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

Imitation Learning and Model Integrated Excavator Trajectory Planning

Qiangqiang Guo[1], Zhixian Ye[2,*], Liyang Wang[2], Liangjun Zhang[2]

Abstract— **Automated excavation is promising to improve the safety and efficiency of excavators, and trajectory planning is one of the most important techniques. In this paper, we propose a two-stage method that integrates data-driven imitation learning and model-based trajectory optimization to generate optimal trajectories for autonomous excavators. We firstly train a deep neural network using demonstration data to mimic the operation patterns of human experts under various terrain states including their geometry shape and material type. Then, we use a stochastic trajectory optimization method to improve the trajectory generated by the neural network to guarantee kinematics feasibility, improve smoothness, satisfy hard constraints, and achieve desired excavation volumes. We test the proposed algorithm on a Franka robot arm equipped with a bucket end-effector. We further evaluate our method on different material types, such as sand and rigid blocks. The experimental results show that the proposed two-stage algorithm by combining expert knowledge and model optimization can increase the excavation weights by up to 24.77% meanwhile with low variance.**

I. Introduction

Excavators have been widely used in the construction and mining industries. Many excavation tasks are challenging and often need to be conducted under hazardous environments, making it hard for humans to safely and efficiently operate the excavators. With the rise of sensor and computation techniques, autonomous excavators [1] have received more and more attention due to the potential to guarantee safety, increase efficiency, and reduce cost [2] [3] [4]. Technically, there are many aspects, which should be considered in order to develop autonomous excavators: perception, planning, control, system integration, etc. As one of the most important components, excavation trajectory planning is considered in this paper.

In general, there are two approaches to generate trajectories for autonomous excavators: 1) model-based trajectory planning, and 2) learning-based trajectory planning. Model-based trajectory planning methods can be further categorized as rule-based and optimization-based. Rule-based planning methods generate trajectories using a set of rules given the current excavation condition [2], [5]. After fine-tuning, rule-based methods can perform pretty well under specific fixed environments. However, it lacks generalization ability, making it hard to be efficiently applied to dynamic environments. Optimization-based planning methods try to generate trajectories that can minimize (or maximize) certain objectives based on the excavator's kinematic and inverse kinematic dynamic models and the environment information [6], [7]. Such methods can work under different excavation environments, provided the excavator model is correct and the environment information is sufficient. However, optimization-based methods suffer from two disadvantages. First and most importantly, it is usually very difficult to model the interaction dynamics between the excavator and the environment, which makes it hard to accurately optimize the trajectory. Although one can fine-tune a set of empirical parameters to adjust the dynamics, such parameters need to be re-tuned when the excavation task changes, e.g., the interaction dynamics should be different between digging sand and digging gravel. Second, an optimization-based trajectory planning problem might have multiple solutions, or the solutions might be different given different initial trajectories, which reduces the stability of the planning process.

Learning-based trajectory planning methods aim to generate excavation trajectories from data, either from self-generated data, i.e., through reinforcement learning (RL) [8], [9], or from real-world expert data, i.e., through imitation learning (IL) [10]- [11]. The RL methods adopt the concept of agent learning actions under different environments with the aim of maximizing a pre-defined expected cumulative reward through interactions with the environment [12]. Trajectory planning and excavator control are usually coupled in RL methods, i.e., the agent learns the optimal actions (i.e., joint position, velocity, acceleration, or even hydraulic cylinder's pressure) for different states, and such actions make up trajectories. [8] proposed a RL-based excavator control method using the Proximal Policy Optimization with Covariance Matrix Adaptation (PPO-CMA) algorithm. Simulation results showed that the poposed method could move up to 97% of a predefined mass. There are two drawbacks of RL-based methods. First, the agent requires numerous data to fully explore possible actions for different environments, thus lacks data efficiency. Second, consequently, RL-based methods are usually trained in simulation, since it is really time and cost consuming, and even dangerous to train an RL agent in real-world. However, there are usually huge gaps between simulation models and real-world physics, making it hard for RL-based methods to be transferred to real-world applications. The IL-based methods aim to mimic

*This work was not supported by any organization

[1]Qiangqiang Guo finished this work during internship in Baidu USA, Email: guoqq77@gmail.com

[2]Zhixian Ye, Liyang Wang, and Liangjun Zhang are with the Robotics and Autonomous Driving Lab., Baidu USA, Sunnyvale, CA, 94089, USA. Emails: zhixianye, liyangwang, liangjunzhang@baidu.com

* Corresponding author

2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-7927-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/IROS47612.2022.9981220

the trajectories generated by human experts [13] - [14]. For example, [11] proposed a linear regression based imitation learning model learned from expert wheel loader drivers, and [10] improved the data efficiency of the imitation-based control using clustering and association analysis. The main disadvantage of IL-based trajectory planning methods is that, if the expert data can not cover all environment conditions, or the data is severely biased, the learned policy map may fail under those environments that are rarely encountered. Besides, the expert operations might not be optimal considering that excavation is a very complicated task. In addition, the trajectories generated by such methods may not be feasible, e.g., do not satisfy the inverse kinematics or violate hard constraints, especially when the policy is learned by a neural network.

In this paper, we propose an IL and model integrated method for excavation trajectory planning. We first collect expert trajectories under various environments and learn a policy network from expert data through imitation learning. Then, we apply and update the learned policy in a DAGGER (Dataset Aggregation, [15]) fashion. If the trajectory generated by the policy network is obviously not optimal (checked by human experts), the human experts will operate the excavator to finish the excavation task, and the expert trajectories will be added to the dataset of IL. If the experts think that the trajectory is good, it will be used as the reference and the initial trajectory for the lower-level model-based trajectory planning algorithm, which will improve such a trajectory in terms of satisfying hard constraints, guaranteeing correct kinematics, increasing smoothness, and minimizing user-defined objectives. Finally, the generated trajectory will be applied to a real excavator robot, which will be evaluated by experts and added to the dataset of IL.

To summarize, our main contributions are:

- We present a two-stage method for excavation trajectory generation. Our method can take advantage of the benefits of both model-based and learning-based methods, and avoid their disadvantages, i.e., we can get expert trajectory as guidance from IL and make the trajectory feasible/optimal using the model-based method;
- Our method can be generalized for handling different geometry and material types of terrain;
- We thoroughly tested this integrated method in real robots across different material environments. The experimental results show that the proposed method can increase the excavation weights by up to 24.77% meanwhile with low variance.

In the following of this paper, we first define the excavation task in Section and present the methodologies used in Section II. In Section III, we show the experiment settings, results, and discussions. Finally, we conclude this paper and discuss future directions in Section IV.

II. Methodology

We consider the trajectory planning for single excavation given the terrain and the point of attack (POA, the point that excavator bucket contacs the material first). Formally, let s be the state of the environment which consists of the terrain information and excavator status, a be the action, i.e., the trajectory under the current state. The goal is to design a policy framework $\pi(s)$ such that $a = \pi(s)$ is the optimal action under s. We propose a two-stage IL and model integrated policy framework $\pi(s) = \pi_{\text{Model}}(\pi_{\text{IL}}(s))$. We show this framework in Fig. 1.

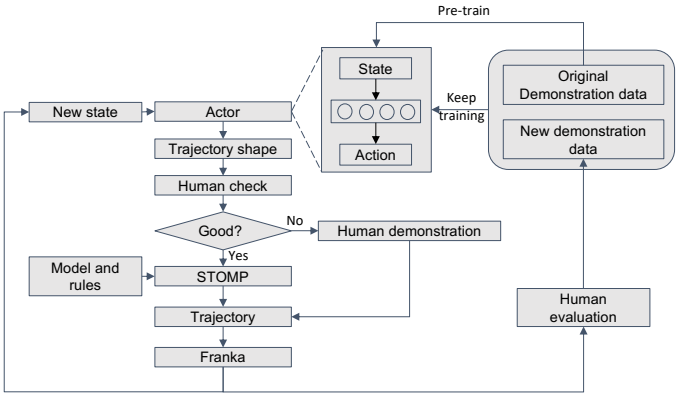


Fig. 1: Algorithm framework

We first use the demonstration data collected from human operations to train an actor neural network, which acts as $\pi_{\text{IL}}(.)$. Given a new state s, we get an initial trajectory shape by $\pi_{\text{IL}}(s)$. Then, we manually check whether this trajectory shape is good or not for the current state. If the trajectory shape is good, we use it as the initial trajectory and the guidance trajectory in the stochastic trajectory optimization for motion planning (STOMP, [16]) optimization module, which acts as $\pi_{\text{Model}}(\pi_{\text{IL}}(s))$. The STOMP considers the excavator's kinematic models to guarantee the feasibility, the hard constraints such as the excavation boundaries to enhance safety, the smoothness of the trajectory to reduce energy, and some user-defined objectives to improve the initial trajectory. The optimized trajectory is then applied to the Franka Panda robot. If the trajectory shape generated by $\pi_{\text{IL}}(.)$ is not good, we will show a trajectory by directly operating the robot. Under both conditions, the robot should execute a trajectory a (either optimized by STOMP or shown by human). After the excavation, we will evaluate the trajectory by a value r from 0 (acceptable) to 10 (very good), then store the (s, a, r) tuple as new demonstration data. The actor-network is then updated using the aggregated demonstration data set.

In the following of this section, we first show the definition of the state s, action a, and the structure of the actor neural network $\pi_{\text{IL}}(.)$. Then, we show the mechanism of the STOMP algorithm and how we integrate the trajectory shape, kinematics, hard constraints, and user-defined objectives into it.

A. First stage: IL-based actor

We use terrain information as the state s, including geometric shape and material type.

1) Geometric shape:

For the terrain shape, it is usually captured by a grid map in real-world applications. A grid map can be represented by a two-dimensional array, with the size determined by precision. One approach to formulate the geometric information s_g is to directly use the entire grid map. However, such an approach will make the actor-network complicated, e.g., the convolution neural network (CNN) might be needed to improve the mapping performance. And the large state space requires a large amount of data to train the actor-network. In fact, since the excavator base usually does not swing during a single excavation, once the POA is fixed, the excavation process is only related to a specific part of the terrain, i.e., the neighbor areas along the straight line between the bucket point and the excavator base. As shown in Fig. 3, the excavation-related terrain is the orange area. Considering this pattern, we propose a sectional-volume-based state representation. We first uniformly segment the orange area to n sections. For each section, we go through each point and calculate the relative height, i.e., the absolute height of the grid map point minus the absolute height of the bucket point, and add them up as the feature for that section. In this way, we can get a $s_g \in \mathbb{R}^{n\times 1}$ with the terrain information relative to the POA point.

2) Material type:

In reality, the material of the terrain will affect the characteristics of the trajectory. Compared with the trajectories for the sand-like material, the trajectories of rocky terrain should be shallower and the trajectory points will be denser (sampling in the time domain with equal time period). In that case, we have to take material type into our consideration when generating the trajectories. As shown in Fig. 2, we incorporate a classification model into the whole pipeline. For simplicity, we only capture one image of the terrain while getting the point-cloud of the terrain at the same time, and then input this image into our pre-trained classifier to get the material feature (or material label) s_m for our next stage.

The state vector now includes both the geometric and the material information about the terrain. As shown in Fig. 2, we concatenate the geometric feature s_g and material feature vector s_m into the state vector s, which is $\{s_g, s_m\}$.

In order to reduce the network size as well as the training data, we uniformly (in the time domain) select m points from an excavation trajectory as the action a. Note that a trajectory can be represented by a sequence of 4 joint states (i.e., swing, boom, arm, and bucket), or a sequence of 4 dimensional bucket positions (i.e., x,y,z, and the angle of the bucket point). For this problem, there are no swing operations, indicating the number of joints can be reduced to 3 and the bucket position can be represented by 3 variables (i.e., distance to the excavator base in the x-y plane, z, and the angle of the bucket point). Therefore, we get an $a \in \mathbb{R}^{3m\times 1}$.

Given the collected demonstration data $D = \{(s_1, a_1), (s_2, a_2), \dots (s_N, a_N)\}$ where N is the number of demonstrations, we train an actor neural network π_{IL} to minimize the mean square error (MSE) between the demonstration actions and the predicted actions, i.e., $\min \frac{1}{N}\sum_i^N (a_i - \pi_{\text{IL}}(s_i))^2$. As shown in Fig. 1, this actor

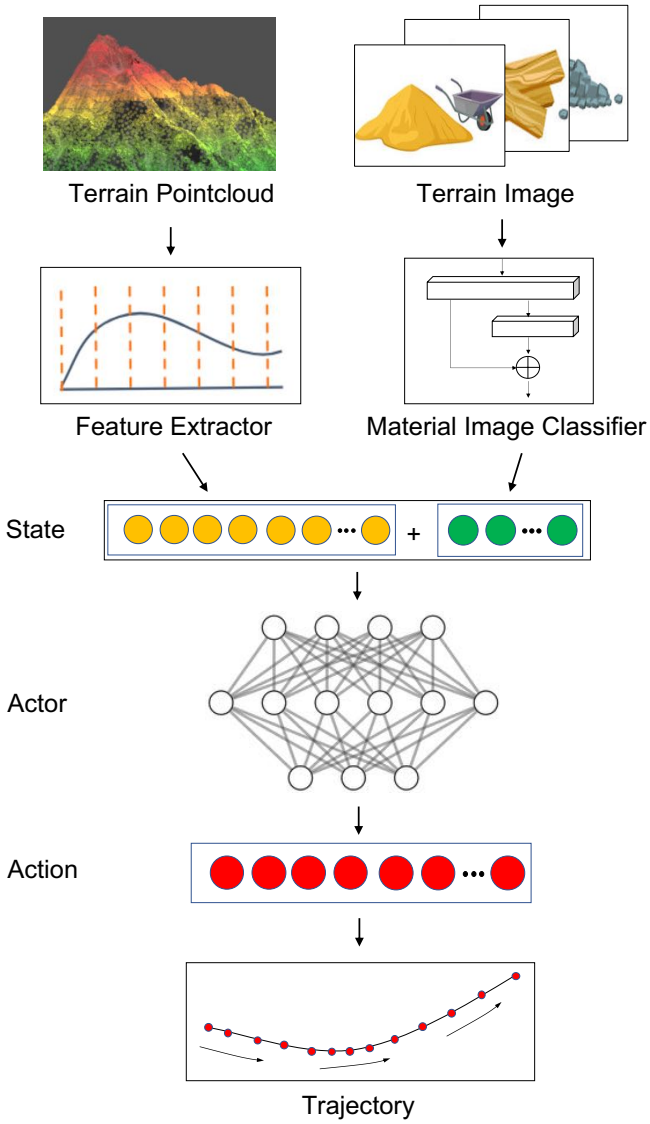


Fig. 2: Architecture of actor trajectory generation

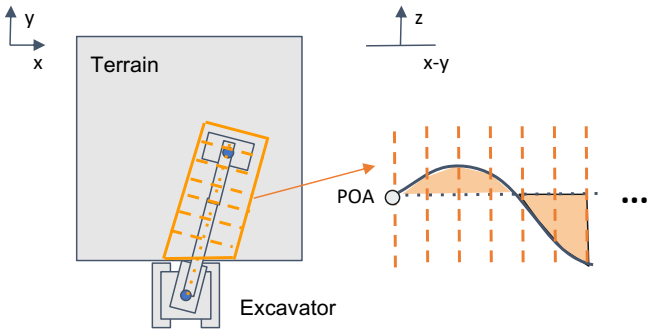


Fig. 3: An illustration of the geometric feature extraction of the excavation-related terrain: top view of the terrain and the excavator (left), and side view of the terrain for feature extraction(right)

network will be re-trained when new demonstration data is available. The new demonstration data come with human evaluation, i.e., a value ranging from 0 to 10 which indicates the trajectory is just "acceptable" to "very good". We use an importance sampling technique to re-train the actor network. Trajectories with higher evaluation values have higher possibilities to be selected.

B. Second stage: STOMP optimizer

STOMP is a stochastic trajectory optimization method. It explores the trajectory space by generating random noisy trajectories around an initial trajectory and updates the trajectory based on the cost of candidate trajectories in a probabilistic way. Details of STOMP can be found at [16]. In this paper, we select STOMP as the second stage optimization method due to the high flexibility (e.g., no gradient information is needed) and efficiency (e.g., can converge in a small number of iterations once the hyper-parameters are fined tuned). The task for the second stage STOMP optimizer is to improve the trajectory shape given by the first stage IL actor such that the excavator kinematics and hard constraints are satisfied, the trajectory is smooth, the initial trajectory generated by the IL actor is considered, and the user-defined objectives are maximized.

To satisfy the excavator kinematics, we use the joint state space as the trajectory space. For each iteration, the STOMP generates an updated sequence of joint states, which are guaranteed to satisfy the kinematics and inverse kinematics. Note that the bucket positions are needed to calculate other cost values, we calculate the bucket position $p_i = [x_i, y_i, z_i, \alpha_i]$ from the joint state of a trajectory point using the kinematics.

We enhance the other three considerations by adding them to the cost functions of STOMP. The hard constraints of the excavation tasks mainly indicate that the materials outside the excavation area should not be touched. For example, if the task is to load a pile of sand standing on the solid ground to a truck, the excavator should not penetrate the ground. If the task is to dig a trench, the excavator should not touch any area outside the interested trench cube. In this paper, we add an exponentially increased penalty to the violation of hard constraints. Mathematically, let $A \in \mathbb{R}^3$ be the 3-dimensional excavation area, the hard constraint cost for a trajectory point p_i is defined as

$$c_i^{\text{hc}} = \begin{cases} e^{w_{\text{hc}} d(p_i, A)}, & \text{if } p_i \notin A \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where $d(p_i, A)$ is the distance from p_i to A and w_{hc} is the weight. The minimum Euclidean distance is used as the $d(.)$ in this paper. Larger w_{hc} indicates more rigorous requirements to satisfy the hard constraints.

The smoothness of a trajectory has already been discussed in [16], we adopt the same method to improve the smoothness, the cost of which is defined as C^{sm}. Note that the capital C indicates this cost is calculated for the entire trajectory.

Although not accurate, the trajectory generated by the IL actor contains valuable guidance for the general trend of the desired trajectory. The goal of STOMP is to improve the trajectory generated by the IL actor. The updated trajectory should not deviate from the initial trajectory too much. To enhance the role of the initial trajectory, we add a penalty for deviating from the initial trajectory. For a trajectory point p_i, let $\hat{p}_i$ be the corresponding initial trajectory point, the cost is defined as

$$c_i^{\text{in}} = w_{\text{in}} \| p_i, \hat{p}_i \| \tag{2}$$

User-defined objectives are used to regulate the initial trajectory generated by the IL actor using empirical knowledge so that the updated trajectory is reliable. In this paper, we add volume-based knowledge to the cost function, i.e., the excavation volume should be a constant value. Ideally, if the interactions between the terrain and the bucket can be accurately modeled, we can precisely calculate the volume for a trajectory. However, it is very difficult to model such interactions, and the computation burden is usually very heavy even if it can be modeled. In this paper, we simply use the volume surrounded by the trajectory and the terrain as the volume that can be excavated. Considering the interaction errors, we add an adjustment factor to the raw volume. After fine-tuning, this method is shown to be quite efficient when the materials are uniform. Let V_{raw} be the raw volume calculated from the trajectory and terrain, $\hat{V}$ be the desired volume, the volume cost is

$$C^{\text{vo}} = w_{\text{vo}} e^{\| V_{\text{raw}} - \hat{V} \|} \tag{3}$$

In summary, the trajectory-based cost function for STOMP is

$$C^{\text{traj}} = C^{\text{sm}} + C^{\text{vo}} \tag{4}$$

For each trajectory point p_i, the point cost function is

$$c_i^{\text{point}} = c_i^{\text{hc}} + c_i^{\text{in}} \tag{5}$$

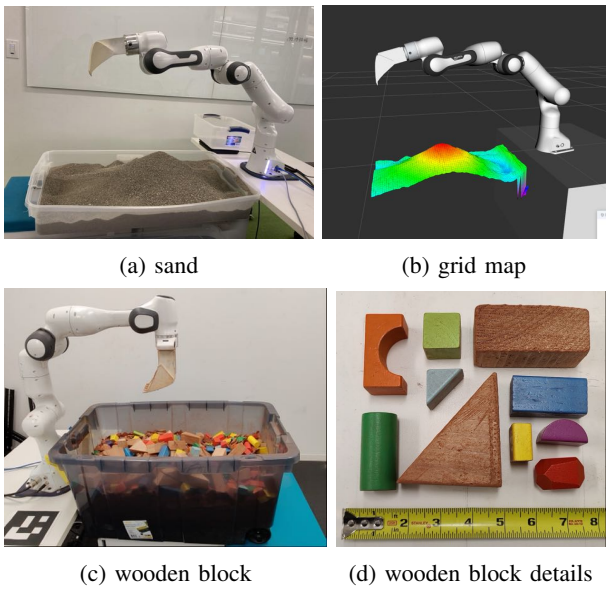
III. Experiment

A. Experiment settings

We test the proposed algorithm on a Franka Panda robot and select sand and wooden block as the excavation material. The experiment platform is shown in Fig. 4a and Fig. 4c. Specifically, we have prepared various sizes of the wooden block to mimic the real environment as we can, and the types and sizes of the wooden block are shown in Fig. 4d.

We use the Microsoft Azure Kinect to get the point-cloud of the terrain and transform it into a grid map. The boundary of the excavation area is set as a $0.8m \times 0.4m$ rectangle in front of the robot base origin with $0.2m$ offset. Fig. 4b

We test three trajectory planning methods: the proposed method (denoted as IL+STOMP), pure imitation learning (denoted as pure-IL), and pure STOMP (pure-STOMP). For the pure-IL learning method, we do not revise the trajectory generated by the IL actor, and directly apply it to the robot. The action is defined at the bucket space, some points of the trajectory might not satisfy the inverse kinematics. We use a linear interpolation method to complete the trajectory. For pure-STOMP, we do not use the initial trajectory generated by the IL actor. Linear initialization is used given the POA



(a) sand (b) grid map

(c) wooden block (d) wooden block details

Fig. 4: Experiment settings: Two types of material for experiment, (a) sand and (c) wooden block; (b) Terrain visualization using grid map [17]; (d) Sizes of the wooden block

and the ending point of the trajectory. During the optimization, we do not take account of the trajectory deviation cost c_i^{in} since there is no reference trajectory. Other costs and settings are the same as those in IL+STOMP.

Apart from testing on the sand material, we also test our method on the wooden block material to verify our various types of actors. We have trained two types of actors, namely the generic actor, and the specialized actor. The generic actor is trained with all the episodes of the data, while the specialized actor is trained with the episodes data containing only one specific type of terrain, such as wooden block material, and its corresponding trajectory. We would like to see how well can our actor generalize in different types of material, where we compare the result of using the pure-STOMP method as our baseline.

For each method, we conduct three episodes of experiments. For each episode, we let each method conduct 15 excavations. This number is based on the observation that it usually takes at least 15 excavations to remove the materials that stands above the zero terrain height (as shown in Fig 4d). The initial terrain for each episode is set as the same by manually modifying the terrain. Then, we design a rule-based POA selection algorithm to automatically find the POA. The key idea is to find the highest point of the terrain and set the POA at a certain distance beyond it. This POA selection algorithm does not change for all experiments.

B. Experiment results and discussions

1) Trajectory geometric comparison:

To better illustrate the difference among the three methods, we show the trajectories given the same terrain state in Fig 5.

In Fig 5, the solid red lines represent the trajectory of the bucket points, solid blue lines are the terrains and dashed

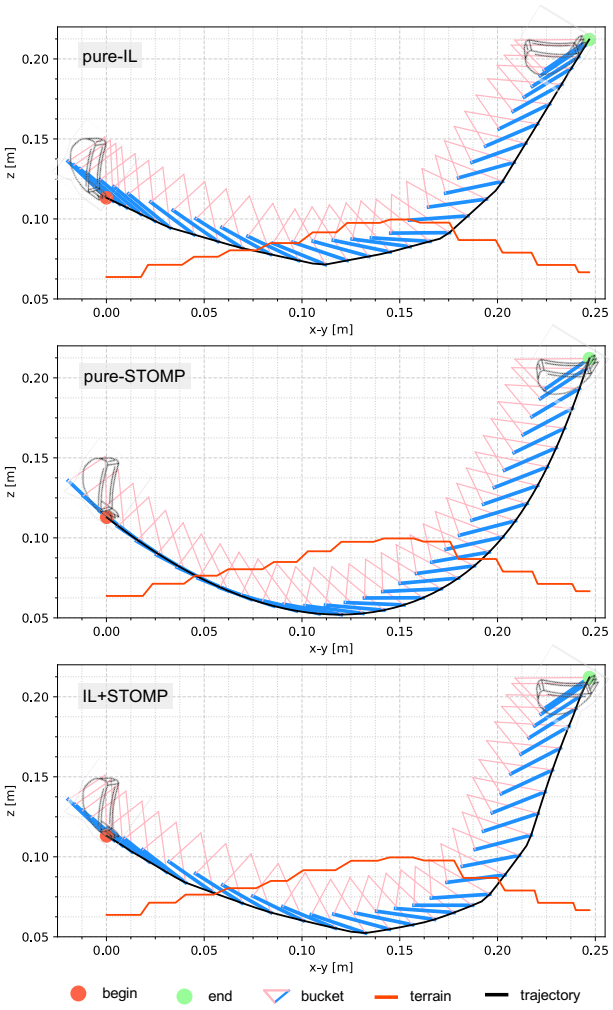


Fig. 5: Trajectories generated by different methods under the same terrain state

blue lines are the tangent lines of the bucket points. For each subplot, the left-most trajectory point is the POA, and the right-most trajectory point is the pre-defined ending point. It can be shown that, compared with pure-IL, the IL+STOMP method can excavate more volumes since the area, surrounded by the trajectory and the terrain, is larger. Compared with pure-STOMP, the IL-STOMP method is also supposed to excavate more volumes if we apply them to the real robot since the bucket angles (i.e., the angle between the dashed blue lines and the $x-y$ axis) of the IL-STOMP method is more reasonable. As shown in the figure, for the IL-STOMP method, the bucket angle reaches 0 at the end of the "drag" operation (i.e., the right most area along the 0 terrain), and quickly increases so that the materials can be excavated into the bucket. However, for the pure-STOMP method, the bucket angle changes almost linearly and the bucket will lift up too late, the materials are more likely to be pushed aside instead of to be excavated into the bucket. This observation can be proved by the experiment results in the next section.

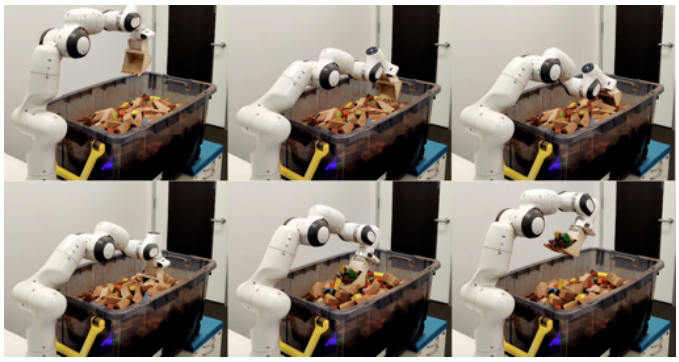


Fig. 6: IL trajectory execution on the robot arm

And in Fig. 5, we plot the final trajectory generated by our IL-STOMP method on the sand piles. The series of triangles in the graph indicate the series of bucket motions of excavation with the blue bold line showing the current angle of the bucket. The visualization graph gives an intuition of the method trying to mimic the human operator's behavior, such as aligning the penetration angle along with the trajectory direction, also adjusting movement pace in the time domain during penetrating and dragging the target material. This type of learned pattern is displayed in another trajectory execution on the robot arm, as shown in Fig. 6. We sample these six snapshots of the entire excavation. Starting from the home position (top-left), the arm first reaches the POA (top-mid), the beginning point of the trajectory. Then it tries to penetrate into the piles (top-right), drag the material inside the bucket (bottom-left), and close the bucket as the bucket is full (bottom-mid). It finishes the trajectory at a fixed given pose (bottom-right).

2) IL and STOMP comparison:

We use the average excavated weight (noted as Avg-W in kg) and the success rate (noted as Suc-R) as the evaluation metrics. For this experiment, we use the sand material as our excavation target. The experiment results are shown in Table I. Note that the "Stdev" is the standard deviation of all 15 excavations for each episode, and the two improvement columns under the IL+STOPM method represent the improvements compared with pure-STOMP and pure-IL, respectively.

TABLE I: Experiment performance of different methods

Epi	Index	pure-STOMP	pure-IL	IL + STOMP		
				value	improvement	
1	Avg-W	0.763	0.905	0.969	26.96%	7.15%
	Stdev	0.176	0.193	0.148	-15.59%	-23.09%
	Suc-R	0.867	1.000	1.000	15.38%	0.00%
2	Avg-W	0.742	0.978	0.964	29.94%	-1.35%
	Stdev	0.146	0.185	0.143	-1.75%	-22.62%
	Suc-R	0.867	1.000	1.000	15.38%	0.00%
3	Avg-W	0.813	0.920	0.960	18.00%	4.30%
	Stdev	0.101	0.236	0.108	6.95%	-54.30%
	Suc-R	0.800	1.000	1.000	25.00%	0.00%
Total	Avg-W	0.773	0.932	0.965	24.77%	3.25%
	Stdev	0.144	0.204	0.131	-8.78%	-35.52%
	Suc-R	0.844	1.000	1.000	18.42%	0.00%

It can be shown that the IL+STOMP method outperforms other methods in terms of both excavation weight and the success rate. Compared with pure-STOMP, IL+STOMP can dramatically improve the excavation weight by up to 29.94% in episode 2 and by 24.24% on average. The main reason is that, for the STOMP optimization stage, the linear initialization strategy used in pure-STOMP is not as good as the initial trajectory generated by the IL actor in IL+STOMP. STOMP is sensitive to the initial trajectory, different initial trajectories usually lead to different exploration directions thus different final trajectories. Meanwhile, for IL+STOMP and pure-STOMP, the overall standard deviations of the excavated weights are almost the same. This is due to the volume cost in STOMP, which is a dominant objective due to the exponential property. The planned trajectories from both pure-STOMP and IL+STOMP tend to surround the same volume, which reduces the planning variance. The IL+STOMP method succeeds in planning all the trajectories, while the average success rate of pure-STOMP is 84.4%. The failure cases result from bad initial trajectories. As an example, the linear initialized trajectory may not penetrate the terrain, leaving no explore space for STOMP to reduce the volume cost. This further indicates the initial trajectories generated by the IL actor are better than the linear initialization strategy used in pure-STOMP.

Same as IL+STOMP, the success rate of pure-IL is also 100%. In addition, the average excavation weight of the pure-IL method is close to IL+STOMP. However, the performance of pure-IL is not stable. The standard deviation of IL-STOMP is 35.52% better than pure-IL. This is due to the fact that the demonstration data is limited. The IL actor trained by the limited data can not cover every terrain state thus introducing errors and disturbances. In fact, the raw trajectory generated by the pure-IL method can not be applied directly to the real robot since there might be trajectory points that do not satisfy the inverse kinematics. We used the linear interpolation method to make the raw trajectories feasible. In addition, without the STOMP optimization, the volumes surrounded by the trajectories generated by the IL actor usually vary a lot, making the standard deviation of excavation weight relatively large (which can be seen from Table I).

TABLE II: Comparison of generic model and specified model of different materials

Model	Baseline	Wooden block		Generic	
Avg-W	0.21248	0.22882	7.69%	0.22754	7.09%
Min-W	0.02530	0.12740	403.56%	0.08190	223.72%
Max-W	0.39370	0.32060	-18.57%	0.32160	-18.31%
Stdev	0.08332	0.05127	-38.47%	0.05391	-35.30%
Suc-R	0.5556	0.8444	52.00%	0.7442	33.95%

3) Material model comparison:

As for the comparison of material type, here we compare our methods with the baseline method. The baseline method is using a fixed template trajectory and later get optimized using pure STOMP for the purpose of kinematic feasibility. Also, we use the same denotation for the weight and success rate like previous experiment.

In Table II, we can see that our method can beat the baseline method easily for 7.69% and 7.09% in terms of average excavation weight accordingly. Both generic model (generic) and specialized model (wooden block) have a lower variance of excavation weight and a better success rate. This is because the baseline model cannot have a good adaption to the terrain when generating the trajectory, while our methods try to learn the hidden characteristics of terrain excavation and express them in the trajectories generated. For example, our models will be more conservative in the rocky terrain because this type of material will incur a higher stuck rate during excavation for the unknown rock distribution under terrain surface, which explains that our method has a lower maximum of excavation weight, but a much higher minimum weight compared to baseline model.

The result shows that the generic model surprisingly achieves a near performance level as the specialized model that has a slightly better average weight. Among our purposed methods, the specialized model displays its proficient capability of finishing the excavation job in its own sophisticated material domain, with a lower excavation weight variance and higher success rate. It shows a stable and strong performance in this relatively difficult material type. The generic model is able to generate similar performance trajectory like the specialized model since it has a close excavation weight, although it still has a small gap in terms of success rate.

IV. Conclusion

In this paper, we proposed an IL and STOMP integrated trajectory planning algorithm for autonomous excavators. We firstly trained an IL actor using demonstration trajectories under various terrain states incorporating geometric and material information. Then, we designed a STOMP algorithm to improve the trajectories generated by the IL actor so that kinematics feasibility is guaranteed, hard constraints are satisfied, smoothness is improved, and the desired volume is approached. Finally, we tested the proposed algorithm on a Franka Panda arm and the results showed that the proposed algorithm could increase the excavation weights by up to 24.77% compared with pure-IL and pure-STOMP methods.

There are several directions that future studies can focus on. First, we used a small demonstration dataset (with 128 trajectories) to train the IL actor. It will be interesting to evaluate the proposed method under a larger and more diversified dataset. We expect that the performances of both the IL+STOMP method and the pure-IL method should increase, especially the latter. Second, after certain episodes of newly collected data, the DAGGER framework could not improve the result any more, at least not in an obvious way. This can be a huge challenge in the real world excavation. We would like to try other more advanced imitation learning or reinforcement learning techniques, such as few-shot learning. Third, it is worthwhile to apply the proposed method to real excavators. To this end, collecting demonstration data, cleaning the data, formulating the objectives in STOMP, applying the trajectory, etc. should be carefully considered.

References

[1] 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. [Online]. Available: https://robotics.sciencemag.org/content/6/55/eabc3164

[2] A. Stentz, J. Bares, S. Singh, and P. Rowe, "A robotic excavator for autonomous truck loading," in *Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No.98CH36190)*, vol. 3, 1998, pp. 1885–1893 vol.3.

[3] 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.

[4] L. Wang, Z. Ye, and L. Zhang, "Hierarchical planning for autonomous excavator on material loading tasks," in *Proceedings of the 38th International Symposium on Automation and Robotics in Construction (ISARC)*. International Association for Automation and Robotics in Construction (IAARC), November 2021.

[5] F.-Y. Wang and P. Lever, "On-line trajectory planning for autonomous robotic excavation based on force/torque sensor measurements," in *Proceedings of 1994 IEEE International Conference on MFI '94. Multisensor Fusion and Integration for Intelligent Systems*, 1994, pp. 371–378.

[6] M. Arsenault, L.-F. Tremblay, and M. Zeinali, "Optimization of trajectory durations based on flow rate scaling for a 4-dof semi-automated hydraulic rockreaker," *Mechanism and Machine Theory*, vol. 143, p. 103632, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094114X19312777

[7] Z. Li, X. Li, S. Liu, and L. Jin, "A study on trajectory planning of hydraulic robotic excavator based on movement stability," in *2016 13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, 2016, pp. 582–586.

[8] B. J. Hodel, "Learning to operate an excavator via policy optimization," *Procedia Computer Science*, vol. 140, pp. 376–382, 2018, cyber Physical Systems and Deep Learning Chicago, Illinois November 5-7, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050918319744

[9] I. Kurinov, G. Orzechowski, P. Hämäläinen, and A. Mikkola, "Automated excavator based on reinforcement learning and multibody system dynamics," *IEEE Access*, vol. 8, pp. 213 998–214 006, 2020.

[10] R. Fukui, T. Niho, M. Nakao, and M. Uetake, "Imitation-based control of automated ore excavator: improvement of autonomous excavation database quality using clustering and association analysis processes," *Advanced Robotics*, vol. 31, no. 11, pp. 595–606, 2017. [Online]. Available: https://doi.org/10.1080/01691864.2017.1297735

[11] S. Dadhich, U. Bodin, F. Sandin, and U. Andersson, "Machine learning approach to automatic bucket loading," in *2016 24th Mediterranean Conference on Control and Automation (MED)*, 2016, pp. 1260–1265.

[12] R. Sutton and A. Barto, *Reinforcement Learning, second edition: An Introduction*, ser. Adaptive Computation and Machine Learning series. MIT Press, 2018. [Online]. Available: https://books.google.com/books?id=uWV0DwAAQBAJ

[13] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation," 2018.

[14] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adversarial inverse reinforcement learning," 2018.

[15] S. Ross, G. J. Gordon, and J. A. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *AISTATS*, 2011.

[16] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 4569–4574.

[17] P. Fankhauser and M. Hutter, "A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation," in *Robot Operating System (ROS) – The Complete Reference (Volume 1)*, A. Koubaa, Ed. Springer, 2016, ch. 5. [Online]. Available: http://www.springer.com/de/book/9783319260525