Granular Loco-Manipulation: Repositioning Rocks Through Strategic Sand Avalanche

Haodi Hu[1], **Yue Wu**[1], **Feifei Qian**[1,†], **Daniel Seita**[1,†]
feifeiqi@usc.edu
† Equal Advising
[1]University of Southern California, United States

Abstract: Legged robots have the potential to leverage obstacles to climb steep sand slopes. However, efficiently repositioning these obstacles to desired locations is challenging. Here we present DiffusiveGRAIN, a learning-based method that enables a multi-legged robot to strategically induce localized sand avalanches during locomotion and indirectly manipulate obstacles. We conducted 375 trials, systematically varying obstacle spacing, robot orientation, and leg actions in 75 of them. Results show that movement of closely-spaced obstacles exhibit significant interference, requiring joint modeling. In addition, different multi-leg excavation actions could cause distinct robot state changes, necessitating integrated planning of manipulation and locomotion. To address these challenges, DiffusiveGRAIN includes a diffusion-based environment predictor to capture multi-obstacle movements under granular flow interferences and a robot state predictor to estimate changes in robot state from multi-leg action patterns. Deployment experiments (90 trials) demonstrate that by integrating the environment and robot state predictors, the robot can autonomously plan its movements based on loco-manipulation goals, successfully shifting closely located rocks to desired locations in over 65% of trials. Our study showcases the potential for a locomoting robot to strategically manipulate obstacles to achieve improved mobility on challenging terrains. Supplementary material is available at https://sites.google.com/view/diffusivegrain/home.

Keywords: Granular media, avalanche dynamics, diffusion models, legged robots

1 Introduction

Natural environments contain deformable sand, steep inclines, and large rocks and boulders, which present significant challenges for terrestrial robot locomotion. Recent robotics research has explored bio-inspired [1, 2] "obstacle-aided locomotion" strategies, enabling robots to utilize interactions and collisions with large rocks and boulders to improve mobility on complex terrains [3, 4, 5, 6, 7, 8]. While obstacle-aided locomotion offers promising opportunities for robots [7, 9, 10] to negotiate challenging terrains, these strategies often rely on specific leg-obstacle contact positions [5, 11]. Mischosen contact positions can lead to catastrophic failures due to slipping, getting stuck, or even flipping over (Fig. 2). As a result, their effectiveness depends heavily on the availability and spatial distribution of rocks and boulders, which can vary unpredictably across natural terrains.

A recent study [12] found that by strategically triggering localized sand avalanches [13, 14, 15], a robotic leg can generate granular flow on steep sand slopes, repositioning large rocks and boulders to desired locations. This introduces a new approach for indirect obstacle manipulation through granular flow. By integrating this capability with obstacle-aided locomotion strategies, legged robots can modify their environments to enhance traversability. However, most existing granular manipulation methods focus either on homogeneous granular media [16, 17, 18, 19] or assume that obstacle movements under granular flow are independent of robot locomotion and movement of other obsta-

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

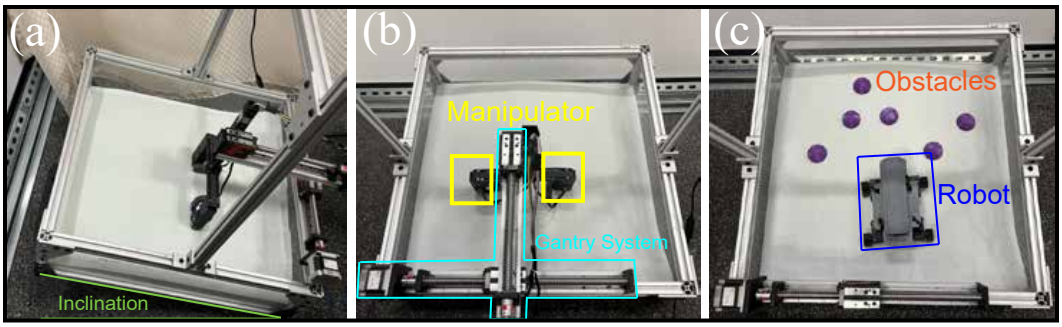

Figure 1: Experiment environment, with (a) a side view of the granular trackway with an inclination angle of $\Phi = 20$ degrees; (b) the granular trackway with two robotic legs mounted on an actuated gantry system; (c) the robot (*not* the manipulator in (b)) in the granular trackway, and 3D-printed obstacles (purple semi-spheres).

cles [12]. How to extend granular flow based manipulation capabilities to locomoting robots and substrates with densely-distributed obstacles remains unexplored.

To address this, we investigate multi-obstacle manipulation on granula r slopes using a locomoting quadrupedal robot, where each leg has 1 DoF and functions as an excavator to interact with sand and indirectly reposition obstacles (Fig. 1C). Our findings reveal two critical challenges: (1) obstacle movement can be influenced by nearby obstacles on the sand, requiring a new method to resolve multiple obstacle movement under leg excavation simultaneously; and (2) the robot state (i.e., its position and orientation) can also be substantially affected by multi-leg excavation actions, necessitating a method that predicts and plans for both environment and robot state changes.

To tackle these challenges, we propose DiffusiveGRAIN, a learning-based method that enables multi-legged robots to reposition densely-distributed rocks on sand slopes to desired locations during its locomotion. DiffusiveGRAIN has a novel granular media dynamics predictor that learns both obstacle and robot movement under leg excavation actions. It takes as input, a depth image of the environment state, and a separate image representation of the leg excavation action. The predictor has two U-Net [20] models. One is a diffusion [21, 22, 23] model that predicts the environment state change, and the other predicts the robot state change. During deployment, the dynamics predictor predicts the robot state change given its action. Since we train the environment predictor using data from a "manipulator" (see Fig. 1B) instead of the robot (Fig. 1C), we adjust the action representation passed to the environment predictor, which then predicts granular surface depth change.

In summary, our contributions are: (i) A novel manipulation predictor using a U-Net with diffusion for a legged robot to predict avalanche dynamics of a sand slope with obstacles via leg-terrain interaction. (ii) A novel locomotion predictor based on a U-Net for a legged robot to predict its state change on a sand slope given the robot's action. (iii) Integration of the proposed manipulation and locomotion predictors to enable a legged robot to achieve loco-manipulation tasks on a sand slope.

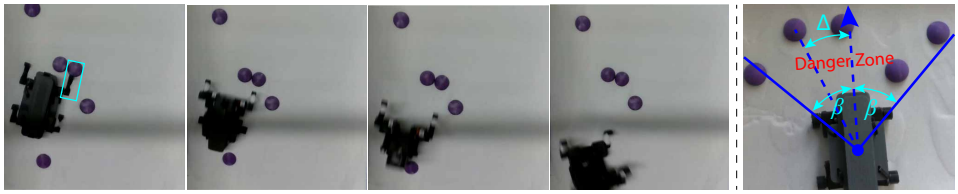

Figure 2: Left (4 images): An example of the robot flipping backward (over its two back legs) on a steep sand slope due to undesired leg-obstacle contact. The cyan rectangle highlights the leg's contact with the obstacle. Right: Stepping on undesired locations can result in a high risk of robot slipping, stuck, or flipping backwards.

2 Preliminaries

Experiment Setup: Fig. 1 illustrates our physical gantry system. The granular trackway is ($60\,\mathrm{cm}$ L $\times$ $60\,\mathrm{cm}$ W $\times$ $20\,\mathrm{cm}$ D) and contains model granular medium (Grainger, $0.3\,\mathrm{mm}$ glass beads). The $0.3\,\mathrm{mm}$ particle size is similar to those observed in natural deserts, and behave qualitatively similar

to natural sand and are widely used in granular robotics studies [24, 25, 26]. The granular trackway can be tilted up to 35 degrees to emulate a wide variety of sand slopes in natural environments [15]. In this experiment, we chose to use a slope angle $\Phi = 20$ degrees, which is close to the angle of repose [27] of our granular material and facilitates the study of avalanche dynamics. We mount a top-down RealSense 435i RGBD camera to record granular flow and obstacle movement.

Manipulator versus Robot: To study avalanche dynamics and object movement upon different leg excavation actions, we build a gantry system with two linear actuators, which move along the x and y axes. This gantry moves a *manipulator* on a 2D surface *above* the granular slope. This manipulator contains two C-shape robot leg motors with rotation centers $1.0\,\mathrm{cm}$ above the granular surface. Each has a diameter of $6.0\,\mathrm{cm}$ and a width of $1.5\,\mathrm{cm}$. In this paper, the "manipulator" is distinct from the actual *robots* (i.e., quadrupeds) we use in experiments. The manipulator's two C-shape leg designs match the front two C-shape legs of the robot, and it can also execute the same excavation action as the robot. The manipulator enables faster and safer data collection compared to using the robot.

Action Space: The action space $\mathcal{A}$ for the robots (and manipulators) has six actions, each containing a different group of the four robot legs, (Fig. 3 bottom left): (1) *Left Front Excavation (LFE)*, where the left front (LF) leg rotates backwards relative to the robot heading; (2) *Right Front Excavation (RFE)* where the right front (RF) leg rotates backwards through the granular medium; (3) *Left Pair (LP)*, where the robot's left front (LF) and left hind (LH) legs rotate backwards synchronously; (4) *Right Pair (RP)*, where the robot's right front (RF) and right hind (RH) legs rotate backwards synchronously; (5) *Front Pair (FP)*, where the robot's left front (LF) and right front (RF) legs rotate backwards synchronously; and (6) *All Four (AF)*, where all four robot legs rotate backwards synchronously. For all actions, the rotating angular speed of each leg is constant, $1\pi\ rad/s$.

3 Relevant Work and Knowledge Gap

Prior work has investigated terrain manipulation techniques, wherein a robotic leg or wheel supports body weight and actively reconfigures, compacts, or fluidizes the substrate to bolster stability and traction [12, 16, 28, 29, 30]. Such innovations leverage terra-dynamics [25], the study of movement in granular substrates to inform real-time control decisions, enabling robots to traverse and reshape their environment for more efficient locomotion [31]. This approach has broad implications for challenging real-world operations, from extraterrestrial exploration to search-and-rescue missions. One particularly interesting work in this domain is GRAIN [12], a learning-based approach that predicts sand slope change based on leg excavations. GRAIN used a Vision Transformer (ViT) [32] to process image representations of granular dynamics and robot excavation actions. The ViT predicts 2D object movement on a granular medium slope corresponding to different excavation actions.

While GRAIN worked well for single leg, single obstacle manipulation, it assumes the following which may be invalid for multi-leg, multi-obstacle scenarios: (i) GRAIN assumed that obstacle movements are independent and not affected by the movement or granular flow of adjacent obstacles. (ii) GRAIN assumed that obstacle movement for each leg excavation could be trained independently and did not consider the effects of simultaneous multi-leg manipulation. (iii) GRAIN assumed that the robot state remained unaffected under leg excavation actions, which may be invalid since a locomoting robot could slip and turn as it performs excavations. In addition, such state change may depend on which leg (or legs) was used for excavation. To test these assumptions under multi-obstacle, multi-leg manipulation tasks, we conducted two sets of experiments for GRAIN.

Multi-obstacle with systematically-varied distance, where we systematically varied the positions between two adjacent obstacles from 0 to 8cm, with a 2cm increment (Fig. 3 Left). Experimentally measured obstacle movement suggested that as the distance between obstacles decreases, the influence on obstacle movement becomes stronger (Fig. 3 top row right, 2cm and 0cm). This effect is particularly significant when additional obstacles are added in the direction of sand inclination (Fig. 3 top row left, y axis), where the displacement of an obstacle was observed to decrease to 67$\%$ and 42$\%$ as compared to no obstacle (Fig. 3 top row right, 2cm and 0cm, respectively). This suggests that treating obstacle movements as independent [12] only works when obstacles are sufficiently far apart. When obstacles are close, adjacent ones significantly affect avalanche propagation [33].

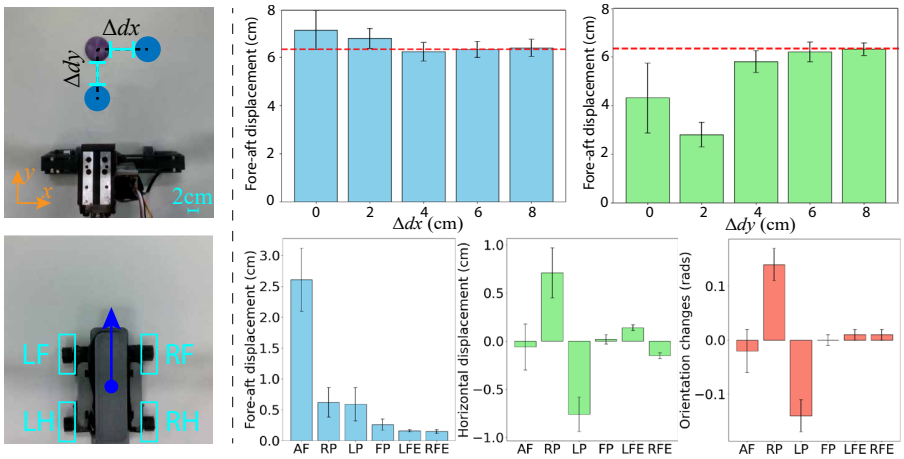

Figure 3: Experiment setup for investigating GRAIN [12] (see Sec. 3). Left top: multi-obstacle manipulation experiment setup, where Δdx is the lateral distance and Δdy is the fore-aft distance between the two obstacles; Left bottom: the robot state change experiment, where we investigate the robot locomotion state change under 6 different combinations of leg excavation actions; Right top: 2 obstacles distance in horizontal and fore-aft directions affects obstacle movement, red dash line represents the obstacle movement without the affect of the other obstacle; Right bottom: statistics of robot state change by different robot actions.

Robot state change under leg excavation actions, where we tested six sets of multi-leg excavation actions (see Sec. 2) to investigate the effect of leg manipulation actions on robot locomotion. Experimental measurements from the 6 multi-leg manipulation actions showed that the *AF* action resulted in the largest displacement in the robot fore-aft position. The *RP* and *LP* actions resulted in the largest change in the robot orientation and lateral position, while not changing much of the robot fore-aft position. The *FP*, *LFE*, *RFE* are manipulation actions that trigger avalanche behavior at different locations while keeping the robot position change small. See Fig. 3 (bottom right) for aggregated results over 3 trials for each robot action, and Fig. 9 in the Appendix for visualizations. These results indicated that the change in robot state can sensitively depend on the group of robot legs used in manipulation. For loco-manipulation tasks, it is critical to consider and plan these robot state changes jointly with manipulation goals.

These experiments demonstrated the limitations of GRAIN, and highlighted two key challenges that require new methods: (i) the need to consider the granular flow influence from adjacent obstacles and (ii) the need to jointly plan manipulation and locomotion actions.

4 DiffusiveGRAIN: Learning to Predict Obstacle Movement

4.1 State Representation and Robot Action Space

A top-down camera provides a depth image $\mathbf{I}_t$ at time t (see Fig. 4 for examples). We denote the *robot state* in $\mathbf{I}_t$ as $\mathbf{x}_t \in \mathbb{R}^3$, consisting of 2D position and 1D orientation. We denote the starting state as $\mathbf{x} = (x^*, y^*, \phi^*)$. The *obstacles' states* in $\mathbf{I}_t$ is represented as $\mathbf{o}_t$ where $\mathbf{o}_t^i \in \mathbb{R}^2$ represents the i-th obstacle state (represented as a 2D position). The robot action space $\mathcal{A}$ has 6 actions as described in Sec. 2. To test DiffusiveGRAIN, we propose to study the following tasks:

Manipulation: A robot starting at $\mathbf{x}$ needs to manipulate N obstacles with initial states $\mathbf{o}^i = (o_x^i, o_y^i)$ to target states $\mathbf{d}_i^o = (d_x^i, d_y^i)$, where i is an integer, i.e., $i \in \{1, N\}$.

Locomotion: A robot starting at $\mathbf{x}$ needs to travel to a desired location $\mathbf{d}^r = (d_x, d_y)$.

Loco-manipulation: A robot at $\mathbf{x}$ needs to travel to a desired location $\mathbf{d}^r = (d_x, d_y)$ and manipulate N obstacles with initial states $\mathbf{o}^i = (o_x^i, o_y^i)$ to targets $\mathbf{d}_i^o = (d_x^i, d_y^i)$, where $i \in \{1, N\}$. We consider two variants: *in-distribution* and *out-of-distribution*. The former uses sand slope angles and objects from training, while the latter tests generalization by using different angles and objects.

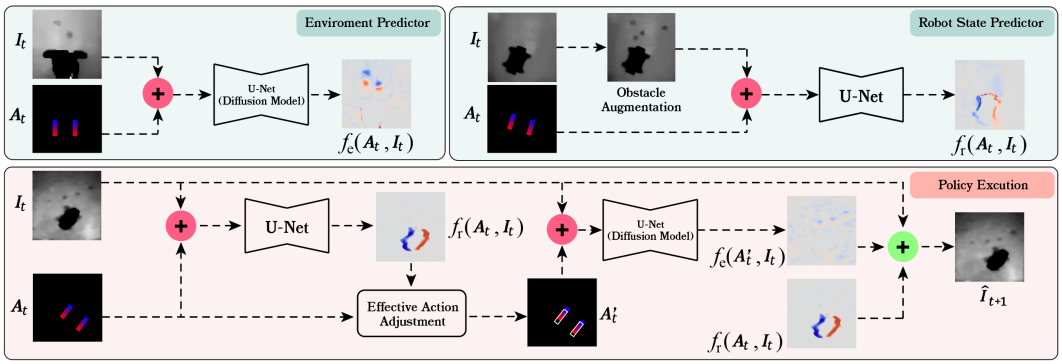

Figure 4: System overview. The *environment predictor* f_e uses a diffusion model (with a U-Net backbone) to predict the depth change of the environment given the depth image and action. The *robot state predictor* f_r uses a U-Net to predict the robot state change given the robot state and action. During *policy execution*, given the predicted robot state change, we introduce an "Effective Action Adjustment (EAA)" (see Sec. 4.3). We then combine the updated robot action image with the depth image to the trained diffusion model and get the predicted depth image change. We combine this with the predicted robot state and the original depth image to get the predicted next depth image. The red addition symbols represent channel-wise image concatenation operation, and the green addition symbols represent the image combination method as described in Sec. 4.5.

4.2 Environment State and Robot State Predictors

Environment State Predictor f_e. We improve over GRAIN [12] by using a diffusion model instead of a Vision Transformer [32] and by directly collecting data with multiple obstacles on the granular slope. Our f_e has a U-Net backbone [20] diffusion model. The inputs are the depth image $\mathbf{I}_t$ and an RGB image $\mathbf{A}_t$ representing the robot action. For $\mathbf{A}_t$, we use a space-aligned gradient color region from blue to red to represent the robot leg interaction area with the granular slope. The color region's length is the effective action length ($12.0\,\mathrm{cm}$) and its width is the robot leg width ($1.5\,\mathrm{cm}$). We train f_e to predict the change in the depth image of the granular slope surface $f_e(\mathbf{I}_t, \mathbf{A}_t)$. The predicted image is converted to a grayscale image, which we then add to the original input depth image to get the predicted depth image of the environment for the *next* state: $\mathbf{I}_t + f_e(\mathbf{I}_t, \mathbf{A}_t)$.

Robot State Predictor f_r. We use a second U-Net, f_r, to predict the robot state change given its action and the environment state. As in f_e, the inputs are the depth image $\mathbf{I}_t$ and the RGB image action representation $\mathbf{A}_t$. During training, we use OpenCV code [34] to augment the input data to f_r by adding obstacles to the depth image of the collected dataset while keeping the same label. As a result, the U-Net learns from the input depth images with extra obstacles. We use this method based on our observation during the experiments that the obstacles do not noticeably affect robot's state change unless the robot leg directly contacts obstacles. The output is the predicted robot state change $f_r(\mathbf{I}_t, \mathbf{A}_t)$. We can similarly convert this to a grayscale and obtain the predicted depth image representing the robot, $\mathbf{I}_t + f_r(\mathbf{I}_t, \mathbf{A}_t)$, for the *next* state. See Fig. 4 for an overview.

4.3 Effective Action Adjustment (EAA)

We observe that *robot* action-triggered sand avalanche behavior can significantly differ from the sand avalanche behavior triggered by the *manipulator* with the same excavation action, especially when the robot executes *AF, LP, RP* actions. The reason is that robot leg excavation actions can lead to a different amount of advancement or slippage in each leg, resulting in significant changes to the robot state during excavation. We propose an Effective Action Adjustment (EAA) method to compensate for this prediction error. Based on the robot action, we know there are two leg-sand interaction events in a full rotation of a robot leg, and the robot state change is because the robot leg rotation provides a robot propulsion and rotation force to change its position and orientation. The robot has an initial state $\mathbf{x}_0$ and during the first leg-sand interaction the robot state changes to $\mathbf{x}_1$ and later changes to $\mathbf{x}_2$ during the second leg-sand interaction. The EAA assumes the leg excavation action triggered sand avalanche during the robot transition from $\mathbf{x}_0$ to $\mathbf{x}_2$ is equal to the leg excavation action triggered a sand avalanche at the fixed robot state $\mathbf{x}_1$. To get $\mathbf{x}_1$ in the DiffusiveGRAIN

policy execution stage, we assume $\mathbf{x}_1 = \frac{\mathbf{x}_0+\mathbf{x}_2}{2}$. As a result, during policy execution, we update the action image A_t by changing the shaded color area corresponding to the robot state $\mathbf{x}_1$, where the $\mathbf{x}_0$ is extracted from $\mathbf{I}_t$ and the $\mathbf{x}_2$ is extracted from $f_r(\mathbf{A}_t, \mathbf{I}_t)$. See the Appendix C for details.

4.4 Cost Functions for Different Modes

Cost function for locomotion: During robot locomotion mode, we consider (i) the distance between the robot CoM and target location and (ii) robot safety. For the distance, given the target location and robot position on the sand, $\mathbf{d}^r$ and $\mathbf{x}_p = (x^*, y^*)$, we define the cost as:

$$\mathbf{C}_{rt} = \|\mathbf{x}_p - \mathbf{d}^r\|_2. \tag{1}$$

For robot safety, we consider (i) the distance of the obstacles and robot CoM and (ii) the relative angle between the line connecting the obstacle and robot heading direction. Given the robot state $\mathbf{x} = (x^*, y^*, \phi^*)$ and an obstacle location $\mathbf{o} = (o_x, o_y)$, the angle difference between the line connecting the obstacle and robot heading direction, Δ, is defined as $\Delta = \phi - \phi^*$, where $\phi = \mathrm{atan2}\big(o_y - y^*,\ o_x - x^*\big)$. We use these quantities because according to a previous study, leg-obstacle contact position can significantly influence robot locomotion outcomes [35]. While stepping on certain locations of obstacles could aid locomotion, the robot can also be at high risk of flipping over or getting high centered on the obstacle when misstepping on undesired obstacle locations. Based on qualitative observations of these failure events (Fig. 2), we introduce a penalty for obstacles within the "danger zone" of the robot heading direction. We define the robot safety cost $\mathbf{C}_{rs}$ as:

$$\mathbf{C}_{rs} = \textstyle\sum_i \dfrac{\mathrm{AngFactor}(\Delta_i)}{\|\mathbf{x}_p - \mathbf{o}^i\|_2}, \qquad \mathrm{AngFactor}(\Delta) = \begin{cases} \exp\big(\alpha(\beta - |\Delta|)\big), & \text{if } |\Delta| \le \beta, \\ 1, & \text{otherwise.} \end{cases} \tag{2}$$

See Fig. 2 for β and Δ visualizations. We formulate the cost for locomotion by linearly combining Eq. 1 and Eq. 2: $\mathbf{C}_l = w_1\mathbf{C}_{rt} + w_2\mathbf{C}_{rs}$, where we use $w_1 = 0.6$, $w_2 = 0.4$, $\alpha = 4$, and $\beta = \frac{\pi}{4}$.

Cost function for manipulation: During robot manipulation mode, we consider (i) the distance between obstacle positions and targets, and (ii) robot safety. For the distance between obstacle positions and targets, given the target locations on the sand slope, $\mathbf{d}_i^o$, the cost is defined as in Eq. 3:

$$\mathbf{C}_o = \sum_i \|\mathbf{o}^i - \mathbf{d}_i^o\|_2. \tag{3}$$

The robot safety cost is the same as in locomotion mode (Eq. 2). The manipulation mode cost, $\mathbf{C}_m$, is a linear combination of Eq. 2 and Eq. 3: $\mathbf{C}_m = w_3\mathbf{C}_o + w_4\mathbf{C}_{rs}$; we use $w_3 = 0.8$ and $w_4 = 0.2$.

4.5 Policy Planning and Execution

Previous work [12] demonstrated the potential of a greedy strategy for certain manipulation tasks. While this can be efficient in simpler scenarios, it often fails in more complex loco-manipulation tasks that require jointly considering locomotion and manipulation. Our preliminary experiments indicate that to effectively leverage sand avalanches—particularly by controlling their impact regions—the robot must plan multiple steps in advance. Specifically, multi-step action planning ensures that locomotion and manipulation are coordinated in ways that reposition obstacles advantageously on the sand slope, which is crucial for successful loco-manipulation. As a result, we adopt a receding-horizon planning method that minimizes the cumulative cost: $\sum_{t=0}^{t=3} \gamma^t \mathbf{C}(\mathbf{x}_t, \mathbf{o}_t, \mathbf{a}_t)$ by planning actions for the next 4 steps, where γ is the discount rate and $\mathbf{C}$ is the cost function for a given task (*i.e.* manipulation or locomotion or both). We set $\gamma = 0.8$ in this paper.

Policy Execution. During deployment, given the depth image $\mathbf{I}_t$ and robot action $\mathbf{A}_t$, we first use f_r (Sec. 4.2) to predict the robot state change, $f_r(\mathbf{I}_t, \mathbf{A}_t)$. We then use our EAA (Sec. 4.3) to update the robot action representation image to $\mathbf{A}'_t$. To combine the predicted robot state change and the predicted environment state change to get the system's predicted next state (i.e., depth image) $\hat{\mathbf{I}}_{t+1}$, we introduce another step to proceed with the image representation of environment state and robot state. Specifically, we identify the pixel set that represents the robot at time t in $\mathbf{I}_t$, $\mathbf{p}_t$, and we use

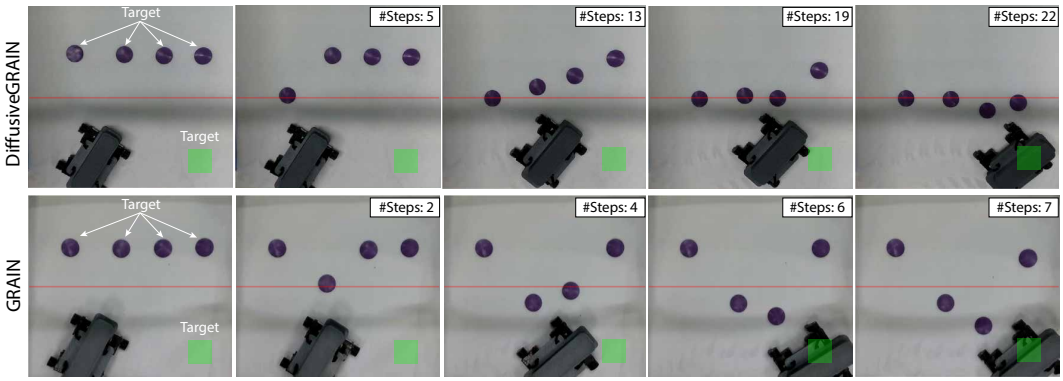

Figure 5: An example robot loco-manipulation trial for DiffusiveGRAIN and GRAIN. The robot must bring 4 obstacles below the red horizontal line while also moving to the target marked with the green square. In DiffusiveGRAIN, the robot achieved both locomotion and manipulation at step 22. In GRAIN, the robot achieved its locomotion task but only moved the middle 2 obstacles below the red line, and thus failed in manipulation.

the same method to identify the pixel set that represents the robot in $\mathbf{I}_t + f_r(\mathbf{I}_t, \mathbf{A}_t)$, $\mathbf{p}_{t+1}$. We use the $\mathbf{p}_{t+1}$ to replace the pixel in the depth image representation of the predicted environment observation image, $\mathbf{I}_t + f_e(\mathbf{I}_t, \mathbf{A}'_t)$, and we replace the pixel in $\mathbf{p}_t$ but not in $\mathbf{p}_{t+1}$ with the mean value of surrounding pixels in $\mathbf{I}_t + f_e(\mathbf{I}_t, \mathbf{A}'_t)$. With this additional step, we combine the diffusion model's predicted environment next state with the U-Net predicted robot next state and get the depth image that represents the system's next state $\hat{\mathbf{I}}_{t+1}$. In this work, the robot only has six actions (see Sec. 2), so we iterate through all possible robot action combinations for the next 4 steps.

5 Evaluation

5.1 Collecting Real-World Training Data

We use the system shown in Fig. 1 (see Sec. 2 for background). To scalably collect training data while reducing the risk of interruptions if obstacles enter the robot's danger zone (see Fig. 2), we separate data collection into (i) *leg manipulation data* using the gantry and (ii) *locomotion data* using the quadruped robot. For *leg manipulation*, we collect 240 ($60 \times 3 + 60$) trials, resulting in 24,590 images. This includes 60 trials for each gantry manipulator manipulating obstacles at orientations of 0, 15, and 30 degrees, with the left and right manipulators executing excavation actions. We also do 10 trials each for the gantry manipulator manipulating obstacles at orientations of 0, 15, and 30 degrees with just the left or right leg. For *robot locomotion*, we detach the gantry system to free space for the robot. We collect 60 trials, resulting in a total of 13,480 images. We do 5 trials for each robot orientation of 0, 15, and 30 degrees with the *AF* action, and 3 trials for each robot orientation of 0, 15, and 30 degrees with the other 5 actions.

5.2 Baselines and Evaluation Protocol

Robot movement prediction Baseline: For the locomotion model, we use GRAIN [12] to predict the robot state change. GRAIN takes the depth image concatenated with the robot action representation image as input and outputs a 1×3 matrix containing the robot's 2D position and 1D orientation. *Manipulation Planning Baseline:* We also adapt GRAIN [12] to this setup. We trained GRAIN with the new dataset and output the coordinates of each obstacle. The GRAIN framework was originally designed to train on depth images with only one obstacle, resulting in a 1×2 matrix corresponding to obstacle positions on the x-axis and y-axis. However, we have 1 to 5 obstacles. Consequently, we provided the ground truth number of obstacles to GRAIN and modified its output to be a 5×2 matrix. When there are $N < 5$ obstacles, GRAIN only considers the first N rows in the matrix.

We evaluate using Euclidean distance. To get the position of each obstacle and robot, we estimate their center of mass. For manipulation, we use the Euclidean distance among obstacle positions and their targets. For locomotion, we measure the distance between the robot and the locomotion target.

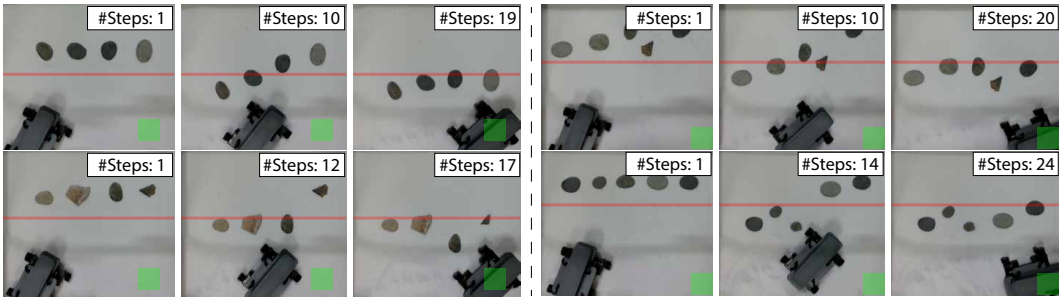

Figure 6: Loco-manipulation (out-of-distribution) experiments with more realistic settings (rocks with different shapes and sizes). We show 4 trials with real rocks, on 16 degrees (top) and 24 degrees (bottom) inclinations.

5.3 Evaluation Results

Manipulation: DiffusiveGRAIN results in 8/10 success, while GRAIN has 6/10 success. See Tab. 1 for results. Fig. 10 (in the Appendix) shows top-view images during policy execution for a robot manipulation trial with 4 obstacles (i.e., objects). In both trials, the robot must indirectly manipulate all 4 obstacles to bring them to a designated goal region near the robot. The improved performance suggested the importance for including training data with multiple obstacles, especially when obstacle distances are small (see Fig. 12, also in the Appendix).

Task	**Method**	**MAE** (cm)	**Success** (%)
Manipulation	GRAIN	2.71($\pm$ 0.91)	60
	Ours	2.03($\pm$ 0.84)	80
Locomotion	GRAIN	1.34($\pm$ 0.46)	80
	Ours	1.21($\pm$ 0.38)	90
Loco-manipulation	GRAIN	3.24($\pm$ 1.24)	20
	Ours	2.31($\pm$ 0.98)	70

Table 1: We compare the performance of DiffusiveGRAIN and GRAIN during policy execution on the tasks we study.

Locomotion: DiffusiveGRAIN obtains 9/10 success, while GRAIN gets 8/10 success. Fig. 11 (in the Appendix) shows 1 locomotion trial per method.

Loco-manipulation (in-distribution): Using DiffusiveGRAIN results in 7/10 success, compared to 2/10 for GRAIN. Fig. 5 shows 1 loco-manipulation trial using DiffusiveGRAIN and GRAIN. The robot must move all 4 obstacles to a target region (below the red line) and navigate to a target location (marked by a green square). The result suggests that DiffusiveGRAIN enables the robot to successfully plan simultaneous manipulation and locomotion, and highlights the importance of combining the environment state and robot state predictors for loco-manipulation tasks, as mischosen leg excavations may lead to undesired robot state change and preclude future manipulation options.

Loco-manipulation (out-of-distribution): To test generalization of DiffusiveGRAIN, we conduct experiments using *real rocks* with variations in shape and angularity (see Fig. 6), without any fine-tuning. We also test 2 inclination angles (16 and 24 degrees) in addition to 20 degrees tested earlier, to evaluate performance on different slope inclination angles. We perform 60 experiment trials with 20 trials for each inclination angle. The experiments show high success rates and small prediction errors (Tab. 2).

Inclination (degree)	**MAE**† (cm)	**Success** (%)
16	2.61($\pm$ 1.02)	65%
20	2.39($\pm$ 0.96)	70%
24	2.57($\pm$ 0.92)	65%

Table 2: DiffusiveGRAIN performance for loco-manipulation (out-of-distribution).

6 Conclusion

We introduce DiffusiveGRAIN, which enables a locomoting legged robot to leverage sand avalanche dynamics to reposition rocks on a sand slope. By integrating a diffusion-based environment state predictor and a robot state predictor, we show that DiffusiveGRAIN is able to achieve a significantly high success rate (70%, as compared to 20% using the baseline method) for robot loco-manipulation tasks where a change of robot state needs to be jointly considered and planned with obstacle movement. Our method opens a new avenue for robots to actively manipulate their locomotion environments to improve mobility on challenging granular slopes.

7 Limitations

One key limitation is that DiffusiveGRAIN is vision-focused and does not capture obstacle properties that may influence avalanche behavior, such as mass. In natural environments, rocks may have significantly larger mass than those used in our experiments, and such properties might not be fully apparent from vision alone. Future work should investigate how avalanche dynamics vary with obstacle mass and how to integrate such information into predictive models. Additionally, the current policy execution performs exhaustive evaluation over all possible action sequences. While effective for small action spaces and short horizons, this brute-force approach becomes computationally infeasible as the problem size grows, and future work could explore approaches to mitigate the computational burden.

Acknowledgments

This work is supported by the National Science Foundation (NSF) CAREER award # 2240075, the NASA Planetary Science and Technology Through Analog Research (PSTAR) program, Award # 80NSSC22K1313, the NASA Lunar Surface Technology Research (LuSTR) program, Award # 80NSSC24K0127, and the NASA Mars Exploration Program (MEP) Technology Development Funding.

References

[1] P. E. Schiebel, J. M. Rieser, A. M. Hubbard, L. Chen, and D. I. Goldman. Collisional diffraction emerges from simple control of limbless locomotion. In *Conference on Biomimetic and Biohybrid Systems*, pages 611–618. Springer, 2017.

[2] P. E. Schiebel, J. M. Rieser, A. M. Hubbard, L. Chen, D. Z. Rocklin, and D. I. Goldman. Mechanical diffraction reveals the role of passive dynamics in a slithering snake. *Proceedings of the National Academy of Sciences*, 116(11):4798–4803, 2019.

[3] F. Qian and D. E. Koditschek. An obstacle disturbance selection framework: emergent robot steady states under repeated collisions. *The International Journal of Robotics Research*, 2020.

[4] R. Othayoth, G. Thoms, and C. Li. An energy landscape approach to locomotor transitions in complex 3d terrain. *Proceedings of the National Academy of Sciences*, 117(26):14987–14995, 2020.

[5] J. M. Rieser, P. E. Schiebel, A. Pazouki, F. Qian, Z. Goddard, K. Wiesenfeld, A. Zangwill, D. Negrut, and D. I. Goldman. Dynamics of scattering in undulatory active collisions. *Physical Review E*, 99(2):022606, 2019.

[6] T. Wang, C. Pierce, V. Kojouharov, B. Chong, K. Diaz, H. Lu, and D. I. Goldman. Mechanical intelligence simplifies control in terrestrial limbless locomotion. *arXiv preprint arXiv:2304.08652*, 2023.

[7] H. Hu and F. Qian. Obstacle-aided trajectory control of a quadrupedal robot through sequential gait composition. *IEEE Transactions on Robotics*, pages 1–15, 2024.

[8] H. Hu, X. Liao, W. Du, and F. Qian. Multi-robot connection towards collective obstacle field traversal, 2024. URL https://arxiv.org/abs/2409.11709.

[9] D. Ramesh, A. Kathail, D. E. Koditschek, and F. Qian. Modulation of robot orientation via leg-obstacle contact positions. *IEEE Robotics and Automation Letters*, 5(2):2054–2061, 2020.

[10] K. Chakraborty, H. Hu, M. D. Kvalheim, and F. Qian. Planning of obstacle-aided navigation for multi-legged robots using a sampling-based method over directed graphs. *IEEE Robotics and Automation Letters*, 7(4):8861–8868, 2022.

[11] F. Qian and D. Goldman. Anticipatory control using substrate manipulation enables trajectory control of legged locomotion on heterogeneous granular media. In *Micro-and Nanotechnology Sensors, Systems, and Applications VII*, volume 9467, page 94671U. International Society for Optics and Photonics, 2015.

[12] H. Hu, F. Qian, and D. Seita. Learning granular media avalanche behavior for indirectly manipulating obstacles on a granular slope. In *8th Annual Conference on Robot Learning*, 2024.

[13] G. Barker and A. Mehta. Two types of avalanche behaviour in model granular media. *Physica A: Statistical Mechanics and its Applications*, 283(3-4):328–336, 2000.

[14] S. P. Pudasaini and K. Hutter. *Avalanche dynamics: dynamics of rapid flows of dense granular avalanches*. Springer Science & Business Media, 2007.

[15] N. Gravish and D. I. Goldman. Effect of volume fraction on granular avalanche dynamics. *Physical Review E*, 90(3):032202, 2014.

[16] C. Pavlov and A. M. Johnson. Soil displacement terramechanics for wheel-based trenching with a planetary rover. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 4760–4766. IEEE, 2019.

[17] Y. Wang, Y. Li, K. Driggs-Campbell, L. Fei-Fei, and J. Wu. Dynamic-Resolution Model Learning for Object Pile Manipulation. In *Robotics: Science and Systems (RSS)*, 2023.

[18] S. Xue, S. Cheng, P. Kachana, and D. Xu. Neural Field Dynamics Model for Granular Object Piles Manipulation. In *Conference on Robot Learning (CoRL)*, 2023.

[19] C. Schenck, J. Tompson, S. Levine, and D. Fox. Learning robotic manipulation of granular media. In *Conference on Robot Learning (CoRL)*, 2017.

[20] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015.

[21] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning (ICML)*, 2015.

[22] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui, and M.-H. Yang. Diffusion models: A comprehensive survey of methods and applications. *ACM Computing Surveys*, 56(4):1–39, 2023.

[23] J. Ho, A. Jain, and P. Abbeel. Denoising Diffusion Probabilistic Models. In *Neural Information Processing Systems (NeurIPS)*, 2020.

[24] R. D. Maladen, Y. Ding, C. Li, and D. I. Goldman. Undulatory swimming in sand: subsurface locomotion of the sandfish lizard. *science*, 325(5938):314–318, 2009.

[25] C. Li, T. Zhang, and D. I. Goldman. A terradynamics of legged locomotion on granular media. *science*, 339(6126):1408–1412, 2013.

[26] J. R. Finn, M. Li, and S. V. Apte. Particle based modelling and simulation of natural sand dynamics in the wave bottom boundary layer. *Journal of Fluid Mechanics*, 796:340–385, 2016.

[27] R. Albert, I. Albert, D. Hornbaker, P. Schiffer, and A.-L. Barabási. Maximum angle of stability in wet and dry spherical granular media. *Physical Review E*, 56(6):R6271, 1997.

[28] S. Shrivastava, A. Karsai, Y. O. Aydin, R. Pettinger, W. Bluethmann, R. O. Ambrose, and D. I. Goldman. Material remodeling and unconventional gaits facilitate locomotion of a robophysical rover over granular terrain. *Science robotics*, 5(42):eaba3499, 2020.

[29] A. Karsai, D. Kerimoglu, D. Soto, S. Ha, T. Zhang, and D. I. Goldman. Real-time remodeling of granular terrain for robot locomotion. *Advanced Intelligent Systems*, 4(12):2200119, 2022.

[30] D. Kerimoglu, D. Soto, M. L. Hemsley, J. Brunner, S. Ha, T. Zhang, and D. I. Goldman. Learning manipulation of steep granular slopes for fast mini rover turning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16985–16990. IEEE, 2024.

[31] M. Schwarz, M. Beul, D. Droeschel, S. Schüller, A. S. Periyasamy, C. Lenz, M. Schreiber, and S. Behnke. Supervised autonomy for exploration and mobile manipulation in rough terrain with a centaur-like robot. *Frontiers in Robotics and AI*, 3:57, 2016.

[32] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.

[33] T. Börzsönyi, T. C. Halsey, and R. E. Ecke. Avalanche dynamics on a rough inclined plane. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 78(1):011306, 2008.

[34] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[35] F. Qian and D. I. Goldman. The dynamics of legged locomotion in heterogeneous terrain: universality in scattering and sensitivity to initial conditions. In *Robotics: Science and Systems (RSS)*, 2015.

[36] A. Daerr and S. Douady. Two types of avalanche behaviour in granular media. *Nature*, 399 (6733):241–243, 1999.

We structure the Appendix as follows:

- App. A presents additional hardware details.
- App. B has additional experimental results.
- App. C presents and evaluates multiple additional baselines.
- App. D has more details of DiffusiveGRAIN, including information about our environment state predictor, robot state predictor, and cost function.

A Additional Details of Hardware Design

The primary hardware components of this work are the trackway manipulation system and the sand-proof robot. In this design, the manipulator is able to perform horizontal rotations on the xy-plane and lock at specific angles within a range of -90° to 90°. Furthermore, the gantry system allows precise positioning of the manipulators within the sand tank for targeted experimental operations. A custom slide bar was 3D-printed with two slots shown in Fig. 7, allowing the motors (Lynxmotion ST) to be mounted to the slide bar and slide within a range of 0–15 cm on either side of the slide bar's center. For the rotation functionality, we used a magic arm with a lock, attached to the center of the slide bar. The magic arm has two DoFs; we used screws to fix one degree and mounted the arm on the XY table's step motor via a metal bar.

We developed the control unit using an ADA Lynxmotion board and a Raspberry Pi. The unit manages the motion of the XY table and the manipulator's excavation actions. To enhance the flexibility of the experimental procedure, we programmed the Raspberry Pi for remote control via SSH.

The sand-proof robot was designed with compactness as a core principle, prioritizing minimal size while retaining functionality. Key design dimensions are as follows: the distance between the two front legs (or the two rear legs), including the width of the legs, is 15 cm. The robot's height ranges from a maximum of 15 cm (in a fully standing position) to a minimum of 10 cm (when the legs are folded). The maximum length, measured with respect to the longest component, is approximately 15 cm. The robot has a total mass of 849.1 g, including the LiPo battery. The frame of the robot is 3D-printed using PLA material. This compact and efficient design enables the robot to operate effectively within constrained environments. Since the glass beads are extremely fine and pose a risk of damaging the motor mechanisms, protecting the motors from sand intrusion was a critical task. To address this issue, we covered the Lynxmotion Standard Torque motors with tape to ensure good sealing.

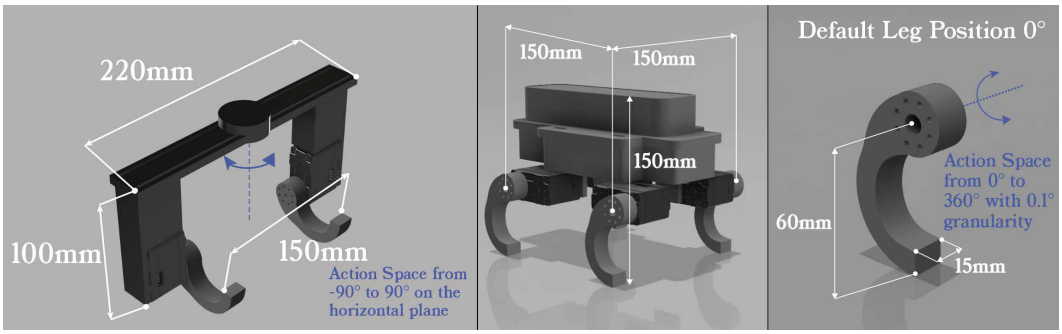

Figure 7: Manipulator and robot design. *Left:* The manipulator is designed to simulate the robot leg action for different robot body sizes. We fixed the distance between the two motors to 150mm in this study to match the robot legs. *Middle :* Sand-proof Robot Design. *Right:* Robot leg design.

B Additional Experiments

Additional experiments for GRAIN: To test GRAIN's assumptions regarding independent obstacle motion, we evaluate how GRAIN performance changes as a function of the obstacle distance on

the sand slope, by calculating its prediction error with different obstacle distances. We found that the prediction error worsens when the two obstacles are closer (Fig. 12). Fig. 8 illustrates the experiments with varied obstacle distances, and shows that GRAIN's prediction error is low in (Fig. 8, c, d, g, h) as obstacle distances were relatively large, but becomes significantly larger as the distance of obstacles decreases (Fig. 8, a, b, e, f). Similarly, to test the effect of different manipulation actions on robot state, we performed experiments with different leg manipulation pairs. Fig. 9 shows the example of the 6 robot actions where the cyan highlighted legs are activated legs and green arrowed circles represent the robot states after the corresponding action. The experiment indicates we need to consider the robot's state change due to the leg excavation; otherwise, the cumulated robot state changes with multiple steps will introduce errors in predicting robot manipulation results.

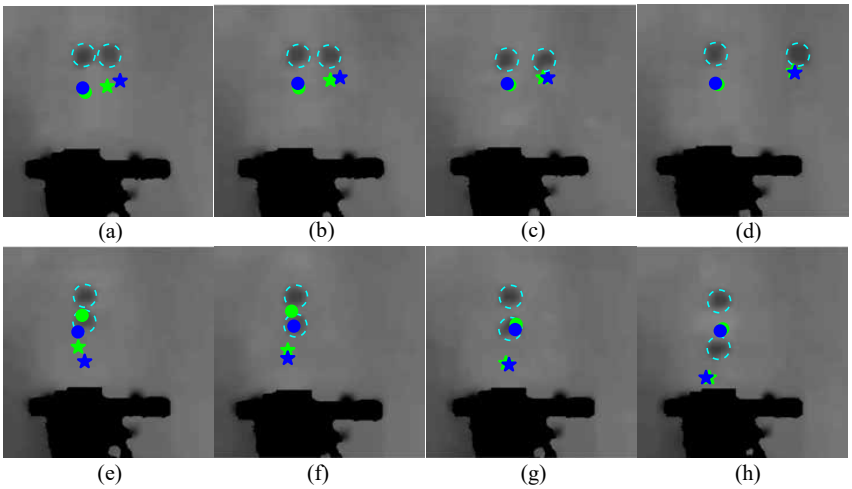

Figure 8: GRAIN performance on obstacles with different lateral or fore-aft distances. (a)(b)(c)(d) are obstacles with a lateral distance of 0cm, 2cm, 4cm, and 8cm respectively; (e)(f)(g)(h) are obstacles with a fore-aft distance of 0cm, 2cm, 4cm, and 8cm respectively. The cyan dashed circles are the obstacle positions before the excavation action, and the green star and circle are the ground truth positions of the obstacles after an excavation action. The blue star and circle are the GRAIN predicted positions of the obstacles after the excavation action. Specifically, the star shape markers are for the reference obstacle and circle shape markers are for the obstacle with different lateral or fore-aft distance to the reference obstacle.

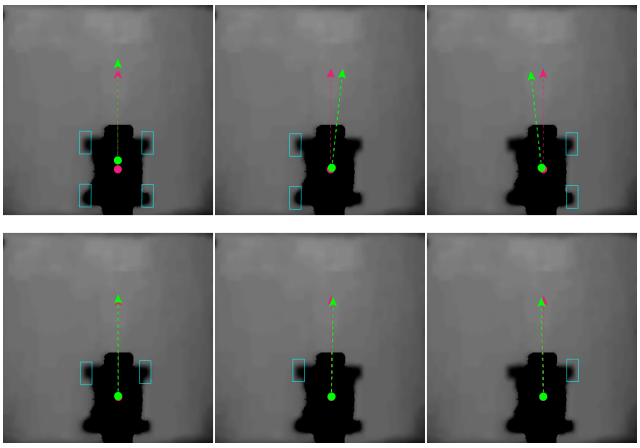

Figure 9: Robot state before and after different robot actions. The cyan rectangle represents the robot action (activated legs), the pink circle with the dashed arrow represents the robot CoM position and orientation before the robot action, and the green circle with the dashed arrow represents the ground truth of the robot CoM position and orientation after the robot action.

Additional experiments for DiffusiveGRAIN: We performed experiments to compare the performance between DiffusiveGRAIN and GRAIN. Fig. 10 shows 1 manipulation trial using DiffusiveGRAIN and GRAIN. In both trials, the robot must (indirectly) manipulate all 4 obstacles to bring

them to a designated goal region near the robot. Fig. 11 shows 1 locomotion trial using DiffusiveGRAIN and GRAIN. We place the robot in the bottom right region, and the robot must move to the area marked by the green rectangle. In both manipulation and locomotion trials, the improved performance suggested the importance for including training data with multiple obstacles, especially when obstacle distances are small (Fig. 12).

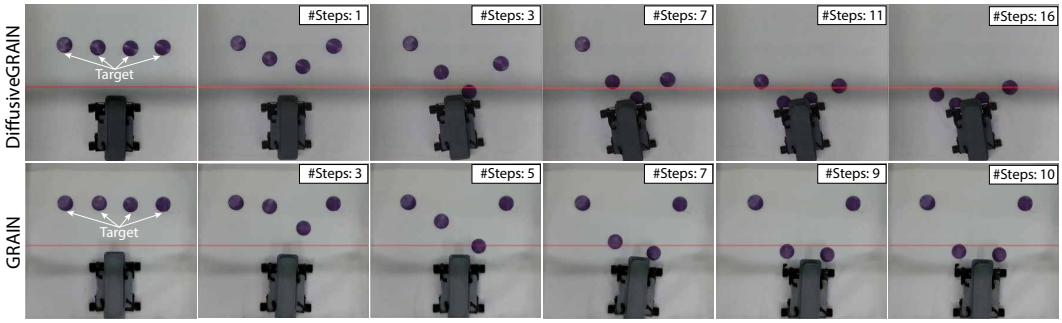

Figure 10: One robot manipulation trial using DiffusiveGRAIN and GRAIN. The robot starts in the bottom middle and must manipulate all 4 obstacles to below the red line. DiffusiveGRAIN results in success, but with GRAIN, the robot only manipulated 2 of 4 obstacles to below the red line, and policy execution terminated as GRAIN predicted no action could further optimize the cost function at Step 10.

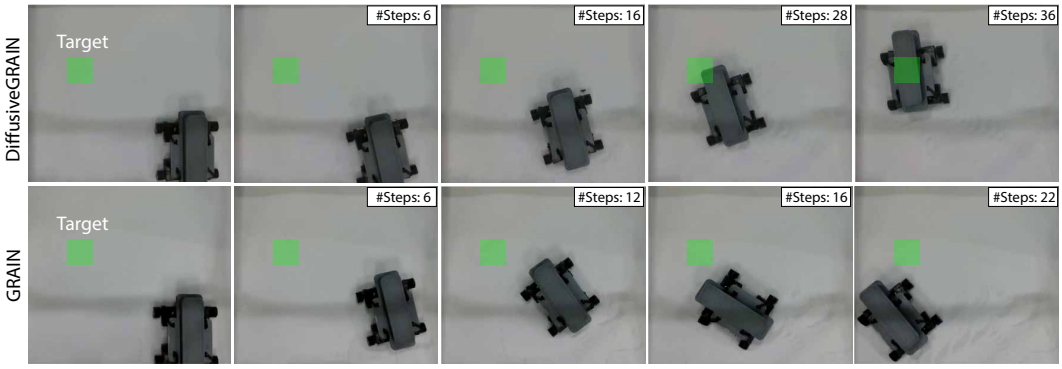

Figure 11: One robot locomotion trial using DiffusiveGRAIN and GRAIN. The robot starts in the bottom right corner and must locomote to the target area marked by a green rectangle. DiffusiveGRAIN results in success, but with GRAIN, the robot reaches the sand tank boundary which is considered a failure.

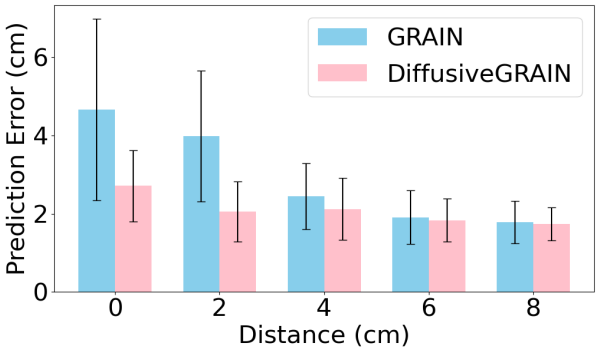

Figure 12: GRAIN prediction errors on obstacles with different distances. The blue and pink bars represent the mean prediction error of GRAIN and DiffusiveGRAIN (our proposed method, see Sec. 4). The error bars indicate the standard deviation.

Additional experiments for EAA: Fig. 13 shows an example of the DiffusiveGRAIN prediction of robot and obstacle states w/o EAA. A statistical evaluation of EAA performance is shown in Tab. 3.

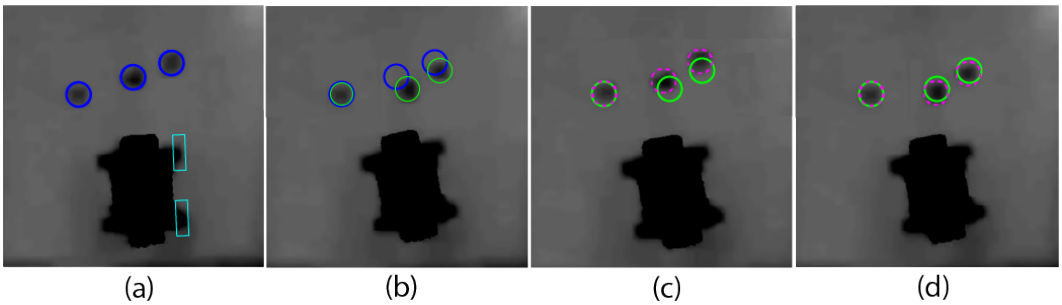

Figure 13: The EAA (see Sec. 4.3). (a) an example of obstacle states before the robot executes a *left turn* action where the blue circles are obstacles and cyan rectangles are the activated legs for the robot; (b) the ground truth of the robot state and obstacle states after the robot executed a *RP* action in (a) where the blue circles and green circles are the obstacle states before and after the robot action; (c) the *Environment Predictor* predicted obstacles states as marked by pink dash circles without EAA; (d) the *Environment Predictor* predicted obstacles states as marked by pink dash circles with EAA.

C Additional Baselines

E2E-Diffusion: In this paper, we use two models: an environment state predictor and a robot state predictor. To investigate if this division is necessary, we design another baseline (E2E-Diffusion) which is a single diffusion model for end-to-end prediction. Specifically, we retrained the same U-Net-based diffusion model in the *Environment State Predictor* on all 300 trials in our data, and have it output the predicted depth image at the next state. We evaluated E2E-Diffusion on the experiment dataset by extracting the predicted obstacle and robot states from the predicted depth image at the next state and comparing them with the ground truth.

Pose-Diffusion: To test our proposed representations of the system, we trained a new version of the environment state predictor and robot state predictor on pose inputs, which are solely the robot and obstacle states. Like DiffusiveGRAIN, this Pose-Diffusion baseline also outputs the predicted robot and obstacle states. The input to the network is a $B * (3 + 2 * N + 1)$ vector, and the output is a vector with the dimension of $B * (3 + 2 * N)$, where the input vector is a concatenation of the robot state, a 1 by 3 vector, and N obstacle states, a 1 by $2N$ vector, and the indicator of robot action, a 1 by 1 vector (an integer from 0 to 5 represents the 6 actions in this study). In addition, B is the batch size; we used $B = 16$.

DiffusiveGRAIN (without EAA): As an ablation, we also test the DiffusiveGRAIN without EAA in the policy execution for the experiment data.

We report the performance of E2E-Diffusion, Pose-Diffusion, GRAIN, DiffusiveGRAIN, and DiffusiveGRAIN (without EAA) in Tab. 3. As shown in Tab. 3, DiffusiveGRAIN outperforms DiffusiveGRAIN (without EAA), E2E-Diffusion, and GRAIN on the experiment dataset. We hypothesize that

Task	**Method**	**MAE** (cm)
Obstacle state	E2E-Diffusion	3.81(± 1.25)
	Pose-Diffusion	5.04(± 1.66)
	GRAIN	2.91(± 0.88)
	DiffusiveGRAIN	2.44(± 0.78)
	DiffusiveGRAIN (without EAA)	2.80(± 0.84)
Robot state (position)	E2E-Diffusion	2.25(± 0.96)
	Pose-Diffusion	2.94(± 1.03)
	GRAIN	1.26(± 0.41)
	DiffusiveGRAIN	1.17(± 0.32)
	DiffusiveGRAIN (without EAA)	1.17(± 0.32)

Table 3: We compare the performance of the E2E-Diffusion, Pose-Diffusion, GRAIN, DiffusiveGRAIN, and DiffusiveGRAIN (without EAA) on the experiment dataset.

the low performance of E2E-Diffusion is because it must learn from a highly complex dataset from combining the manipulator and robot datasets. Consequently, this leads to the current dataset size being insufficient to achieve the same performance level as compared to DiffusiveGRAIN (which may be more data-efficient). We plan to investigate this hypothesis in the future by collecting more training data in experiments.

The performance of Pose-Diffusion implies that solely relying on the pose representation does not work well. Our experiments showed that an obstacle with the same initial pose could have significantly different displacements in response to the same manipulation action, *depending on the terrain state*. This aligned with previous literature [36] indicating that the terrain surface significantly affects obstacle displacement with the same manipulation action.

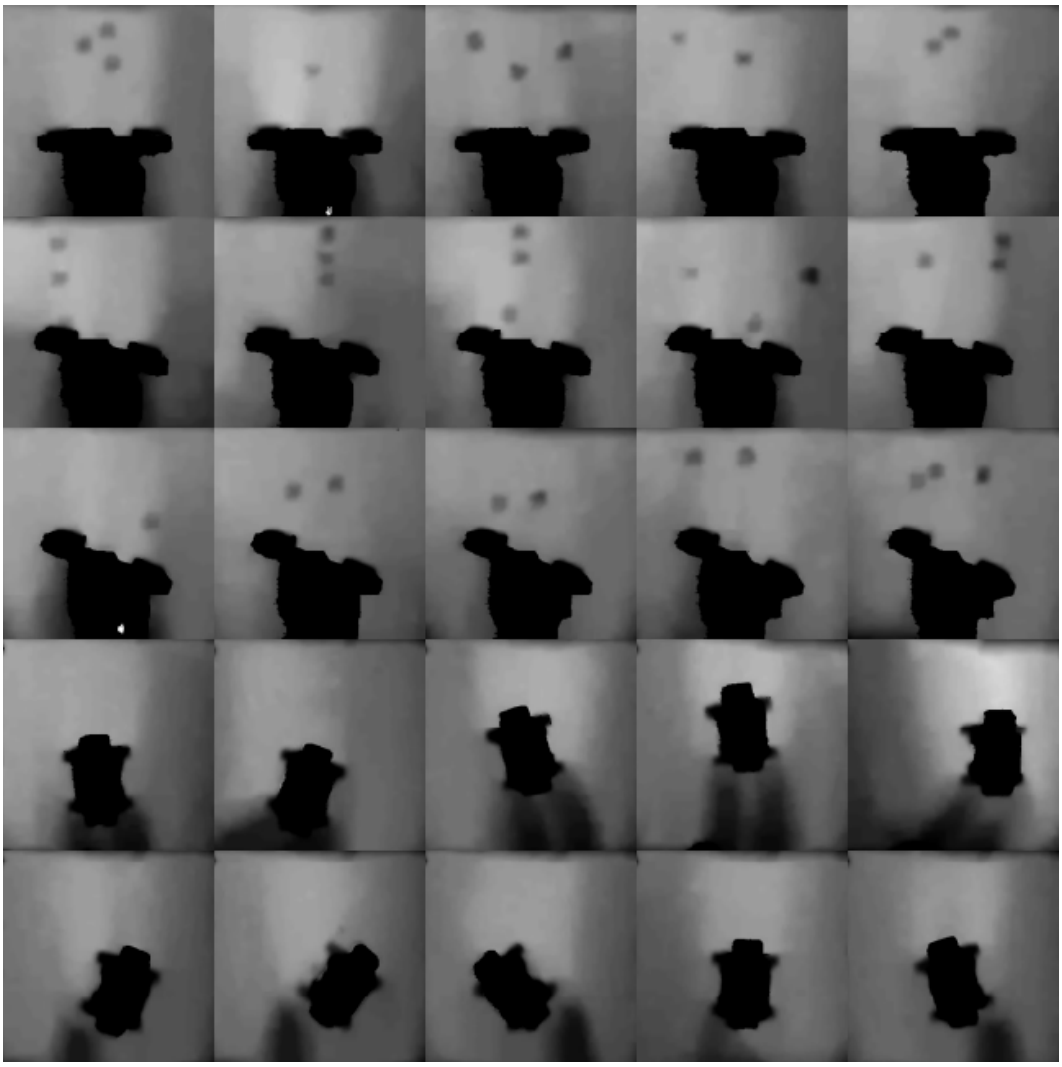

Figure 14: Samples of the dataset.

D Additional Details of DiffusiveGRAIN

In DiffusiveGRAIN, we trained the environment state predictor f_e using the dataset collected by the manipulator. During policy execution, f_e takes depth image inputs from the robot. Although the manipulator and the robot differ in size and appear slightly different in the depth images, they share common characteristics. We believe the diffusion model learns to ignore these appearance differences and focuses on predicting changes in the environment state.

The robot state predictor f_r is trained on the dataset collected by the robot. To make the network robust to inputs during policy execution, we augmented the training data by adding obstacles into the depth images. The primarily consideration to decouple f_e and f_r is that it significantly reduces the complexity of the loco-manipulation problem, which in turn lowers the amount of data needed to train the networks, which is demonstrated in Tab. 3. This is critical in our study, as all data were collected from real physical experiments. Our preliminary experiments suggested that obstacle positions on the granular slope have little effect on the robot's state change under a given action when the distance between robot leg and obstacle is relatively large, which supports the chosen method of obstacle augmentation. We acknowledge that in the case of direct leg-obstacle contact, the current method may not accurately predict robot and obstacle state changes, which is a limitation of this study that future work should further explore.

We show some examples of the training dataset in Fig. 14, and we also show some output images of the environment state predictor and robot state predictor on the validation dataset in Fig. 15 and Fig. 16, respectively. We report the key hyperparameters of f_e and f_r in Tab. 4 and Tab. 5, respectively.

In addition, we provide additional details on the cost function used for loco-manipulation (Sec. 4.4): For a robot that does both locomotion and manipulation, we linearly combine the cost of locomotion and manipulation to get the cost for the robot "loco-manipulation" task: $\mathbf{C}_{lo} = w_5\mathbf{C}_l + w_6\mathbf{C}_m$, where in this study, we use $w_5 = 0.4$ and $w_6 = 0.6$.

Category	**Hyperparameter**	**Value**
Diffusion Process	# of Diffusion Steps	150
	Beta Schedule	Linear (0.0001 $\rightarrow$ 0.025)
	Sampling Algorithm	DDPM
U-Net Architecture	Base Channels	64
	Channel Multiplier	[1, 2, 4, 8, 16]
	# of Res. Blocks	2
	Dropout	0.1
Time Embedding	Embedding Dim	32
	Activation	ReLU
Optimizer and Training	Optimizer	AdamW
	Learning Rate	1×10^{-5}
	Weight Decay	1×10^{-5}
	Batch Size	1
Data	Resolution	400×400
	Normalization	$[-1, 1]$

Table 4: Hyperparameters of the Environment State Predictor f_e.

Category	**Hyperparameter**	**Value**
U-Net Architecture	Base Channels	64
	Channel Multiplier	[1, 2, 4, 8]
	# of Res. Blocks	2
	Activation	ReLU
	Dropout	0.1
Optimizer and Training	Optimizer	AdamW
	Learning Rate	1×10^{-4}
	Weight Decay	1×10^{-4}
	Batch Size	1
Data	Resolution	400×400
	Normalization	$[-1, 1]$

Table 5: Hyperparameters of the Robot State Predictor f_r.

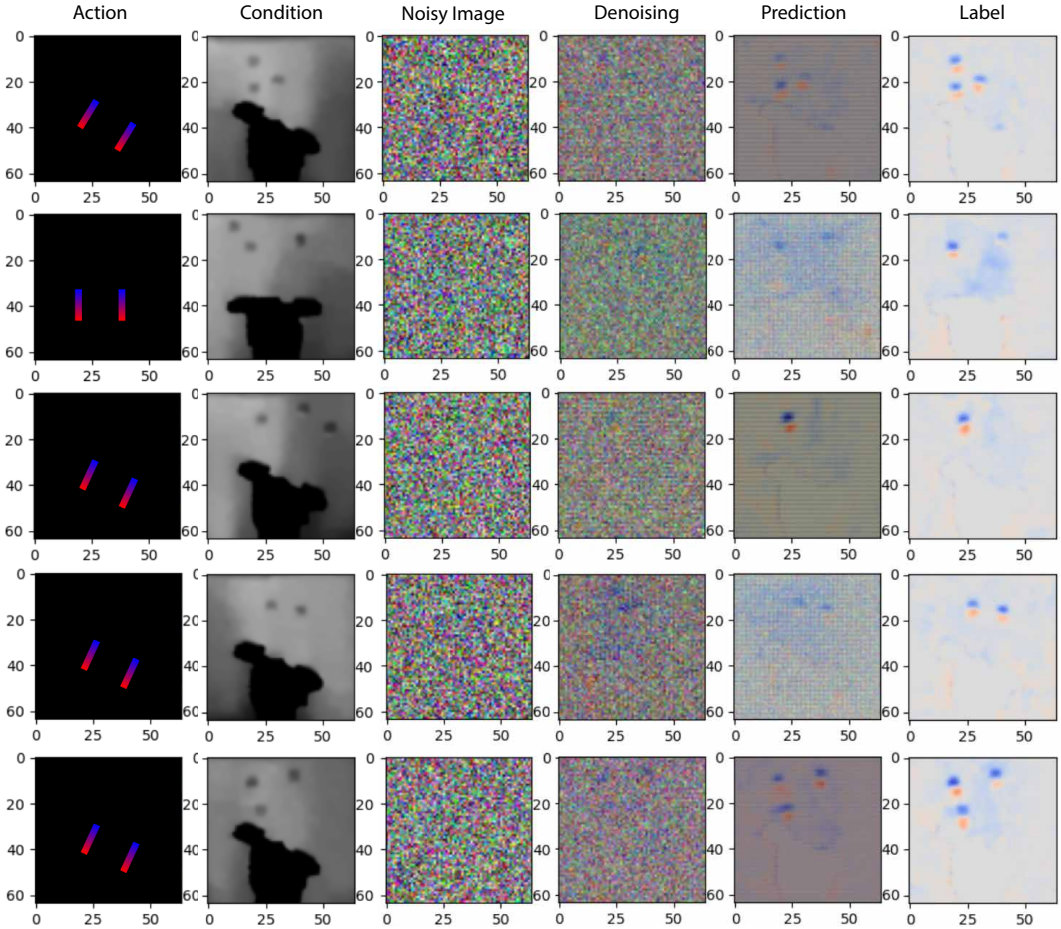

Figure 15: Examples of *Environment Predictor* prediction (one per row). From left to right are the action image $\mathbf{A}_t$, input image $\mathbf{I}_t$, pure noise image, the denoised image at diffusion step 100, output image, and label image respectively.

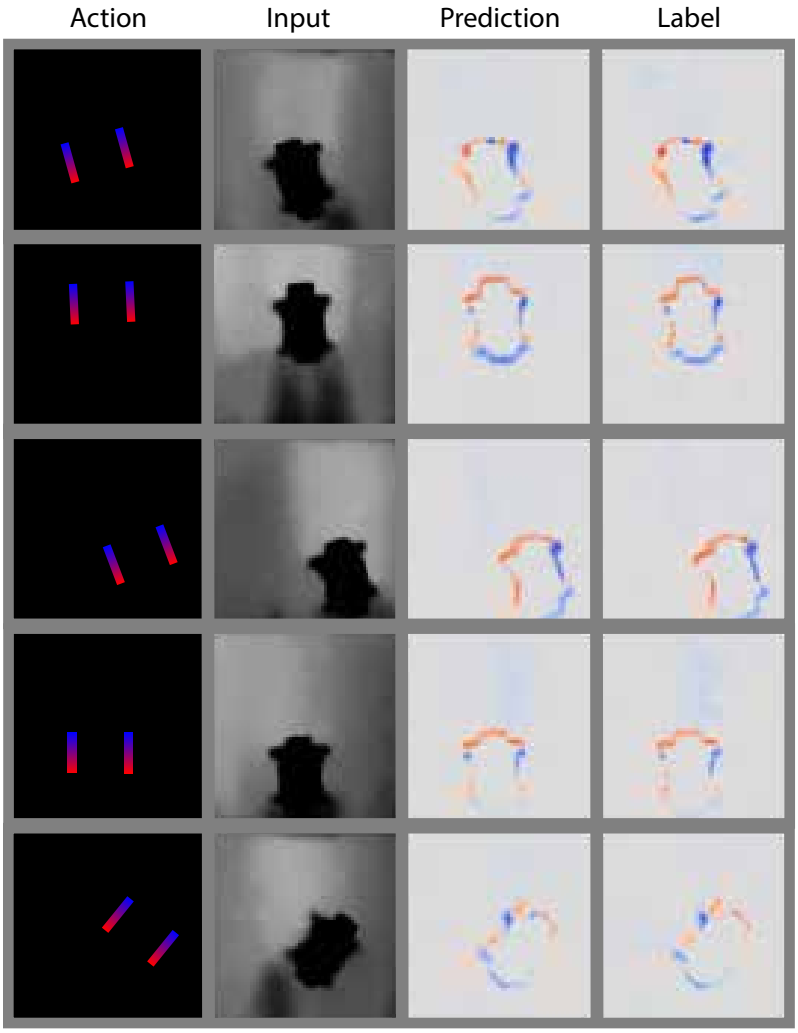

Figure 16: Examples of *Robot State Predictor* prediction (one per row). From the left to right are the action image $\mathbf{A}_t$, input image $\mathbf{I}_t$, output image, and label image respectively.