IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Deformable Cluster Manipulation via Whole-Arm Policy Learning

Jayadeep Jacob[1,2], Wenzheng Zhang[1], Houston Warren[1],
Paulo Borges[3], Tirthankar Bandyopadhyay[2], Fabio Ramos[1,4]

***Abstract*—Manipulating clusters of deformable objects presents a substantial challenge with widespread applicability, but requires contact-rich whole-arm interactions. A potential solution must address the limited capacity for realistic model synthesis, high uncertainty in perception, and the lack of efficient spatial abstractions, among others. We propose a novel framework for learning model-free policies integrating two modalities: 3D point clouds and proprioceptive touch indicators, emphasising manipulation with full body contact awareness, going beyond traditional end-effector modes. Our reinforcement learning framework leverages a distributional state representation, aided by kernel mean embeddings, to achieve improved training efficiency and real-time inference. Furthermore, we propose a novel context-agnostic occlusion heuristic to clear deformables from a target region for exposure tasks. We deploy the framework in a power line clearance scenario and observe that the agent generates creative strategies leveraging multiple arm links for de-occlusion. Finally, we perform zero-shot sim-to-real policy transfer, allowing the arm to clear real branches with unknown occlusion patterns, unseen topology, and uncertain dynamics.**
Website: https://sites.google.com/view/dcmwap/

***Index Terms*—Dexterous Manipulation, Reinforcement Learning, Simulation and Animation**

I. INTRODUCTION

Learning to manipulate deformable objects poses a formidable challenge in robotics [1]; however, operating with clusters of deformables, such as cable bundles, multi-branch tree canopies, and cloth piles, is significantly harder. Although imprecise physics knowledge [2] 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 [3]. Furthermore, with deformable clusters, tiny differences in contact locations, material 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 [4].

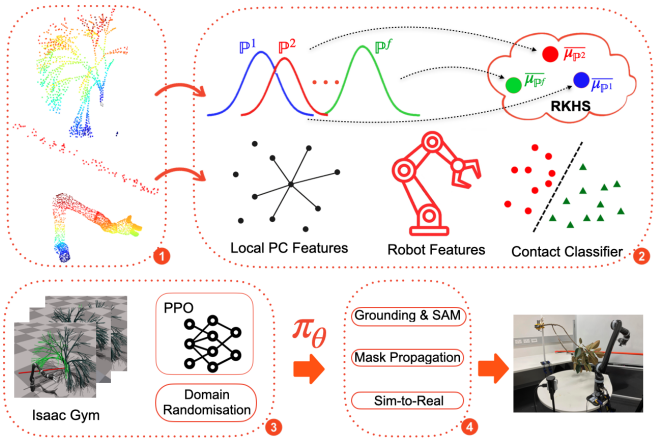


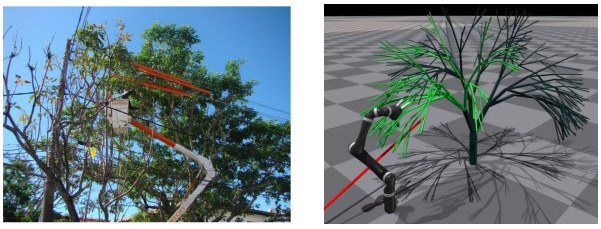
Fig. 1: Overview: (1) Segmented point clouds corresponding to the clustered deformable branches, clearance region, and the robot are captured from the scene. (2) Distribution embeddings representing global scene are generated via kernel mean operator. Additional features include neighbourhood point features, 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.

Novel deformable works manipulating cables [5], dough [6], and clothes [7] 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 [8] or too slow [9] for low-latency applications.

Addressing these challenges, we propose a model-free, non-prehensile, multi-modal, reinforcement learning (RL) approach operating on the cluster as a whole with full-body contact, but without external tactile sensing. We use a distributional representation [3] of the cluster state generated by embedding point cloud into a Reproducing Kernel Hilbert Space (RKHS), without key-point intermediaries. Compared to traditional

Manuscript received: July, 04, 2025; Accepted December, 19, 2025.
This paper was recommended for publication by Editor Julia Borras Sol upon evaluation of the Associate Editor and Reviewers' comments.
[1,2]Jayadeep Jacob, [1]Wenzheng Zhang, and [1]Houston Warren are with the School of Computer Science, The University of Sydney, NSW, Australia. Jayadeep Jacob is also with Data61, CSIRO, Pullenvale, QLD, Australia. jjac4485@sydney.edu.au; jay.jacob@data61.csiro.au; {wzha2981, houston.warren}@sydney.edu.au
[2]Tirthankar Bandyopadhyay is with the CyberPhysical Systems Program, Data61, CSIRO, Pullenvale, QLD, Australia. tirtha.bandy@csiro.au
[3]Paulo Borges is with Orica. paulo.borges@orica.com
[1,4]Fabio Ramos is with the School of Computer Science, The University of Sydney, NSW, Australia, and with the NVIDIA Corporation, Seattle, USA fabio.ramos@sydney.edu.au
Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.



(a) (b)

Fig. 2: (a) Manual pruning of overhanging branches impeding power lines, from [11] (b) Our simulation setup with an L-system structure for the power line clearance scenario.

point cloud feature extractors, a distribution embedding is notably lightweight, enabling faster training and real-time inference. It eliminates the need for points to track identical cluster locations, is invariant to permutations, and can handle variable input sizes due to members moving out of camera view between frames. Further, the observations include a whole-arm contact classifier [10] output from proprioceptive input to flag contacts. Finally, we address the high sample requirements of on-policy RL by leveraging the parallelism of physics simulators during training and transferring it zero-shot to the real-world.

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. We implement two illustrative applications of such a skill abstraction:

1) **Clearing Power Lines**: Hazardous tree branches encroaching on overhead power infrastructure is a leading cause of forest fires and outages [12][13]. Pruning foliage near overhead lines (Fig. 2a). is dangerous for humans due to risks like falls, shocks, and tool accidents [11]; therefore, robotic assistive systems that can clear the branches away from the lines before pruning can enhance safety while reducing outages.
2) **Agricultural Exposure**: Modern harvesting systems suffer from lower success rates with partially (50%-75%) and fully occluded (5%) fruits [14]. Manipulating plant stems for exposing fruits or a diseased region to an external inspection camera requires a line-of-sight restoration through multi-branch clearance.

Among these, the first problem presents the greatest challenge due to additional constraints imposed by the rigid power line on robot's motion, and thus serves as the primary focus of this paper in both simulation and real settings. While our aim is not to provide an industry-grade system, it is notable that no 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 method. Specifically, our contributions are:

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 generalisable 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.

Our first two contributions are technical and apply to deformable manipulation in general, while the third and fourth are system-level, focusing on de-occlusion, but deployable beyond trees. Subsequent sections specifically focus on the power line clearance problem; however, the methodology is readily adaptable to others. We encourage readers to view the supplementary videos to see our approach in action.

II. RELATED WORK

For deformable manipulation with high-dimensional geometries, point clouds offer a rich sensor-agnostic scene mapping. Notable 1D implementations for tracking [15] and learning dense depth object descriptor features [16] leverage point representations. Point clouds have been used to learn policies with 2D and 3D deformables as well for cloth manipulation [7], liquid scooping [17], and dough cutting [6]. On the policy front, these deformable works use either Motion Planning with primitives [7], Task and Motion Planning [6], or Behavioural Cloning [16]. In contrast, model-free RL works on deformables like ours are scarce, largely due to poor sample efficiency. With rigid body point clouds, model-free policies have been proposed [18] for interactive tasks, including sim-to-real transfer.

Post-scene capture, deformable representation strategies include keypoints [15], graphs [19], and dense-descriptors [16]. Relevant to our work, the deformable state can be represented as as a distributional embedding with a kernel mean operator [3]. 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 hooks and props, leveraging the whole arm for clearance. While such formations are less prevalent, contact-aware reaching has been achieved with external full-body tactile sensors, for rigid clutter [8], or with internal sensors for tree branches [10]. However, we go beyond simple reach to sophisticated clearance tasks and implement a multi-modal policy integrating touch and vision feedback.

Among deformables, plant interaction is substantially harder owing to the non-uniformity in geometry, dynamics, and visual features. While [20] gently moves a leaf aside to estimate hidden fruit pose, [21] clears multiple leaves simultaneously with the end-effector using large quantities of real-world data. Both works assume soft leafy resistance, which is typically addressed in farms with simpler tricks, such as blowing compressed air [22] to separate leaves and fruits. From a learning perspective, [23] and [24] employ vision-based RL for reaching vine pruning locations and for exposing fruits, respectively; however, both assume known plant topology while

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

focusing on the end-effector. In comparison, our approach accommodates stiff resistance from multiple branches, uses the whole arm to maximise applied torques, and is trained in simulation without 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.

III. APPROACH

A brief overview of our approach (Fig. 1) is as follows. First, we generate parallel simulations of deformable tree-branches (Sec. III-A) to train an RL policy (Sec. III-B). To guide policy learning, we compute an occlusion metric from the scene, which serves as a reward signal (Sec. III-C). The policy is trained in a physics simulator with domain randomisation applied to the geometry and dynamics of trees (Sec. III-D). The learning framework leverages a kernel mean operator (Sec. III-E) to encode the segmented point clouds as RL features (Sec. III-F). In addition, the features include robot metrics and the output of a touch classifier. For sim-to-real transfer (Sec. IV-B), we use several vision-based segmentation and masking methods to process camera images, generate object masks and extract the vision features necessary for real-world policy deployment (Sec. IV-C).

A. Deformable Simulation

Realistic simulation of the intricate tree-branch geometry is challenging; fortunately, this is well-studied in literature under the L-system paradigm [25], 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. Recently, [10] has shown that this paradigm can be exported to 3D physics simulators, 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 [26] actuated with proportional-derivative controllers. We borrow the recursion rules, growth attributes, and L-system parameters from [10]; however, we extend it with spherical joints to enable multi-axis deformation and enhance compliance, refer Fig. 2(b). We run thousands of such tree models in parallel on the distributed physics simulator Isaac Gym [27].

B. Policy Learning

We formulate the de-occlusion task as a discrete-time Markov Decision Process (MDP), 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 RL algorithm choice is Proximal Policy Optimisation [28].

C. Occlusion Heuristic as Reward Signal

A key consideration for de-occlusion is determining an effective yet simple measure to capture the temporal progression in occlusion level from manipulation, for instance, as a reward signal to guide the policy learning. Common metrics include visible surface ratio, pixel counts [29] or labelled occlusion levels. However, these metrics depend either on prior knowledge of the target geometry or partial observability at start, and 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 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 $D \in \mathbb{R}^{N_1 \times N_2}$:

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

Let $\mathcal{S} \subset \{1, \ldots, N_1\} \times \{1, \ldots, N_2\}$ 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 $\mathcal{D}_k := \{D_{i,j} \mid (i, j) \in \mathcal{S}\}$ and

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

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. Larger the number of branch segments covering the clearance region, higher the h_t. The threshold d_{th} (set to 10cm) is the safe distance for an auxiliary robot to prune the cleared branches. The k (set to 200) balances the branch-line interaction fidelity against computational cost; a lower k risks skipping critical contact points, while a higher k significantly increases processing time. We emphasise 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.

D. Domain Randomisation for RL training

This work aims to generalise a learned policy to deformables with unseen geometry, undetermined dynamics, and devoid of chromatic affinity, by not relying on digital twins [24], parameter inference [26][3], or RGB images during training. Instead, we perturb simulation parameters, for example, L-system traits such as branch divergence angle and elongation rate, and dynamics parameters such as spring stiffness and damping. Further, we subsample and permute all point clouds while adding noise at each time-step.

E. Kernel Mean Embedding for Observation Encoding

In this section, we briefly examine distribution embedding and its spectral approximation; for a thorough explanation, see [30][31]. An RKHS is a Hilbert space of functions uniquely determined by a positive definite kernel function

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, that guarantees both an implicit feature mapping $\varphi : \mathcal{X} \to \mathcal{H}$ from the input space and the higher-dimensional target space $\mathcal{H}$. Akin to point mappings, a kernel mean operator lifts a probability distribution $\mathbb{P}$ to a single mean function in $\mathcal{H}$, however, without any loss of information for a characteristic kernel such as RBF: $k(x, x') = \exp(-\frac{||x-x'||^2}{2\gamma^2})$. We imagine x to be points from the 3D point clouds. The lifted kernel mean is defined as:

$$\varphi(\mathbb{P}) = \mu_{\mathbb{P}} := \mathbb{E}_{x\sim\mathbb{P}}[k(\cdot, x)] = \int k(\cdot, x)\, d\mathbb{P} \tag{2}$$

Intuitively, formulation (2) could be viewed as the average kernel similarity of parameter x to all domain points according to the distribution $\mathbb{P}$ in RKHS. 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 $\mu_{\mathbb{P}}$ from just the i.i.d samples $\{x_1, \ldots, x_N\}$ drawn from $\mathbb{P}$.

$$\overline{\mu_{\mathbb{P}}} := \frac{1}{N}\sum_{i=1}^{N} \varphi(x_i) = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} \langle \varphi(x_i), \varphi(x_j)\rangle_{\mathcal{H}} \tag{3}$$

While the inner product $\langle ., . \rangle_{\mathcal{H}}$ in eqn. (3) can indeed be computed with a finite-dimensional gram matrix, leveraging the reproducing property $\langle \varphi(x_i), \varphi(x_j)\rangle_{\mathcal{H}} = k(x_i, x_j)$, but this is still problematic for large N. On the other hand, representing kernel functions through their spectral representations, particularly with Random Fourier Features (RFF) [31], significantly reduces the computational overhead. Through Bochner's theorem, any kernel can be defined as a Fourier transform of a non-negative measure, subject to a stationarity constraint $k(x, x') = k(x - x')$.

A Monte Carlo estimate of the kernel can then be written as $k(x, x') \approx \frac{1}{R}\sum_{r=1}^{R} \cos(\omega_r^\top (x - x'))$ where the frequencies ω_r are sampled from a spectral density, $\omega_r \sim p(\omega)$. The above estimate becomes computationally efficient for $R << N$, which we exploit. If $k(x, x') \approx \hat{\varphi}^\top(x)\hat{\varphi}(x')$, the empirical kernel embedding can be further simplified as:

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

A higher R indicate more informative features, but an excessive value explodes the observation space, demanding more samples during training. As a trade-off, we choose $R = 16$ and use the RBF kernel, which results from the Fourier transform of a Gaussian $p(\omega) \sim \mathcal{N}(0, 1)$.

F. Distribution Embedding for Deformable Clearance

Given a point cloud $P_t \in \mathbb{R}^{N\times 3}$ at time-step t, we assume it as samples from an underlying distribution. These 3D points are elevated to RKHS with an RFF approximated kernel mean to form the key constituent of RL observations. 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. While the robot points are sampled from URDF mesh based on link positions at each step, the cylindrical clearance region points are sampled from its surface, given a fixed pose. 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. 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++ [32], 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 III-E, 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.

G. Observation & Reward Space

TABLE I: 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**

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

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

A robot-power line contact doesn't compromise the validity of our approach, as the overhanging branches act as conductors already[13]; nevertheless, a safety margin ($d_{sm} = 4$cm, to account for motion and sensor uncertainties) is introduced to reduce power line wear and tear.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

A novel aspect of our method is its multi-modality, attributable to a whole-arm touch detector. Recently, [10] has shown that in the absence of expensive tactile sensors, time-series sliding window torque measurements: $[\tau_t, \tau_{t-1}, \tau_{t-2}, ..., \tau_{t-(m-1)}]$ 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 labeled touch data. While [10] 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: a) an occlusion clearance reward r_h, b) a smoothness reward r_q on the joint velocities $\dot{q}$, and c) a safety bonus r_{sm} to keep the arm clear of the power line safety margin,

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

The clearance reward $r_h \in [0,1]$ is a monotonically decreasing function of the occlusion heuristic h_t. The r_{sm} coefficient (0.4), is hand-tuned to make the sparse safety bonus noticeable without overshadowing clearance reward, whereas r_q is a penalty term to encourage smoother trajectories.

IV. EXPERIMENTAL SETUP

A. Simulation Setting

For power line clearance (Fig. 2b), we run 6144 parallel Gym environments, each simulating a tree with deformable branches (in green), a rigid power line (in red), and a Kinova arm actuated via velocity control of its six 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. 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. 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. In contrast, during simulation tests, all contacts are in place to accurately reflect real-world, but the environment count is set low. This approach, along with careful tuning of the Temporal Gauss-Seidel (TGS) contact solver parameters, enables us to run large-scale training without compromising representational accuracy.

B. Hardware Design & Sim-to-Real

In this work, 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. 3:(c)-(e). 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 (SSED)[10]. 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.

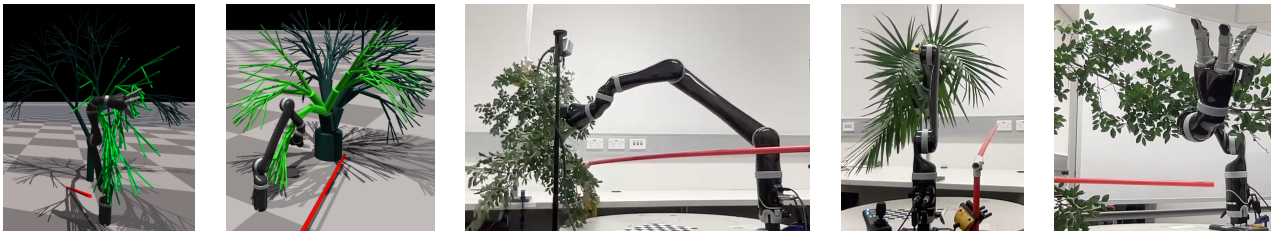
C. Real Vision Pipeline

This section outlines the key transformation steps required to convert the real-time streaming images into RL features. First, we use Grounding DINO [33] to locate the objects in the scene with simple text prompts. The resulting bounding boxes are passed to a Segment Anything SAM-HQ [34] model to generate object masks. The masks are then de-projected to the robot frame, leveraging the depth channel input. Crucially, manipulators operating in velocity drive 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 compromising the mask quality. Therefore, we use a video segmentation framework Cutie [35], to propagate the mask across time-steps, occasionally replacing it with updated masks to prevent quality loss.

V. EXPERIMENTS AND RESULTS

Experiment 1: First, we establish quantitative non-RL baselines by executing hand-tuned, multi-step, inverse kinematics

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.



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

Fig. 3: Terminal state poses: (a)(b): Simulation trajectory states showing the whole arm being utilised to shield the power line. (c)-(e): Similar real strategies using various arm links for clearance, executed on branches of different tree species.

TABLE II: Baseline Comparison & Feature Ablation

	Description	$d_{\max}$	Train Rew	Trials	Test Rew	Test SR%	Occ Drop%	Touch%	Steps in Succ
(a)	Baseline: Guided IK - Lift	-	-	1536	-	33.4 ± 2.7	12.7 ± 1.4	11.3 ± 0.4	179.1 ± 13.5
(b)	Baseline: Guided IK - Sweep	-	-	1536	-	41.1 ± 1.7	17.2 ± 2.2	11.0 ± 0.6	193.5 ± 17.5
(c)	Baseline: GNN Features	0.02	1613.9	1536	1511.7 ± 21.0	55.5 ± 2.9	33.6 ± 1.7	10.5 ± 0.1	291.4 ± 14.6
(d)	Proprioceptive + Heuristic: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
(e)	(d) + Local PC Features	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
(f)	(e) + 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
(g)	(e) + 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
(h)	All Features (RBF)	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
(i)	All Features (Laplace)	0.02	1592.8	1536	1559.0 ± 35.0	52.5 ± 1.8	36.0 ± 2.4	4.0 ± 0.2	315.9 ± 21.0
(j)	All Features (Matérn-3/2)	0.02	1662.5	1536	1620.0 ± 25.5	59.8 ± 1.3	43.5 ± 1.5	3.2 ± 0.2	362.9 ± 16.7
(k)	**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**

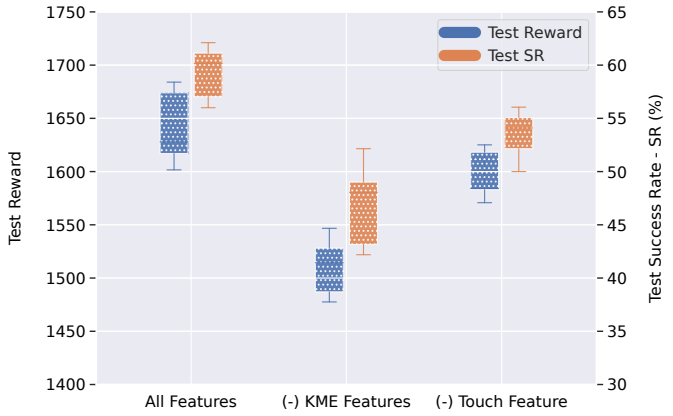


Fig. 4: Ablation showing the relevance of key feature groups in our multi-modal policy. (-) indicates the removal of the single specified group. Each data point is a trained policy with a varying noise level. Boxes indicate median and IQR.

(IK) strategies in simulation. The results are shown in Table II, rows (a)-(b). In the first case (Guided IK - Lift), we drive the end-effector to a location just above the power line midpoint (where the occlusion is maximum, on average), and then lift the arm up, to push the branches away. In the next case (Guided IK - Sweep), the arm first aligns the gripper to the axis connecting the robot base and tree trunk, then executes a sweeping motion to the left, if power line is on the left of the robot, or to the right otherwise, clearing branches on the way.

As a second baseline, in Table II: (c), we leverage a GNN model [36], where nodes represent branch fork positions and edges denote parent–child cylindrical links. However, unlike their graph-to-graph contact policy, which requires a predefined target deformation state (unknown in our context), we feed all node, edge, and global attributes to an RL policy, using only the representation component of the graph.

Experiment 2: Second, we perform a simulation feature ablation study, with RL policies, to justify our choices. 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., $\sigma = d_{\max}/3$, $d_{\max}$ being the maximum deviation representative of the camera depth inaccuracy. The feature ablation results are shown in Fig. 4 and Table II: (d)-(h). 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 II is an individual policy tested on multiple test sets, listing a variety of additional evaluation metrics, with rows (e)-(j) showing the results applying a median noise $d_{\max} = 0.02$, representative of the ablation study. By contrast, row (k) shows our best model with a lower $d_{\max} = 0.005$, the depth inaccuracy of the D405 camera used for real tests. Furthermore, to justify our choice of RBF kernel, we ablate against two alternatives: the Laplace kernel $k_{\mathrm{L}}(x, x') = \exp(-\|x - x'\|/\gamma)$ and Matérn-3/2 kernel $k_{\mathrm{M}}(x, x') = \big(1 + \sqrt{3}\|x - x'\|/\gamma\big)\exp\big(-\sqrt{3}\|x - x'\|/\gamma\big)$, where, $\gamma > 0$ is a length-scale hyper-parameter. The kernel ablation results are in Table II: (i)-(j).

Experiments 1 & 2 Results: The Table II result metrics are described as follows: Applicable only to RL, Training Reward (**Train Rew**) and Test Reward (**Test Rew**) assess the agent's performance during training and on unseen test scenarios. Test Success Rate (**Test SR %**) quantifies the

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

TABLE III: Representational Efficiency

	$\boldsymbol{GF_t^{(1024)}}$ (secs)	$\boldsymbol{GF_t^{(1)}}$ (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**

TABLE IV: Real Results

Branches	**Attempts**	**Test SR**
Branch 1	9	44.4%
Branch 2	26	42.3%
Branch 3	18	55.5%

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 %**) is 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 power line safety margin δ_{sm}. The metric (**Steps in Succ**) measures the efficiency of successful trajectories; as an example, the final row (k) implies that, 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.

From Table II: (a)-(c), the human-specified trajectories consistently underperformed our learning strategy. Our approach also outperformed the GNN simulation baseline; moreover, the graph method is less transferable to real, as it relies on true branch node positions that must be tracked across time-steps. Predictably, both baselines resulted in significantly higher power line contacts due to the absence of a touch indicator or a contact penalty. The improved Steps in Succ for RL policies indicate that, following successful de-occlusion, the RL agent attempts to stabilise the scene by holding the branches steady, unlike guided swipes.

From the feature ablation study, the plot in Fig. 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. Table. II: (d)-(h) demonstrates that despite the stochasticity of individual policies and high noise injection, progressive addition of feature groups induces improvement across all evaluation metrics. To highlight, we report all success rates without distinguishing between feasible and infeasible tasks, i.e., all occlusion patterns are randomly generated and contain scenarios that cannot be solved by a single arm. In some unsolvable failure instances, the arm cannot generate sufficient torques (due to joint limits) to overcome the collective branch resistance; in others, multiple disconnected branches can occlude the line at locations not closely spaced, only one of which can be removed at a time.

Experiment 3: Third, we compare the representational efficiency of our approach to other point cloud feature extractors, namely, PointNet++ [32] and the Point2Vec [37]. Specifically, we replace the four global point features from Table I, 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 from PointNet++ and Point2Vec. In each case, we use pre-trained checkpoints, down-sample the input, and employ PCA post-feature extraction for consistency with our approach. We train each set for 250 epochs with 1024 environments, tabulating the time taken (**Train Time**) and the accumulated reward (**Train Rew**). 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. $\boldsymbol{GF_t^{(1024)}}$ is indicative of the representation efficiency while training on 1024 environments and $\boldsymbol{GF_t^{(1)}}$ represents the inference performance for one environment. The results are in Table III, where lower is better for the time metrics, such as, $\boldsymbol{GF_t^{(1024)}}$, $\boldsymbol{GF_t^{(1)}}$, and **Train Time** while higher values are better for others.

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

Experiment 4: Finally, we go beyond simulation validation to evaluate our approach in a real-world laboratory setting. We test three 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 (Table IV) and the corresponding videos (supplement & Fig. 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 unactuated end-effector. 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 approach in discovering efficient strategies and adapting to complex scenarios, well beyond what humans

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

can demonstrate.

VI. LIMITATIONS AND FUTURE WORK

A few directions for future work to reduce the sim-to-real gap and improve the success rates are as follows. First, the quality of perception can be enhanced with multiple camera views or with partial-to-complete point cloud models for surface reconstruction. Second, to introduce additional real-world complexities during simulation, integrating with system identification methods [26] that infers parameters governing branch dynamics, may be necessary. Third, external whole-arm tactile sensors [9] could be leveraged instead of our proprioceptive-only model. Finally, our approach could extend beyond agriculture; for instance, to clear deformable cable bundles obstructing inspection cameras in data-centres, reposition flexible ducting during aircraft maintenance, or manipulate soft tissue clusters under occlusion in robotic surgery using whole-arm contact.

REFERENCES

[1] H. Yin, A. Varava, and D. Kragic, "Modeling, learning, perception, and control methods for deformable object manipulation," *Science Robotics*, vol. 6, no. 54, p. eabd8803, 2021.

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

[3] 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*, vol. 7, no. 3, pp. 5819–5826, 2022.

[4] O.-A. Maillard, D. Ryabko, and R. Munos, "Selecting the state-representation in reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 24, 2011.

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

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

[7] Y. Deng and D. Hsu, "General-purpose clothes manipulation with semantic keypoints," *arXiv preprint arXiv:2408.08160*, 2024.

[8] 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*, vol. 32, no. 4, pp. 458–482, 2013.

[9] B. Huang, Y. Wang, X. Yang, Y. Luo, and Y. Li, "3d-vitac: Learning fine-grained manipulation with visuo-tactile sensing," *arXiv preprint arXiv:2410.24091*, 2024.

[10] J. Jacob, S. Cai, P. V. K. Borges, T. Bandyopadhyay, and F. Ramos, "Gentle manipulation of tree branches: A contact-aware policy learning approach," in *8th Annual Conference on Robot Learning*.

[11] L. C. Siebert, L. F. Toledo, P. A. Block, D. B. Bahlke, R. A. Roncolatto, and D. P. Cerqueira, "A survey of applied robotics for tree pruning near overhead power lines," in *Proceedings of the 2014 3rd International Conference on Applied Robotics for the Power Industry*. IEEE, 2014, pp. 1–5.

[12] T. Lowe, S. Lichman, J. Pinskier, C. Sun, and S. Dunstall, "The identification and management of hazard trees to mitigate bushfire risk," 2022.

[13] S. Guggenmoos, "Effects of tree mortality on power line security," *Arboriculture & Urban Forestry (AUF)*, vol. 29, no. 4, pp. 181–196, 2003.

[14] H. Zhou, X. Wang, W. Au, H. Kang, and C. Chen, "Intelligent robots for fruit harvesting: Recent developments and future challenges," *Precision Agriculture*, vol. 23, no. 5, pp. 1856–1907, 2022.

[15] T. Tang and M. Tomizuka, "Track deformable objects from point clouds with structure preserved registration," *The International Journal of Robotics Research*, vol. 41, no. 6, pp. 599–614, 2022.

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

[17] D. Seita, Y. Wang, S. J. Shetty, E. Y. Li, Z. Erickson, and D. Held, "Toolflownet: Robotic manipulation with tools via predicting tool flow from point clouds," in *Conference on Robot Learning*. PMLR, 2023, pp. 1038–1049.

[18] M. Liu, X. Li, Z. Ling, Y. Li, and H. Su, "Frame mining: a free lunch for learning robotic manipulation from 3d point clouds," *arXiv preprint arXiv:2210.07442*, 2022.

[19] X. Ma, D. Hsu, and W. S. Lee, "Learning latent graph dynamics for deformable object manipulation," *arXiv preprint arXiv:2104.12149*, vol. 2, 2021.

[20] S. Yao, S. Pan, M. Bennewitz, and K. Hauser, "Safe leaf manipulation for accurate shape and pose estimation of occluded fruits," *arXiv preprint arXiv:2409.17389*, 2024.

[21] X. Zhang and S. Gupta, "Push past green: Learning to look behind plant foliage by moving it," *arXiv preprint arXiv:2307.03175*, 2023.

[22] S. Yamamoto, S. Hayashi, H. Yoshida, and K. Kobayashi, "Development of a stationary robotic strawberry harvester with a picking mechanism that approaches the target fruit from below," *Japan Agricultural Research Quarterly: JARQ*, vol. 48, no. 3, pp. 261–269, 2014.

[23] 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)*. IEEE, 2021, pp. 2400–2406.

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

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

[26] J. Jacob, T. Bandyopadhyay, J. Williams, P. Borges, and F. Ramos, "Learning to simulate tree-branch dynamics for manipulation," *IEEE Robotics and Automation Letters*, vol. 9, no. 2, pp. 1748–1755, 2024.

[27] 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.

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

[29] F. Bai, Y. Li, J. Chu, T. Chou, R. Zhu, Y. Wen, Y. Yang, and Y. Chen, "Retrieval dexterity: Efficient object retrieval in clutters with dexterous hand," *arXiv preprint arXiv:2502.18423*, 2025.

[30] K. Muandet, K. Fukumizu, B. Sriperumbudur, B. Schölkopf, *et al.*, "Kernel mean embedding of distributions: A review and beyond," *Foundations and Trends® in Machine Learning*, vol. 10, no. 1-2, pp. 1–141, 2017.

[31] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," *Advances in neural information processing systems*, vol. 20, 2007.

[32] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

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

[34] L. Ke, M. Ye, M. Danelljan, Y.-W. Tai, C.-K. Tang, F. Yu, *et al.*, "Segment anything in high quality," *Advances in Neural Information Processing Systems*, vol. 36, pp. 29 914–29 934, 2023.

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

[36] C. H. Kim, M. Lee, O. Kroemer, and G. Kantor, "Towards robotic tree manipulation: Leveraging graph representations," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 11 884–11 890.

[37] K. A. Zeid, J. Schult, A. Hermans, and B. Leibe, "Point2vec for self-supervised representation learning on point clouds," in *DAGM German Conference on Pattern Recognition*. Springer, 2023, pp. 131–146.