances can be placed into each parallel environment. As for the

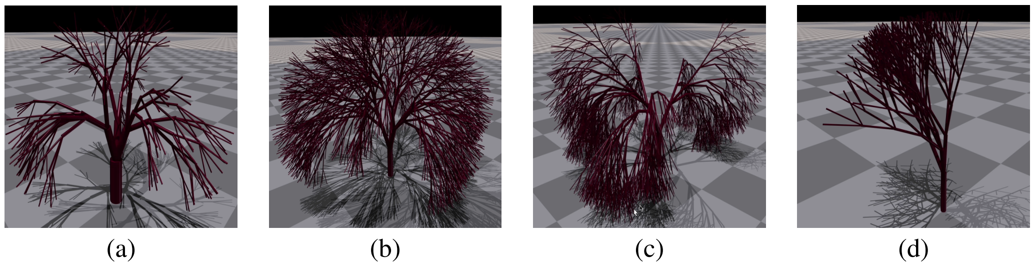

Figure 1: L-system derived branching structure on Isaac Gym, ternary classes Ta, Tb, Tc, & Td

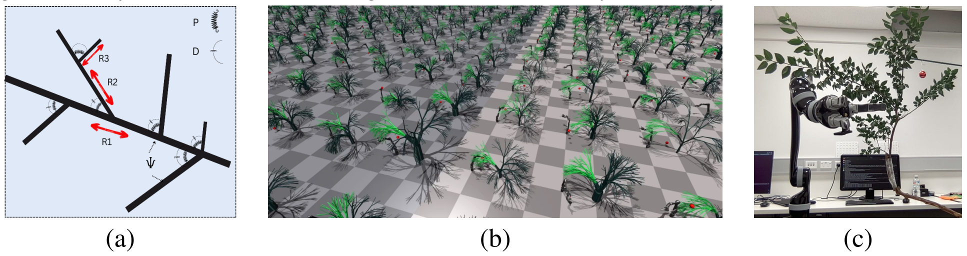

Figure 2: (a) Mass-Spring-Damper branch system from [8] (b) Training with our domain randomised forest generator (ternary class Ta). (c) Real setup: Kinova arm reaching for a previously unseen target (in red), obstructed by a real tree branch of unknown topology and dynamics.

dynamics, while the deformation parameters ϕ of the rudimentary model are randomised [25] with a Gaussian ($\sigma = 1.0$), we do not perturb the Beam 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.

3.2 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 [26][27]. 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 (section 3.3) 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.

3.3 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 (Markov Decision Process) framework, 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 Optimisation(PPO)[28] variation from the rl-games library[29], 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 formulation (2), over episodic rewards for faster learning and easier credit assignment during each exploration step.

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

Reward r_t at each training time step t consists of five distinct components along with their corresponding scaling factors, which are: a) 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, b) a bonus distance reward r_b for close target proximity required for grasps, c) a smoothness reward r_s on the six joint velocities, d) a collision reward r_c to encourage contact evasion or prefer low impact collisions, and finally, e) 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.

3.4 Sim-to-Real

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.

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 [4]. Moreover, accurately replicating the complex contact dynamics between the branches and arm is challenging [25][30], especially when the robot operates in a velocity control mode. As an alternative to parameter estimation, [4] proposes to use Policy Level Action Integrator (PLAI) by integrating policy actions over time, effectively applying them to the desired state s^d_t in contrast to the current state s_t, in other words, $s^d_{t+1} = s^d_t \oplus a_t = s^d_t \oplus \pi(o_t)$, 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, $s^d_{t+1} = s_t \oplus \pi(o_t)$, 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.

$$s^d_{i+1} = s^d_i \oplus \frac{\pi(o_t)}{n}, \quad s^d_i = \begin{cases} s_t & \text{if } i\%n = 0 \\ s^d_i & \text{otherwise} \end{cases}, \quad \text{for } i = 0, 1, 2, \ldots, n \times t \tag{3}$$

Intuitively, formulation (3) could be viewed as local disturbance rejection over the short term, with periodic syncing to the current state to offset accumulated deviations.

Robot Parameters: The Domain Randomisation in section (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 Experiments & Results

We evaluate the effectiveness of our proposed approach through the following experimental setups:

Experiment 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 through branch occlusions, disregarding any plant damage. Second, a Contact Penalty Only (CPO) policy uses our reward formulation (2), but that does not take in the classifier predictions towards o_t. Third, a Model Predictive Control (MPC) baseline from [31], 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 formulation (2). Specific policy details are presented in Table 1, where $\theta_t, \dot{\theta}_t, q_t,$ & d_t represent the joint angle (6D), joint velocity (6D), the gripper

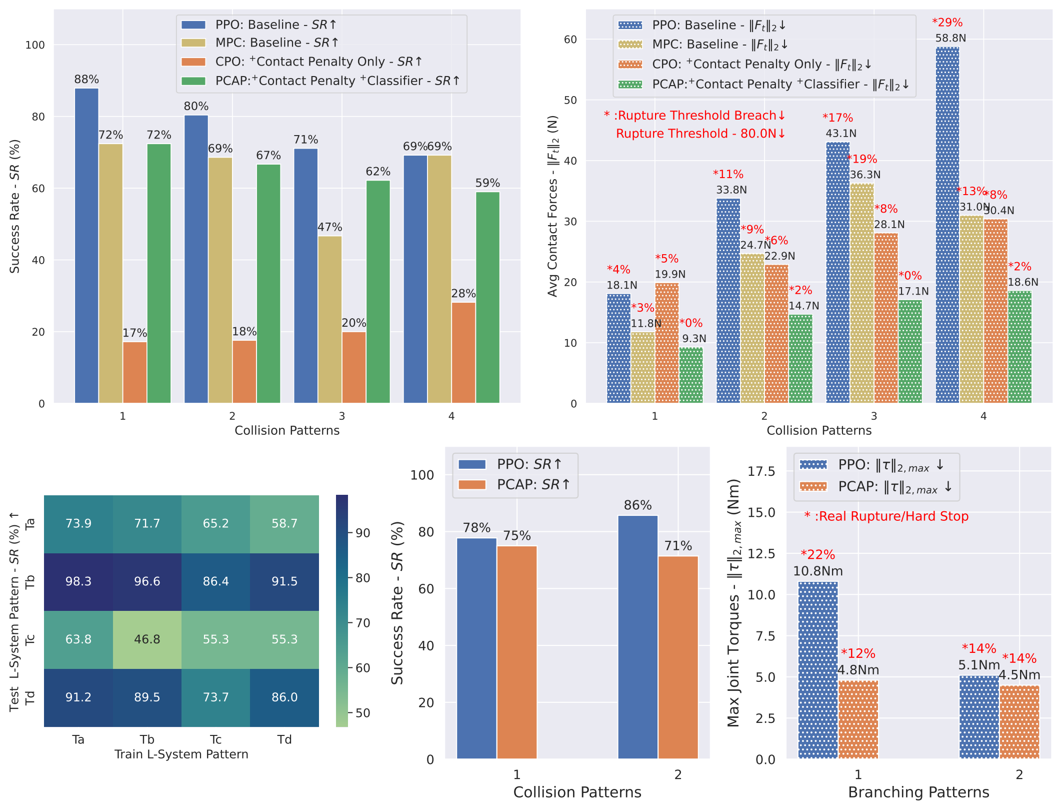

Figure 3: **(Top) Exp 1:** Comparison of Success Rates (SR) and Average Contact Forces across four policies: PPO, MPC, CPO, and PCAP (ours). For SR, higher values are better, for $\|F_t\|_2$ & Rupture values (red%), the lower the better. **(Bottom Left) Exp 2:** SR for various L-System ternary classes. 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.

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.(3: 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 section A.3.

Experiment 2 (L-System Variability): We generate four distinct sets of ternary forests, each corresponding to classes Ta, Tb, Tc, & Td (Fig.1). Then, we train PCAP policies on each forest and test its performance on all classes. The result heat map, presented in Fig.(3: 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.

Experiment 3 (Real-world Tests): 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 3.2. 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.(3: 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 (A.4), geometry ablation with and without L-system (A.5), dynamics ablation (A.7), and the classifier performance (A.8).

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 supplementary material 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 Appendix (A.1).

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.

5 Conclusion and Limitations

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.

Limitations to the method include the reliance on simulated environments, which may not capture all real-world complexities. We acknowledge that the [purposeful] absence of vision-based feedback could restrict performance in highly unpredictable scenarios, but the objective was to push the limits of the proprioceptive approach. Future work should explore integrating multi-modal sensory inputs and improving the fidelity of simulated environments to enhance the transferability and efficacy of contact-aware policies in diverse applications.

Acknowledgments

This work was partially supported the SILVANUS Project through European Commission Funding on the Horizon 2020 call number H2020-LC-GD-2020, Grant Agreement number 101037247. The authors are also grateful to multiple groups at CSIRO, Data61 and the University of Sydney who supported this work.

References

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

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

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

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

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

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

[7] P. Prusinkiewicz and A. Lindenmayer. *The algorithmic beauty of plants*, chapter 2, pages 58–61. Springer Science & Business Media, 2012.

[8] J. Jacob, T. Bandyopadhyay, J. Williams, P. Borges, and F. Ramos. Learning to simulate tree-branch dynamics for manipulation. *IEEE Robotics and Automation Letters*, 2024.

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

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

[11] P. Prusinkiewicz. Graphical applications of l-systems. In *Proceedings of graphics interface*, volume 86, pages 247–253, 1986.

[12] P. Prusinkiewicz, M. Hammel, J. Hanan, and R. Mech. L-systems: from the theory to visual models of plants. In *Proceedings of the 2nd CSIRO Symposium on Computational Challenges in Life Sciences*, volume 3, pages 1–32. Citeseer, 1996.

[13] J. Weber and J. Penn. Creation and rendering of realistic trees. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 119–128, 1995.

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

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

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

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

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

[19] C. H. Kim, M. Lee, O. Kroemer, and G. Kantor. Towards robotic tree manipulation: Leveraging graph representations. *arXiv preprint arXiv:2311.07479*, 2023.

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

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

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

[23] H. Honda. 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, 1971.

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

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

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

[27] Z. Zhang, K. Qian, B. W. Schuller, and D. Wollherr. 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, 2020.

[28] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[29] D. Makoviichuk and V. Makoviychuk. rl-games: A high-performance framework for reinforcement learning. *Denys88/rl_games*, 2022.

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

[31] M. Bhardwaj, B. Sundaralingam, A. Mousavian, N. D. Ratliff, D. Fox, F. Ramos, and B. Boots. Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation. In *Conference on Robot Learning*, pages 750–759. PMLR, 2022.

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

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

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

A Appendix

A.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 [32], slows down in response to external contact. Similarly, many organisms, such as rodents and ground beetles, exhibit both negative and positive thigmotaxis [33], 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) [34] 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.

A.2 Observation/Reward Table

The observation and reward combinations for each policy type are presented in Table 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 (3.3)

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 (ours)	$\theta_t, \dot{\theta}_t, q_t, d_t, \mathbb{I}(\|F_t\|_2 > f_u)$	$\{g_d \cdot (r_d + r_b) + g_s \cdot r_s\} \cdot r_c + r_p$

Table 1: Training observations & rewards provided for each policy

A.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.(3: 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 Success Rate (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 $\|F_t\|_2$, 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 (color, 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.

A.4 Time Efficiency

In every experiment throughout this work, we limit the number of steps (simulation: 800, real: 1000) the agent can take. 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.(4) 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.

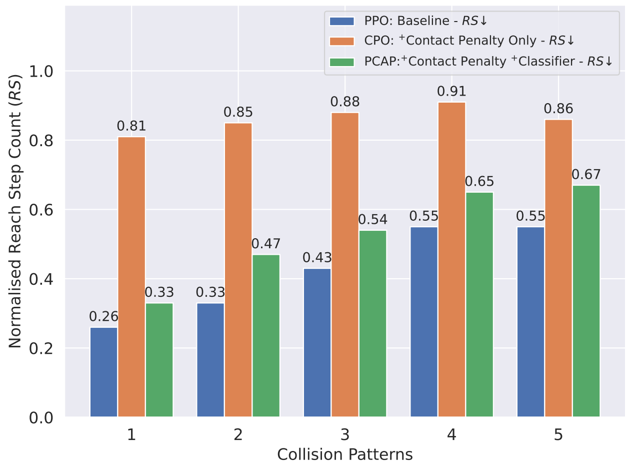

Figure 4: Comparison of the number of RL steps taken to reach the target (or timeout) across three policies: PPO, CPO, & PCAP (ours). For RS, lower values are better, indicating lower time taken.

The 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 maneuvers such as rolling and retractions (rather 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.

A.5 Geometry Ablation

This section demonstrates the benefits of the novel L-System-based forest generator compared to the simpler branching structure proposed in [8]. While it is well known that L-system generated topology can generate realistic branching patterns compared to naive models (sample images presented in Fig.5.), 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 [8] 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 3.1. The results in Fig. 6 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.

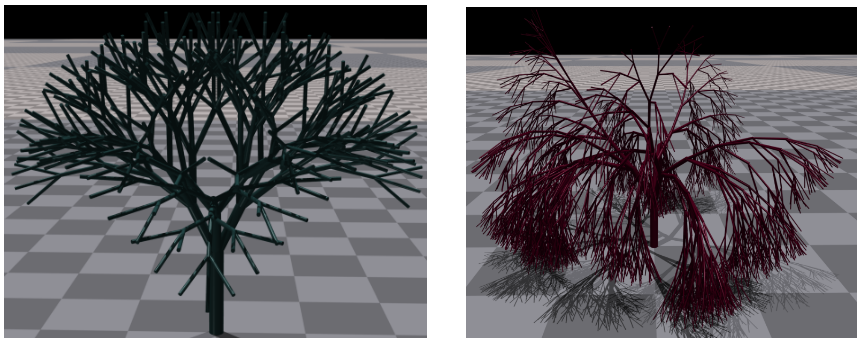

Figure 5: **(Left):** A simplistic tree procedurally generated from [8] **(Right):** An L-System generated tree from the ternary classes Tc, based on our method. Both simulations are presented without any randomisation, to highlight the intrinsic advantage of L-system to generate diverse occlusion patterns, asymmetric arching angles and non-uniform parent-child splits across the topology.

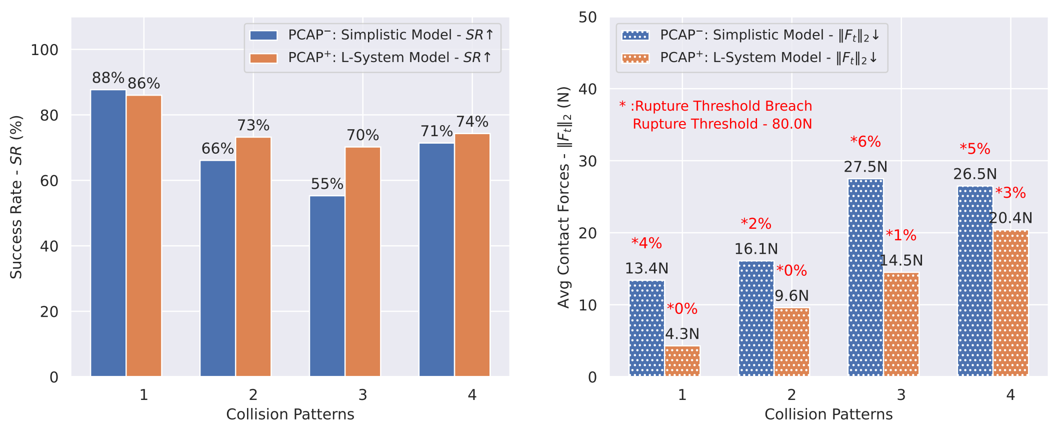

Figure 6: Ablation study of the geometric models comparing the simplistic branching model from [8] and the L-system model (ours), keeping the Dynamics fixed. The comparison is performed by domain randomising both base models. For SR, higher values are better, for $\|F_t\|_2$, the lower the better.

A.6 Explicit Classifier vs Raw Joint Torques

As illustrated in Table 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.8) 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 [8]. Second, we perform an additional ablation study (Fig. 7), 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 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.

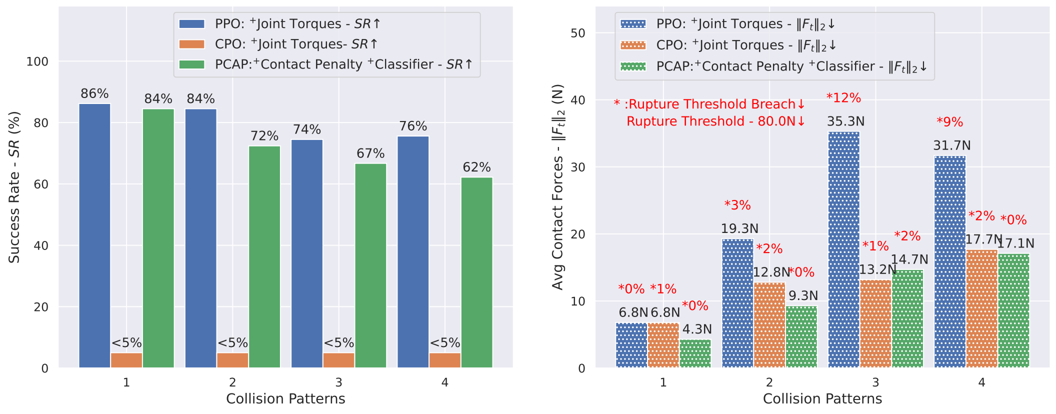

Figure 7: Ablation results comparing PCAP with the explicit classifier to PPO$^+$ and CPO$^+$ with τ_t included as part of o_t.

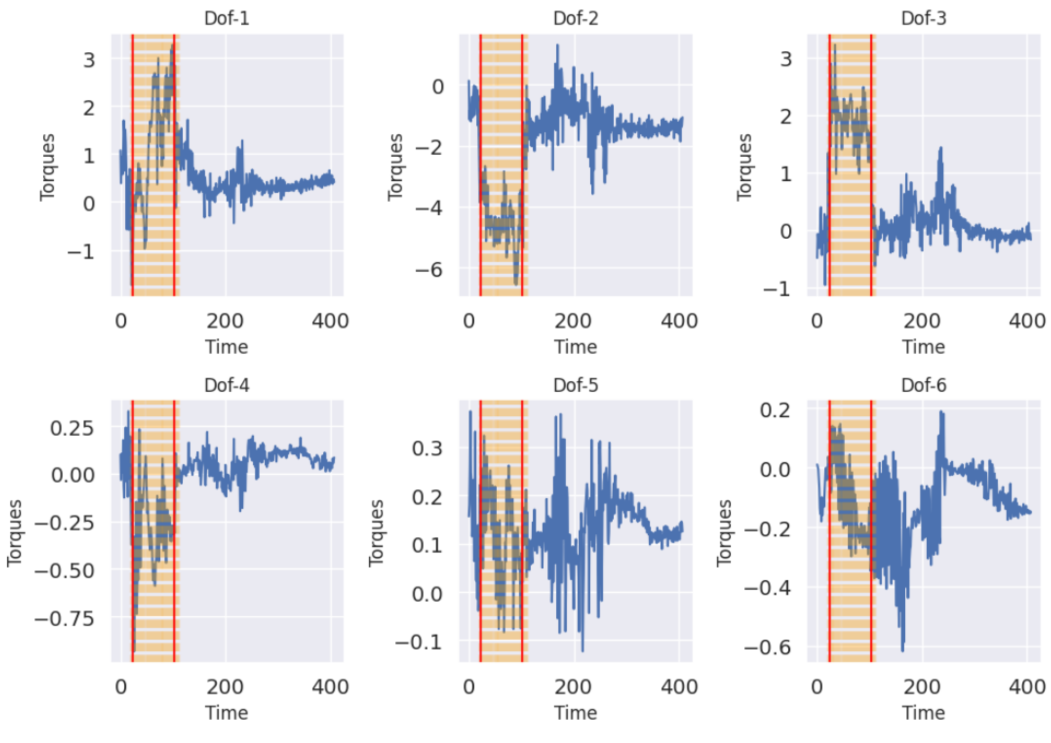

Figure 8: A sample prediction given by the classifier. The input in blue is the time series noisy joint torques of the six dofs. Red solid lines represent the ground truth, i.e., the start and end of the obstruction. Yellow dashed lines represent the predicted contact at each time-step.

A.7 Dynamics Ablation

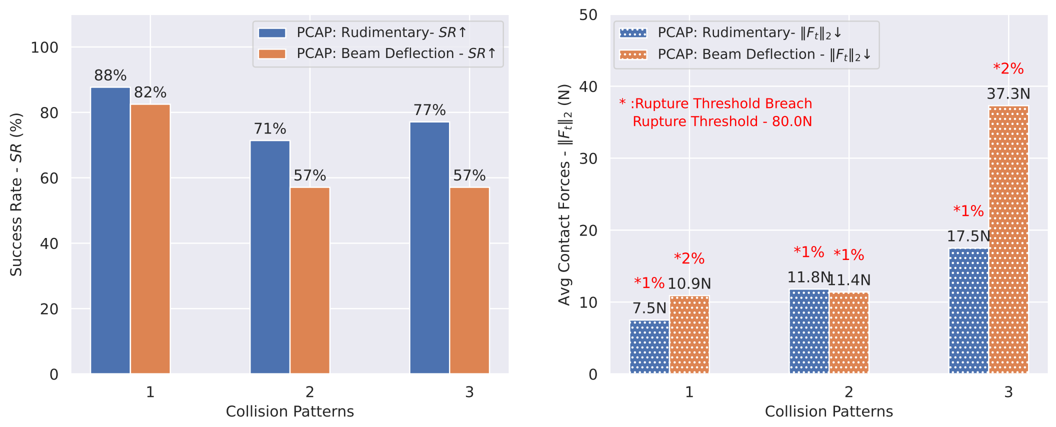

Figure 9: Ablation on the tree dynamics models, keeping the L-system geometry fixed. Comparison of Success Rates (SR) and Average Contact Forces across two dynamics models: Rudimentary and Beam Deflection. For SR, higher values are better, for $\|F_t\|_2$, the lower the better.

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 3.1. From Fig.(9), 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 perturbations while training. This result strongly suggests that integrating system identification for dynamic parameters [8] can significantly boost policy learning.

A.8 Classifier Metrics

Metrics for our contact detection classifier with two off-the-shelf classifiers are given in Fig.(10): 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.

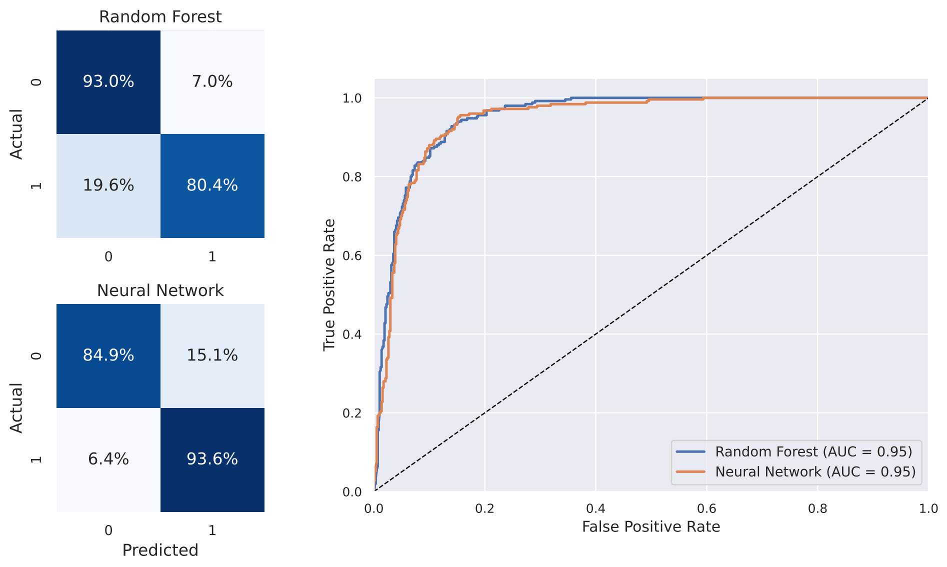

Figure 10: Confusion matrix & ROC for the two contact detection classifiers

A.9 L-System Perturbation

We domain randomise L-system morphological parameters Σ by injecting Gaussian Noise while recursing through the production rules. From Fig.11, we observe that low-level perturbations (σ = 0.01) create structures with limited diversity in topology, identical branches, and similar spans. By contrast, high noise levels (σ = 0.25) generate infeasible spans with few prospects for robot engagement. We settle for a Goldilocks level (σ = 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, move the trees along the gravity axis, and rotate them to maximise occlusions and arm interactions.

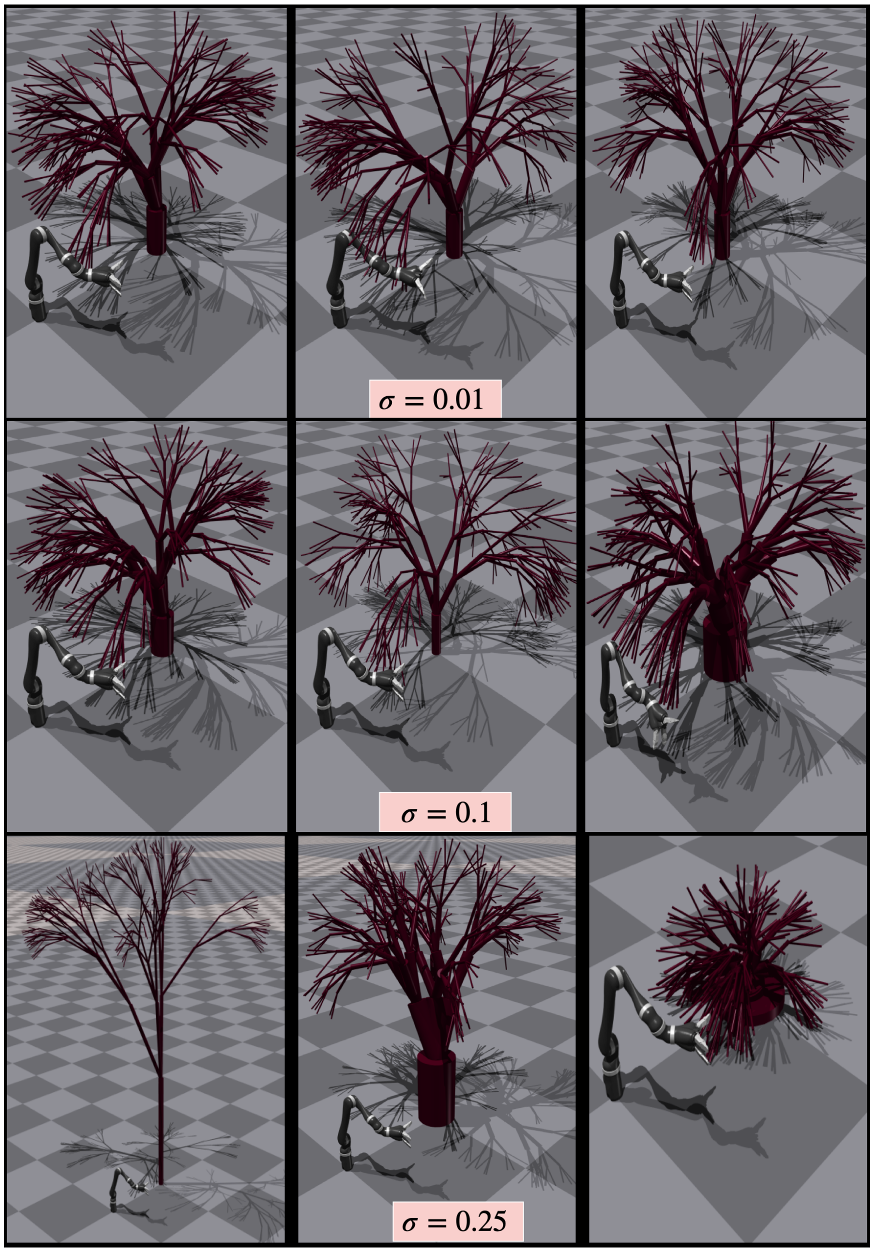

Figure 11: Effects of perturbations on topology: Samples generated by varying noise levels injected in L-System parameters. **(Top Row)**: too low (σ = 0.01) , **(Middle Row)**: our choice (σ = 0.1), **(Bottom Row)**: too high (σ= 0.25)

A.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 2: Additional hyper parameter and configuration settings