Disentangling Planning and Control for Non-prehensile Tabletop Manipulation

Vishal Reddy Mandadi[1], Kallol Saha[1], Dipanwita Guhathakurta[1], M. Nomaan Qureshi[1], Aditya Agarwal[1], Bipasha Sen[1], Dipanjan Das[2], Brojeshwar Bhowmick[2], Arun Singh[3], and Madhava Krishna[1]

***Abstract*— Manipulating a target object using non-prehensile actions presents a complex problem that requires addressing multiple constraints, such as finding collision-free trajectories, modeling the object dynamics, and speed of execution. Previous approaches have relied on contact-rich manipulation, where the object moves predictably while attached to the manipulator, thereby avoiding the need to model the object's dynamics. However, this assumption may not always hold, and different end-effectors may be necessary depending on the use case. In place of contact-rich manipulation, we implement the pushing-by-striking method (without tactile feedback) by explicitly modeling the object dynamics. Our novel framework disentangles planning and control enabling us to operate in a context-free manner. Our method consists of two components: an A* planner and a low-level RL controller. The low-level RL controller feeds on purely object-centric representations and has an object-centric action space, thus making it agnostic of the scene context. On the other hand, the planning module operates idependently of the low-level control and only takes scene context into account, making the approach quick to adapt and implement. We demonstrate the performance of our algorithm against a global RL policy which is computationally expensive and unreliable to train or fine-tune. To the best of our knowledge, a model-free RL policy without tactile feedback on push-by-strike problems has not been addressed before.**

I. INTRODUCTION

Non-prehensile manipulation is an important aspect that involves manipulating without grasping the object is essential in many places. For instance, manipulating objects that are too heavy to pick or too broad to grasp. However, non-prehensile manipulation has received less attention in comparison to grasp manipulation, and a standard mechanism to robustly push an object from start to goal position while avoiding colliding with objects on the table does not exist.

A recent work [9] tackles this problem but assumes contact-rich manipulation in which the target object is attached to the manipulator. This reduces the complexity of the problem by assuming predictable dynamics - the object moves along with the manipulator. Contact-rich manipulator, however, assumes a certain type of gripper or a way of contact [20] that may not be feasible for a given use case. We concentrate on non-prehensile manipulation in an end-effector agnostic manner that assumes the object dynamics to be independent of the end-effector dynamics. The primary challenge in developing a policy to execute such an action is the stochastic outcome of pushing an object: in the absence of privileged information like friction, and the weight of the target object, it is impossible to accurately predict the dynamics of the object on a push action. To tackle this, several works have tried to explicitly model the outcome by training a deep neural network on a large amount of push-outcome pairs collected via simulation [11], [12], [5]. However, such explicit modeling results in low generalization making multi-step long-horizon planning of pushing an object from start to goal challenging.

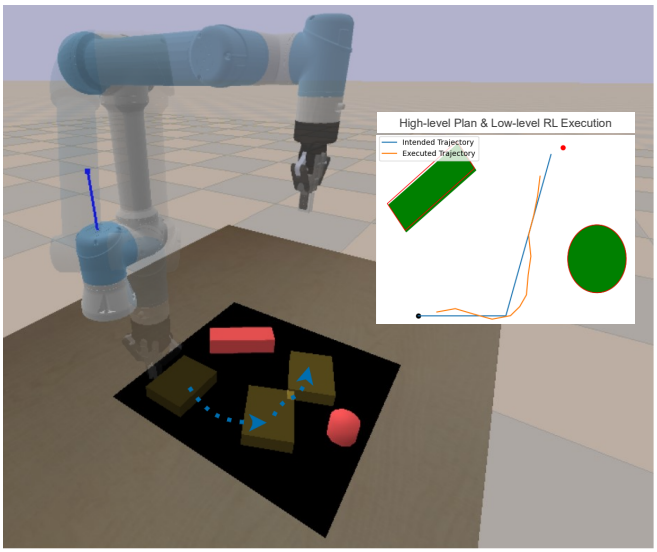

Fig. 1: We tackle object manipulation from start to goal position using non-prehensile actions. Non-prehensile action is necessary in many scenarios such as manipulating objects that are too heavy to be picked. To tackle this, we propose a novel method that disentangles planning from control by relying on a planning algorithm A* and a context-agnostic low-level control. Here we showcase one such trajectory planned and executed by our proposed algorithm.

In recent years, deep reinforcement learning (RL) has shown promising results in various robotics applications [3], [14], [19], [25], [7], [8], [2], [16], [4], [21], [13], [6]. RL algorithms learn to perform a task by maximizing a reward signal, which can be a scalar value reflecting the task's success; or a dense reward computing the distance from the goal. Taking inspirations from these works, we devise an end-to-end RL (global RL) policy to manipulate an object from start to goal in a collision-free manner. We design an object-centric action space suitable for the task in hand and a set of reward components for fast convergence.

However, we observe that such a design results in low-generalization across different contexts whilst making the

[1]Robotics Research Center, IIIT-H
[2]TCS Research
[3]University of Tartu

framework hard to train and debug. To tackle this, we try to disentangle the problem into a set of small independent problems of high-level planning and low-level control. The path planning module generates a feasible trajectory for the robot's end-effector; the low-level RL controller learns to control the robot's movements to achieve the desired task. In our framework, RL acts as an efficient mechanism for understanding the dynamics of an object on different push actions. To achieve this, we train the controller using a simple objective to push the object quickly to a given goal location. In other words, we train it to simply push a given object quickly to a goal. As a result, the control is unaware of the scene context (such as the collision objects) and is dependent on a high-level planning module to obtain the *collision-free* trajectory. Consequently, this results in a smaller state space making it easy for RL to converge.

Compared to training a global RL model, the disentangled framework has several advantages:

(1) Improved sample efficiency: Non-prehensile object manipulation tasks typically require a large number of samples to train an RL model effectively. A modular framework can reduce the stress on the RL model by reducing the size of the problem it has to tackle.

(2) Better interpretability: A modular approach separates the control of the task-specific plan and the low-level actions, making it easier to understand how the system operates and diagnose issues. In contrast, a global RL model can be more opaque and difficult to interpret.

(3) Ability to handle constraints: Non-prehensile object manipulation tasks often have constraints, such as avoiding collisions or maintaining balance. A modular approach can incorporate these constraints into the high-level planning module to guide the low-level actions. In contrast, a global RL model may struggle to handle all the constraints effectively, as designing an appropriate reward can be tricky.

Summary of our contributions:

1) We propose a novel framework that disentangles control and planning for non-prehensile manipulation making our framework easy to train and adapt to unseen contexts.
2) A novel global RL model which plans and executes a collision-free path to reach a specified goal from the start in an end-to-end way.
3) We extensively evaluate our proposed framework against the global RL policy and showcase that our approach outperforms the global RL despite being trained for just a fraction of global RL's train time.

II. Method

Push-based non-prehensile manipulation is an important area in tabletop rearrangement and planning [1] and can be divided into push-to-grasp, such as pushing objects in clutter to make them graspable [11], [12] or sliding an object to the edge of the table [10], [15] and push-to-goal to push an object from a start to a goal position [9], [24], [18], [5]. The latter line of work can further be classified as contact-rich manipulation either by sliding by the top [23], [22], [24], [9] or by the side [18].

We aim to tackle push-to-goal through push-by-striking manipulation. Push-by-striking loses the contact-rich assumption and thus disentangles the dynamics of the object from the manipulator dynamics removing the constraints in the type of end-effector – as the end-effector strikes an object, the outcome of the object state is less susceptible to the end-effector state or type. To tackle this, we propose two approaches: an end-to-end RL policy (global RL), and a disentangled framework that incorporates a high-level planning algorithm, A*, and a low-level RL-based control.

The global RL policy operates on an object-centric action space. We later improve this framework by disentangling it into two small and independent components, high-level A* planner and context-agnostic low-level RL planner. We show that this disentanglement helps improve the training speed, thus making it easier to adapt to any new context. The high-level planning component of our approach involves the prediction of an optimal set of collision-free waypoints for the concerned object. On the other hand, the low-level planning and control module is responsible for executing short-term movements between consecutive waypoints without explicit knowledge of the scene.

Our proposed disentangled framework benefits from a modular design simplifying the training, execution, and interpretability by decoupling the planning and control.

A. Task Setup

We address the problem of achieving collision-free object manipulation from an initial to a final location using push-by-strike actions, in the context of a tabletop scene with several collision objects. We define a euclidean distance threshold $thresh = 5cm$ to determine if the goal has been reached.

The environment is a planar surface of dimensions of 0.5m $\times$ 0.5m, upon which two movable collision objects are randomly placed. The target object is a rectangular block with varying dimensions, where $length, breadth \in [0.08, 0.12]$, $height \in [0.02, 0.05]$, and the robot manipulator is a Universal Robots (UR5) arm with a closed two-fingered gripper. The manipulator interacts with the environment by striking the target object, using the tip of the arm. The goal is specified as a 3D pose of the pushed object.

B. Proposed Framework 1: Global RL

Our global RL planner is a closed-loop goal-conditioned planner, which learns to push an object from a given initial position to a final desired position in a collision-free way. We define this as a Markov Decision Process (MDP) with states $s_t \in S$, actions $a_t \in A$, goals $\eta \in G$, and reward function $r : S \times A \rightarrow R$. The state space $s_t = (h_t, l_t)$ consists of a height map h_t of the scene and a three-dimensional vector $l_t = (x_c, y_c, yaw_c)$, denoting the initial position, of the object that needs to be pushed, with respect to the global frame. The action space consists of 16 discrete actions, where, the i^{th} action denotes a push vector of fixed length in free space, which makes an angle $\frac{2\pi i}{16}$ with the x-axis of the

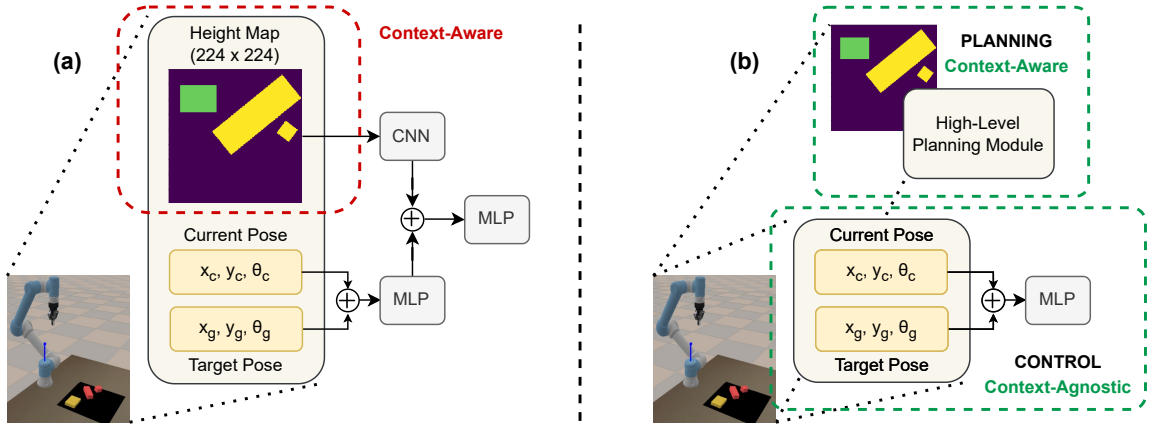

Fig. 2: (a) shows the architecture of global RL, which takes a height map, the current state of the object, and the goal state of the object as inputs, and returns the Q values for each of the 16 actions. (b) shows the architecture of disentangled framework that incorporates a high-level planning module and a low-level RL controller. (a) is aware of the scene-context making it hard to generalize to arbitrary scenes. On the other hand, the control in (b) is agnostic of the context.

global frame. We train this policy on cuboidal objects, where each cuboid is represented as a set of 8 points (4 corners, and 4 mid-points). When a push vector is chosen, the point among the 8 points which offers the lowest torque about the object's center of mass when pushed in that direction, is chosen as the point for pushing. A goal $\eta \in G$ is defined by a 3-dimensional vector (x_g, y_g, yaw_g) denoting the object's goal pose with respect to the global frame.

The objective is to find a goal-conditioned policy $\pi(s_t, \eta)$ that maximizes the return $R_t = \sum_{k=t}^{\inf} \gamma^{k-t} r(s_k, a_k)$. We find the optimal policy by modeling the Q function using a deep Q-learning network (DQN). Since our policy is goal-conditioned, the Q-function must also be goal-conditioned, given by $Q^\pi(s_t, a_t, \eta)$. Therefore, the DQN takes both the state s_t and goal η as input and returns the Q-value for each of the 16 actions. The policy is then given by:

$$\pi(s_t, \eta) = \arg\max_{a_t} Q^\pi(s_t, a_t, \eta) \tag{1}$$

Fig. 2(a) illustrates the proposed architecture. The architecture consists of three main modules: CNN1, MLP1, MLP2. We first pass the height map of dimensions (224, 224, 1) to CNN1, a three-layered convolutional network, which processes it and outputs a (13, 13, 64) map. The MLP1 module, which is a two-layered MLP, takes a 6-dimensional $(l_t, \eta) = (x_c, y_c, yaw_c, x_g, y_g, yaw_g)$ as input and gives a 16-dimensional output. The output from CNN1 is flattened and appended to the output from MLP1, to form a unified 10832-dimensional vector, which is then passed into MLP3, a three-layered MLP, to get a 16-dimensional vector as output, which denotes the q-values for each of the 16 actions.

Each convolutional layer in CNN1 is followed by batch normalization, activation by the ReLU function, and max pooling. All the linear layers in MLP1 and MLP2, except for the output layer, also use the same ReLU as their corresponding activation functions.

We use Huber loss to train the DQN. We follow the double DQN algorithm, where we use two identical networks, one for collecting the data (target network) and the other for the training (policy network). Instead of using a hard-update rule, we instead use a soft-update rule as used in DDPG [17], given by:

$$tNet = pNet * \tau + tNet * (1 - \tau) \tag{2}$$

where $tNet$ = target network, $pNet$ = policy network, and τ is set to 0.005 to reduce the variance in training data.

The reward function consists of four components:

1. Goal Distance reward (GDR) is the difference between the previous and the current distance to the goal, given by:

$$GDR(s_n, s_{n-1}, \eta) = D_2G(s_{n-1}, \eta) - D_2G(s_n, \eta) \tag{3}$$

Here, $D_2G(s_k, \eta) = \|l_k(0:2) - \eta(0:2)\|$

2. Collision Reward (CR) is the difference between the obstacle positions before and after the push, given by:

$$CR(s_{n-1}, s_n) = \sum_i \|obs_{n-1}(i) - obs_n(i)\| \tag{4}$$

Here, $obs_k(i)$ denotes the pose of i^{th} obstacle at state s_k.

3. Goal Reward (GR) is a reward given when the object reaches its goal state. This is given by:

$$GR(s_n) = \begin{cases} +100, & \|s_n - \eta\| \leq 0.05 \\ 0, & otherwise \end{cases} \tag{5}$$

4. Out-of-bounds penalty (OOB) is a reward given when the object goes out of bounds. This is given by:

$$OOB(s_n) = \begin{cases} -100, & \text{if object goes out of bounds} \\ 0, & otherwise \end{cases} \tag{6}$$

C. Proposed Disentangled Framework: A* + Low-level RL

Our proposed approach involves two modules: a path-planning module and a low-level RL controller. The path planning module generates an optimal collision-free trajectory for the robot's end-effector, while the low-level RL controller controls the robot's movements to accomplish the task objectives. Fig. 3 illustrates the proposed approach.

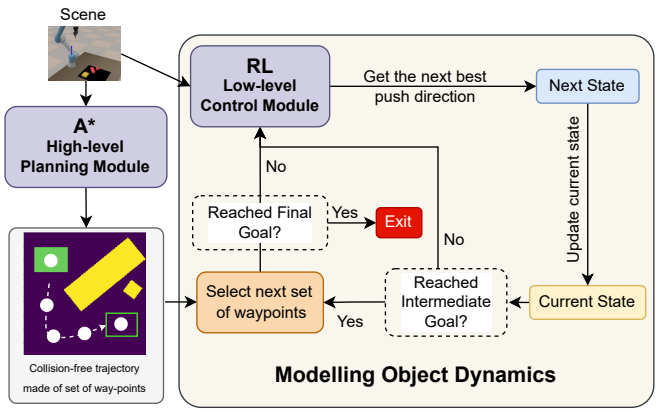

Fig. 3: Modular planning framework to tackle non-prehensile object manipulation in a cluttered scene. We make use of a low-level RL controller trained to reach a short goal. The high-level planning module predicts a set of collision-free waypoints for the object. These waypoints are fed as goals to the low-level RL that pushes the objects between each intermediate goals.

1) Low-level RL Controller: Our RL controller learns to simply push a target object from a start to a goal location on a tabletop scene. Our controller is trained without any obstacles on the table and is optimized to approach the goal location quickly. The MDP formulation is identical to the one designed for global RL. We also use the same action space. However, the state is simpled from $s_t = (h_t, l_t)$ to $s_t = (l_t)$. The reward function for low-level RL involves only the GDR and the GR mentioned in the previous subsection. We use a discount factor $\gamma = 0.9$, which instructs the policy to reach the goal as quickly as possible.

We train a DQN in the same manner as global RL, with a target network for exploration and a policy network for learning from experience. The optimal policy is given by:

$$\pi(s_t, \eta) = \arg\max_{a_t} Q^{\pi}(s_t, a_t, \eta) \tag{7}$$

Fig. 2 (b) illustrates the proposed architecture for low-level RL. We use a simple three-layered MLP, with 128 units per layer. The first two layers are activated by ReLU, while the third layer does not use any activation function.

2) A Path Planning Module:* The high-level A* planner generates a feasible minimum collision trajectory for the pushed object, given the initial and final configurations, along with shapes, locations, and sizes of the obstacles in the scene.

We define a graph with a minimum distance of 0.05 meters between adjacent nodes, bounded by the limits of the workspace. Every node has 5 equally spaced adjacent neighbours separated by an angle of $\frac{360^\circ}{5} = 72^\circ$. These parameters have been set to improve convergence speed for a given scene and can be further tuned based on the trade-off between accuracy and speed.

We use Euclidean distance to the goal as our heuristic function. The edge cost between two nodes is the sum of the Euclidean distance between them and the collision cost, if any. The collision radius R_c^i for collision object i is the maximum distance from its centroid to its boundary added to the maximum distance between the pushed object's centroid and boundary. d_i is the maximum distance between the pushed object's centroid to its boundary. The collision cost is therefore given as:

$$CollisionCost = \sum_{i}^{n} \begin{cases} R_c^i - d_i & d_i \leq R_c^i \\ 0 & otherwise \end{cases} \tag{8}$$

where, n = number of collision objects in the scene Therefore, we predict waypoints that reach the goal with minimum distance covered and with the lowest collision rate possible

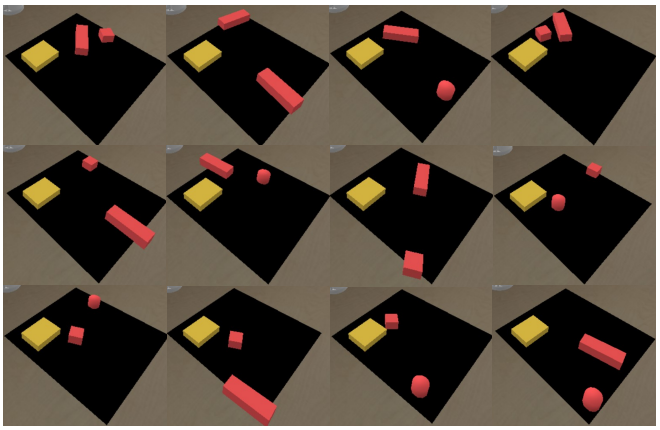

Fig. 4: Images of 12 different experiments used for comparing our disentangled framework against global RL

	Success Rate (%)	Avg APD ($\times 10^{-3}$)	Avg AAD
Global RL	62.5	53.07	0.53
Ours	**100**	**6.06**	**0.49**

TABLE I: Comparison of our proposed disentangled framework (A*+Low-level RL) against global RL in terms of the Success Rate(↑), Average APD(↓) across the 12 experiments presented in Fig. 4, and Average AAD(↓) across all experiments.

III. Experiments

We build the simulation environment for this task using PyBullet[1]. The environment contains a UR5 arm with a closed two-fingered gripper, a table, and a set of objects on top of the table. We focus on pushing standard cuboidal objects, with obstacles in three different shapes: cubes, cuboids, and cylinders, placed in valid randomized locations to create 12 distinct scenes as shown in Fig. 4. The global RL is a Deep Q-learning Network (DQN) trained in simulation for 10000 episodes, equivalent to 183000 environment steps. The low-level RL in the proposed disentangled framework, with zero discount factor, is also a DQN trained on 500 episodes, equivalent to 5350 environment steps. The training was done on a single NVIDIA GeForce GTX 1080 Ti GPU, with 10 CPUs.

A. Metrics

In this section, we define the metrics. We focus on the collisions and the success rate by defining three metrics: Av-

[1]https://pybullet.org/wordpress/

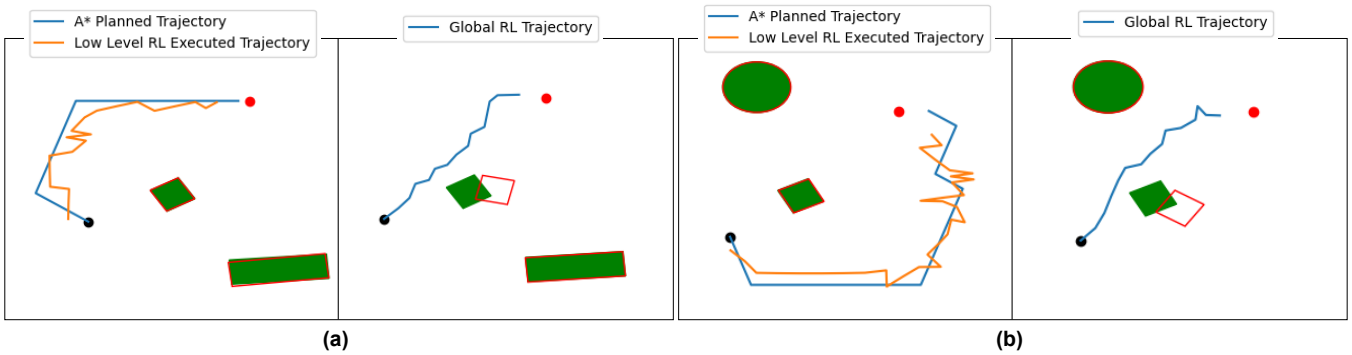

Fig. 5: We qualitatively compare the trajectories predicted and executed by our disentangled framework against the global RL. The green objects in the figure denote the obstacles, and the red borders denote the final positions and orientations of the obstacles once the full path has been executed. Black and red dots indicate the starting position and goal positions of the target object, respectively. The orange paths in the left cells of (a) and (b) indicate the trajectory taken by the low-level RL policy, and the blue lines indicate the trajectory predicted by A*. In the right cells, the blue lines indicate the trajectory executed by the global RL policy. As can be seen in the disentangled framework, the A* algorithm predicts a collision-free trajectory and the low-level RL executes the predicted trajectory with no collision. On the other hand, although the global RL finds a shorter trajectory, it fails to avoid collisions, significantly moving the obstacles (as indicated by the displacement of the collision objects) while executing the path.

erage Positional Disturbance, Average Angular Disturbance, and Success rate.

1. Average Positional Disturbance (APD): For a given scene, let $pose_{init}(i)$ be the initial 2D cartesian pose of the object i, and $pose_{final}(i)$ be the final 2D cartesian pose calculated at the end of the episode. Then, the average positional disturbance for a given scene j is calculated as follows:

$$APD(j) = \sum_{i} \|pose_{final}(i) - pose_{init}(i)\|$$

We then calculate the average of APD over a set of scenes and report it as Average APD.

2. Average Angular Disturbance (AAD): For a given scene, let $yaw_{init}(i)$ be the initial yaw of the object i, and $yaw_{final}(i)$ be the final yaw calculated at the end of the episode. Then, the average angular disturbance for a given scene j is calculated as follows:

$$AAD(j) = \sum_{i} \|yaw_{final}(i) - yaw_{init}(i)\|$$

We then calculate the average of AAD over a set of scenes and report it as Average AAD.

3. Success Rate: If the concerned object reaches its goal without going out of bounds, then, we call this a success. We report the success rate, which is the number of successful episodes divided by the total number of episodes.

B. Qualitative Analysis

We demonstrate the qualitative results in Fig. 5. Here, we showcase the trajectory predicted by A* (a) and (b), left cells in blue. The path executed by the low-level RL is denoted by the orange lines. The shape of the A* trajectory is a consequence of collision constraints between the target object and the obstacles (green boxes) on the table. The low-level RL planning controller effectively moves the item to the objective without colliding by following the path fed by A*. (a) and (b), right cells depict the global RL trajectory for the same arrangement of start, goal, and obstacle poses. While the global RL seems to compute shorter plans to the goal, there are collisions between the pushed object and other obstacles on the table. This demonstrates that the proposed disentangled framework, A* + low-level RL, performs better than global RL, despite being trained for a fraction of global RL's training time.

C. Quantitative Analysis

In this section, we quantitatively compare global RL and our proposed disentangled framework, A* + low-level RL, on the test set made of 12 environments, as shown in Fig. 4. The metrics are defined in Sec. III-A. As can be seen, A* + low-level RL outperforms global RL on all the metrics, despite being trained for just a few hours, as opposed to global RL trained for $\sim$4 days.

IV. CONCLUSIONS

In this work, we first proposed a novel object-centric end-to-end global RL solution to solve the problem of pushing-by-striking without tactile feedback. We then disentangled the problem into two small and independent modules, A* and low-level RL, and showed that such a disentanglement helps speed up training and adaptation to new contexts. Disentangling the collision-free path planning + object control into two separate modules helped make the task easy for the RL model. Thus, they were able to achieve a performance better than the global RL with just a few hours of training. Such a disentanglement also makes the model more transparent and interpretable. However, this is still the beginning of interpretability. Replacing A* with a learning-based module that takes the behavior of low-level RL into account while planning, would form an important future direction, as this would move us in the direction of interpretable and transparent entanglement, which combines the advantages of global RL and the disentangled framework.

REFERENCES

[1] A. Agarwal, B. Sen, S. Narayanan V, V. R. Mandadi, B. Bhowmick, and K. M. Krishna, "Approaches and challenges in robotic perception for table-top rearrangement and planning," *arXiv preprint arXiv:2205.04090*, 2022.

[2] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, "Legged locomotion in challenging terrains using egocentric vision," *CoRL*, 2022.

[3] ——, "Legged locomotion in challenging terrains using egocentric vision," in *Proceedings of The 6th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, K. Liu, D. Kulic, and J. Ichnowski, Eds., vol. 205. PMLR, 14–18 Dec 2023, pp. 403–415. [Online]. Available: https://proceedings.mlr.press/v205/agarwal23a.html

[4] S. Bahl, M. Mukadam, A. Gupta, and D. Pathak, "Neural dynamic policies for end-to-end sensorimotor learning," in *NeurIPS*, 2020.

[5] F. Bai, F. Meng, J. Liu, J. Wang, and M. Q.-H. Meng, "Hierarchical policy for non-prehensile multi-object rearrangement with deep reinforcement learning and monte carlo tree search," *arXiv preprint arXiv:2109.08973*, 2021.

[6] ——, "Hierarchical policy with deep-reinforcement learning for nonprehensile multiobject rearrangement," *Biomimetic Intelligence and Robotics*, vol. 2, no. 3, p. 100047, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667379722000134

[7] T. Chen, M. Tippur, S. Wu, V. Kumar, E. Adelson, and P. Agrawal, "Visual dexterity: In-hand dexterous manipulation from depth," *arXiv preprint arXiv:2211.11744*, 2022.

[8] T. Chen, J. Xu, and P. Agrawal, "A system for general in-hand object re-orientation," *Conference on Robot Learning*, 2021.

[9] N. Dengler, D. Großklaus, and M. Bennewitz, "Learning goal-oriented non-prehensile pushing in cluttered scenes," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 1116–1122.

[10] K. Hang, A. S. Morgan, and A. M. Dollar, "Pre-grasp sliding manipulation of thin objects using soft, compliant, or underactuated hands," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 662–669, 2019.

[11] B. Huang, S. D. Han, A. Boularias, and J. Yu, "Dipn: Deep interaction prediction network with application to clutter removal," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 4694–4701.

[12] B. Huang, S. D. Han, J. Yu, and A. Boularias, "Visual foresight trees for object retrieval from clutter with nonprehensile rearrangement," *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 231–238, 2021.

[13] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," 2018. [Online]. Available: https://arxiv.org/abs/1812.03201

[14] S. Joshi, S. Kumra, and F. Sahin, "Robotic grasping using deep reinforcement learning," *CoRR*, vol. abs/2007.04499, 2020. [Online]. Available: https://arxiv.org/abs/2007.04499

[15] J. King, M. Klingensmith, C. Dellin, M. Dogar, P. Velagapudi, N. Pollard, and S. Srinivasa, "Pregrasp manipulation as trajectory optimization," in *Proceedings of Robotics: Science and Systems (RSS '13)*, June 2013.

[16] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," *RSS*, 2021.

[17] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[18] J. Moura, T. Stouraitis, and S. Vijayakumar, "Non-prehensile planar manipulation via trajectory optimization with complementarity constraints," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 970–976.

[19] A. Orsula, S. Bøgh, M. Olivares-Mendez, and C. Martinez, "Learning to Grasp on the Moon from 3D Octree Observations with Deep Reinforcement Learning," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 4112–4119.

[20] R. Papallas and M. Dogar, "Non-prehensile manipulation in clutter with human-in-the-loop," 05 2020, pp. 6723–6729.

[21] M. N. Qureshi, B. Eisner, and D. Held, "Deep sequenced linear dynamical systems for manipulation policy learning," in *ICLR 2022 Workshop on Generalizable Policy Learning in Physical World*, 2022. [Online]. Available: https://openreview.net/forum?id=rF-fT4pN1Wc

[22] C. Song and A. Boularias, "Object rearrangement with nested non-prehensile manipulation actions," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 6578–6585.

[23] Z. Xu, W. Yu, A. Herzog, W. Lu, C. Fu, M. Tomizuka, Y. Bai, C. K. Liu, and D. Ho, "Cocoi: Contact-aware online context inference for generalizable non-planar pushing," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 176–182.

[24] W. Yuan, J. A. Stork, D. Kragic, M. Y. Wang, and K. Hang, "Rearrangement with nonprehensile manipulation using deep reinforcement learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 270–277.

[25] W. Zhou and D. Held, "Learning to grasp the ungraspable with emergent extrinsic dexterity," 2022. [Online]. Available: https://arxiv.org/abs/2211.01500