2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

Robots of the Lost Arc: Self-Supervised Learning to Dynamically Manipulate Fixed-Endpoint Cables

Harry Zhang, Jeffrey Ichnowski, Daniel Seita, Jonathan Wang, Huang Huang, and Ken Goldberg[1]

2021 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9077-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICRA48506.2021.9561630

***Abstract*— We explore how high-speed robot arm motions can dynamically manipulate ropes and cables to vault over obstacles, knock objects from pedestals, and weave between obstacles. In this paper, we propose a self-supervised learning framework that enables a UR5 robot to perform these three tasks. The framework finds a 3D apex point for the robot arm, which, together with a task-specific trajectory function, defines an arcing motion that dynamically manipulates the cable to perform a task with varying obstacle and target locations. The trajectory function computes minimum-jerk motions that are constrained to remain within joint limits and to travel through the 3D apex point by repeatedly solving quadratic programs to find the shortest and fastest feasible motion. We experiment with 5 physical cables with different thickness and mass and compare performance against two baselines in which a human chooses the apex point. Results suggest that a baseline with a fixed apex across the three tasks achieves respective success rates of 51.7 %, 36.7 %, and 15.0 %, and a baseline with human-specified, task-specific apex points achieves 66.7 %, 56.7 %, and 15.0 % success rate respectively, while the robot using the learned apex point can achieve success rates of 81.7 % in vaulting, 65.0 % in knocking, and 60.0 % in weaving. Code, data, and supplementary materials are available at `https://sites.google.com/berkeley.edu/dynrope/home`.**

I. Introduction

Dynamic manipulation and management of linear deformable objects such as ropes, chains, vacuum cords, charger cables, power cables, tethers, and dog leashes are common in daily life. For example, a person vacuuming may find that the vacuum power cord is stuck on a chair, and use dynamic manipulation to "vault" the cord over the chair. If the first motion does not succeed, the human can try again, adapting their motion to the failure. In this paper, we explore how a robot can teach itself to perform three tasks:

Task 1: Vaulting The robot dynamically manipulates a cable to move from one side of an obstacle to another.
Task 2: Knocking The robot dynamically manipulates a cable to knock a target object off an obstacle.
Task 3: Weaving The robot dynamically manipulates a cable to weave it between three obstacles.

To accomplish these tasks, we propose to generate trajectories using a parameterized function. To reduce the complexity of parameterizing actions, we compute trajectories using DJ-GOMP [1], [2] that take as input the apex point in the trajectory. This point, combined with task-specific starting

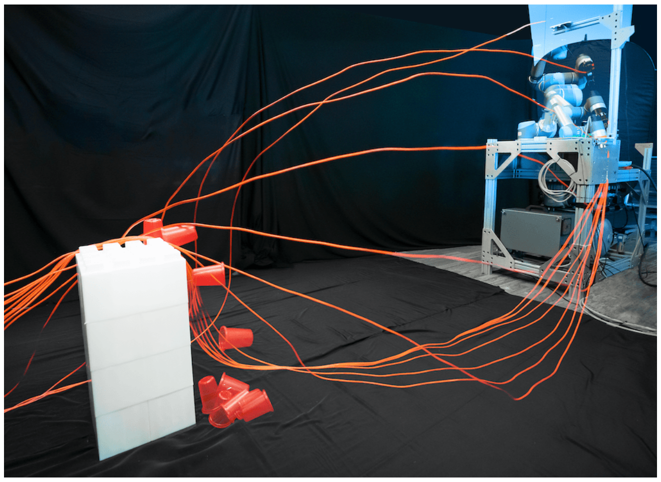


Fig. 1: Long exposure photo of a UR5 robot dynamically manipulating an orange cable fixed at one end (far left) to knock the red target cup off the white obstacle using the learned 3d apex point and minimum-jerk robot trajectory.

and ending arm configurations, are used to generate a single high-speed trajectory for the manipulator arm of a physical UR5 robot. We present a self-supervised deep imitation learning pipeline that learns to predict the apex point given target positions. Using a simulator we develop for vaulting, with a simulated UR5, we obtain an example action that completes the vaulting task. For any of the three tasks in the real world (see Fig. 1), we then have the physical UR5 continually execute actions based on the action computed in simulation, with noise added to increase dataset diversity. While the robot performs actions, one end of the cable is attached to its gripper, and the other is fixed to a wall. (Hereafter, we use *cable* to refer to any 1D deformable object with low stiffness, such as ropes, cords, and strings.) This process results in a self-supervised procedure for collecting a large dataset. We use the resulting data to train a deep neural network policy to predict an action (i.e., the apex point) given a target position.

This paper contributes:

1) Three novel dynamic cable manipulation tasks: vaulting, knocking, and weaving.
2) INDy, a self-supervised framework for collecting data and learning a dynamic manipulation policy using parameterized min-jerk trajectories.
3) A simulation platform to experiment with novel tasks and policy learning.
4) Physical experiments evaluating the policies with a UR5

[1]All authors are affiliated with the AUTOLab at UC Berkeley (automation.berkeley.edu). {`harryhzhang, jeffi, seita, jnwang19, hh19971229, goldberg`} `@berkeley.edu`

robot on 5 different cables.

II. Related Work

Manipulation of 1D deformable objects has applications in surgical suturing [3], [4], knot-tying [5], [6], [7], [8], untangling ropes [9], [10], [11], deforming wires [12], [13], and inserting cables into obstacles [14]; Sanchez et al. [15] provide a survey.

Various approaches have been proposed for quasistatic manipulation of cables, such as using demonstrations [16], [17], [18], self-supervised learning [19], [20], model-free reinforcement learning [21], [22], contrastive learning [23], dense object descriptors [24], generative modeling [25], and state estimation [26]. These prior works generally assume quasistatic dynamics to enable a series of small deformations, often parameterized with pick-and-place actions. Alternative approaches involve sliding cables [27], [28] or assuming non-quasistatic systems that enable tactile feedback [11]. This paper focuses on dynamic actions to manipulate cables from one configuration to another, traversing over obstacles and potentially knocking over objects.

More closely related is the work of Yamakawa et al. [29], [30], [31], [27] that focuses on dynamic manipulation of cables using robot arms moving at high speeds, enabling them to model rope deformation algebraically under the assumption that the effect of gravity is negligible. They show impressive results with a single multi-fingered robot arm that ties knots by initially whipping cables upwards. Because we use fixed-endpoint cables moving at slower speeds, the forces exerted on the rope by gravity and the fixed endpoint prevent us from assuming a model where each rope joint follows the robot motion with constant time-delay, as in Yamakawa et al. Instead, we use a learning-based method and do not use specialized end-effectors. Kim and Shell [32] use a small-scale mobile robot with a cable attached as a tail to manipulate small objects by leveraging inter-object friction to drag or strike the object of interest. While their work aims to tackle dynamic cable control, it relies heavily on physics models for motion primitives that are only applicable to dragging objects via cables, and uses a RRT-based graph search algorithm to address the stochasticity in the system. Other studies have developed physics models to analyze dynamic cable motion. For example, Zimmermann et al. [33] use the finite-element method to model elastic objects for dynamic manipulation of free-end deformable objects, and Gatti-Bonoa and Perkins [34] and Wang and Wereley [35] analyze fly fishing rod casting dynamics. We focus on fixed-end cable manipulation and do not attempt to model cable physics, and we aim to control cables dynamically via vaulting, knocking, and weaving, and cannot rely on the same dragging motion primitives as in Kim and Shell [32].

We propose an approach that is related to TossingBot [36] where a robot learns to toss objects to target bins. TossingBot uses RGBD images and employs fully convolutional neural networks [37] to learn dense, per-pixel values of tossing actions, and learns a residual velocity of tossing on top of a physics model in order to jointly train a grasping and tossing policy through trial and error. We take a similar approach in dynamically manipulating by reducing the learning process to learn only the joint angles of the apex of the motion. We do not rely on a physics model of the dynamic manipulation motion, and instead, leverage a quadratic programming-based motion-planning primitive to generate a fast ballistic motion that minimizes the jerk while moving through the learned apex point.

III. Problem Definition

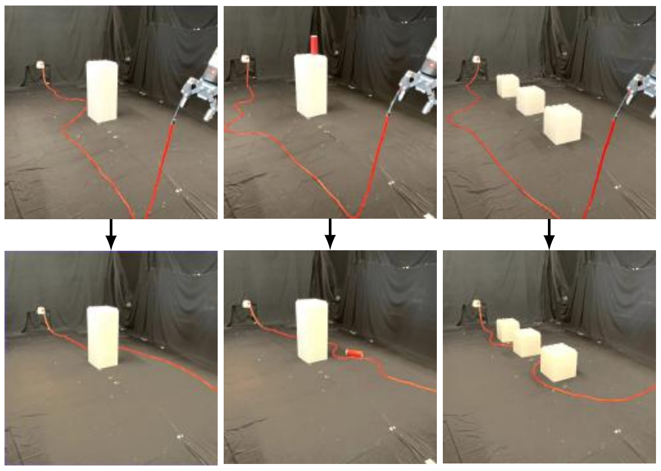
Given a manipulator arm holding one end of a cable with the other end fixed (e.g., plugged into a wall), the objective is to compute and execute a single high-speed arm motion from a fixed far-left starting location, to a fixed far-right ending location that moves the cable from an initial configuration to a configuration that matches a task-dependent goal constraint.

Let $s \in \mathcal{S}$ be the configuration of the cable and task-specific objects, including obstacles or targets, where $\mathcal{S}$ is the configuration space. Since $\mathcal{S}$ is infinite-dimensional, we consider observations of the state, and focus on tasks for which the observations are sufficient to capture progress. Let $o \in \mathcal{O}$ be the observation, where $\mathcal{O}$ is the observation space and $f_c : \mathcal{S} \to \mathcal{O}$ map a state to an observation (e.g., via a camera view). Let $\tau_a \in \mathcal{A}$ be the action the manipulator arm takes, where τ_a is a complete open-loop trajectory and $\mathcal{A}$ is the action space. Let $f_{\mathcal{S}} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ be the stochastic state transition function of the cable.

The observation space needs to encode sufficient information for the policy to infer actions, such as the dimension and location of the target object and/or the obstacle. Thus, in our problem, a camera positioned slightly above the manipulator arm should contain sufficient information to cover the minimum observation space. Thus, $\mathcal{O} = \mathbb{R}^{w \times h \times 3}$, where $w \times h$ are the dimensions of an RGB image in our experiments.

Formally the objective is: for each task $i \in \{\text{vaulting}, \text{knocking}, \text{weaving}\}$, to learn a task-specific policy $\pi_i : \mathcal{O} \to \mathcal{A}$, that when given the observation $o_0 = f_c(s_0)$ of an initial configuration $s_0 \in \mathcal{S}$ and a goal set of observations $\mathcal{O}_{\text{goal}} \subset \mathcal{O}$, computes an action $\tau_a \in \mathcal{A}$ such that $f_c(f_{\mathcal{S}}(s_0, \tau_a)) \in \mathcal{O}_{\text{goal}}$. For each task, we define the goal set in terms of constraints, such as requiring an observation with the cable on the right side of an obstacle (Fig. 2 (a)).

Since the action space $\mathcal{A}$ is also infinite-dimensional, we define a parametric trajectory function $f_{\mathcal{A}_i} : \mathbb{R}^3 \to \mathcal{A}$, that, when given an apex point, computes complete arm trajectory. While the exact function is a design choice, we propose that the function should have the following features: the arm motions are fast enough to induce a motion over the length of the cable, the apex point of the motion can be varied based on an $\mathbb{R}^3$ input, the other points in the trajectory vary smoothly through the apex, and the motions are kinematically and dynamically feasible. This converts the problem to learning $\bar{\pi}_i : \mathcal{O} \to \mathbb{R}^3$, where $\mathcal{O}$ contains position and dimension information, such that for the action $a \in \mathbb{R}^3$ computed by the policy, $f_c(f_{\mathcal{S}}(s_0, f_{\mathcal{A}_i}(a))) \in \mathcal{O}_{\text{goal}}$.



(a) Task 1: Vaulting (b) Task 2: Knocking (c) Task 3: Weaving

Fig. 2: Illustration of the 3 tasks: Vaulting, Knocking, and Weaving in real experiments with an orange 18-gauge cable.

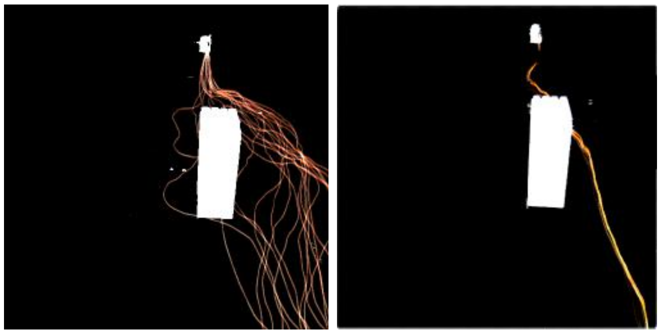


Fig. 3: **Repeatability without (left) and with (right) taut-pull reset.** Both images show an overlay of 20 ending cable configurations after sequentially applying the same left-to-right vaulting trajectory 20 times. **Left:** without applying resets before the motion, there is high variation in ending configurations. **Right:** resetting the cable via a taut-pull before the motion results in nearly identical configurations across 20 left-to-right attempts.

IV. Method

We propose a deep-learning system that takes as input an image, and produces a sequence of timed joint configurations; when the robot to follows the motion, it results in the dynamic manipulation of a cable that accomplishes the task.

A. Hypothesis

Repeatability: A cable with fixed length is attached to the wall and with a gentle pull, the robot can reset the cable to a consistent starting state before each motion. The same arm trajectory will result in a similar end configuration of the cable (Fig. 3).

Parameterized Trajectory Function: We hypothesize that a high-speed minimum-jerk ballistic manipulator-arm motion can solve many left-to-right vaulting problems by varying the midpoint location in (x, y, z) (Fig. 5 left). This reduces the problem to finding which $\mathbb{R}^3$ input to a trajectory function $f_{\mathcal{A}_i}$ solves the task. This is akin to the $\mathbb{R}^2$-regression TossingBot [36] uses to throw object.

Algorithm 1 Iterative traiNing for Dynamic manipulation (INDy)

```
Require: Base action of task $i$ $a_{\beta,i}$, a task-trajectory function $f_{\mathcal{A}_i}$,
    a goal observation set $\mathcal{O}_{\text{goal}}$, number of data points $N$, the
    task's time horizon $T$.
 1: Empty dataset $\mathcal{D} = \{\}$.
 2: for $t \in \{1 \ldots N\}$ do
 3:     Randomize environment (obstacle size and location)
 4:     Pull taut to reset to cable's state (updates $s_t$)
 5:     $o_t \leftarrow f_c(s_t)$ {record initial observation}
 6:     Set starting action $a_{0:T} = a_{\beta,i}$
 7:     loop
 8:         $\tau_a \leftarrow f_{\mathcal{A}_i}(a_{0:T})$ {compute trajectory}
 9:         execute $\tau_a$ on the robot (updates $s_t$)
10:         if $f_c(s_t) \in \mathcal{O}_{\text{goal}}$ then
11:             break {success observed}
12:         Pull taut to reset to cable's state (updates $s_t$)
13:         $a_{0:T} \leftarrow a_{0:T} +$ (random noise)
14:     $\mathcal{D} \longleftarrow \mathcal{D} \cup \{(o_t, a_{0:T})\}$
15: Train policy $\pi_i(a|o)$ with $\mathcal{D}$ to minimize Eq. 1.
16: return Trained policy for task $i$, $\pi_i(a|o)$
```

B. Deep-Learning Training Algorithm

We propose a self-supervised training algorithm, INDy (Alg. 1), to learn a policy π_i for task i. The main inputs are: a base action for the task $a_{\beta,i}$, a task-trajectory function $f_{\mathcal{A}_i}$, and a goal observation set $\mathcal{O}_{\text{goal}}$.

To train the policy, INDy first forms a dataset $\mathcal{D}$ of successful observations and actions by having the robot attempt the base action $a_{\beta,i}$. On failure, it repeatedly tries random perturbations on top of the base action until it successfully completes the task. Before each attempt, the robot self-resets the cable with a taut-pull, which makes the system repeatable (Fig. 3). INDy add successful observation-action pairs to $\mathcal{D}$ until the data contains a user-specified number of points N. INDy then uses $\mathcal{D}$ to train a convolutional neural network (CNN) [38] policy π_i via behavior cloning [39], [40] to output motion parameters given image observations. We parameterize the policy to produce the mean and covariance of a multivariate Gaussian, and for each data sample $(o_t, a_{0:T}) \sim \mathcal{D}$, optimize:

$$\mathcal{L}_{\text{MSE}} = ||\hat{a}_{0:T} - a_{0:T}||_2^2 \tag{1}$$

where $\hat{a}_{0:T} \sim \pi_i(a|o_t)$, and optimization is done using the reparameterization trick [41].

C. Min Jerk Trajectory Generation Function

INDy uses a trajectory generation function $f_{\mathcal{A}_i} : \mathbb{R}^3 \rightarrow \mathcal{A}$ to reduce the output dimension of the neural network, while still being able to achieve a high success rate on tasks. In initial experiments, we observed that for the trajectory to be dynamically feasible, the trajectory generation must also limit and minimize jerk. Otherwise, acceleration changes would often be too fast for the manipulator arm to follow while burdened by the inertia of the cable.

We propose a function based on a quadratic program (QP) that has these features and empirically achieves high success rates in the 3 tasks. We first discretize a trajectory to $H+1$ time steps, each with h seconds apart. At each time step $t \in [0\,..\,H]$, the discretized trajectory has a configuration

$\mathbf{q}_t$, and a configuration-space velocity $\mathbf{v}_t$ and acceleration $\mathbf{a}_t$. With constraints in the QP, we enforce that the trajectory is kinematically and dynamically feasible, starts and ends at 2 fixed arm configurations, and reaches an apex configurations defined by the output of the learned policy. The objective of the QP minimizes jerk. To make the trajectory as fast as possible, in a manner similar to GOMP [1], we repeatedly reduce H by 1 until the QP is infeasible, and use the shortest feasible trajectory. The QP takes the following form:

$$
\begin{aligned}
\underset{(\mathbf{q},\mathbf{v},\mathbf{a})_{[0..H]}}{\operatorname{argmin}} \quad & \frac{1}{2h}\sum_{t=0}^{H-1}(\mathbf{a}_{t+1}-\mathbf{a}_t)^\intercal Q(\mathbf{a}_{t+1}-\mathbf{a}_t) \\
\text{s.t.} \quad & \mathbf{q}_0 \text{ is the start configuration} \\
& \mathbf{q}_{\lfloor H/2 \rfloor} \text{ is the apex configuration} \\
& \mathbf{q}_H \text{ is at the end configuration} \\
& \mathbf{v}_0 = \mathbf{v}_H = \mathbf{0} \\
& \mathbf{q}_{t+1} = \mathbf{q}_t + h\mathbf{v}_t + \tfrac{1}{2}h^2\mathbf{a}_t \quad \forall t \in [0\,..\,H-1] \\
& \mathbf{v}_{t+1} = \mathbf{v}_t + h\mathbf{a}_t \quad \forall t \in [0\,..\,H-1] \\
& \mathbf{q}_{\min} \le \mathbf{q}_t \le \mathbf{q}_{\max} \quad \forall t \in [1\,..\,H-1] \\
& \mathbf{v}_{\min} \le \mathbf{v}_t \le \mathbf{v}_{\max} \quad \forall t \in [1\,..\,H-1] \\
& \mathbf{a}_{\min} \le \mathbf{a}_t \le \mathbf{a}_{\max} \quad \forall t \in [0\,..\,H-1] \\
& h\mathbf{j}_{\min} \le \mathbf{a}_{t+1} - \mathbf{a}_t \le h\mathbf{j}_{\max} \quad \forall t \in [0\,..\,H-1].
\end{aligned}
$$

The Q matrix is a diagonal matrix that adjusts the relative weight of each joint. The QP constraints, in order, fix the start, midpoint, and end configurations ($\mathbf{q}_0$, $\mathbf{q}_{\lfloor H/2 \rfloor}$, $\mathbf{q}_H$); fix the start and end velocity ($\mathbf{v}_0$, $\mathbf{v}_H$) to zero; enforce configuration and velocity dynamics between time steps ($\mathbf{q}_{t+1}$, $\mathbf{v}_{t+1}$); keep the joints within the specified limits of configuration ($\mathbf{q}_{\min}$, $\mathbf{q}_{\max}$), velocity ($\mathbf{v}_{\min}$, $\mathbf{v}_{\max}$), acceleration ($\mathbf{a}_{\min}$, $\mathbf{a}_{\max}$), and jerk ($\mathbf{j}_{\min}$, $\mathbf{j}_{\max}$). Unlike GOMP, we do not consider obstacle avoidance in the trajectory generation, and instead ensure that there are no obstacles in the path of the arm to avoid. In practice, we fix h to be an integer multiple of the robot's control frequency, and the initial value of H to be sufficiently large such that there is always a solution.

The input to the trajectory function constrains the midpoint configuration of the trajectory. As most trajectories lift then lower the end effector through the midpoint, the input is often the apex of the motion. With the UR5 robot, we observe that varying the first 3 joints (base, shoulder, elbow) moves the end-effector's location in (x, y, z). The base joint rotates left and right, while the shoulder and elbow joints lift and extend. We thus use these 3 joints to define the midpoint, while having the remaining joint configurations depend linearly on the first three. We make this function a task-specific function by constraining the start and end configurations to depend on the task. For example, for the left-to-right vaulting task, we set the start configuration to be down and left, and the end configuration to be down and right.

V. Simulatior

We use simulation to experiment with training algorithms, and to find a feasible base action a_β to bootstrap the data collection process in the real world. Using simulation enables fast prototyping to efficiently assess qualitative and quantitative behavior of multiple base actions, but can suffer from a sim-to-real gap, motivating research teams to collect data from running robots in the real world [42], [43]. Similarly, we collect data by running a physical UR5 robot, but use simulation as a means to obtain a functional action that can be improved upon when collecting data.

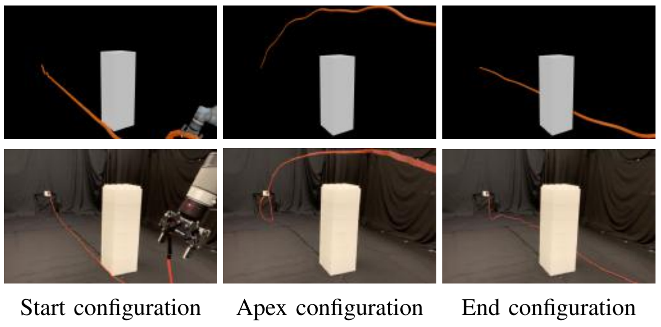


Fig. 4: Cable trajectory for data collection in **simulation (top)** vs. cable trajectory in **real (bottom)** for the vaulting task after applying the same apex point configuration.

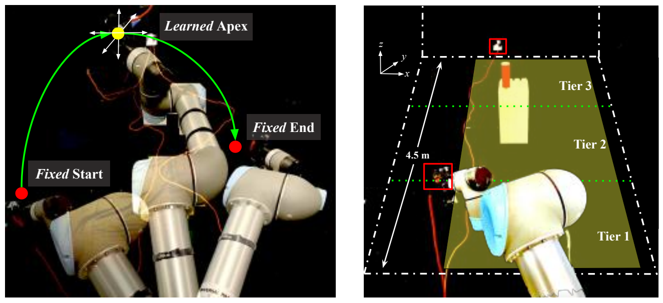


Fig. 5: **Left:** Using three points in a curve to control the trajectory of the cable. The start and end points are fixed using the arm configurations. We change the trajectory by varying the apex configuration. **Right:** Design of physical experiments. We randomize the white Lego bricks' location within the yellow shaded area. For vaulting and knocking, the width of the yellow area is 1.5 m, and 1.0 m for weaving. The yellow area is divided into three difficulty tiers.

To model cables in simulation, we use a Featherstone [44] rigid-body simulator from BulletPhysics via Blender [45]. Cables consist of a series of small capsule links connected by spring and torsional forces. We tune the spring and torsional coefficients by decreasing bending resistance and stopping before the cable noticeably behaves like individual capsules rather than a cable, and increasing twist resistance until excessive torsion causes the capsules to separate.

When finding a feasible a_β to transfer to real, we only simulate vaulting. The obstacle height, radius, and location are randomized for each trial, along with the initial cable state. We use 10 different obstacle and initial cable settings in simulation, and select the input apex point with highest success rate out of 60 random points. Then, we take this apex point directly to physical experiments as the base action a_β to collect data, as the physical UR5 exhibits similar behavior to that in simulation after applying the predefined configurations from simulation, as seen in Fig. 4.

VI. Physical Experiments

We use physical UR5 robot grasping to dynamically manipulate a cable attached to a wall 4.5 m away, on the three

TABLE I: Physical cables used in experiments and the number of successes in the three tasks (Vaulting, Knocking, Weaving) using policies trained with the 18 AWG orange cable. For each cable, we evaluate with 20 trials for each task.

Type	kg	m	Vaulting	Knocking	Weaving
18 AWG orange cable	0.65	5.5	16	13	12
16 AWG white cable	0.70	5.2	16	12	12
16 AWG orange cable	0.90	6.2	4	2	2
22 AWG blue network cable	0.48	5.5	14	9	12
12 AWG blue jump rope	0.45	5.1	15	10	8

tasks, with 5 different cable types. The system obtains observations from a Logitech C270 720p webcam placed 0.75 m above the robot arm base. It scales and crops the image to a 512x512x3 segmented RGB image that feeds into a ResNet-34 [46] deep neural network implemented with PyTorch [47]. We initialize weights using He initialization [48].

In physical experiments of the 3 tasks (see Fig. 5 for layout), we use large, white plastic bricks as the obstacle(s), which can vary in size and location. The obstacles we use for experiments are 0.15 m to 0.75 m in width and 0.3 m to 1.5 m in height. To generate training data, we randomize the location and size of the obstacles in each trial. We test the system with 5 cables listed in Table I.

We define *difficulty tiers* based on the distance of the obstacle to the base of the robot.

Tier 1	**Tier 2**	**Tier 3**
$\leq$ 1.5 m	1.5 to 3 m	$\geq$ 3 m

We evaluate INDy along with two baselines:

Baseline with Fixed Apex. In the three tasks, for each configuration, a human annotator sets one set of apex joint angles that can accomplish the task. The apex joint angles are fixed with respect to the obstacle itself, and they remain the same across different locations of the same obstacle.

Baseline with Varying Apex. In vaulting and weaving, for each configuration (including obstacle size and location), a human annotator sets the apex joint angles such that the robot arm's end effector is 15 cm above the center of mass (COM) of the obstacle, and aligned horizontally with the COM of the obstacle. For knocking, a human annotator makes the robot arm end 10 cm above the COM of the target object, and aligned horizontally with the COM.

We find the baselines from simulation, where we observe that the fixed and varying apex motions could induce enough velocity to make the cable vault the obstacle.

VII. Results

We measure the correlation between simulated and real setups, benchmark the three dynamic cable tasks, and investigate generalization to different cables. The project website supplements these results with videos.

A. Comparisons between Simulation and Reality

To observe correlations between cable behavior in simulation and real, we evaluate vaulting on the same set of apex point and obstacle settings in both setups. Each trial consists of a random obstacle location from all 3 difficulty tiers and a different apex point. In simulation, we use a cube mesh of the same dimension as the plastic bricks obstacle in real, and a cable of the same length and mass as the 18 AWG orange cable. Table II reports the frequency of success and failure cases across simulation and real. While the cable trajectory in simulation tends to match the trajectory of real for successes, 50 % of all failed cases in real succeed in simulation under the same setting. The discrepancies in cable behavior between simulation and real are most apparent for Tier 3 scenarios, where the height achieved by the cable at the opposite end from the robot differ in the two environments. Because the cable properties cannot be perfectly tuned to match the physical properties in real, these experiments motivate the decision to use real experiments to collect data.

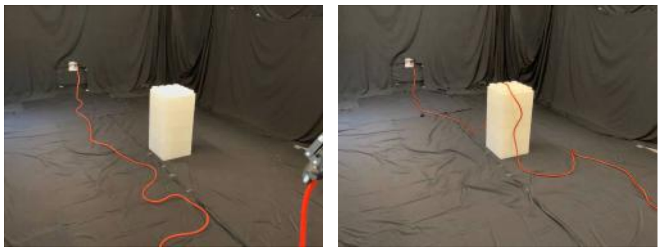


Fig. 6: **Failure mode**. The 16-gauge 6.2 m orange cable is too long for the vaulting motion. **Left:** before the motion there is too much slack. **Right:** the vaulting motion that works for shorter cables does not clear the obstacle.

TABLE II: Number of successes and failures for vaulting in simulation and physical environments across 15 trials, evaluating a different apex point on a different obstacle setting for each trial. Listed are the number of cases that succeed in both sim and real, fail in sim but succeed in real, succeed in sim but fail in real, and fail in both sim and real. The correlation between success rates in the two environments is surprisingly high.

	#Success (Simulation)	#Failure (Simulation)
#Success (Physical)	7	2
#Failure (Physical)	3	3

B. Results for Vaulting Task

For vaulting, we perform 180 physical trials from manipulating the orange 5.5 m 18-gauge power extension cable, with 60 trials for each of the 3 difficulty tiers; this process took 8 hours. In each tier, we shift the obstacle vertically within the bounds of the difficulty tier and horizontally within a range of 1.5 m. We use plastic bricks to construct 6 combinations to build 6 objects of different sizes, and for each obstacle, we vary its location 10 times. Table III shows results of the trained policy on vaulting evaluated with 20 trials per tier.

C. Results for Knocking Task

For knocking, we perform 180 physical trials from manipulating the orange 5.5-m 18-gauge power extension cable, with 60 trials for each of 3 difficulty tiers; this process took 9 hours. In each difficulty tier, we use the plastic bricks as the base upon which we place the target object to be knocked,

TABLE III: Physical Vaulting Results: Number of successes of the learned policy for vaulting evaluated against two baseline methods across 20 trials per difficulty tier using the 18 AWG orange cable.

Method	Tier 1	Tier 2	Tier 3
Baseline with Fixed Apex	16	13	2
Baseline with Varying Apex	19	14	7
Learned Apex	**20**	**18**	**11**

TABLE IV: Physical Knocking Results: Number of successes of the learned policy for knocking evaluated against two baseline methods across 20 trials per difficulty tier using the 18 AWG orange cable.

Method	Tier 1	Tier 2	Tier 3
Baseline with Fixed Apex	12	6	4
Baseline with Varying Apex	14	14	6
Learned Apex	**15**	**14**	**10**

and we shift the plastic bricks base vertically within the bounds of the difficulty tier and horizontally within a range of 1.5 m. We train the policy using four different target objects: a cylinder, a tennis ball, a cup, and a rectangular box. We use plastic bricks to construct 3 combinations to build 3 bases of different sizes. For each base, we vary its locations 5 times, and for each location of the base, we vary the target on top 4 times. Table IV shows the results for the knocking trained policy evaluated with 20 trials for each difficulty tier.

D. Results for Weaving Task

We perform 100 physical trials from manipulating the orange 5.5-m 18-gauge power extension cable, which took 7.5 hours. We make the 3 obstacles identical in size. In our training data for this task, we vary the space between each object by 15 to 50 cm and we shift the obstacles horizontally within a range of 1 m. Table V shows the result of the weaving trained policy evaluated with 20 trials.

E. Generalization to Different Cables

While the policies for the three tasks are trained using the 18-gauge orange cable, we experiment with different types of cables. The results suggest that the policy trained on the 18-gauge orange cable can transfer well to other cables of similar lengths. We observe that the same reset action can bring these cables to nearly identical starting configurations in observation space, making the policy agnostic to cable type. We use the learned policy from the 18-gauge orange cable without retraining or finetuning to run the same experiments (same object sizes and locations for each task) for the 3 tasks with different cables and show results in Table I.

F. Limitations and Failure Modes

While the framework can be robust to object locations and shapes, it has limitations. First, the policies are learned in a highly controlled environment. Many failure cases in vaulting and knocking are from placements outside the training area. For example, in Tier 3 vaulting, 7 of 9 failure cases are due to the horizontal shift of the obstacle exceeding the 1.5 m bound. In weaving, the horizontal shift of the obstacles is bounded into a smaller 1.0 m range.Second, learned policies have difficulty generalizing to cables of different lengths and masses. We conjecture that with different lengths, the amount of slack in the cables should also be different, thus decreasing the repeatability of applying the same action. With the 6.2 m cable, Tier 2 and 3 vaulting and knocking always fail in that the segment of the cable near the wall end can never be accelerated up due to the fact that the slack near the wall end cannot be eliminated by the resetting action. To show this failure mode with a longer cable, we run the policies for vaulting, knocking, and weaving trained with the 5.5-m 18-gauge cable on the 6.2-m 16-gauge cable, and we achieve 20 % success rate for vaulting, 10 % success rate for knocking, and 10 % success rate for weaving. Fig. 6 shows this failure case. Additionally, Table I suggests that cables that are too light are likely to yield lower success rates, which may be because they often travel slower during descent, limiting the distance the cable traverses. Finally, the data generation part in Alg. 1 is inefficient since when the obstacle size and location change, the algorithm needs to search for successful apex point again, so data generation in real is unable to record a wider variety of obstacle placements in an efficient manner.

TABLE V: Physical Weaving Results: Number of successes of the learned policy for weaving evaluated against two baseline methods across 20 trials using the 18 AWG orange cable.

Method	#Success
Baseline with Fixed Apex	3
Baseline with Varying Apex	3
Learned Apex	**12**

VIII. Conclusion

This paper explores learning three dynamic cable manipulation tasks: vaulting, knocking, and weaving. Experimental results suggest that the proposed procedure outperforms two baseline methods.

In future work, we will generalize the learned policy to different cable lengths and masses, and more complex obstacles and target objects. We will apply the parameterized trajectory method to other dynamic cable manipulation tasks without fixed-endpoints of the cable, insired by the adventures of Dr. Indiana Jones.

IX. Acknowledgments

This research was performed at the AUTOLAB at UC Berkeley in affiliation with the Berkeley AI Research (BAIR) Lab, the CITRIS "People and Robots" (CPAR) Initiative, and the Real-Time Intelligent Secure Execution (RISE) Lab. The authors were supported in part by donations from Toyota Research Institute and by equipment grants from PhotoNeo, NVidia, and Intuitive Surgical. Daniel Seita is supported by a Graduate Fellowship for STEM Diversity (GFSD). We thank our colleagues who provided helpful feedback, code, and suggestions, in particular Francesco Borrelli, Adam Lau, Priya Sundaresan, and Jennifer Grannen.

References

[1] J. Ichnowski, M. Danielczuk, J. Xu, V. Satish, and K. Goldberg, "GOMP: Grasp-Optimized Motion Planning for Bin Picking," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[2] J. Ichnowski, Y. Avigal, V. Satish, , and K. Goldberg, "Deep Learning can Warm Start Grasp-Optimized Motion Planning," *Science Robotics*, 2020, to appear.

[3] J. Schulman, A. Gupta, S. Venkatesan, M. Tayson-Frederick, and P. Abbeel, "A Case Study of Trajectory Transfer Through Non-Rigid Registration for a Simplified Suturing Scenario," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[4] S. Sen, A. Garg, D. V. Gealy, S. McKinley, Y. Jen, and K. Goldberg, "Automating Multi-Throw Multilateral Surgical Suturing with a Mechanical Needle Guide and Sequential Convex Optimization," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[5] J. van den Berg, S. Miller, D. Duckworth, H. Hu, A. Wan, X.-Y. Fu, K. Goldberg, and P. Abbeel, "Superhuman Performance of Surgical Tasks by Robots Using Iterative Learning from Human-Guided Demonstrations," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[6] J. Hopcroft, J. Kearney, and D. Krafft, "A Case Study of Flexible Object Manipulation," in *International Journal of Robotics Research (IJRR)*, 1991.

[7] T. Morita, J. Takamatsu, K. Ogawara, H. Kimura, and K. Ikeuchi, "Knot Planning from Observation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2003.

[8] W. Wang and D. Balkcom, "Tying Knot Precisely," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[9] W. H. Lui and A. Saxena, "Tangled: Learning to Untangle Ropes with RGB-D Perception," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[10] 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," in *Conference on Robot Learning (CoRL)*, 2020.

[11] Y. She, S. Dong, S. Wang, N. Sunil, A. Rodriguez, and E. Adelson, "Cable Manipulation with a Tactile-Reactive Gripper," in *Robotics: Science and Systems (RSS)*, 2020.

[12] H. Nakagaki, K. Kitagi, T. Ogasawara, and H. Tsukune, "Study of Insertion Task of a Flexible Wire Into a Hole by Using Visual Tracking Observed by Stereo Vision," in *IEEE International Conference on Robotics and Automation (ICRA)*, 1996.

[13] ——, "Study of Deformation and Insertion Tasks of Flexible Wire," in *IEEE International Conference on Robotics and Automation (ICRA)*, 1997.

[14] W. Wang, D. Berenson, and D. Balkcom, "An Online Method for Tight-Tolerance Insertion Tasks for String and Rope," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[15] 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," in *International Journal of Robotics Research (IJRR)*, 2018.

[16] J. Schulman, J. Ho, C. Lee, and P. Abbeel, "Learning from Demonstrations Through the Use of Non-Rigid Registration," in *International Symposium on Robotics Research (ISRR)*, 2013.

[17] A. X. Lee, S. H. Huang, D. Hadfield-Menell, E. Tzeng, and P. Abbeel, "Unifying Scene Registration and Trajectory Optimization for Learning from Demonstrations with Application to Manipulation of Deformable Objects," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014.

[18] D. Seita, P. Florence, J. Tompson, E. Coumans, V. Sindhwani, K. Goldberg, and A. Zeng, "Learning to Rearrange Deformable Cables, Fabrics, and Bags with Goal-Conditioned Transporter Networks," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[19] A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine, "Combining Self-Supervised Learning and Imitation for Vision-Based Rope Manipulation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[20] D. Pathak, P. Mahmoudieh, G. Luo, P. Agrawal, D. Chen, Y. Shentu, E. Shelhamer, J. Malik, A. A. Efros, and T. Darrell, "Zero-Shot Visual Imitation," in *International Conference on Learning Representations (ICLR)*, 2018.

[21] Y. Wu, W. Yan, T. Kurutach, L. Pinto, and P. Abbeel, "Learning to Manipulate Deformable Objects without Demonstrations," in *Robotics: Science and Systems (RSS)*, 2020.

[22] X. Lin, Y. Wang, J. Olkin, and D. Held, "SoftGym: Benchmarking Deep Reinforcement Learning for Deformable Object Manipulation," in *Conference on Robot Learning (CoRL)*, 2020.

[23] W. Yan, A. Vangipuram, P. Abbeel, and L. Pinto, "Learning Predictive Representations for Deformable Objects Using Contrastive Estimation," in *Conference on Robot Learning (CoRL)*, 2020.

[24] 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 *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[25] A. Wang, T. Kurutach, K. Liu, P. Abbeel, and A. Tamar, "Learning Robotic Manipulation through Visual Planning and Acting," in *Robotics: Science and Systems (RSS)*, 2019.

[26] M. Yan, Y. Zhu, N. Jin, and J. Bohg, "Self-Supervised Learning of State Estimation for Manipulating Deformable Linear Objects," in *IEEE Robotics and Automation Letters (RA-L)*, 2020.

[27] Y. Yamakawa, A. Namiki, M. Ishikawa, and M. Shimojo, "One-Handed Knotting of a Flexible Rope With a High-Speed Multifingered Hand Having Tactile Sensors," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007.

[28] J. Zhu, B. Navarro, R. Passama, P. Fraisse, A. Crosnier, and A. Cherubini, "Robotic Manipulation Planning for Shaping Deformable Linear Objects with Environmental Contacts," in *IEEE Robotics and Automation Letters (RA-L)*, 2019.

[29] Y. Yamakawa, A. Namiki, and M. Ishikawa, "Motion Planning for Dynamic Knotting of a Flexible Rope With a High-Speed Robot Arm," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010.

[30] ——, "Simple Model and Deformation Control of a Flexible Rope Using Constant, High-speed Motion of a Robot Arm," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[31] ——, "Dynamic High-Speed Knotting of a Rope by a Manipulator," in *International Journal of Advanced Robotic Systems*, 2013.

[32] Y.-H. Kim and D. A. Shell, "Using a compliant, unactuated tail to manipulate objects," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 223–230, 2016.

[33] S. Zimmermann, R. Poranne, and S. Coros, "Dynamic Manipulation of Deformable Objects with Implicit Integration," in *IEEE Robotics and Automation Letters (RA-L)*, 2021.

[34] C. Gatti-Bonoa and N. Perkins, "Numerical Model for the Dynamics of a Coupled Fly Line / Fly Rod System and Experimental Validation," *Journal of Sound and Vibration*, 2004.

[35] G. Wang and N. Wereley, "Analysis of Fly Fishing Rod Casting Dynamics," *Shock and Vibration*, vol. 18, no. 6, pp. 839–855, 2011.

[36] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "TossingBot: Learning to Throw Arbitrary Objects with Residual Physics," in *Robotics: Science and Systems (RSS)*, 2019.

[37] J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Neural Information Processing Systems (NeurIPS)*, 2012.

[39] D. A. Pomerleau, "Efficient Training of Artificial Neural Networks for Autonomous Navigation," *Neural Comput.*, vol. 3, 1991.

[40] S. Ross, G. J. Gordon, and J. A. Bagnell, "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[41] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," in *International Conference on Learning Representations (ICLR)*, 2014.

[42] A. Gupta, A. Murali, D. Gandhi, and L. Pinto, "Robot Learning in Homes: Improving Generalization and Reducing Dataset Bias," in *Neural Information Processing Systems (NeurIPS)*, 2018.

[43] R. Lee, D. Ward, A. Cosgun, V. Dasagi, P. Corke, and J. Leitner, "Learning Arbitrary-Goal Fabric Folding with One Hour of Real Robot Experience," in *Conference on Robot Learning (CoRL)*, 2020.

[44] R. Featherstone, "Robot Dynamics Algorithms," 1984.

[45] B. O. Community, *Blender – a 3D modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. [Online]. Available: http://www.blender.org

[46] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[47] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Neural Information Processing Systems (NeurIPS)*, 2019.
[48] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.