DeformX: A Versatile Co-Simulation Framework for Deformable Linear Objects

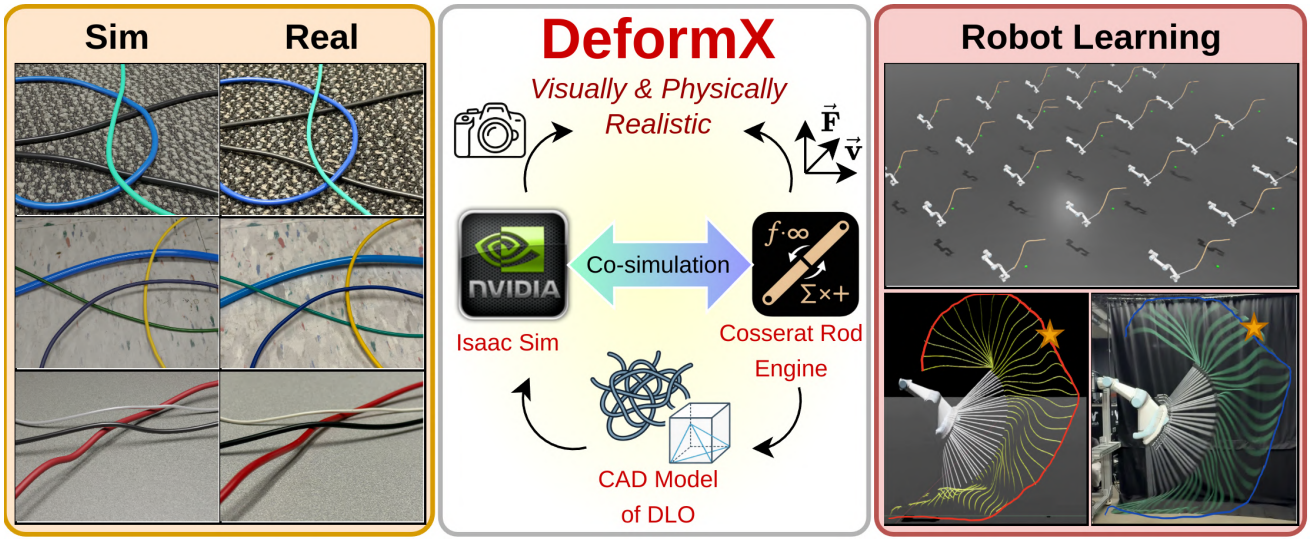

Fig. 1: DeformX is a co-simulation framework for deformable linear objects (DLOs) that integrates NVIDIA Isaac Sim with a dedicated Cosserat rod engine, providing visually and physically realistic simulation. The framework enables scalable data generation and robot learning, supporting DLOs perception and manipulation.

***Abstract*— Deformable linear objects (DLOs) such as wires, cables, and ropes are common in robotic manipulation tasks, yet simulating them with both visual realism and physical accuracy remains challenging. Existing visual simulation methods typically rely on procedural geometric primitives that lack physically grounded deformation behavior, while physics-based approaches with robot learning support often approximate DLOs as rigid-link chains or generic soft bodies, failing to accurately capture the bending, twisting, and shear mechanics of slender elastic structures. In this work, we introduce DeformX, a co-simulation framework that integrates a dedicated Cosserat rod physics engine with NVIDIA Isaac Sim, enabling DLO simulations that are both physically faithful and visually realistic. Our Cosserat rod engine simulates the dynamics and self-collisions of DLOs, and contact interactions with arbitrary free-form meshes. To achieve high-fidelity visualization, we employ mesh skinning to map discrete rod deformations onto imported CAD models. To the best of our knowledge, DeformX is the first framework for DLO simulation that unifies realistic visualization, principled physics, and compatibility with robot learning pipelines. We demonstrate its versatility across synthetic data generation and policy learning for DLO manipulation, and validate visual and physical fidelity through comparisons against real-world experiments. Notably, fine-tuning Segment Anything Model 3 (SAM3) on DeformX-generated data yields a 10.2% mAP@75 improvement in real-image wire segmentation, and a rope-swinging policy trained entirely in DeformX achieves a mean target-hitting error of 6.6 cm on a UR5e manipulator in real-world trials, highlighting its strong sim-to-real transfer capability.**

I. INTRODUCTION

Deformable linear objects (DLOs), including wires, cables, and ropes, are pervasive in robotics and industrial settings. They appear in applications such as wire harness assembly, cable routing, infrastructure inspection, and textile manipulation [1]. Accurate DLO simulation is therefore important for perception, planning, and control: a realistic simulator can enable scalable synthetic data generation for visual learning, support robot learning for manipulation policies, and reduce dependence on expensive and time-consuming real-world experimentation. Despite growing interest in DLO manipulation, existing approaches rarely satisfy three requirements simultaneously: (i) *visual realism* for perception, (ii) *physical fidelity* under gravity, contact, and manipulation, and (iii) *compatibility with robot learning frameworks* for closed-loop training and evaluation. Most prior systems emphasize either visually plausible rendering with simplified dynamics or physically motivated models without an integrated, photorealistic robotics stack, which limits their utility for end-to-end perception and robot learning.

Existing approaches to visually simulating DLOs primarily rely on procedural modeling tools such as Blender, where DLOs are generated as random Bézier curves or as chains of simple geometric primitives like connected cylinders [2], [3]. These methods can produce visually plausible cable shapes under controlled conditions and are often used to generate synthetic datasets for training perception models [2]–[6], given the difficulty of collecting and annotating large-scale real-world wire data. For example, HANDLOOM [2] synthesizes 30,000 cable images for visual learning, but represents cables as random Bézier curves without physically grounded deformation and restricts appearance (e.g., uniformly white cables), limiting both mechanical realism and visual diversity. Similarly, [3] constructs synthetic cables as

connected cylinders and uses collision modifiers to approximate interactions, which increases appearance variation but still relies on highly simplified mechanics. These procedural representations are typically not mesh-based, which prevents direct import of realistic cable CAD assets and reduces geometric fidelity and asset reusability. In addition, because the generated shapes are not governed by physically meaningful rod models, they may appear plausible in static renders yet fail to produce consistent dynamics under contact and self-interaction, conditions that matter for robot manipulation.

On the physics side, common DLO simulators approximate deformable objects as chains of rigid capsules connected by ball joints [7]–[9]. While computationally convenient and widely used in manipulation studies, rigid-link models oversimplify continuum mechanics (e.g., bending–twisting coupling and shear deformation), and often depends on extensive manual tuning of joint stiffness/damping parameters rather than directly interpretable material properties. An alternative is to model DLOs as volumetric soft bodies within general deformable-body frameworks, including finite-element and related methods [10]–[13]. These approaches can express richer material behavior, but are typically inefficient for representing long, slender objects: key rod mechanics arise only implicitly through 3D discretization, computational cost can scale poorly with resolution, and results can be sensitive to meshing choices. In contrast, Cosserat rod theory provides a principled formulation specialized for DLOs by modeling a DLO as a one-dimensional continuum that explicitly captures stretching, shearing, bending, and twisting [14]–[16]. Its dynamics are governed by physically meaningful parameters (e.g., Young's modulus and shear modulus), establishing a clearer link between simulation behavior and real material properties and enabling more accurate and interpretable modeling of DLOs [17]. However, existing Cosserat rod solvers are often used in isolation from photorealistic simulators and robotics middleware. For example, libraries such as PyElastica [18] provide physically realistic rod dynamics, but lack native support for complex interactions with free-form meshes and do not directly provide a high-fidelity rendering and robotics stack, limiting their applicability in robotic manipulation.

These limitations become especially consequential for robot learning with DLOs. DLO tasks remain bottlenecked by data acquisition and sim-to-real discrepancies [9], [19]. Many systems still rely heavily on real-world demonstrations because collecting reliable simulated rollouts with realistic dynamics and contacts is difficult. For instance, [19] reports collecting 1,442 teleoperated demonstration trajectories for cable routing, underscoring the cost of scaling real-world data. Some works attempt to train DLO manipulation policies purely in simulation, such as Iterative Residual Policy (IRP) [8], but their linked-capsule dynamics necessarily simplify elastic behavior, which can widen the sim-to-real gap and push algorithmic burden onto policy design rather than the simulator. Overall, there remains a gap in simulation infrastructure: a unified framework that jointly delivers physically grounded rod dynamics, visually realistic rendering with reusable CAD assets, and seamless integration with robot learning pipelines.

To address this gap, we propose **DeformX**, a co-simulation framework that couples a discrete Cosserat rod engine with Isaac Sim to achieve both physically and visually realistic DLO simulation for perception and robot learning. DeformX combines (i) physically interpretable Cosserat rod dynamics, (ii) realistic interactions with complex environments via mesh-based contact, and (iii) high-fidelity visualization and sensor simulation through CAD-compatible mesh skinning and Isaac Sim rendering. Our main contributions are:

- **DeformX Framework:** A co-simulation system integrating a Cosserat rod physics engine with Isaac Sim to produce visually and physically realistic DLO simulations suitable for perception and robot learning.
- **Free-Form Mesh Contact:** Support for DLO interactions with free-form meshes within the Cosserat rod engine, enabling realistic rod-object contact in cluttered environments.
- **Mesh-Skinned Visualization with CAD Support:** We propose to utilize mesh skinning to map discrete Cosserat rod deformations onto meshes, allowing import of DLO CAD assets for high-fidelity visualization.
- **WireSeg-32k Dataset:** A synthetic wire instance segmentation dataset containing diverse RGB images, depth maps, and ground-truth annotations. Fine-tuning SAM3 on our dataset yields a 10.2% improvement in mAP@75 on real-world experiments.

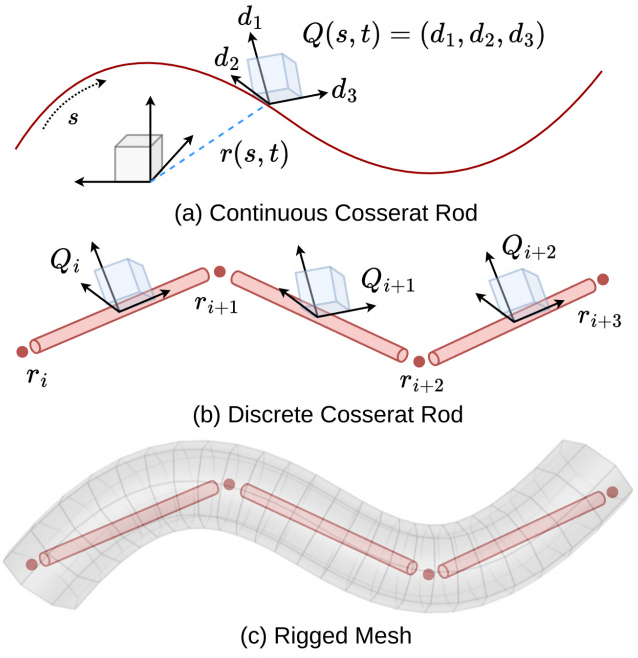

Fig. 2: Cosserat rod modeling of DLOs. (a) Continuous Cosserat rod representation; (b) Discrete Cosserat rod representation; (c) Mesh skinned to the discrete Cosserat rod for realistic visualization.

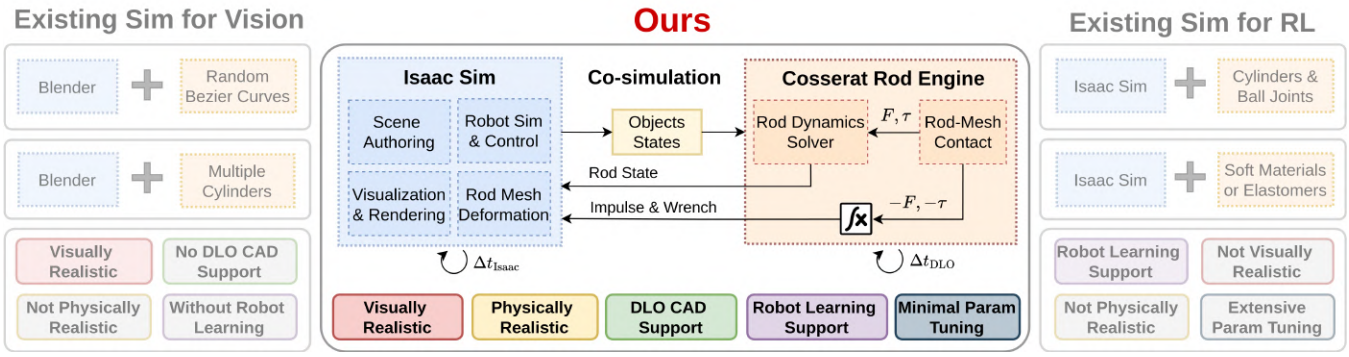

Fig. 3: Overview of our co-simulation framework. Compared to existing simulators for vision (left) and RL (right), our framework jointly achieves visual realism, physical accuracy, DLO CAD support, and robot learning support with minimal parameter tuning by combining Isaac Sim and a dedicated Cosserat rod engine.

II. NOTATIONS AND PRELIMINARY

We model a deformable linear object (DLO) as a slender elastic rod using the Cosserat rod theory [20], [21], which captures all deformation modes of a 1D continuum including stretching, shearing, bending, and twisting.

In the continuous representation (Fig. 2a), the rod is parameterized by arc-length s and time t, with centerline $\mathbf{r}(s,t) \in \mathbb{R}^3$ and an orthonormal material frame $\mathbf{Q}(s,t) = \{\mathbf{d}_1(s,t), \mathbf{d}_2(s,t), \mathbf{d}_3(s,t)\} \in \mathrm{SO}(3)$ attached to each cross-section, where the directors $\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3$ are the columns of $\mathbf{Q}$.

In the discrete representation (Fig. 2b), the centerline is sampled into vertices $\{\mathbf{r}_i(t)\}_{i=0}^{n}$ and orientations are stored as segment frames $\{\mathbf{Q}_j(t)\}_{j=0}^{n-1}$ associated with edges $\boldsymbol{\ell}_j = \mathbf{r}_{j+1} - \mathbf{r}_j$, yielding a finite-dimensional state $\{\mathbf{r}_i, \mathbf{Q}_j\}$ [21]–[23]. Time evolution is advanced numerically by discretizing t into steps $t^k = k\Delta t$, so all state variables are updated from $(\mathbf{r}_i^k, \mathbf{Q}_j^k)$ to $(\mathbf{r}_i^{k+1}, \mathbf{Q}_j^{k+1})$ at each timestep Δt [18].

Cosserat rod dynamics are computed by conservation of linear and angular momentum along the rod. Material properties are incorporated through linear elastic constitutive laws, where Young's modulus E and shear modulus G determine the rod stiffness in stretching/shearing and bending/twisting [14]. External interactions are incorporated through forcing and boundary-condition terms, including gravity, endpoint loads/torques, and contact with friction.

III. METHODS

A. Framework Overview

We propose a co-simulation framework that tightly integrates a Cosserat rod engine with Isaac Sim to achieve physically accurate and visually realistic simulation of DLOs. The core challenge lies in bridging two systems with fundamentally different functionalities and time scales. To address this, we design a multi-rate scheme that handles state updates, cross-engine interactions, and ensures stable bidirectional coupling between the Cosserat rod engine and Isaac Sim. The proposed framework enables simulation of DLOs that is both visually and physically realistic, facilitating downstream visual perception learning and robot learning tasks. An overview of our framework is illustrated in Fig. 3.

Algorithm 1 Multi-rate co-simulation

Require: Macro step Δt_{Isaac}, substeps N, with $\Delta t_{\text{DLO}} = \Delta t_{\text{Isaac}}/N$

for each Isaac step k **do**
  Get objects states from Isaac Sim (pose/velocity/acceleration)
  for $n = 1$ to N **do**
    Estimate motions of interacting objects
    Integrate DLO dynamics for Δt_{DLO} (Cosserat rod + contact)
  end for
  Export DLO skeleton poses for rendering
  Return reaction wrenches/impulses to Isaac Sim
  Isaac Sim advances rigid-body + control for Δt_{Isaac}
end for

B. Co-simulation Framework

Division of Functionality: Our framework partitions functionalities between the Cosserat rod engine and Isaac Sim while maintaining tightly coupled interaction. The Cosserat rod engine computes all DLO dynamics and contacts between DLOs and other objects, ensuring physically consistent deformation and contact responses, while Isaac Sim handles the rigid-body environment, including robot manipulators, fixtures, and other scene elements, providing rigid-body simulation, scene management, control interfaces, and photorealistic rendering. This division ensures that robot dynamics and non-DLO interactions remain within Isaac Sim's domain, while deformation physics and rod-driven contacts are governed by the specialized rod engine. As a result, our framework preserves high physical fidelity for deformable objects while leveraging the robustness and scalability of a mature rigid-body simulation platform.

Multi-rate Co-simulation: A fundamental challenge in our co-simulation framework stems from the mismatch in time scales between the Cosserat rod engine and Isaac Sim modules. Stable DLO dynamics computation typically necessitates finer time steps, $\Delta t_{\text{DLO}} \sim 10^{-5}$ s, whereas Isaac Sim commonly advances rigid-body dynamics and control

at coarser time steps, $\Delta t_{\text{Isaac}} \sim 10^{-2}$ s. Consequently, the DLO states are updated multiple times within a single Isaac Sim step, during which no direct interaction between these two modules is available.

To address this issue, we replicate the semi-implicit Euler integration used by Isaac Sim PhysX within the Cosserat rod engine. This approach enables us to estimate the motion of interacting rigid bodies (e.g., robot grippers or end-effectors) during fine-grained DLO updates, using only the states provided by Isaac Sim at coarse-grained time steps even when intermediate inputs from Isaac Sim are unavailable.

Specifically, we represent the pose of an interacting rigid body as a rigid transform $T \in SE(3)$ and parameterize incremental motion using a 6D minimal coordinate vector $\boldsymbol{\xi} \in \mathbb{R}^6$, where $\boldsymbol{\xi} = [\mathbf{x}; \boldsymbol{\theta}]$ stacks translation $\mathbf{x} \in \mathbb{R}^3$ and a 3D rotation coordinate $\boldsymbol{\theta} \in \mathbb{R}^3 \cong \mathfrak{so}(3)$. The corresponding generalized velocity and acceleration are denoted by $\dot{\boldsymbol{\xi}}$ and $\ddot{\boldsymbol{\xi}}$, respectively. At each time step Δt, we first update the generalized velocity using semi-implicit Euler:

$$\dot{\boldsymbol{\xi}}_{n+1} = \dot{\boldsymbol{\xi}}_n + \ddot{\boldsymbol{\xi}}_n \Delta t, \tag{1}$$

and then update the pose by composing the current transform with the incremental motion on $\mathrm{SE}(3)$ induced by the updated velocity:

$$T_{n+1} = T_n \,\mathrm{Exp}(\dot{\boldsymbol{\xi}}_{n+1} \Delta t), \tag{2}$$

where $\mathrm{Exp}(\cdot)$ denotes the exponential map that converts a 6D motion vector in $\mathbb{R}^6 \cong \mathfrak{se}(3)$ into an element of $\mathrm{SE}(3)$.

By replicating Isaac Sim's integration scheme within the rod engine, we ensure consistent state evolution across simulation modules during multi-rate updates, while allowing contact interactions to be evaluated against continuously updated rigid-body motion at the DLO time scale. These contact interactions must also be propagated back to Isaac Sim. To this end, we accumulate contact forces computed at the DLO time scale into integrated impulses or wrenches, which are then fed back to the corresponding rigid bodies in Isaac Sim at each coarse simulation step. This strategy enables stable and physically consistent bidirectional coupling across disparate time resolutions.

Modular Interface: We have implemented the Cosserat rod engine as a Python module that integrates directly into Isaac Sim's scripting environment, supporting both interactive UI-based workflows and headless execution. Users can construct scenes within Isaac Sim, after which geometry, poses, and simulation parameters are exported as initial conditions for the rod engine. The same interface is reused for large-scale dataset generation and robot learning, providing a unified pipeline from scene authoring to physics simulation and photorealistic rendering.

C. Interaction with Free-Form Meshes

For DLO simulation in robotic settings, interaction with surrounding objects is essential. However, current PyElastica and other Cosserat rod libraries [18], [24], [25] lack native support for contact with free-form meshes. Building on PyElastica's penalty-based contact formulation, we compute the closest-point distance d between rod nodes $\{\mathbf{r}_i\}$ and the mesh surface, and apply a repulsive normal contact force parameterized by stiffness k and dissipation ν. The Cosserat rod engine applies the resulting contact forces to the rod internally, and returns the equal-and-opposite reaction impulses/wrenches to the mesh, where the corresponding rigid body dynamics are resolved [24].

To improve computational efficiency, we prebuild a Bounding Volume Hierarchy (BVH) over the mesh and employ AABB-based broad-phase pruning to reduce distance queries. To enhance robustness under larger time steps, we introduce a repulsion distance for watertight meshes to prevent deep penetration and keep the rod outside the surface.

D. DLO Visualization in Isaac Sim

For realistic visualization in Isaac Sim, we represent each DLO with a smooth tubular mesh that is skinned to the discrete Cosserat rod vertices. In this way, the motion of the Cosserat rod smoothly deforms the mesh. During simulation, the rod dynamics solver outputs the rod centerline and local orientation for each element, and at every Isaac Sim step we update the tubular mesh accordingly using these rod states.

Mesh skinning is a technique widely used in animation and character modeling [26]. We incorporate this technique with a discretized Cosserat rod model to achieve high visual fidelity in Isaac Sim while maintaining full consistency with the underlying rod dynamics, as illustrated in (Fig. 2c). Moreover, because DLOs are visualized using skinned meshes, we can directly import CAD models of DLOs with pre-defined skins. This allows us to leverage a wide range of existing DLO CAD assets to create diverse and highly realistic DLO representations.

IV. Experiment

A. Validation of DLO Physics Simulation

To validate physical fidelity, we perform a set of physics sanity checks and quantitative comparisons against observed DLO motion. Unlike Isaac Sim's built-in DLO approximations, our Cosserat rod model explicitly captures stretching, shearing, bending, and twisting, enabling characteristic behaviors such as gravity-induced equilibria, curvature propagation, and torsion-bending coupling. We validate the physical fidelity of our simulation through two real-world experiments:

(1) Knot deformation under twist: We design the first experiment to assess the simulator's ability to reproduce realistic quasi-static deformations given real-world measurements. The physics-based parameters in our simulator are computed from factory-provided material specifications [27]. We tie the DLO into a simple (trefoil) knot and apply controlled twisting [22]. As shown in Fig. 4(a), the simulated knot undergoes the same qualitative shape transitions as those observed in real world.

(2) Free-form contact modeling: We evaluate our implementation of contact handling by wrapping a flexible rope (scarf) around a rigid bunny model. Starting from an initially draped configuration, we progressively pull and slide the

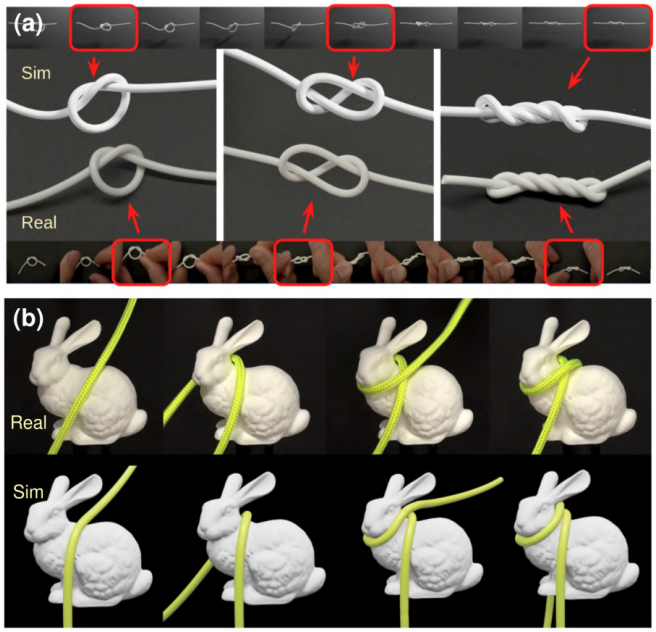

Fig. 4: Sim2real comparison for DLOs. (a) We tie a trefoil knot on a wire, apply continuous twists to induce shape transitions, and qualitatively compare the representative configurations between simulation (top) and real experiments (bottom). (b) shows flexible rope (scarf) wrapping around a bunny with multi-point contact, sliding, and self-contact.

scarf to induce multi-point, sliding contact and self-contact. Fig. 4 visualizes representative snapshots, showing that the simulated scarf conforms to the bunny geometry, preserves stable contact without interpenetration, and produces realistic winding and settling behavior under gravity and friction.

For both experiments, our simulator yields physically consistent DLO behavior, supporting its use for realistic data generation and sim-to-real analysis.

B. Applications: Data Generation

Overview: A key application of a visually and physically realistic DLO simulator is to generate high-quality synthetic data for training perception models without expensive manual annotation. Wire instance segmentation is fundamental to many tasks, yet large-scale annotated datasets are scarce due to the difficulty of labeling thin, deformable objects in cluttered scenes. With our co-simulation framework, we construct *WireSeg-32k*, a wire instance segmentation dataset designed with three principles: (1) high diversity in configurations, (2) a broad spectrum of difficulty levels for targeted training and evaluation, and (3) scenario-specific categories aligned with real-world use cases. A comparison between our dataset and existing datasets is shown in Table I.

Data Generation Pipeline: The data generation pipeline is illustrated in Fig. 5. We assemble a library of base scenes from publicly available 3D assets and then import the resulting geometry and materials into Isaac Sim. For each scene instance, we procedurally generate wire configurations (e.g., number of wires, length, radius, and initial shapes)

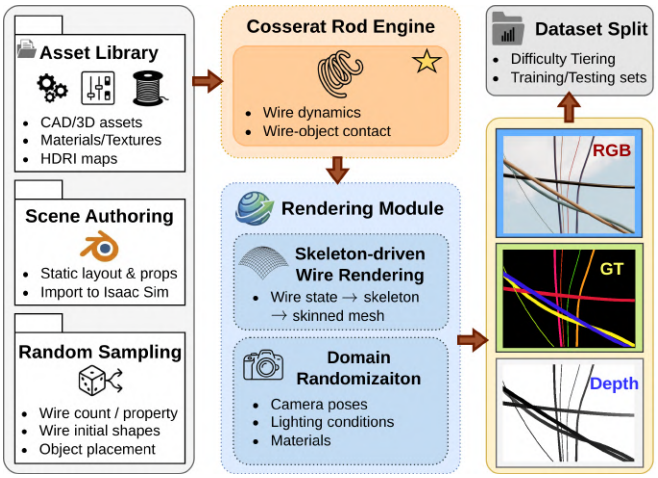

Fig. 5: Synthetic wire segmentation dataset generation pipeline. Randomized scenes are built from asset libraries. Wires are simulated with a Cosserat rod engine, and visualized as skinned meshes. RGB images, segmentation masks, and depth maps are generated and split by difficulty levels.

Dataset	Physics Realism	Instance Label	CAD Support	Visually-grounded Scenes	Num. Images
HANDLOOM [2]	×	×	×	×	30k
FASTDLO [4]	×	✓	×	×	32k
Fresnillo et al. [3]	✓	×	×	✓	25k
Zanella et al. [5]	×	×	×	×	28.5k
ISCUTE [6]	×	✓	✓	✓	28k
WireSeg-32k (Ours)	✓	✓	✓	✓	32k

TABLE I: Binary comparison between existing DLO datasets and our WireSeg-32k dataset. "Physics realism" indicates whether data generation uses physics-based simulation of DLO deformation (e.g., gravity, wind, and collisions). "Instance label" indicates whether per-object instance masks are provided (trace-only supervision is counted as ×). "CAD support" indicates whether DLOs can be represented as free-form meshes (enabling direct CAD import). "Visually grounded scenes" indicates whether cables are rendered within realistic, image-based backgrounds rather than on plain or purely synthetic backdrops.

and sample rigid objects/assets and their placements in Isaac Sim. During data generation, Isaac Sim simulates the robot and all non-wire rigid bodies, while our Cosserat Rod Engine solves the wire dynamics and computes wire-object interaction forces that are exchanged with Isaac Sim through the coupling interface. The simulated wire state is streamed to Isaac Sim for skeleton-driven rendering, together with the static scene geometry and asset materials. We further randomize camera viewpoints, lighting conditions, and scene textures to increase visual diversity and improve robustness to domain shifts. Sample images from our dataset is shown in Fig. 6.

Categories: We organize the dataset into three scenario categories: (1) **wire-on-plane**, a canonical tabletop setup; (2) **flying wires**, focusing on gravity-driven dangling dynamics; and (3) **data center**, featuring cables arranged within data-

Model	Hard (Syn)		Medium (Syn)		Easy (Syn)		Total (Syn)		Total (Real)	
	F1@75	mAP@75	F1@75	mAP@75	F1@75	mAP@75	F1@75	mAP@75	F1@75	mAP@75
SAM3 (Base)	0.179	0.066	0.446	0.310	0.803	0.735	0.409	0.290	0.296	0.157
SAM3 + LoRA	0.225	0.102	0.512	0.404	0.850	0.816	0.465	0.365	0.314	0.173
Δ (LoRA–Base)	+25.7%	+54.5%	+14.8%	+30.3%	+5.9%	+11.0%	+13.7%	+25.7%	+6.1%	+10.2%

TABLE II: SAM3 performance with and without LoRA fine-tuning. For each subset, we report dataset-level F1@75 and COCO mAP@75 (higher is better) on the Easy/Medium/Hard tiers, the full synthetic set, and a held-out real test set. The last row reports the percent improvement from LoRA fine-tuning.

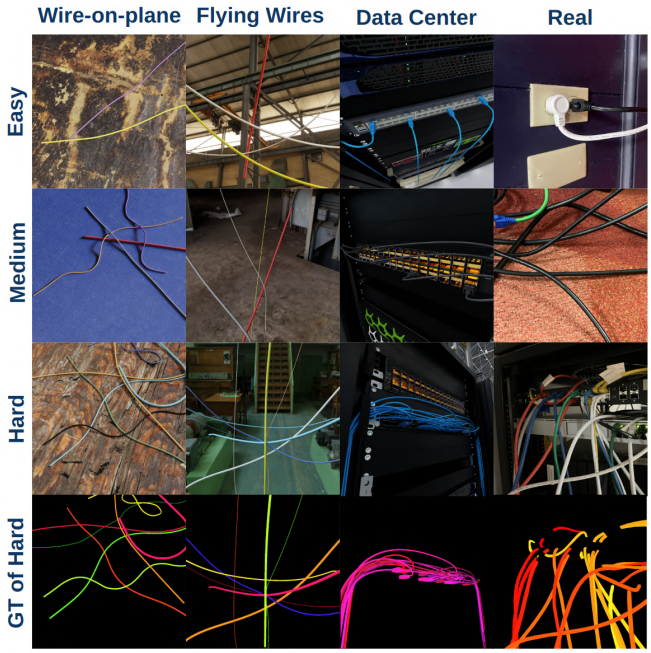

Fig. 6: Representative images from three settings across Easy/Medium/Hard, with Hard ground-truth instance masks.

center racks, using photorealistic assets from NVIDIA Omniverse [28]. In total, we collect 32,000 rendered images from 300+ independent simulation runs.

Annotations, Depth, and Difficulty: We provide per-wire instance masks for multi-wire segmentation, along with per-pixel depth maps to support high-resolution 3D perception beyond typical RGB-D limitations on thin cables [29], [30]. We also stratify the dataset into easy/medium/hard splits by thresholding its per-image AP@75 with SAM3 performance on generated images (Fig. 6).Images with AP@75 < 0.3 are labeled hard, those with AP@75 > 0.6 are labeled easy, and all others are labeled medium.

Complementary Real-World Test Set: To support real-world evaluation, we additionally create a separate test set of 300 in-the-wild images with manually annotated ground truth segmentations. Images are collected across diverse backgrounds, lighting conditions, and cable types to reflect real deployment variation.

Model Fine-tuning: To demonstrate that our dataset can directly support training wire instance segmentation models, we perform a simple LoRA fine-tuning experiment on SAM3. We adapt the vision encoder and mask decoder by inserting lightweight LoRA adapters into the attention projections (`q_proj`, `k_proj`, `v_proj`; rank 16, $\alpha = 32$, dropout 0.05), while keeping all other components frozen to maintain parameter efficiency and training stability. We train for 5 epochs on our synthetic wire dataset using a learning rate of 1×10^{-5} and an effective batch size of 16.

As a strong foundation-model baseline, we also evaluate off-the-shelf SAM3 on our dataset in text-prompt mode using the prompt `''cable''`. We report dataset-level F1@75 (instance F1 with one-to-one matching at IoU ≥ 0.75) and mAP@75 (COCO-style AP at IoU $= 0.75$). Across both the synthetic benchmark and our held-out real-world test set, LoRA fine-tuning improves performance over the off-the-shelf baseline (Table II).

C. Applications: Robot Learning

Motivation: Dynamic rope swinging amplifies modeling errors: small inaccuracies in bending/torsion and contact can cause large deviations in the rope-tip trajectory, leading to a pronounced sim-to-real gap in fast "whipping" regimes. We therefore use a planar hit-target rope-swinging benchmark inspired by Chi *et al.* [8] as a stress test to quantify whether improved DLO physics yields more reliable sim-to-real policy transfer.

Task and Metric: A UR5e robotic arm swings a rope so that its tip passes as close as possible to a fixed target point. Following [8], we evaluate performance using the minimum tip-to-goal distance over a rollout,

$$d_{\min} = \min_{t \in [0,T]} \|\mathbf{p}_{\text{tip}}(t) - \mathbf{p}_{\text{goal}}\|_2 . \tag{3}$$

We parameterize the robot motion with a low-dimensional joint-space action

$$a_t = (\Delta J_1, \Delta J_2, \Delta J_3), \tag{4}$$

where ΔJ_1–ΔJ_3 denote bounded per-step increments for the three actuated UR5e joints. These increments are accumulated into joint position targets at each control step. **Planar Motion Constraint:** We constrain the robot motion to be planar, and define the target location in-plane. This restriction is sufficient for the hit-target task, since out-of-plane targets can be reached by a trivial rotation of the robot base; we therefore fix the base rotation and focus on planar whipping dynamics.

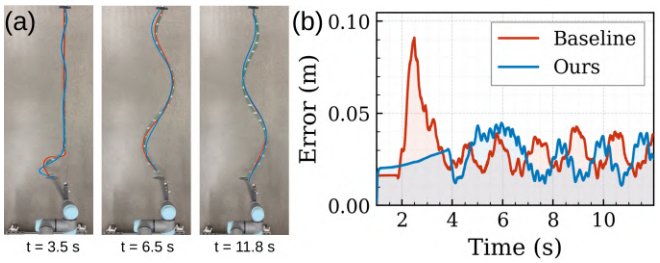

Fig. 7: (a) Robot-driven rope experiment with one end anchored to the ground and the other attached to the robot end-effector, which executes a sinusoidal trajectory; overlays are shown with time stamps. (b) Trajectory error over time, as defined in Eq. 5.

Parameter Settings of Simulated Ropes: The choice of rope parameters can significantly affect experimental outcomes. To ensure a fair comparison between DeformX and the baseline simulator, we calibrate the rope parameters using a robot-driven rope motion experiment. Specifically, we attach a 2 m rope to a UR5e robot arm and command a sinusoidal end-effector trajectory (1.2 Hz frequency and 0.2 m amplitude), while tracking the rope in 3D using a motion-capture system. The identical boundary motion is then replayed in simulation. We tune the rope parameters so that the trajectory error between simulated and real rope configurations is minimized for both simulators:

$$Error(t) = \frac{1}{N}\sum_{i=1}^{N} \left\| \mathbf{x}_i^{\text{sim}}(t) - \mathbf{x}_i^{\text{real}}(t) \right\|_2, \qquad (5)$$

where N is the number of motion-capture markers placed along the rope (here $N = 20$).

The resulting trajectory errors are shown in Fig. 7. The calibrated rope parameters produce small and comparable errors for both simulators. These parameters are subsequently used for RL training.

Controlled Comparison and Training: To isolate the effect of the DLO simulator, we keep the PPO algorithm, observations/actions, reward, training budget, and all non-DLO task components fixed, and vary only the DLO backend: (i) DeformX and (ii) an Isaac Sim linked-capsule baseline. PPO is trained with identical rollout, parallelization, and state/goal inputs for both backends. The policy controls a planar 3-DoF subset of UR5e joints via bounded incremental position commands, with the same action space and command limits for both backends.

Sim-to-real Evaluation: We deploy the learned open-loop motion on a real UR5e and track the rope tip using an OptiTrack system. For a fixed goal, we repeat the same execution n=10 times and report the mean and standard deviation of $d_{\min}$. Representative rollouts are shown in Fig. 8, and results are summarized in Table III. While both simulators obtain low in-simulation errors, only DeformX transfers reliably to the real robot. Across all three targets, DeformX reduces the real-world minimum tip-to-goal distance from 15.1/25.9/30.4 cm to 6.6/7.3/5.8 cm, respectively. Since both simulators are calibrated on the same robot-driven rope experiment and all PPO settings are kept fixed, this result indicates that physically accurate modeling of bending, torsion, and contact is critical for sim-to-real transfer in dynamic rope swinging.

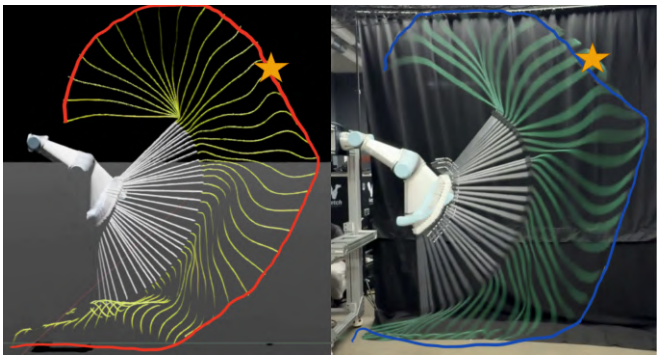

Fig. 8: Goal-conditioned dynamic manipulation of a DLO. Our RL policy reaches the goal. Left: simulation rollout. Right: real-world rollout. The goal location is marked by a star. Colored curves visualize the DLO configuration over time during a rollout.

Target Point	Method	Sim $d_{\min}$	Real (n=10) $d_{\min}$
(0, 200, 230)	Baseline	4.9	15.1 ± 6.1
	DeformX (Ours)	4.2	**6.6** $\pm$ 4.7
(0, 200, 150)	Baseline	4.4	25.9 ± 8.9
	DeformX (Ours)	1.4	**7.3** $\pm$ 1.2
(0, 170, 50)	Baseline	4.3	30.4 ± 14.3
	DeformX (Ours)	3.3	**5.8** $\pm$ 3.2

TABLE III: Hit-target task results. $d_{\min}$ (unit: cm) is defined in Eq. (3) and reported in centimeters. Real-world results are computed over n=10 repeated executions for the same goal; we report mean and standard deviation.

V. Conclusion and Discussion

We introduced *DeformX*, a co-simulation framework that integrates a dedicated Cosserat rod engine with NVIDIA Isaac Sim to enable DLO simulation that is both physically faithful and visually realistic, while remaining compatible with scalable data generation and robot learning. Our framework supports contact interactions with free-form meshes, multi-rate coupling with rigid-body dynamics, and CAD-quality visualization through mesh skinning. Leveraging *DeformX*, we further constructed *WireSeg-32k*, a wire instance segmentation dataset that provides RGB images, depth maps, and ground-truth annotations.

Despite these advantages, there are several limitations. First, our current Cosserat rod engine is implemented entirely on the CPU, which limits computational efficiency and scalability. Second, in the present multi-rate co-simulation scheme, the influence of the DLO on contacting rigid bodies is approximated by integrating contact forces into a single impulse or wrench within each Isaac Sim time step. This approximation may introduce inaccuracies in the bidirectional coupling between deformable and rigid objects. Additionally,

recent advances in stable Cosserat rod formulations [25] propose a split position-rotation optimization scheme with a closed-form Gauss-Seidel quasi-static orientation update, achieving high accuracy under large stiffness parameters while maintaining stability with larger time steps. Such methods also support GPU parallelization. Incorporating these techniques would not only improve computational efficiency but also potentially eliminate the time-scale discrepancy between Isaac Sim and the rod engine.

Future work will therefore focus on upgrading *DeformX* to adopt a stable Cosserat rod formulation, aiming to fundamentally resolve time-scale discrepancy issue while leveraging GPU acceleration for improved performance and scalability.

VI. Acknowledgment

Generative AI tools were used for language refinement and code assistance. All content was reviewed, validated, and integrated by the authors. The authors take full responsibility for the originality and correctness of this work.

References

[1] J. Sanchez, J.-A. Corrales, B.-C. Bouzgarrou, and Y. Mezouar, "Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey," *The International Journal of Robotics Research*, vol. 37, no. 7, pp. 688–716, 2018. [Online]. Available: https://doi.org/10.1177/0278364918779698

[2] V. Viswanath, K. Shivakumar, M. Parulekar, J. Ajmera, J. Kerr, J. Ichnowski, R. Cheng, T. Kollar, and K. Goldberg, "Handloom: Learned tracing of one-dimensional objects for inspection and manipulation," in *Proceedings of The 7th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Tan, M. Toussaint, and K. Darvish, Eds., vol. 229. PMLR, 06–09 Nov 2023, pp. 341–357. [Online]. Available: https://proceedings.mlr.press/v229/viswanath23a.html

[3] P. MalvidoFresnillo, W. M. Mohammed, S. Vasudevan, J. A. PerezGarcia, and J. L. MartinezLastra, "Generation of realistic synthetic cable images to train deep learning segmentation models," *Machine Vision and Applications*, vol. 35, no. 4, p. 84, Jun 2024. [Online]. Available: https://doi.org/10.1007/s00138-024-01562-y

[4] A. Caporali, K. Galassi, R. Zanella, and G. Palli, "Fastdlo: Fast deformable linear objects instance segmentation," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9075–9082, 2022.

[5] R. Zanella, A. Caporali, K. Tadaka, D. D. Gregorio, and G. Palli, "Auto-generated wires dataset for semantic segmentation with domain-independence," *2021 International Conference on Computer, Control and Robotics (ICCCR)*, pp. 292–298, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:231915434

[6] S. Kozlovsky, O. Joglekar, and D. D. Castro, "Iscute: Instance segmentation of cables using text embedding," 2024. [Online]. Available: https://arxiv.org/abs/2402.11996

[7] S. Zhaole, J. Zhu, and R. B. Fisher, "Dexdlo: Learning goal-conditioned dexterous policy for dynamic manipulation of deformable linear objects," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 16 009–16 015.

[8] C. Chi, B. Burchfiel, E. Cousineau, S. Feng, and S. Song, "Iterative residual policy: for goal-conditioned dynamic manipulation of deformable objects," 2022. [Online]. Available: https://arxiv.org/abs/2203.00663

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

[10] P. Kaufmann, S. Martin, M. Botsch, and M. Gross, "Flexible simulation of deformable models using discontinuous galerkin fem," *Graphical Models*, vol. 71, no. 4, pp. 153–167, 2009.

[11] Q. Wang, H. Fang, N. Li, D. C. Weggel, and G. Wen, "An efficient fe model of slender members for crash analysis of cable barriers," *Engineering structures*, vol. 52, pp. 240–256, 2013.

[12] N. Lv, J. Liu, X. Ding, J. Liu, H. Lin, and J. Ma, "Physically based real-time interactive assembly simulation of cable harness," *Journal of Manufacturing Systems*, vol. 43, pp. 385–399, 2017.

[13] P. P. Valentini and E. Pennestrì, "Modeling elastic beams using dynamic splines," *Multibody system dynamics*, vol. 25, no. 3, pp. 271–284, 2011.

[14] D. Cao and R. W. Tucker, "Nonlinear dynamics of elastic rods using the cosserat theory: Modelling and simulation," *International Journal of Solids and Structures*, vol. 45, no. 2, pp. 460–477, 2008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020768307003253

[15] J. Linn and K. Dreßler, "Discrete cosserat rod models based on the difference geometry of framed curves for interactive simulation of flexible cables," in *Math for the Digital Factory*. Springer, 2017, pp. 289–319.

[16] F. Renda, F. Boyer, J. Dias, and L. Seneviratne, "Discrete cosserat approach for multisection soft manipulator dynamics," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1518–1533, 2018.

[17] N. Lv, J. Liu, H. Xia, J. Ma, and X. Yang, "A review of techniques for modeling flexible cables," *Computer-Aided Design*, vol. 122, p. 102826, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010448520300191

[18] N. Naughton, J. Sun, A. Tekinalp, T. Parthasarathy, G. Chowdhary, and M. Gazzola, "Elastica: A compliant mechanics environment for soft robotic control," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3389–3396, 2021.

[19] J. Luo, C. Xu, X. Geng, G. Feng, K. Fang, L. Tan, S. Schaal, and S. Levine, "Multistage cable routing through hierarchical imitation learning," *IEEE Transactions on Robotics*, vol. 40, pp. 1476–1491, 2024.

[20] M. Renardy, "Nonlinear problems of elasticity (stuart s. antman)," *SIAM Review*, vol. 37, no. 4, pp. 637–637, 1995. [Online]. Available: https://doi.org/10.1137/1037152

[21] M. Gazzola, L. H. Dudte, A. G. McCormick, and L. Mahadevan, "Forward and inverse problems in the mechanics of soft filaments," *Royal Society Open Science*, vol. 5, no. 6, p. 171628, 06 2018. [Online]. Available: https://doi.org/10.1098/rsos.171628

[22] M. Bergou, M. Wardetzky, S. Robinson, B. Audoly, and E. Grinspun, "Discrete elastic rods," in *ACM SIGGRAPH 2008 Papers*, ser. SIGGRAPH '08. New York, NY, USA: Association for Computing Machinery, 2008. [Online]. Available: https://doi.org/10.1145/1399504.1360662

[23] Gazzola Lab, "Elastica: Cosserat rods," Website, developed and maintained by the Gazzola Lab at the University of Illinois at Urbana-Champaign. [Online]. Available: https://www.cosseratrods.org/

[24] A. Tekinalp, S. H. Kim, Y. Bhosale, T. Parthasarathy, N. Naughton, A. Albazroun, R. Joon, S. Cui, I. Nasiriziba, M. Stölzle, C.-H. C. Shih, and M. Gazzola, "Gazzolalab/pyelastica," 2024. [Online]. Available: https://doi.org/10.5281/zenodo.7658871

[25] J. Hsu, T. Wang, K. Wu, and C. Yuksel, "Stable cosserat rods," in *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, ser. SIGGRAPH Conference Papers '25. New York, NY, USA: Association for Computing Machinery, 2025. [Online]. Available: https://doi.org/10.1145/3721238.3730618

[26] D. L. James and C. D. Twigg, "Skinning mesh animations," *ACM Transactions on Graphics (TOG)*, vol. 24, no. 3, pp. 399–407, 2005.

[27] COEX 3D, "White coexflex 60a tpu," accessed: 2026-03-05. [Online]. Available: https://coex3d.com/products/white-coexflex-60a-tpu

[28] NVIDIA Corporation, "Data center assets pack," Online documentation, NVIDIA Omniverse, accessed: 2026-03-06. [Online]. Available: https://docs.omniverse.nvidia.com/usd/latest/usd_content_samples/downloadable_packs.html

[29] J. Xiang, H. Dinkel, H. Zhao, N. Gao, B. Coltin, T. Smith, and T. Bretl, "Trackdlo: Tracking deformable linear objects under occlusion with motion coherence," *IEEE Robotics and Automation Letters*, vol. 8, no. 10, pp. 6179–6186, 2023.

[30] H. Lin, S. Chen, J. Liew, D. Y. Chen, Z. Li, G. Shi, J. Feng, and B. Kang, "Depth anything 3: Recovering the visual space from any views," 2025. [Online]. Available: https://arxiv.org/abs/2511.10647