2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 23-27, 2022, Kyoto, Japan

Excavation of Fragmented Rocks with Multi-modal Model-based Reinforcement Learning

Yifan Zhu[1], Liyang Wang[2], and Liangjun Zhang[2]

***Abstract*— This paper presents a multi-modal model-based reinforcement learning (MBRL) approach to the excavation of fragmented rocks, which are very challenging to model due to their highly variable sizes and geometries, and visual occlusions. A multi-modal recurrent neural network (RNN) learns the dynamics of bucket-terrain interaction from a small physical dataset, with a discrete set of motion primitives encoded with domain knowledge as the action space. Then a model predictive controller (MPC) tracks a global reference path using multi-modal feedback. We show that our RNN-based dynamics function achieves lower prediction errors compared to a feed-forward neural network baseline, and the MPC is able to significantly outperform manually designed strategies on such a challenging task.**

I. Introduction

Excavators are versatile earth-moving machines that are widely used in mining, agriculture, and construction. Excavator operation requires highly skilled operators and oftentimes happens in hazardous environments that could cause injuries [1]. This has led to an increasing amount of effort in excavation automation [2], but one of the key challenges of excavation automation is that it is impossible to model the machine-earth interaction accurately [2], especially for complex, non-homogeneous terrains such as fragmented rocks.

In this work, we focus on the problem of excavation of fragmented rocks, and a real-world example is shown in Fig. 1. Compared to homogeneous materials such as sand, the interaction between an excavator bucket and a pile of fragmented rocks is very challenging to model because the rocks are highly variable in sizes and shapes, and they are only visible from the surface. When following an arbitrary digging trajectory, an excavator bucket tends to get stuck because *jamming* among the rocks occurs frequently and generates large interaction forces. When human operators excavate fragment rocks, they need to find gaps between rocks to insert the bucket and continuously adjust and react to the change in the visual scene, interaction forces, and even engine noises of the machines. Another strategy they use when the bucket gets stuck is to wiggle the bucket to disturb and loosen up the rocks. However, this type of strategy is challenging to discover automatically by a planner. Previous works have adopted admittance control [3, 4] and trajectory learning [5] for wheel loaders, but can only handle limited scenarios. Recently, model-free reinforcement learning approaches have been applied to finding excavation policies for wheel loaders [6] and excavators [7], but the training is done in simulation due to the requirement of large data. The bottleneck here is that accurate and efficient simulators are currently not available for complex terrains such as fragmented rocks.

Fig. 1. Fragmented rocks excavation using an excavator.

We aim to address these challenges associated with excavation of fragmented rocks, including the difficulty in modeling terrain-bucket interactions and acquiring a large dataset, and the requirement for dexterous excavation skills. Towards equipping autonomous excavators with similar capabilities as humans, we propose a multi-modal model-based reinforcement learning (MBRL) approach to fragmented rocks excavation, that enables data-efficient learning using a small dataset of physical data and significantly outperforms manually tuned excavation strategies.

In our method, the excavation domain knowledge is encoded into a discrete set of primitive motions, and a recurrent neural network (RNN) learns the dynamics of bucket-terrain interaction with visual, tactile, and proprioceptive modalities using data collected in physical experiments. During online excavation, a global reference excavation path is given, which is tracked with a model predictive controller (MPC). The MPC follows the path whenever possible to maximize the excavated mass, and deviates from the path to avoid excessive contact forces that lead to jamming. We show that our RNN-based model significantly outperforms a feed-forward neural network baseline. We demonstrate from extensive experiments that our system is capable of completing excavations with high success rates of avoiding jamming, and outperforming manually designed strategies in tracking error and excavated mass.

[1]: Y. Zhu is with the Departments of Computer Science, University of Illinois at Urbana-Champaign, IL, USA. yifan16@illinois.edu. Work done as an intern at Baidu Research, Sunnyvale, CA, USA.

[2]: L. Wang, and L. Zhang are with the Robotics and Auto-Driving Lab, Baidu Research, Sunnyvale, CA USA. {liyangwang, liangjunzhang}@baidu.com.

2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-7927-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/IROS47612.2022.9981537

II. Literature Review

A. Excavation Automation

There has been plenty of work in control [8, 9, 10], trajectory planning [11, 12], and task-level planning [13, 14, 15] for autonomous excavators and earth-moving machines. However, the majority of these works focus on excavation of homogeneous materials, such as sand and soil, and can not be directly applied to fragmented rocks. Here we review works on the excavation of fragmented rocks.

Control-based methods have been proposed for fragmented rocks excavation. Dobson et al. adopt an admittance control framework for fragmented rocks excavation by wheel loaders [3]. Fernando et al. extend this work by adapting the admittance control parameters online with iterative learning [4]. Compared to control-based approaches, our method performs local planning that gives rise to more versatile and optimal behaviors.

Dadhich et al. adopt imitation learning for wheel loaders by simply performing regression on expert demonstrations [5], but this approach could only handle simple scenarios. Model-free [6, 7] and model-based [16] RL approaches have been adopted in this problem domain as well, but the learning is performed in simulation, which deviates from reality. For example, when applying trajectories planned with a planner based on simulation data, failures due to jamming have been reported [16]. Our proposed method enables data-efficient learning from physical data and incorporates multi-modal sensory information to avoid jamming.

B. Contact-rich Manipulation with Multi-modal Observations

Humans are capable of seamlessly fusing multiple sensory modalities including visual, tactile, and proprioceptive information while performing contact-rich manipulation tasks, and it is a natural strategy for robots as well because each of these modalities is able to provide unique information and complement each other. In the recent years, it has become increasingly common to consider multi-modal observations for contact-rich manipulation tasks. Kappler et al. propose a method that would learn a set of manipulation skill with multi-modal sensory input, formulated as dynamic motion primitives, and store them in a manipulation graph used for planning [17]. Fazeli et al. demonstrate a robot that leverages hierarchical Bayesian representation with multi-sensory observations to play the game of Jenga. End-to-end learning has also been adopted [18], and Lee et al. utilize auxiliary goals to learn a latent state representation that fuses all modalities [19]. In addition, imitation learning [20, 21] and inverse RL [22] approaches have also been adopted for contact-rich manipulation tasks with multi-modal observations. However, the majority of these methods focus on relatively simple tasks such as peg-in-hole, Jenga, and wiping, while we tackle a very challenging problem of fragmented rocks excavation.

C. Model-based Reinforcement Learning

MBRL approaches have been surveyed and compared recently in a survey [23] and a benchmark [24]. Compared to model-free methods, a model-based approach is more data-efficient and makes it possible to directly learn from real-world data, which is usually expensive to collect for robotics tasks. Compared to most of the MBRL methods for control and planning [24], our method adopts a discrete action space with primitives designed with domain knowledge, and uses discrete planning methods for the MPC.

III. Problem Definition

In the real world, e.g. construction sites, automation of excavators aims to maximize the amount of materials excavated per unit time. In the case of fragmented rocks, the excavator would execute a trajectory that allows the digging bucket to collect as much material as possible, while avoiding getting stuck since that would require the excavator to abort the current trajectory. In this work, we simplify the problem by giving a reference path that is designed to excavate a large volume if followed precisely, and aim to plan actions to follow the path, using multi-modal feedback. Ideally, the executed trajectory would follow the reference path when possible, but also deviate from the path to avoid getting stuck due to large contact forces.

We use a 7-DOF Franka Panda arm and irregular wooden blocks to emulate an excavation scene of fragmented rocks. The excavation setup and the frame definition are in shown in Fig. 2, where we use an overhead RGB camera and mount a digging bucket to the robot arm end-effector. We note that although as explained in the next paragraph, the emulated excavator has 4 DOFs, the same as an actual excavator, the geometries of the configuration space are different. However, this would not affect the generality of the proposed method itself. We represent the state of the robot and the environment at time t as $s_t \in \mathcal{S} = \mathcal{P} \times \mathbb{R}^3 \times \mathcal{I}$, which includes the bucket pose, the contact force at the bucket and an RGB image from the overhead camera. Since an excavator arm only has 4 degrees of freedom, we restrict the robot's bucket pose to only the 3 translations and rotation around the world y-axis, i.e. $\mathcal{P} = \mathbb{R}^3 \times SO(2)$. For notation convenience, we also represent the state without the visual information at time t as $q_t \in \mathcal{Q} = \mathcal{P} \times \mathbb{R}^3$.

When an excavator bucket encounters large resistive forces from the substrate, a human operator often adopts a strategy to wiggle the buckle to disturb and loosen the rocks. We take advantage of such domain knowledge and instead of using continuous controls as the action space, we define a set of 9 discrete action primitives $\mathcal{A}$ as the action space. The primitives include Cartesian movements and rotation of the bucket, and wiggling of the bucket:

- Cartesian movement of the bucket in both the positive and negative x-, y-, and z- directions in the world frame. Each movement is 1.5 cm in length with orientation fixed, executed at 1 cm/s.
- Positive and negative bucket orientation change in the y-axis for 0.1 rad (5.7°) at 1 rad/s.

- Wiggling of the bucket in the x-axis for 4 seconds, where the displacements follow a sine wave with magnitude 2 cm and frequency 1 Hz.

We empirically chose these actions and their parameters for our excavation scene. In the future, it would be beneficial to investigate how to systematically choose the action parameters based on the size of the fragmented rocks. To execute an action a_i, a joint-space trajectory is first generated using inverse kinematics (IK) and sent to the joint impedance controller of the Franka arm with impedance set at 100 Nm/rad.

Throughout this project, we follow the common practice in the excavation literature and divide a reference path into 4 phase: digging, dragging, closing, and lifting [16, 25]. We assume that the bucket goes toward the base of the robot, and to simplify the problem without the loss of generality, we only consider trajectory in the x-z plane of the robot. With these assumptions, the reference path is defined as $f_{ref} : [0, 1] \rightarrow SE(2)$, an example is shown in Fig. 3. Simply tracking such an trajectory on fragmented rocks often results in the bucket getting jammed. In our experiment setup, due to the use of a safe collaborative robot, the robot controller would take over and issue an halt to the robot when a large contact force is detected, which we use as a criterion for whether the robot has jammed or not throughout the paper.

(a) Excavation setup (b) Excavation scene

Fig. 2. Excavation setup with the world frame labeled and the excavation scene with scale.

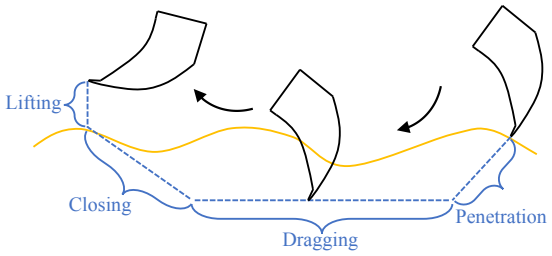

Fig. 3. Reference digging trajectory.

Throughout the paper, due to the use of a discrete action space, a trajectory of length T is defined as T actions, along with the $T + 1$ states with or without visual information before and after executing the action, and we ignore the states of the robot and the environment while the primitive action is being executed.

IV. Dynamics Learning

We adopt a MBRL formulation, where we first collect a dataset of trajectories offline on the physical setup, train a dynamics function for a sequence of actions, and use an MPC to track a reference path online. We shall discuss dynamics learning in this section, and MPC in Section V.

Since only the surface of a pile of rocks is visible, we hypothesize that in order to accurately predict the excavation dynamics, a robot needs not only the current state of the system, but also state history, which is shown to be true through experiments in Section VI. Therefore, the goal of dynamics learning is to, at time t, predict a horizon k of future states without visual information $\tilde{q}_t = [q_{t+1}, \ldots, q_{t+k}]$ given a history of states of length l: $\bar{s}_t = [s_{t-l+1}, \ldots, s_t]$, and future actions $\tilde{a}_t = [a_t, \ldots, a_{t+k-1}]$, where each a_i is an integer. Since this is a sequence-to-sequence prediction task, we propose to use an RNN to learn such a dynamics function, which encodes visual, proprioceptive, and tactile modalities.

The entire architecture is shown in Fig. 4, where the sizes of the fully-connected layers are annotated. Each of the three modalities in each state is first encoded into a single latent feature. The RGB image is encoded with a ResNet50 convolutional neural network (CNN) model [26]. The bucket pose and contact wrench are encoded with separate fully-connected layers. These three modalities of encoded features are then concatenated into a single vector that passes through another fully-connected layer to a feature vector z. A sequence of length l of z vectors is then input into the RNN encoder.

Actions are one-hot encoded and they pass through a fully-connected layer before inputting into the decoder. The dimension of the one-hot encoding is 10, which is the size of the action space plus a null action. Separate fully-connected layers are used to obtain the pose and force from the decoder latent state. We apply batch normalization and the leaky ReLU activation function [27] to each fully-connected layer. In addition, We use a gated recurrent unit (GRU) [28] RNN model with additive attention [29]. Compared to a typical sequence-to-sequence machine translation, the main difference is that we have control inputs in the RNN decoder.

We collect training data by executing random and manually designed trajectories while recording visual, proprioceptive, and tactile data, detailed in Section VI-A. The trajectories are post-processed into training data points, where the i-th data point is denoted as $(\bar{s}_t^{(i)}, \tilde{a}_t^{(i)}, \tilde{q}_t^{(i)})$, where $\tilde{q}_t^{(i)}$ is the prediction target. In addition, we pad the start of each trajectory with $l-1$ null actions and copies of the first state in the trajectory. Let the dynamics function be $\hat{q}_t^{(i)} = f(\bar{s}_t, \tilde{a}_t)$, where $\hat{q}_t^{(i)}$ is the predicted future states, then for N data points, we minimize the following loss:

$$\sum_{i=0}^{N-1} \sum_{j=0}^{k-1} \left\| c^T (\tilde{q}_t^{(i)}[j] - \hat{q}_t^{(i)}[j]) \right\|_2^2, \tag{1}$$

where c is a weight vector, and we give the bucket pose a weight of 10 and the wrench a weight of 1 .

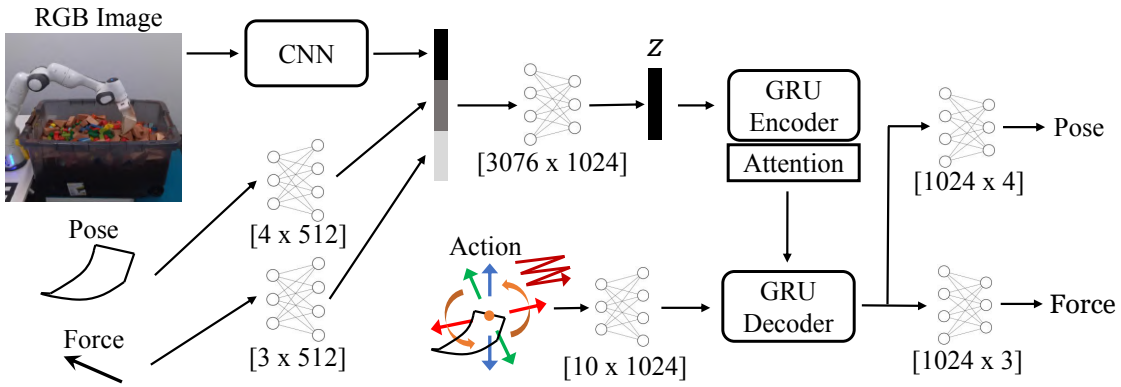

Fig. 4. Dynamics learning architecture.

V. Planning

We adopt the MPC scheme to reactively plan actions to track a reference path, where the planner repeatedly plan a trajectory of length k based on a cost function and execute the first action. This allows the planner to adjust to feedback when applying each action and be robust to dynamics prediction error. The cost function we aim to minimize is a weighted sum of three terms:

$$J(\tilde{a}_t, \tilde{q}_t) = \\ w_1 \cdot g_{prog}(f_{ref}, \tilde{q}_t) + w_2 \cdot g_{track}(f_{ref}, \tilde{q}_t) + w_3 \cdot g_{col}(\tilde{q}_t), \tag{2}$$

where g_{prog} measures the progress along the reference path, g_{track} weighs the tracking performance of the planned trajectory, g_{col} calculates the penalty given to large contact forces. We set w_1, w_2, and w_3 to 0.8, -1.05, and -1/1500, respectively, which are tuned empirically. The goal is to maximize g_{prog} while minimizing g_{track} and g_{col}. The exact definitions of these 3 terms are introduced in Section V-B.

The planning problem is then defined as:

$$\begin{aligned} \tilde{a}_t \quad & J(\tilde{a}_t, \tilde{q}_t) \\ \text{s.t.} \quad & \tilde{q}_t[\cdot] \in \mathcal{Q}, \tilde{a}_t[\cdot] \in \mathcal{A}, \\ & \tilde{q}_t = f(\bar{s}_t, \tilde{a}_t), \end{aligned} \tag{3}$$

where $\mathcal{Q}$ is the space of feasible robot configurations and $f(\cdot, \cdot)$ is the dynamics function.

A. Planning Algorithm

Due to the discrete nature of the actions, we consider 3 well-known methods for discrete planning, including brute-force search, random shooting (RS), and Monte Carlo tree search (MCTS). In brute-force search, each possible trajectory is evaluated and the optimal trajectory is selected. However, this method is limited by the planning horizon since the number of possible trajectories increases exponentially with the prediction horizon. For RS, a fixed number of random trajectories are sampled and evaluated. This number is the computation budget. MCTS is a heuristic search algorithm, where a search tree is iteratively expanded by balancing exploration and exploitation while each node is assigned a utility that is the average of the rewards, i.e. negative costs, of randomly sampled future trajectories starting from this node. In our implementation, a fixed number of random trajectories are evaluated at each expansion step of MCTS. The computation budget for MCTS equals this fixed number multiplied by the total number of expansion steps.

B. Excavation Specific Details

As defined earlier, q_t contains both the bucket pose and contact force, which we denote as $p = [x, y, z, \alpha]$ and $F = [F_x, F_y, F_z]$. For a bucket pose p, we evaluate its progress along the reference path by first finding the closest point p_{closet} on the path in terms of Euclidean distance, ignoring the angular component. Then the progress $\in [0, 1]$ is calculated as the path length, considering only the position, from the start of the path to p_{closet} divided by the entire length of the path. g_{prog} simply averages the progresses for the k future bucket poses.

To calculate g_{track}, for a pose p, we alternatively calculate its distance to the reference path by calculating the Euclidean distance between p and $p_{closest}$ including the angle, where the position and angle use meter and radian as units, and the angles are weighted by 0.15. g_{track} could be defined as the average of the distances of the k future poses. However, due to the granularity of the discrete actions, we allow small deviations from the reference trajectory and use the average of the adjusted distances, where each adjusted distance scales the portion of the original distance that is below a threshold of 0.015 by 0.2.

g_{col} is used to penalize large contact forces, and is an average of k future contact force penalties, where for each contact force, zero penalty is given when the contact force is below a threshold of 17.5 N, and is penalized with a coefficient of 1 over the threshold.

Finally, since large contact forces during lifting rarely occur, the reference path does not contain the lifting phase. When the robot finishes closing, we simply execute lifting by sending open-loop commands. We also note that the parameters described in this section are empirically determined.

VI. Results

A. Data Collection

One goal of data collection is to obtain data that cover a wide distribution. To do this, we adopt 3 strategies, including applying random actions, following scripted policies, and following scripted policies with stochasticity.

For random actions, the robot randomly executes the 9 actions with equal probability, with boundaries on the position and orientation of the digging bucket to ensure safety. We consider 3 types of scripted policies. In the first policy, denoted as "Follow", the robot executes the action that minimizes its distance from a moving reference point on the reference path, where the distance is defined as that of g_{track} without scaling in V-B. In particular, once the distance in the x position between the bucket and the reference point is smaller than 0.04, the reference point moves along the trajectory for 2 cm. Therefore, this policy does not consider the wiggling action. The second policy (denoted as "Closed-loop") is the same as Follow, but wiggles every 2 actions. The third policy (denoted as "Open-loop") is an open-loop version of Closed-loop, where the entire action sequence along the path is first planned in a similar fashion by assuming perfect execution of the actions, also with wiggling every 2 other actions.

To introduce randomness into the scripted policies, we use a modified version of Follow. Instead of picking actions in a deterministic way, we assign nonzero probabilities to select each of the actions and bias the action that would minimize the distance to the reference point.

Using these strategies, we collect a total of about 300 trajectories, and split them into training, validation, and testing data with a ratio of 8:1:1.

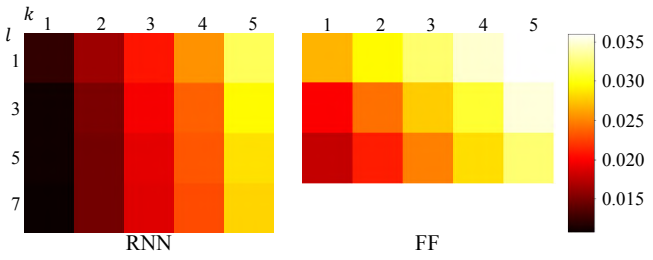

Fig. 5. Loss of RNN and FF with different history lengths and prediction horizons on the testing data. [Best viewed in color.]

B. Dynamics Learning

We compare our method (denoted as RNN) against a feed-forward neural network baseline (denoted as FF). FF also uses the same multi-modal encoding layers as RNN to obtain the history features. The model concatenates the history features and the future controls, and predict the next k states all at once. Note that we cannot adopt the scheme where we predict only the next state, move the window of history forward one time step and predict again such that a variable number of future states can be predicted. This is because we would not be able predict the future images. For the architecture, we use a total of 5 fully-connected layers of sizes $(l+k)\times 512$, $(l+k)\times 1024$, $(l+k)\times 512$, $k\times 512$, and $k\times 64$. We also use batch normalization and the leaky ReLU activation function on each layer.

For RNN, we train a model with history length l = 7 and prediction horizon k = 7. For the FF baseline, we train three models with the same prediction horizon of 5, and history lengths of 1, 3, and 5. We do not train FF with l or k = 7 due to neural network model size and computer hardware limits.

One difficulty in predicting a horizon of future states with RNN is that errors in predictions accumulate and could blow up. This is a problem for the initial stage of training where the 1-step prediction error is big. Therefore, we perform RNN training in 2 stages, similar to the strategy used by [30]. In stage 1, we train the entire pipeline with a prediction horizon length of 1. In stage 2, we train with the full horizon length k. For both FF training and stage 1 of RNN training, we set the initial learning rate to 1e-4, with an exponential decay rate of 0.7 every 10 epochs. We train with a total of 200 epochs with the Adam optimizer and batch size of 16, and set gradient clipping at 1. For stage 2 of RNN training, we use the same settings, except for the initial learning rate of 1e-5 and a total of 100 epochs.

For our method, and both baselines, we compare the loss on the testing dataset, for different history length and prediction horizon, shown in Fig. 5. The trend in the loss on the rows and columns of the grid shows that a longer history length leads to a smaller prediction error and prediction errors increase with prediction horizon, for both RNN and FF. In addition, RNN outperforms FF consistently, while using fewer neural network parameters (excluding the multi-modal encoding layers, RNN is approximately 1/3 of the size of FF with l = 1). To give a sense of the scale of the prediction error, an average prediction error of 4.5 mm in position, 0.18 rad in angle, and 0.10 N in force is equal to a loss of about 0.3. Finally, we demonstrate the dynamics prediction and its quality along a trajectory in Fig. 6.

1) Ablation studies: In the ablation studies, we intend to answer whether the visual and tactile modalities help improve state prediction. We also look into whether we could use pretrained visual features on other tasks to improve training efficiency. We train separate RNN models for each of the methods listed in Table I for $l = k = 7$, and report the loss on the testing dataset. For pretrained visual features, we use pretrained RNN weights on the ImageNet dataset [31]. The results show that the visual features learned on the ImageNet are not suitable for this task. At the same time, each modality decreases the loss, although it seems that the vision plays a smaller role than tactile information. This could indicate that the surface of the object pile is less useful. However, we suspect that this is more likely due to the fact that end-to-end learning does not lead to the best visual features, and we would like to pretrain the visual features with auxiliary tasks designed for excavation in the future.

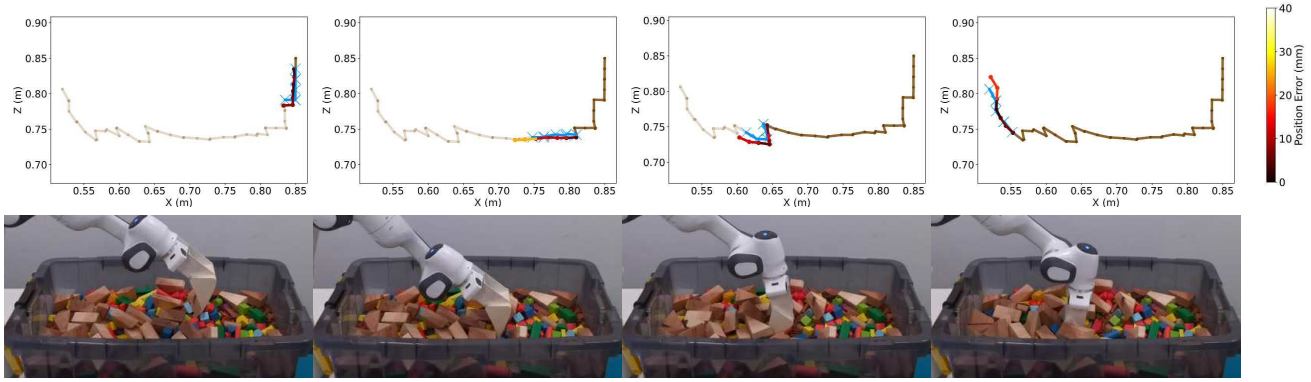

Fig. 6. Four snapshots of position prediction errors along a testing trajectory with RNN. In every picture of the top row, the brown line represents the x-z position of the entire trajectory, with future trajectory shown with transparency. The blue crosses are the ground-truth positions for the 7 future states. The colored line are the 7 positions predicted by the learned dynamics, with colors representing position error. The bottom row contains the corresponding excavation scene. [Best viewed in color.]

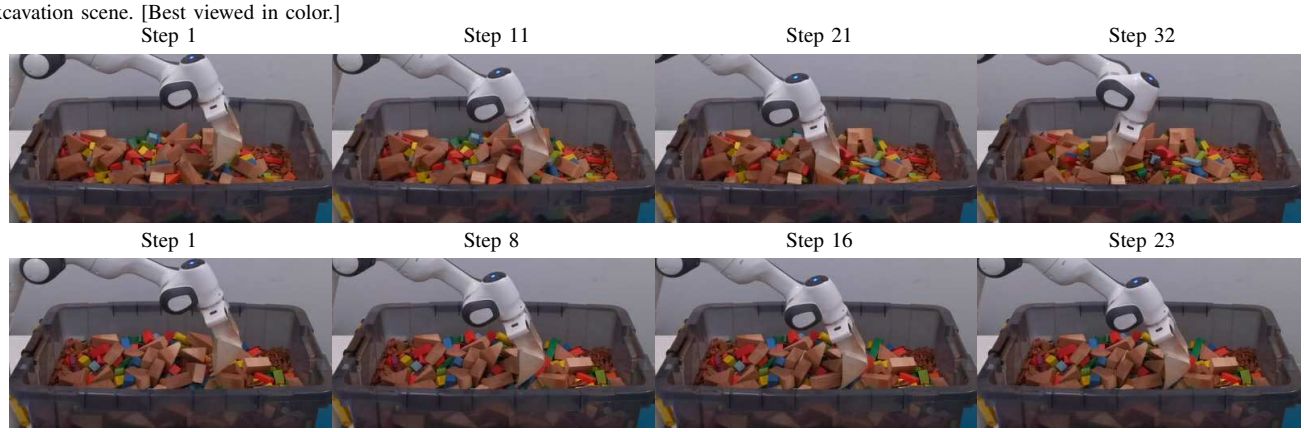

Fig. 7. The top and bottom rows show two example trajectories executed with MCTS, $k = 5$, tracking the deep trajectory. In the top row, a few objects are successfully excavated, while the excavation is aborted due to large contact forces in the bottom row.

TABLE I
DYNAMICS ABLATION STUDIES

Method	Testing Loss
all modalities	**0.02191**
all modalities w/ pretrained CNN	0.02606
no vision	0.02217
no tactile	0.02356
no vision nor tactile	0.02370

Fig. 8. Two cases that led to jamming. On the left, big blocks prevented penetration. On the right, the digging bucket teeth caught a red concave object. [Best viewed in color.]

C. Excavation Experiments

For the excavation experiments, in addition to evaluating the performance of the 3 different planners, we look into whether using wiggling helps excavation, and whether using a longer prediction horizon improves planner performance despite the larger errors in prediction. We compare the planners against 3 baselines, Follow, Closed-loop, and Open-loop, defined in Section VI-A. We use three previously unseen trajectories to evaluate the methods. Relative to the surface of the fragmented rocks pile, the trajectories are shallow, medium, and deep in depth, with the deepest trajectory about 8-10 cm beneath the surface during the dragging phase. We repeat 10 trials on each of the 3 trajectories, and we manually randomize and reset the object pile after each trial. If the robot gets jammed during any of the trials, we abort the current trial. Throughout the experiments in this subsection, we use the same dynamics model architecture with the best performance, which is RNN with all modalities. Due to the small size of the dataset, we train a final model on both the training and testing data, and stop training when the validation error is minimized. We use a history length $l = 7$ for all the experiments.

We use 4 metrics to compare the performance of different planners, which are the number of trajectories that lead to jamming, the average position and angle tracking errors for the trajectories that do not jam, and the amount of material removed per action. The first three metrics are directly related to the terms in the cost function for tracking error and large contact forces. Since the ultimate goal in the real

TABLE II
EXCAVATION EXPERIMENTS FOR SHALLOW/MEDIUM/DEEP TRAJECTORIES

Method	% of Jam	Pos. Err. (cm)	Ang. Err. (°)	Weight per Action (g)
Follow	50% / 60% / 100%	0.61 / 0.64 / -	2.2 / 2.2 / -	1.00 / 2.10 / 0.0
Closed-loop	10% / 20% / 60%	1.5 / 1.3 / 1.4	9.1 / 8.5 / 7.7	0.202 / 0.688 / 1.55
Open-loop	10% / 10% / 30%	1.2 / 1.4 / 2.0	5.4 / 6.4 / 6.5	0.341 / 0.669 / 1.40
Brute-force, k = 3	0% / 0% / 20%	1.6 / 2.0 / 1.6	2.9 / 2.8 / 2.5	0.971 / 1.35 / 1.47
Brute-force, k = 5	10% / 10% / 20%	1.6 / 1.7 / 2.2	2.1 / 2.4 / 2.7	0.954 / 2.37 / 3.13
RS, k = 5	30% / 10% / 30%	1.5 / 2.0 / 2.4	2.3 / 2.6 / 3.1	1.67 / 2.29 / 2.53
RS, k = 7	10% / 10% / 10%	1.4 / 1.9 / 2.8	2.9 / 3.2 / 3.5	1.65 / 2.64 / 2.90
MCTS, k = 5	10% / 10% / 20%	1.4 / 1.8 / 1.9	3.2 / 3.1 / 2.7	1.34 / 2.23 / 2.77
MCTS, k = 7	40% / 10% / 20%	1.2 / 2.3 / 2.0	3.3 / 3.3 / 3.2	1.84 / 1.45 / 3.11

world is to maximize the amount of materials removed per excavator action, we also use the weight per action metric. For the trajectories that lead to jamming, we simply count the actions that have been executed before jamming happens, and consider the excavated weight as 0. The results are shown in Table II. We experiment with prediction horizons of 3 and 5 for brute-force, and do not experiment with 7 due to computational tractability. For RS and MCTS, we experiment with prediction horizons of 5 and 7, all with a computation budget of 20,000 simulations. We show in Fig. 7 two excavation trajectories, one with successfully excavated objects and one that jammed, by MCTS with $k = 5$ tracking the deep trajectory.

Comparing Follow against the other two baselines, it is obvious that the use of wiggling helps reduce large contact forces, without which excavation always fails for a deep trajectory. All of our planners, with different settings, are able to significantly outperform the baselines. The results for brute-force demonstrate that using a prediction horizon of 5 instead of 3 improves excavation performance. Using an even longer prediction horizon of 7 shows different results for RS and MCTS, where RS is improved and MCTS is slightly worse. Overall, brute-force with k = 5, RS with k = 7, and MCTS with k = 5 achieve the best performance, but with brute-force using a greater amount of computation (59,049 compared to 20,000 simulations). We expected MCTS to outperform RS, a simple strategy, but we do not see such results from the experiments. We think that for the task of excavation, during the search in MCTS, the node with the lowest average cost might not be along the best trajectory. For example, an action of going down when the contact force is large, could lead to large costs on average for all possible future trajectories due to penalty for contact forces. However, the optimal trajectory might require the action of going down first.

One advantage of the method is data efficiency, where using only a total of about 300 trajectories result in policies that perform significant better than manually designed strategies. The data efficiency of method mainly comes from a model-based formulation and a well-designed action set of motion primitives where we encode domain knowledge.

We show two common cases that lead to jamming in Fig. 8, including a big object on the way of penetration and a concave object that catches the digging bucket teeth and results in excessive contact forces. In the first case, our planner is unable to plan a local trajectory to get around it, due to limitation in the planning horizon as a longer planning horizon leads to increasingly inaccurate dynamics predictions. We believe that, in this case, a re-planning of the global reference path is necessary and we plan to investigate this in the future.

VII. CONCLUSION

In conclusion, we propose a multi-modal MBRL approach for the task of fragmented rocks excavation. With a discrete action space of excavation primitives encoded with domain knowledge and the multi-modal RNN-based dynamics architecture, we show that we can learn the dynamics function from a small real-world dataset reasonably well to be used in planning. In addition, our MPC is able to significantly outperform manually designed strategies for tracking a global reference path, while brute-force, RS, and MCTS do not differ significantly in performance from each other.

In the future, we would like to extend our work to combine the re-planning of global reference path when our local planner is struggling. The results also show that the visual features improve prediction accuracy marginally, and we would like to investigate better learning of the visual features, possibly by pretraining them with auxiliary tasks. Finally, we would like to implement our method on real excavators. One foreseeable challenge is the amount of training data needed compared to a controlled experiment in this paper, and we intend to collaborate with industrial partners to scale up data collection.

REFERENCES

[1] S. M. Marsh and D. E. Fosbroke, "Trends of occupational fatalities involving machines, United States, 1992-2010," *Am J Ind Med*, 2015.

[2] S Dadhich, U Bodin, and U Andersson, "Key challenges in automation of earth-moving machines," *Automation in Construction*, vol. 68, pp. 212–222, 2016.

[3] A. A. Dobson, J. A. Marshall, and J. Larsson, "Admittance Control for Robotic Loading: Design and Experiments with a 1-Tonne Loader and a 14-Tonne Load-Haul-Dump Machine," *Journal of Field Robotics*, vol. 34, no. 1, pp. 123–150, 2017.

[4] H. Fernando, J. A. Marshall, and J. Larsson, "Iterative learning-based admittance control for autonomous excavation," *Journal of Intelligent & Robotic Systems*, vol. 96, no. 3, pp. 493–500, 2019.

[5] S. Dadhich, U. Bodin, F. Sandin, and U. Andersson, "Machine learning approach to automatic bucket loading," in *24th Mediterranean Conference on Control and Automation*, 2016.

[6] O. Azulay and A. Shapiro, "Wheel Loader Scooping Controller Using Deep Reinforcement Learning," *IEEE Access*, vol. 9, pp. 24 145–24 154, 2021.

[7] Q. Lu, Y. Zhu, and L. Zhang, "Excavation Reinforcement Learning Using Geometric Representation," *arXiv*, 2022.

[8] P. Egli and M. Hutter, "Towards RL-Based Hydraulic Excavator Automation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2020)*, 2020.

[9] G. J. Maeda, I. R. Manchester, and D. C. Rye, "Combined ILC and disturbance observer for the rejection of near-repetitive disturbances, with application to excavation," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 5, pp. 1754–1769, 2015.

[10] F. E. Sotiropoulos and H. H. Asada, "A model-free extremum-seeking approach to autonomous excavator control based on output power maximization," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1005–1012, 2019.

[11] Y. Yang, P. Long, X. Song, J. Pan, L. Zhang, X. Song, and L. Zhang, "Optimization-Based Framework for Excavation Trajectory Generation," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1479–1486, 2021.

[12] R. J. Sandzimier and H. H. Asada, "A Data-Driven Approach to Prediction and Optimal Bucket-Filling Control for Autonomous Excavators," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2682–2689, 2020.

[13] D. Jud, G. Hottiger, P. Leemann, and M. Hutter, "Planning and Control for Autonomous Excavation," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2151–2158, 2017.

[14] J. Seo, S. Lee, J. Kim, and S. K. Kim, "Task planner design for an automated excavation system," *Automation in Construction*, vol. 20, no. 7, pp. 954–966, 2011.

[15] L. Zhang, J. Zhao, P. Long, L. Wang, L. Qian, F. Lu, X. Song, and D. Manocha, "An autonomous excavator system for material loading tasks," *Science Robotics*, vol. 6, no. 55, 2021.

[16] Q. Lu and L. Zhang, "Excavation Learning for Rigid Objects in Clutter," *IEEE Robotics and Automation Letters*, 2021.

[17] D. Kappler, P. Pastor, M. Kalakrishnan, M. Wüthrich, and S. Schaal, "Data-Driven Online Decision Making for Autonomous Manipulation," in *Robotics: Science and Systems*, 2015.

[18] R. Calandra, A. Owens, D. Jayaraman, J. Lin, W. Yuan, J. Malik, E. H. Adelson, and S. Levine, "More than a feeling: Learning to grasp and regrasp using vision and touch," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3300–3307, 2018.

[19] M. A. Lee, Y. Zhu, P. Zachares, M. Tan, K. Srinivasan, S. Savarese, L. Fei-Fei, A. Garg, A. Garg, and J. Bohg, "Making Sense of Vision and Touch: Learning Multimodal Representations for Contact-Rich Tasks," *IEEE Transactions on Robotics*, vol. 36, no. 3, pp. 582–596, 2020.

[20] Y. Liu, D. Romeres, D. K. Jha, and D. Nikovski, "Understanding Multi-Modal Perception Using Behavioral Cloning for Peg-In-a-Hole Insertion Tasks," *arXiv*, 2020.

[21] M. V. Balakuntala, U. Kaur, X. Ma, J. Wachs, and R. M. Voyles, "Learning Multimodal Contact-Rich Skills from Demonstrations Without Reward Engineering," in *IEEE Int'l. Conf. on Robotics and Automation*, 2021.

[22] Z. Wu, W. Lian, V. Unhelkar, M. Tomizuka, and S. Schaal, "Learning Dense Rewards for Contact-Rich Manipulation Tasks," in *IEEE Int'l. Conf. on Robotics and Automation*, 2021.

[23] F. Yi, W. Fu, and H. Liang, "Model-based Reinforcement Learning: A Survey," in *Int'l Conf. on Electronic Business*, vol. 2018-Decem, International Consortium for Electronic Business, 2020, pp. 421–429.

[24] T. Wang, X. Bao, I. Clavera, J. Hoang, Y. Wen, E. Langlois, S. Zhang, G. Zhang, P. Abbeel, and J. Ba, "Benchmarking Model-Based Reinforcement Learning," *arXiv*, 2019.

[25] S. Sing, "Synthesis of tactical plans for robotic excavation," Ph.D. dissertation, Carnegie Mellon University, 1995.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016.

[27] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[28] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Conference on Empirical Methods in Natural Language Processing*, 2014.

[29] D. Bahdanau, K. H. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *International Conference on Learning Representations*, 2015.

[30] I. Lenz, R. Knepper, and A. Saxena, "DeepMPC: Learning deep latent features for model predictive control," in *Robotics: Science and Systems*, 2015.

[31] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Conference on Computer Vision and Pattern Recognition*, 2010.