Towards Autonomous Lunar Resource Excavation via Deep Reinforcement Learning

Joseph M. Cloud*, Rolando J. Nieves†, Adam K. Duke‡, Thomas J. Muller§, Nashir A. Janmohamed¶
Brad C. Buckles‖, Michael A. DuPuis**
NASA Kennedy Space Center, KSC, FL, 32899

To support sustainable infrastructure on the Moon, NASA needs to leverage lunar resources for in-situ processing and construction. NASA's Regolith Advanced Surface Systems Operations Robot (RASSOR) is principally designed to mine and deliver regolith for these tasks. To reliably perform these operations on the lunar surface, RASSOR's sensors and control systems need to be robust and maximize information extracted from a reduced sensor payload. Herein, we present our findings from the Intelligent Capabilities Enhanced RASSOR project. We created reduced-order simulation environments in which we applied reinforcement learning algorithms to learn autonomous trenching controllers and produced state estimation architectures. We developed two simulations: a 2D excavation simulation used to facilitate parameter selection, and a 3D simulation developed using a game physics engine to simulate simplified soil interactions and incorporate robotic agents parameterized by dynamic models. Within these simulations, we learned autonomous excavation routines that exceed excavation efficiency measures as compared against RASSOR's existing control and teleoperation-based methods.

I. Introduction

The affordable establishment of a sustainable human presence on the lunar surface necessitates means of extracting, processing, manufacturing, and constructing using resources available in-situ. In-Situ Resource Utilization (ISRU) is the practice of making use of local resources to create useful products [1]. The lunar surface contains several vital minerals which can be used to address the needs of a sustainable habitat, including: mining water [2], extracting oxygen for use as a rocket propellant and for sustaining a breathable habitat [3], and leveraging the physical properties of lunar regolith for manufacturing and construction activities [4]. Several technologies have been proposed for construction within the concept of operations of a lunar habitat, such as landing pads, berms [5, 6], trenches [7], additive manufacturing using regolith [8, 9], and radiation shields among other structures.

A key engineering challenge within ISRU systems is the excavation and transport of regolith. Low gravity environments such as the Moon lack the tractive force that excavators on Earth use to extract material without slipping or tipping. Additionally, the high cost of delivering payloads to the Moon renders low mass and low excavation force capabilities essential aspects of the design of space excavation systems [10].

Over the last several years, various excavation platforms have been proposed [10]. The Swamp Works rapid development laboratory at NASA's Kennedy Space Center developed the Regolith Advanced Surface Systems Operations Robot (RASSOR) [7] to address these needs. RASSOR is designed to mine and deliver regolith in low gravity environments. It has two sets of counter-rotating drums, enabling it to maintain the reaction force necessary to excavate effectively. Some ISRU mission models [7, 11] estimate that to fulfill mission design criteria for a habitat, excavators will need to operate near continuously to sustain ISRU processing plants. Consequently, manual control of excavators becomes impractical. Other issues such as communications delay from Earth-based control [12] and limited astronaut time on the surface [13] further highlight the need for excavators that are not only physically robust to the environment but also capable of operating autonomously.

In this paper, we present results from the Intelligent Capabilities Enhanced RASSOR (ICE-RASSOR) project to address challenges of autonomous lunar excavator control in simulation. We use reinforcement learning algorithms to

*Robotics Engineering Trainee, Advanced Engineering Development Branch, NE-L6, and AIAA Student Member.
†AST, Exploration Systems & Development Office, UB-E0.
‡Intern, Advanced Engineering Development Branch, NE-L6.
§Intern, Exploration Systems & Development Office, UB-E0.
¶Intern, Advanced Engineering Development Branch, NE-L6.
‖AST, Exploration Systems & Development Office, UB-E0.
**Principal Investigator - Robotics and Autonomous Systems, Advanced Engineering Development Branch, NE-L6.

learn excavation strategies by abstractly rewarding a simulated robot for achieving desirable excavation behavior and penalizing it otherwise. We benchmark these learning methods against controllers implemented analogously to routines available on the physical platform. We developed our simulations to be fast to decrease learning time and facilitate the prototyping of embedded sensor capabilities.

Our first simulation was modeled in two dimensions to facilitate sensor and learning parameter selection. The second simulation was developed using a 3D game physics engine to approximate wheel-soil interaction and increase the fidelity of the dynamic models of the robots within. The development of the 3D simulation has enabled the training of additional sensing capabilities and research both at mechanics and operations levels. We experimented with various virtual sensor payloads to identify a configuration that equipped RASSOR with the perception means to efficiently learn to excavate.

The rest of the paper is structured as follows: in Section II we describe RASSOR 2.0 in more detail, machine learning preliminaries, and existing simulation tools. In Section III we provide an overview of our 2D simulation and reinforcement learning results within it. In Section IV we explain how our 3D simulation works and its modeling effort. Section V builds upon Section IV by describing how the environment is configured for learning and the reward scheme. Next, in Section VI, we present the results of our deep reinforcement learning agent and compare its performance against semi-automated and teleoperated means of control. Finally, in Section VII, we summarize our findings and discuss avenues of future work.

II. Related Work

A. RASSOR 2.0

1. Robot Design

RASSOR 2.0 (referred to simply as *RASSOR*) is a TRL 4 (*Component/subsystem validation in laboratory environment*) lightweight planetary excavation robot. It weighs 66 kg and has an 80 kg regolith payload capacity. It is designed to perform deep regolith excavation and slot trenching in low gravity environments. The symmetric counter-rotating bucket drum design enables RASSOR to manipulate regolith efficiently. RASSOR is equipped with four wheels and two arms, each supporting a set of two bucket drums which are used to excavate the terrain. The drums protrude the wheels enabling RASSOR to dig trenches and then drive into them [14].

In contrast to scientific exploration rovers that are designed for shorter mission durations, excavators such as RASSOR need to be able to continuously operate in adverse conditions and handle abrasive materials for extended periods of time. As such, RASSOR's drums can be used as a contingency mobility system to get the robot unstuck [7]. RASSOR also has the ability to use the combination of its arms and drums to right itself and use contingency formations to traverse steep terrain and deposit regolith, such as in 'Z' formations, and use the weight of the loaded drums to shift the center of gravity. It can increase its traction with loose soil by lowering the arms such that the weight of the robot is distributed between the eight drums and wheels contacting the surface.

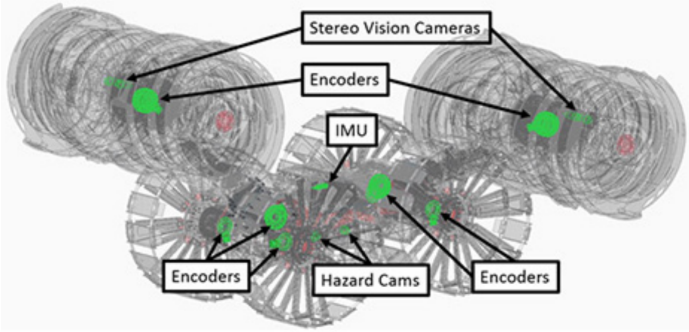

Fig. 1 Rendering of RASSOR 2.0 highlighting sensors and their placement.

RASSOR possesses a limited set of on-board sensors. Each actuator contains an encoder and torque sensor. The robot also contains stereo vision cameras mounted to the end of each of the arms, hazard cameras, and an inertial measurement unit (IMU). As part of this work, we explored the inclusion of additional sensors such as time-of-flight

measurements in simulation. This is described further in Section IV.D.

2. Auto Dig Routine

During trenching operation, RASSOR 2.0 employs a semi-autonomous digging routine called *auto dig*. Auto dig is a proportional-integral-derivative (PID)-based controller conceived to perform linear trench excavation. The controller tracks a drum motor current setpoint which is selected (and updated) by the robot operator throughout the execution of the routine. As the drum motor current draw increases with ingestion of regolith, the robot operator can increase (or decrease) the setpoint to maintain consistent engagement with the soil. The robot's wheels and drums are set to a constant velocity. In practice, the controller's D term is not used.

The process variable is the motor drum current (in amperes). We measure the error between the setpoint and the drum current as an average for the combined front (front left and front right) and combined rear (rear left and rear right) sets of drums, independently. The controller effectors are the arms of the robot. We employ separate PID loops for each arm and each error term is encompassed by a deadband.

The starting setpoint will force RASSOR to lower the arms until it engages with the soil. The engagement with the surface corresponds to increased drum current draw. If the setpoint current target is set too high, the arms may push too hard into the surface and attempt to dig too deep. Consequently, as the soil is ingested and drum current exceeds the setpoint, the arms will be forced up until the setpoint is increased or regolith expelled.

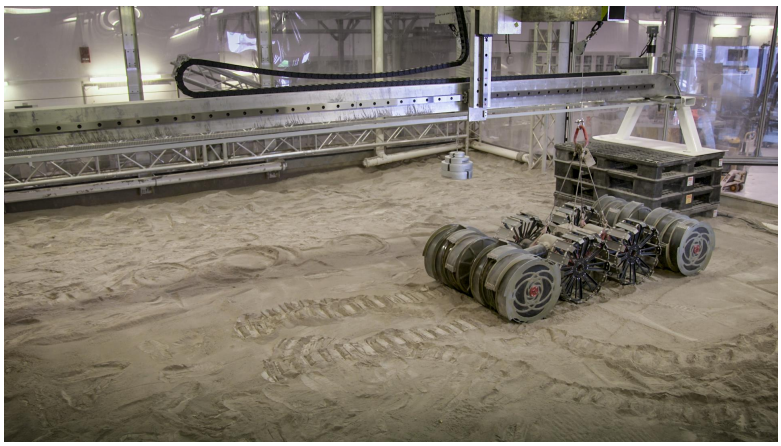

Fig. 2 RASSOR 2.0 performing dig routine while attached to the gravity off-loader in the regolith bin at Kennedy Space Center.

We devised an analogous version of auto dig in our 3D simulation to compare the results of our learner. These results are presented in section VI.B. In our simulated auto dig, we used a PD controller to achieve excavation behavior similar to operation in the physical environment.

B. Reinforcement Learning

In recent years, advancements in computing have enabled the widespread application and success of machine learning. Machine learning is a data-driven artificial intelligence methodology and is generally divided into three categories: supervised, unsupervised (or self-supervised), and reinforcement learning [15]. Supervised machine learning algorithms rely on defined (*labeled*) datasets consisting of known input and output relationships. This enables supervised algorithms to learn to approximate functions using some quantity of the data during a training procedure. Unsupervised learning algorithms learn to recognize patterns within unlabeled data. Finally, reinforcement learning algorithms learn decision making processes through trial-and-error interactions in an environment.

Fundamentally, reinforcement learning problems are modeled by Markov Decision Processes (MDPs). The algorithm consists of a learner, called the *agent*, which learns a task by performing actions in an environment and sensing rewards. A learned strategy is called a *policy*, π, that maps states to actions. The goal of reinforcement learning is to learn the optimal policy, π^*. When an action, a, is performed, a change to the state in the environment is measured, s, and a quantitative reward, r, is returned to inform the agent how good or bad the action was. The agent selects actions to

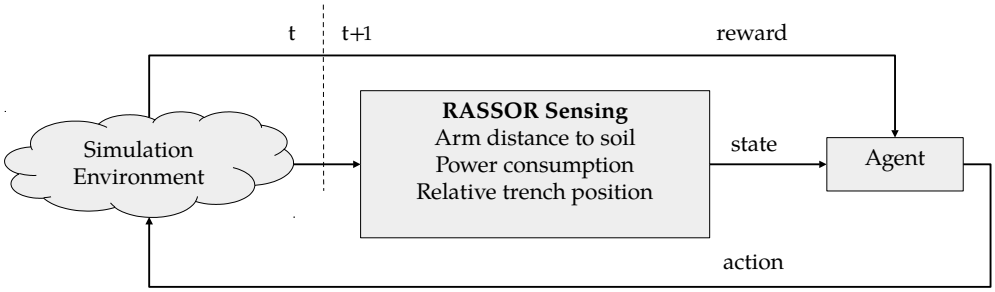

Fig. 3 Reinforcement learning process as applied to RASSOR.

maximize its reward over the period of an episode. In our work, the agent is the robot and the actions are movements of the robot's actuators. State is measured via sensors that track the environment. The reward function is specified by the machine learning engineer to encourage the agent to learn a desirable policy. It does this by learning the value (or utility) of being in a state s using a value function and selecting actions that maximize its expected reward for an episode given its state.

In this work, we employed two reinforcement learning algorithms. For our 2D simulation, we used the Q-Learning algorithm [16]. The results of this algorithm are described further in Section III. For our 3D simulation, we trained with the Deep Q-Learning algorithm [17]. We present our findings using this algorithm further in Section VI.A.

1. Q-Learning

The goal of Q-learning is to learn policies through estimation of the value of performing actions in a particular state, also known as the Q-value. Q-values represent the expected total reward from a particular state forward to the end of an episode. Q-values are calculated with the following equation:

$$Q^{\pi}(s,a) = r(s,a) + \gamma \max_{a'} Q^{\pi}(s',a') \tag{1}$$

To calculate the value of being in a state s after taking action a, we take the reward returned via $r(s,a)$ and then select the action that returns the maximum Q-value of the following state, s'. In Equation 1, γ is the discount factor and is selected to be between 0.0 and 1.0. Discount factors are selected to encourage an agent to behave myopically or maximize for long-term rewards.

The value of a state-action combination is calculated through recursive calls within Equation 1 until the maximum value is returned after reaching a terminal state. Building upon this, the Bellman optimality equation [15] is used to update Q-values throughout training with the following rule:

$$Q^{\pi}_{t+1}(s,a) = Q^{\pi}_{t}(s,a) + \alpha[r(s,a) + \gamma \max_{a'} Q^{\pi}_{t}(s',a') - Q^{\pi}_{t}(s,a)] \tag{2}$$

From Equation 2, the agent iteratively reaches new states by selecting actions, after each time step, t. In each step, a new reward is sensed and is used to update the estimate of the Q-value of the policy. This is referred to as bootstrapping. Q-values are stored in a Q-Table which is of size $S \times A$, where S is the set of all states, and A the set of all actions. At the conclusion of each step, the state is updated and thus s' becomes s and the agent continues to loop (selecting actions based on the policy, observing the reward, new state, and consequently updating the state-action values) until reaching the terminal state. α is the learning rate, selected between 0.0 and 1.0. A high learning rate weights new rewards more heavily and is thus more sensitive to new information.

An important consideration of reinforcement learning algorithms is the trade-off between exploring and exploiting the agent's knowledge within an environment. In order to learn, the agent must explore new states and update the Q-Table. However, the agent must also act rationally upon its existing policy to remain efficient as a learner. To balance exploration and exploitation, ϵ-greedy action selection is typically used. Within ϵ-greedy, the action with the highest Q-value is selected most of the time. Though we also randomly select a random action based on a probability threshold denoted by ϵ. ϵ-greedy action selection is generally used with a decay so that the probability of random actions being selected decreases gradually.

The use of a max function in Equation 1 means that the agent must iterate through all possible actions to compute a Q-value for a particular state-action. In learning problems with large or continuous action spaces, Q-Learning is no

longer a tractable approach. When applicable, quantization of the action space can address this problem. However, another challenge arises in large state spaces, where quantization may not be appropriate due to the complexity of the environment. In large state spaces, the learner may not (or will not) reach all possible states. Consequently, the policy may fail during execution in the environment due to the lack of experience in regions of the state space. In these cases, a method of mapping the state space to a discrete abstraction of state is necessary.

2. Deep Q-Learning

Deep Q-Learning [17] addresses Q-Learning's weakness in larger state spaces, such as in continuous domains. Instead of learning the value function through direct updates of Equation 2, Deep Q-Learning approximates the value function through supervised learning, typically using neural networks. The neural networks consist of weights, θ, which are learned by optimizing for a loss function. Within this framework, a value function is initialized, $Q^{\pi}(s, a; \theta)$, with random weights for θ. Through training the neural network, the goal is to approximate the optimal value function:

$$Q^{\pi}(s, a; \theta) \approx Q^{\pi^*}(s, a) \tag{3}$$

Though Deep Q-Learning extends the state representation, using neural networks to approximate the value function comes at the cost of learning stability. To mitigate this, a technique called *experience replay* is often used [18]. Experience replay updates the neural network in batches. Updating the value function in batches improves the stability of the agent as it learns using uncorrelated data from previous experience stored in a replay buffer instead of using the latest state-to-state transition data. Using experience replay can also make the training process more efficient as the updates reuse previously seen data to increase retention.

$$\theta_{t+1} = \theta_t - \alpha \nabla_{\theta} [(r(s, a) + \gamma \max_{a'} Q^{\pi}(s', a'; \bar{\theta}) - Q^{\pi}(s, a; \theta_t))^2] \tag{4}$$

Equation 4 shows gradient descent as applied to the neural network parameters. To further increase stability, two neural networks are used called the *target* and *local* networks. Both networks share parameters, θ, though the target network is updated less frequently than the local network so that θ does not converge erroneously. From Equation 4 we see the target network with older parameters $\bar{\theta}$. To update the parameters, we use soft polyak updating:

$$\bar{\theta}_{t+1} = \tau \theta_t - (1 - \tau) \bar{\theta}_t \tag{5}$$

From Equation 5, τ is a weighting term selected to update the neural network parameters. This is done analogously to the learning rate α. Reinforcement learning algorithms provide rich structure for constructing and incorporating new learning methods and parameters. Deep reinforcement learning extends this framework to include deep neural networks. Deep Q-Learning is just one of many deep reinforcement learning algorithms [18]. We include the parameters used in our work in Table 2.

C. Lunar Environment Simulations

Designing a comprehensive robotic simulation is difficult as fidelity comes at the cost of performance, and both must be carefully balanced. Several robotic simulation tools are under active development [19, 20], though to the best of our knowledge, none support terrain deformation which is a necessary feature for simulating excavation and low-level trenching operation. Alternatively, discrete element method (DEM) simulation tools have been applied to simulating excavation [21] and modeling ISRU excavation needs [22]. However, DEM simulations are computationally demanding and are not currently tractable for real-time or super-real-time robotic simulations.

Prior work within Swamp Works has made use of Gazebo [23], an open source 3D robotics simulation. Gazebo is frequently used for robotics simulations as it integrates well with the Robot Operating System (ROS). ROS-controlled robots and sensors are found in several NASA projects including Robonaut [24], Astrobee [25], and RASSOR, to name a few. Gazebo has been used to simulate the lunar environment, including a simulation that was extended to support mission software, increased visual fidelity of the terrain, a terrain generation system for leaving "tracks" in the soil [19], and an open-source ISRU autonomy simulation based on RASSOR [20]. However, Gazebo does not natively support terrain deformation and thus cannot be readily used to simulate trenching routines in high fidelity.

Other work on ISRU-based simulations for excavation include [26], which used the Digital Spaces simulation and focused on applying evolutionary algorithms to multi-robot ISRU systems. Digital Spaces was a 3D simulation tool used to simulate a deformable lunar surface. In the following section we describe our preliminary development of a terrain deformation system and the incorporation of learning algorithms.

III. Prototyping Learning and Sensor Capabilities in a Reduced-order Simulation

A. Simulation Modeling

As a precursor to our 3D excavation simulation, we developed a reduced-order 2D simulation as a proof-of-concept for learning and sensor selection. Our 2D simulation was designed to be compatible with OpenAI Gym, a reinforcement learning toolkit that facilitates the integration of learning algorithms with simulation environments [27]. We prioritized simplicity and performance during the development of this simulation to support the rapid deployment and training of different learning algorithms and terrain parameters.

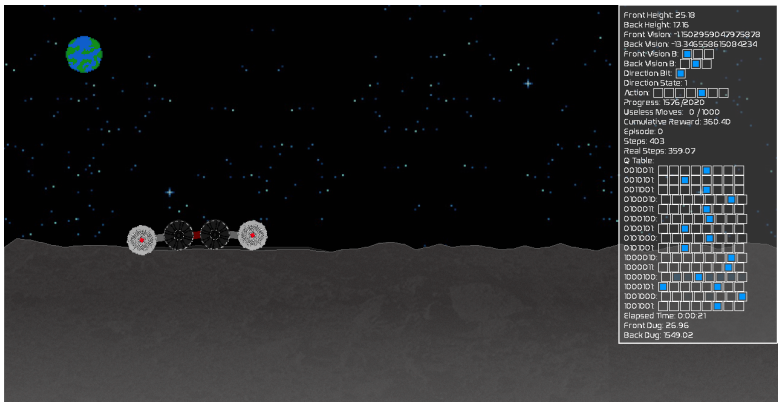

Fig. 4 ICE-RASSOR 2D simulation with Heads-Up Display depicting learning status and vehicle information.

In the simulation, we implemented a line-based mineable surface. The terrain is parameterized so that its topography is varied using perlin noise terms. Additionally, the simulation terrain supports configuration of the dig depth of the scoops and a wall sloping effect during deep trenching operation. We did not model gravity in this simulation. Instead, the surface is represented by a spline that is perturbed when intersected by RASSOR's drum. The robot wheels permanently intersect with the ground so the robot is not capable of "acrobatic" or contingency maneuvers. However, motion is visualized when the robot is moving (wheels and drums rotate). This environment enabled us to deterministically test several learning algorithms. Over the course of development, several optimizations were made to the simulation to improve speed. On an Intel Core i9-10900X workstation, the system can process 105 simulation steps/second with headless mode enabled in single-threaded mode, and over 2000 steps/second in multi-threaded mode.

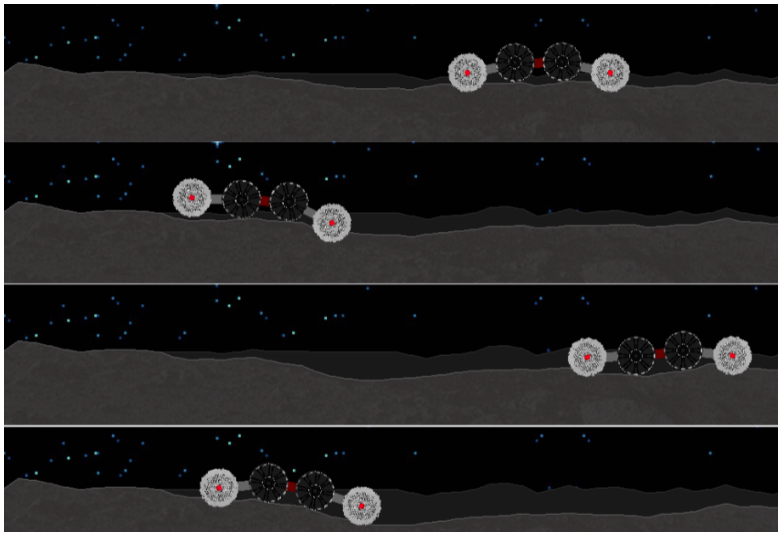

Fig. 5 ICE-RASSOR 2D simulation through execution of the Q-Learning policy.

During the development of the 2D simulation, we implemented and augmented simulated sensors as we applied the

learning algorithms. With a sensor framework limited to knowledge of the kinematics of the robot, the learner was unaware of the surrounding surface topography. To produce cohesively excavated trenches, we implemented pseudo time-of-flight sensors to report distance from each drum to the ground and the overall height of the robot above the surface plane. We assume the availability of reliable odometry methods since accurate feedback of movement across the soil is necessary to the robot's ability to achieve desired states and final trench configurations.

B. Learning in the 2D Simulation

We constructed the reward function to positively reward RASSOR for the amount of regolith excavated per step and penalize it for the following conditions: drums not engaging soil, the robot attempts to break kinematic limits (e.g. knock drums of opposing arms together), or reaching a position where the robot becomes stuck. Finally, for the purpose of limiting the dig site, RASSOR is penalized for going near the edges of the simulation window.

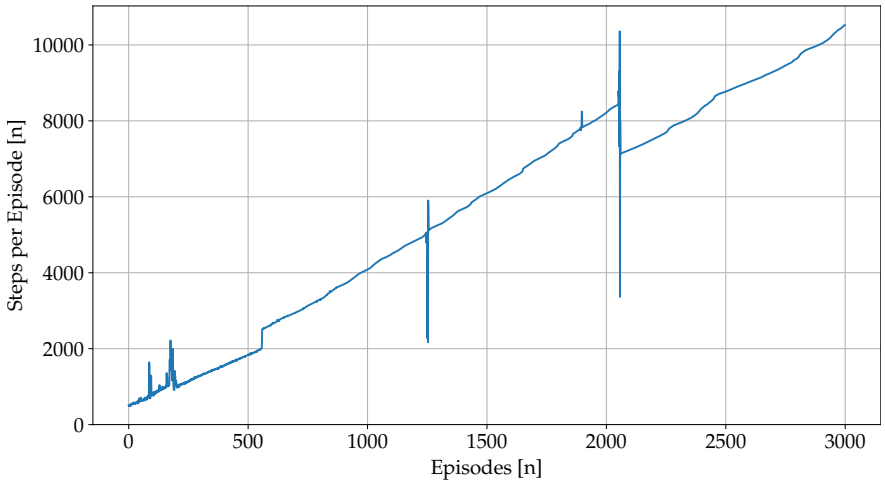

Fig. 6 RASSOR was rewarded for continuing to excavate the entire trench. It maintained a steady increase in excavation steps throughout training.

The observation function returns a seven bit binary state to represent the time-of-flight sensors for both front and rear of the robot, and a directional bit to determine the movement direction based on the bounds of the dig site.

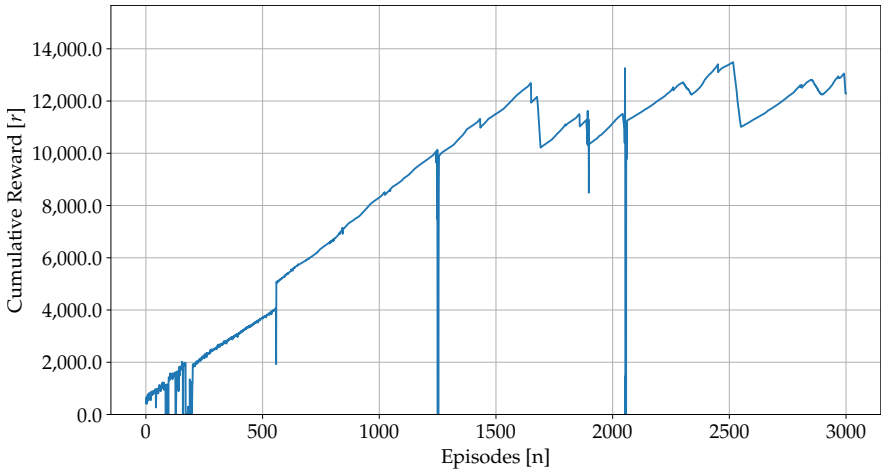

Fig. 7 The cumulative episodic reward continued to increase as RASSOR learned to dig deeper and deeper.

RASSOR has six possible actions in the environment: drive forward, drive backward, raise front arm, raise back arm, lower front arm, and lower back arm. Each of these commands are done in position-space as increments per time step

when an action is performed. The simulation can also be configured to set bucket drum angular velocities, though the drums were ultimately left at a constant velocity to simplify the learning process (and is similarly done with auto dig).

In Figure 6, we see the steps per episode increase throughout training of the Q-Learning algorithm. Similarly in Figure 7, we show the reward per episode throughout training. These figures indicate the success of learning as a function of the state representation, reward function, and learning parameters selected for the environment. The developed solution enabled RASSOR to trench the environment in levels and effectively track state. We explored the use of existing on-board voltage/current sensors to approximate sensor information where vision or other advanced sensors would have otherwise been appropriate. The pseudo sensor capabilities, actions, and reward schemes developed for the 2D environment and learner inspired the development of our 3D simulation described in the following section.

IV. Simulating Lunar Excavation in 3D

A. Environmental Modeling

The environment for the 3D simulation is an arbitrarily generated terrain surface designed to mimic the lunar surface. For simplicity the dig site was kept flat. Two versions of the terrain excavation system were developed.

In the first version, we devised a mesh system consisting of a set of vertices, edges, and triangles to describe the surface of desired resolution. When the teeth on the bucket drums of RASSOR collide with the terrain, the parameters of the collision are returned. The collision information is then used to calculate an offset for the nearest vertices to the collision point and reduce their height value by the scoop height in the world coordinate space to imitate physical scoop engagement with the soil (further described in Section IV.D).

This approach was subject to several performance constraints. One of the most computationally demanding calculations are collision checks. The computation time needed to calculate the updated collision boundaries for the terrain exceeded our requirements for simulating in super-real-time. For a desired work area of 100m x 100m with a resolution of 50mm, 4,000,000 vertices would be needed to achieve excavation resolution. At a rotation rate of five radians per second, collisions with RASSOR's bucket drums would generate several collision updates per second rendering this approach intractable. To resolve this, the terrain was further divided into smaller tiles. By creating 1m x 1m tiles, the number of vertices per tile could be reduced to 400. However, the tile-based system presented other issues when manipulating the corner edges of adjacent tiles. These issues, in part, led us to the second version of the terrain system.

The second version of the terrain system uses the built-in Unity terrain class which enables us to use a third party asset called Digger PRO. Digger PRO takes advantage of Unity's built-in burst compiler and job system. The burst compiler converts .NET code to optimized native code using a low-level virtual machine, and the job system allows safe multi-threading capabilities within the Unity framework. We applied our existing collision method for detecting scoop events. Using these collision events between the bucket drum teeth and the terrain, Digger PRO was configured to modify the terrain.

B. Robot Modeling

The visual and reference geometry for RASSOR in the 3D simulation comes from the EZ-RASSOR simulation [20] developed by the Florida Space Institute. EZ-RASSOR is an open source mining robot designed to mimic the characteristics of the RASSOR 2.0 platform with smaller geometries. We tuned the parameters provided with EZ-RASSOR to match the needs of our simulation environment.

Within Unity, RASSOR was modeled as a set of components connected via actuation joints. These include the chassis, wheels, arms, and bucket drums. The NVIDIA PhysX engine was used to represent these components in the simulation. PhysX provides rigid-body and joint classes which we used to model RASSOR's dynamics. Each fundamental component was modeled in Unity as a distinct rigid-body. These components were connected by a joint class, with constraints defined for each joint. The wheels and arms are locked in position relative to the chassis, but are allowed to freely rotate about their primary axis. Likewise, the bucket drums are locked in position relative to their respective arms. For the arms, minimum and maximum angles can be applied to each joint to act as pseudo mechanical limitations. PhysX also offers simulated motors for each joint using a proportional-derivative drive model. The motors can be configured to apply a specific force and target velocity for each joint which allows control of the robot.

C. Actuator Modeling

Actuator models were defined within Unity's PhysX joint system. Power consumption models were then defined based on empirical data and robot kinematics. An energy storage model was also developed to track amp-hours (Ah) used and remaining. RASSOR's battery pack was set to a constant voltage.

Total power consumption is calculated by summing the power consumed by individual wheel, arm, and drum actuators, plus a buffer for the power consumption of other electrical components not associated with robot motion (hotel power).

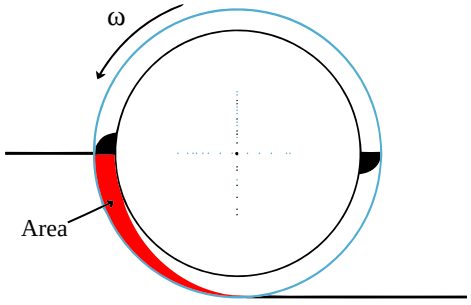

Fig. 8 Cross section of bucket drum interacting with regolith.

Bucket drum current draw, i_{drum}, is modeled as the superposition of the current draw when the drum is freely spinning, i_{f}, and the current draw while digging, i_{d}.

$$i_{drum} = i_f + i_d \tag{6}$$

Our simulation models actuators that run in constant speed mode, where control is performed by setting desired angular velocities. A closed-loop controller within each actuator's electronic speed controller (ESC) is responsible for metering power to the actuator to achieve and maintain the set speed.

When the ESCs are in speed control mode, an appropriate first-order approximation of the drum actuator's free-spinning current draw while rotating is the sum of a constant-speed friction term, and a term proportional to the drum's total mass (which includes the mass of the regolith in the drum).

$$i_f = B\omega + C(m_d + m_r) \tag{7}$$

In Equation 7, B and C represent proportional terms adjusted to approximate empirical data, m_d is the mass of the empty drum, m_r, the mass of regolith within the drum, and ω is the arm angular rate. A more accurate freely rotating drum power consumption model would consider higher-order terms such as regolith tumbling dynamics, variable and non-linear friction terms, ESC dynamics, etc. However, appropriate definitions of coefficients B and C were tuned to sufficiently reproduce free spinning drum power consumption in the quasi-steady state.

The additional current draw due to excavation is modeled by Equation 8, where α and γ are empirically determined and A_{area} is the cross-sectional area of regolith engaged by the bucket drum scoop as shown in Figure 8. Equation 8 assumes the scoops are the only drum structure contacting the regolith. Additional terms would be required to represent current consumption if excessive speed of the chassis or arm motors resulted in outer drum surfaces contacting regolith. Just as in Equation 7, α and γ were tuned to sufficiently approximate excavation current draw for the purposes of the reinforcement learning environment.

$$i_d = (\gamma\omega A_{\text{area}})^{-\alpha\omega} \tag{8}$$

When the arm actuator ESCs are in speed control mode, Equation 9 approximates their free-spinning current draw, with appropriate definitions of coefficients D and E. In this model, I is the moment of inertia about the actuator rotational axis, and θ represents the arm's pitch angle relative to the horizontal plane (orthogonal to the gravity vector). Forces reacted by arm actuators due to drum-soil interaction were neglected since the motor brake is engaged when the arm's ESC is not actively commanding a rotation. As with our other reduced order models, this model neglects higher order effects encountered on the physical hardware and is the topic of future work.

$$i = D\omega + EI\cos(\theta) \tag{9}$$

The simplified wheel actuator power consumption model expressed in Equation 10 ignores wheel-soil interaction effects. F is defined such that its product with v, the robot's linear velocity, matches the physical robot's power consumption under similar operational conditions.

$$P = Fv \tag{10}$$

An important aspect of the power modeling simplifications presented above are the context wherein they are used. None of the above equations perfectly reproduce RASSOR's power consumption, yet it is not essential they do so. The critical feature of the above equations is that they reflect the relative magnitude of power consumption caused by various robot actions. When so proportioned, robot performance gains realized by learned excavation policies trained in the simplified environment suggest that these gains are extensible to both higher fidelity simulation environments and the physical robot.

D. Soil-Drum Interaction

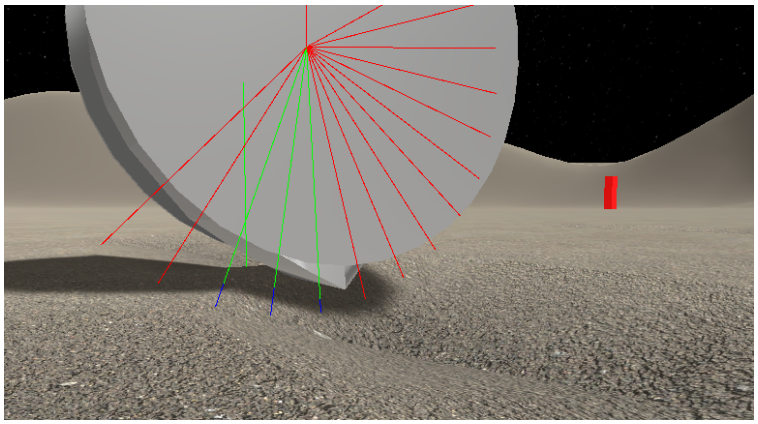

Fig. 9 Soil-drum interaction through scoops of bucket drum.

The soil interaction model was simplified to improve collision performance. We constrained kinematic rates of the wheels and arms to prevent stalling or undesirable dynamical events. Doing so enabled us to simplify the dynamics of the soil interaction between the drums and the terrain and avoid edge cases. For example, if the angle of the drum with respect to the ground is known, then the angular rate the motors can send to the arms can be limited to avoid collisions that would cause the drums to get stuck.

To determine the amount of overlap between the tooth collider on the bucket drum teeth and the soil, we use ray-tracing, as depicted in Figure 9. The collider checks a set of rays extending outward from the center of the bucket drum to the farthest tooth edge. In Figure 9, red rays represent positions along the drum which do not interact with soil, green represent valid collisions with the terrain, and blue represents the amount of overlap between the tooth and the surface. The area of overlap is extended along the length of the drum tooth to compute the volume of regolith ingested; we then use Digger PRO to modify the terrain and depress it to account for the excavated regolith.

E. Robot Sensing and Control

Our simulated version of RASSOR possesses several on-board sensors. Akin to the physical robot, we measure power consumed at the main bus (which accounts for all actuators and simulated hotel power due to compute and wireless communication power requirements). The simulated robot also provides instantaneous current readings for each motor, motor encoder positions for odometry, and the distance measures based on the time-of-flight sensors described previously, see Figure 10. Finally, we directly measure the quantity of regolith excavated based on the volume subtracted from the environment. In environments where the excavated regolith cannot be directly measured in real-time, RASSOR could estimate this information using motor current data [28, 29].

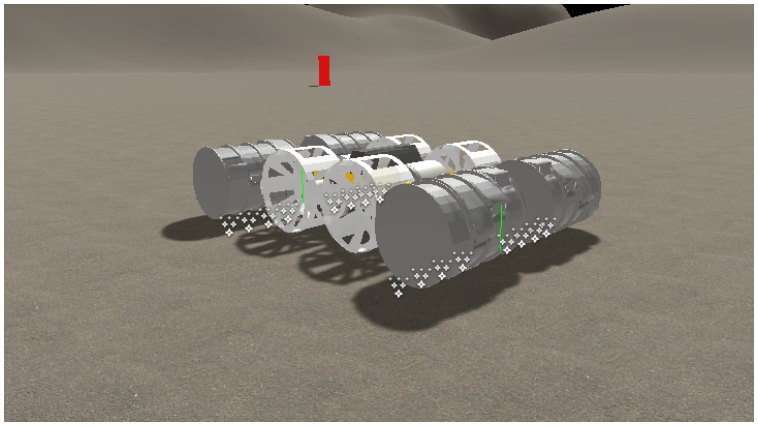

Fig. 10 Pseudo time-of-flight sensors as depicted by the green lines. The green lines measure height of drums above surface.

RASSOR is equipped with velocity controllers which have been extended to support three modes of operation. The first mode is an application programming interface (API) devised to support Unity ML-Agents. Unity ML-Agents facilitates the integration with learning algorithms, though it also supports custom communication channels which are used to transmit state data for control outside the purview of the learner. The second mode enables teleoperation using a physical gamepad connected to the simulation. The last is an analogous implementation of the auto dig PID controller, as described in Section II.A.2.

The implementation of these three modes of operations enables us to observe unique behavior and control under different simulation configurations. The learning mode facilitated development as it would reach states that induced run-time errors. Auto dig was used to tune both the controller gains as well as environmental and actuator parameters.

For an experienced operator, teleoperated control is a direct method of achieving certain behaviors. However, teleoperated control of a dual arm bucket drum excavator such as RASSOR is challenging. To maximize the efficiency of the platform (especially for maintaining traction in low-g environments), movements between both arms must be coordinated in response to the elevation of the soil and the movement of the robot. As the robot moves forward while digging a trench, the arm in the direction of movement must be cautiously servoed to maintain soil engagement yet avoid stall conditions encountered when the drums are pushed forcefully into the soil. We implemented teleoperated control for a few reasons. Teleoperated control enabled us to directly test the feasibility of contingency maneuvers when interacting with the simulation environment, but also served as a baseline to which our automated methods are compared against.

V. Learning Excavation in the 3D Simulation Environment

A key function of the simulated robot is its ability to dig efficiently. The expectation holds whether the strategic goal is to shape the terrain, gather material for processing, or a combination thereof. The general goal of digging on its own, however, is insufficient for creating an RL model that converges on an optimum policy. There must be specific rules and goals governing RASSOR's behavior such that the RL algorithm can find an optimal solution. We focus on the fundamental digging task, and leave strategic goals for future work. For this problem, there are three governing tenets by which the RL-based solution must abide. The RASSOR robot must:

1) learn to gather as much material as possible from the surrounding environment. The amount of material RASSOR can gather is directly proportional to the capacity of its digging drums and available battery charge.
2) maximize its availability to carry out its tasks. Thus, it would be unacceptable for RASSOR to become stuck as a result of its digging operations.
3) dig and gather material from a specific, prescribed area; perhaps an area deemed ideal for digging operations as a result of precursor surveying or prospecting.

These governing tenets in part originate from requirements on the physical platform [7]. Secondarily, they establish constraints that are used as a basis for the metrics in our comparison.

A. Environment Requirements

Normally, the region where RASSOR is allowed to dig may be defined as broad enough to require navigation using two degrees of freedom. In the experiments of this work, the dig site is confined such that RASSOR is only required to move forwards and backwards to produce linear trenches, without the need to yaw using skid steering (similar to auto dig).

The RL agent requires accurate and reliable position information, both while learning a digging policy during training and when utilizing said policy during inferencing. The prevailing assumption in this experiment is that the position is provided by an external service, such as other surface assets and fusion of several sensor modalities. The agent is afforded both its up-to-date position as well as the position and size of the designated digging region using a common reference frame relative to the starting dig position of the robot. We treat this excavation reference frame as global for the purposes of the learner.

B. Intrinsic Requirements

During the training phase, the agent expects RASSOR to be able to measure its pose (Euler angles), with respect to the global reference frame. In addition, during training and inferencing, the agent expects RASSOR to measure the height of the digging drums with respect to the surface. Finally, both during training and inferencing, the agent expects RASSOR to properly estimate the amount of material captured by the digging drums at any point in time.

The capabilities listed up to this point suffice to meet two of the three governing tenets for the RL solution: track the amount of gathered material and remain within a prescribed region. In order to prevent the situation where the robot becomes stuck, RASSOR keeps track of how many times it has removed surface material at a specific spot. During training, the RL agent is penalized from extracting too much material from a particular spot; this encourages smooth contouring of the surface.

C. Action Space

The action space available to the agent consists of seven distinct actions. At each step, the agent is allowed to perform only one of these actions. The list of agent actions is as follows:

1) Remain idle (no action)
2) Drive forward
3) Drive backward
4) Raise the front digger arm
5) Lower the front digger arm
6) Raise the back digger arm
7) Lower the back digger arm

D. Observation Space

In order to both accommodate the need for tracking the amount of times RASSOR has extracted material from a particular region and make the resulting RL-derived policy more generally applicable, RASSOR's position was reduced to a single value along the axis of motion, normalized with respect to the designated digging region, and digitized to a discrete set of possible values.

For this experiment, the eventual digitization of RASSOR's position in the environment was done by splitting the digging region into 20 equally-sized segments. Given the movement limitation in this experiment, only the position along a single axis was digitized. RASSOR's world coordinate frame position $p_w \in \mathbb{R}$ is transformed into a digital position $p_d \in \{0, 1, 2, \ldots, 19\}$ is shown in Equation 11. The equation shows two intermediate values: $p_n \in [-0.5, 0.5)$ which is the 0.0-centered normalized coordinate value, and $p_b \in [0.0, 1.0)$ which is p_n biased by 0.5 so the position center is at 0.5. The equation also involves two constants: 1) D_x, the digging region's center coordinate value along the travel axis, x, in the world frame, as well as the digging region's size, D_l, along the same x-axis.

$$\begin{aligned} p_n &= \frac{(p_w - D_x)}{D_l} \\ p_b &= p_n + 0.5 \\ p_d &= \lfloor p_b \times 20.0 \rfloor \end{aligned} \tag{11}$$

Similar to the auto dig routine, the learner focuses exclusively on digging. We leave planning to other tasks such as choosing when and where to empty the digging drums to future work. The ultimate goal of the policy learned by the agent is to fill the digging drums to a predetermined level as calculated by the aforementioned dig volume perception logic.

Given the environment characteristics and governing tenets, the following observations lead to an optimal RL digging policy:

- **Normalized and digitized position in digging region** - Keeps RASSOR within the established bounds.
- **Front/rear drum height** - Helps the agent find the optimal height at which to gather material.
- **Dig count at current digitized position** - Helps prevent digging too deep and rendering RASSOR stuck.

The environment observations were encoded into a four-dimensional vector that included the digitized position, the digging drum heights (front and rear), and the corresponding position dig count $c_i \in \{0, 1, 2, \ldots, 100\}$. The agent was responsible for keeping track of the dig count for each position using a 0-initialized value array, C, large enough to hold a dig value for each position (in this case, an array with 20 members). Each time RASSOR detected that it had captured material at a position i, it would increment c_i by 1.

E. Rewards

In order to properly influence the learned policy such that it comports to the governing tenets, the environment disburses rewards according to the following schedule:

- Null reward by default.
- Positive, albeit diminishing, reward for acquiring surface material.
  - The decay in reward for a particular spot in the digging area is directly proportional to the number of times RASSOR acquired material at that area.
- Negative reward for leaving the designated digging region (i.e., an out of bounds condition).
- Negative reward for morphing terrain to the point where traversal induces excessive chassis angle, increasing the chances that RASSOR becomes stuck.
- Positive reward for filling the digging buckets to a predetermined level.

F. Training

The training phase of the experiment runs until the average cumulative reward and value function loss visibly settle into a restricted range. During this phase, episodes end if any of the following conditions are met, RASSOR:

- fills both sets of front and rear drums.
- ventures outside the digging region (i.e., goes out of bounds).
- detects chassis angle deviation from level in excess of 20° with respect to the gravity vector.

G. Reward Values

The reward values disbursed by the environment were kept to a maximum magnitude of 1.0. Based on the reward conditions listed in sub-section V.E, the reward values were set as shown in Table 1.

Table 1 RASSOR policy training reward values

Reward	Value
Null reward	0.0
Dig reward	$1.0 - (0.0001 \times c_i)$
Out of bounds	−1.0
Excessive angle	−1.0
Full digging drums	1.0

VI. Results

A. Deep Q-Learning Training

In all, 189,000 steps were performed during the training of the Deep Q-Learning agent across 963 episodes. Post-training, we compare our Deep Q-Learned policy to the auto dig and teleoperation results. Once training concluded, the learned policy was installed into the 3D simulation environment in order to validate its inference capability. Over the course of 20 episodes, the agent executed the policy to completely fill its buckets in every case. Observing the actions taken by the trained policy, it was evident that the training led it to choose to move constantly while only skimming the surface of the terrain. As described in Section II.B.2, we tuned the learning algorithm parameters based on the needs of the digging task. The final hyperparameters used are shown in Table 2.

Table 2 Deep Q-Learning Hyperparameter Configuration

Parameter	Value
Learning Rate (α)	2.7×10^{-4}
Exploration Factor (ϵ)	0.1
Reward Discount (γ)	0.99
Polyak update (τ)	0.01
Buffer Size	50,000
Policy Network	2 layers of 64 nodes

The overall training behavior can be roughly divided into three stages. In the first stage, the learner explored the space to identify rudimentary relationships between actions and reward signals. In this stage, the learner often failed by getting stuck, exceeding slope angle limits, and other undesirable behavior such as waving the arms. Eventually, RASSOR would reach the full drum terminating state and in doing so reached the second observed phase of learning.

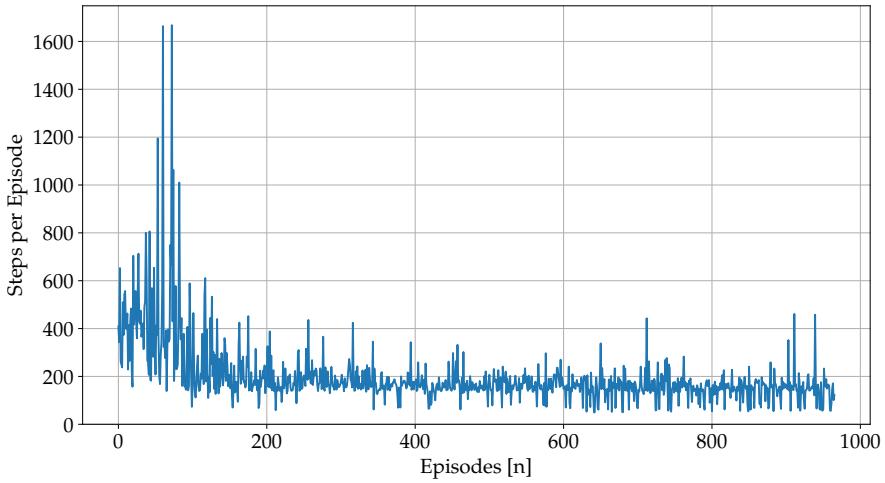

Fig. 11 Through learning, RASSOR was able to reach the terminal excavated volume target in fewer steps.

During the second phase of learning, RASSOR exploited its digging behavior to reach the terminal fill state. However, it would often get stuck in doing so. RASSOR would bury both sets of drums in the surface and excavate while minimizing movement of its drivetrain. This resulted in two trenches, one on each side of the robot drivetrain. Doing so also consumed more power since the robot would servo the arms more in the deeper trenches with the added mass of the regolith in the drums. Further optimizing upon these strategies gave rise to the third observed phase. In the third phase, the learner optimized its strategy to increase excavation efficiency. RASSOR leveraged the low gravity environment by exploiting the drums as a mobility platform. In sloped trenches, RASSOR would find opportunities to lower its drum beneath the center of mass of the robot. Generally, this was done for a short period of time, and eventually, the learner moved towards more conventional bucket drum excavation behavior with consistent and controlled application of force to the surface (similar to RASSOR 2.0 behavior in Kennedy Space Center's regolith test bin).

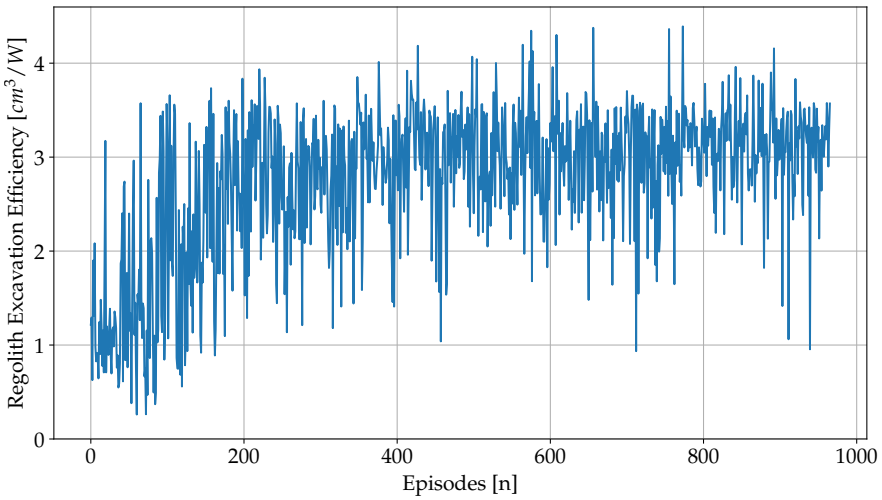

Fig. 12 As the learner trains, the efficiency indirectly increases as a consequence of the reward function.

Maximal excavation efficiency is our key comparison metric. In Figure 12, we plot the volume of regolith excavated per unit of power over the course of training. We see a significant improvement in the terminating episode in efficiency. In a trial of 20 executions of the policy, we saw an average efficiency across the execution of the task of $3.80 \pm 1.47\ cm^3/W$. In the simulation, we logged several performance markers including instantaneous battery measurements, instantaneous motor currents, instantaneous and cumulative regolith excavated (by volume), and other measurements as reported by our pseudo sensors. Using this information, we compute efficiency by dividing cumulative regolith excavated by power.

B. Performance Comparison

In this section, we present our learning results within the 3D lunar simulation and contrast it with an analogous implementation of RASSOR's auto dig routine and human teleoperation (without delay) via gamepad control. All experiments conducted within the simulation used the same environmental conditions and robot configuration. As part of the experiment, we performed 20 trial runs with the Deep Q-Learned policy, 20 with the auto dig routine, and 10 with the teleoperation model operated by a trained robotics engineer. As auto dig is a controller with a static mode of operation, both the learner and teleoperated control was focused on completing an identical task: fill both sets of drums during linear trenching as efficiently as possible (measured by volume of regolith ingested per unit power).

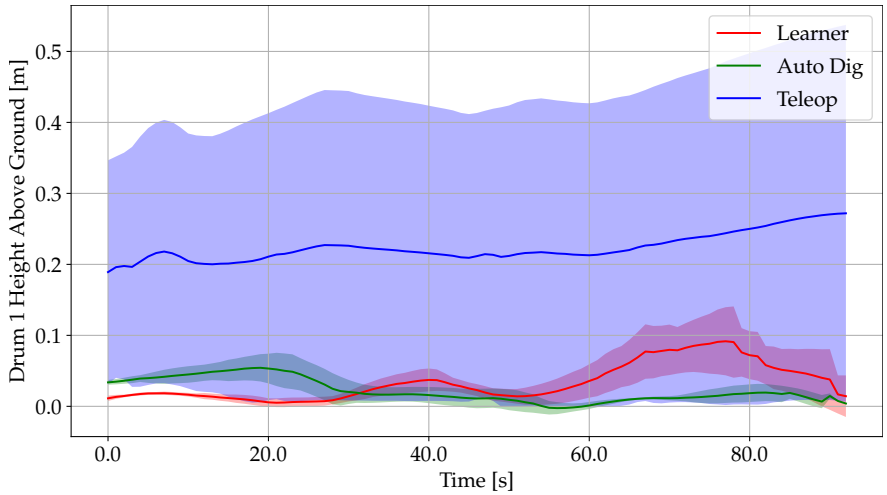

Fig. 13 The learner and auto dig perform similarly whereas teleop has high variance.

Over the course of this work, we identified a pseudo time-of-flight capability that was used to inform the state of the learners. This capability has also helped visualize the efficiency of the different methods by quantifying engagement with the soil over time. As such, our first comparison figures, Figure 13 and 14 highlight a stark difference between the teleoperated user's ability to control the platform consistently versus that of the learned and PID-based controllers. In each of the figures in this section, we show the results of the learner in red, the auto dig routine in green, and the teleoperated results in blue. The dark center line for each method corresponds to the mean of all the trials for a particular method. The lighter infill area represents one standard deviation from the data.

As established in Section V, all learning within the 3D simulation was completed in an end-to-end manner. The learner was responsible for the guidance and control of the robot based on the policy achieved. Learning could be implemented as part of the auto dig controller instead, though this remains the focus of future work.

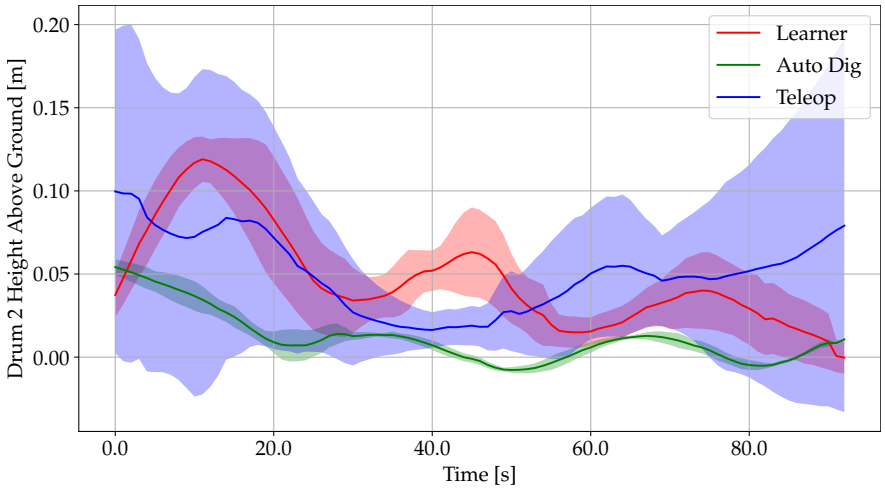

Fig. 14 The auto dig routine maintains the lowest average drum above height for the second set of drums.

Figures 13 and 14 depict the height in meters of the low edge of the drums above the surface of the dig site. Although the original measurement is taken from the center of the arm as shown in Figure 10, we subtract the radius of the drum from all measurements. Our primary comparison is between the learner and auto dig with the teleoperated results provided as contrast to the algorithmic-based approaches. Since the performance of the teleoperated results depends on human skill it is included as a loose benchmark for reference, as indicated by the variance shown in the figures of this section.

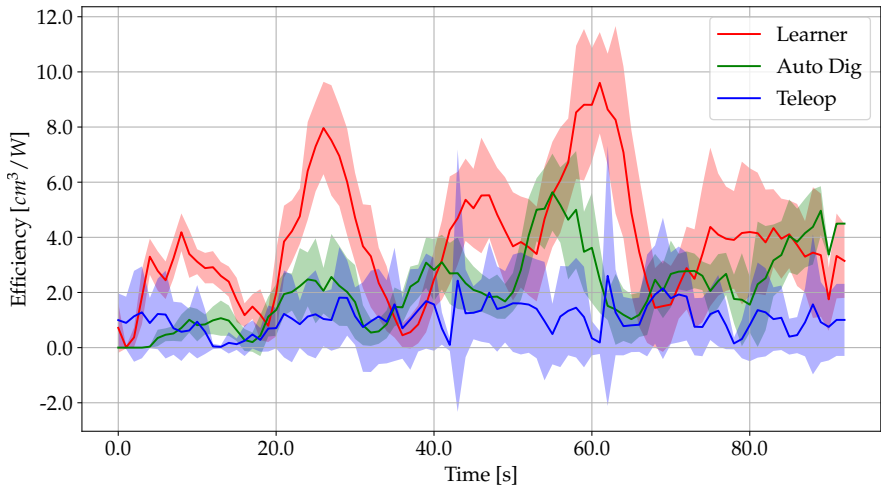

Fig. 15 Measured efficiency of the three modalities throughout execution of trenching task.

During excavation in both the simulation and the physical environment, it is important that the engagement brought

by the drums on the soil and the corresponding reaction force is equalized by controlling the arms. Pressing too hard with the drums could force the robot to lift. We visualize the scoop heights for each of the three methods, in Figures 13 and 14. The mean height above soil corresponds directly to how often the drums are engaged and therefore ingesting regolith; the standard deviation indicates engagement consistency. The teleoperated data had the least consistent height above soil (especially in the first front set of drums as shown in Figure 13). The results with the learner and auto dig were similar to each other. In some cases, the auto dig routine would go below zero meters indicating that it was pushing the scoops below the level of the excavation surface, though it was also consistently lower than the learner and nearer to the surface plane. If we instead consider the balance as a ratio between the sets of front and rear drums, the learner performs slightly better with a deviation error of 3.64% on average, and an auto dig deviation error of 4.39%, also on average. The ratio of front to rear drums for teleoperation produced an error of 62.64%, which is a detriment to efficiency.

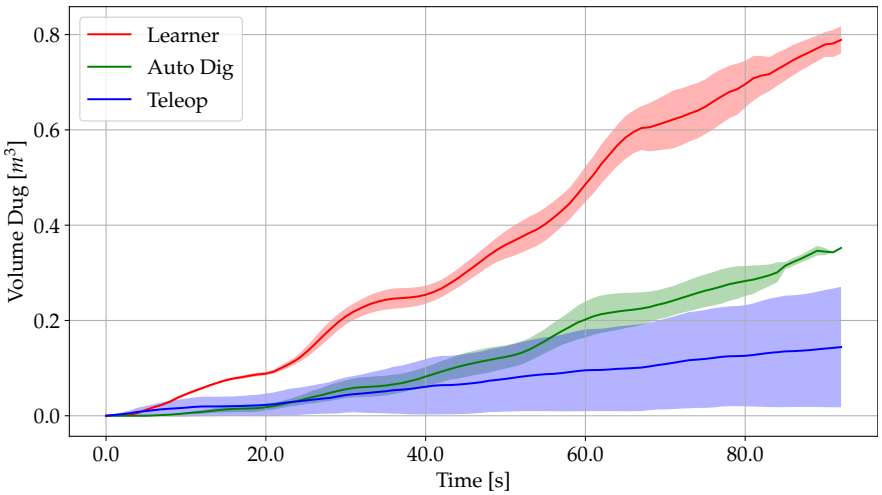

Fig. 16 Cumulative regolith excavated (by volume).

In Figure 15, we see a comparison of the average efficiency (temporally) for each of the methods. This shows the learned policy outperforms both the auto dig routine and teleoperated user by our key comparison metric. On average, the learner achieved $3.80 \pm 1.47\ cm^3/W$, whereas the auto routine achieved $2.18 \pm 0.89\ cm^3/W$, and the teleoperated user, $1.06 \pm 1.33\ cm^3/W$. In full excavation cycles, we see that the learner optimizes for minimizing the number of steps to reach the terminal full state. This is why we see a decrease, on average, in the steps per episode in Figure 11. However, in our 2D simulation, we see an average increase in the number of steps per episode as shown in Figure 6, as there was no positive terminal state. Instead the learner would simply dig until it was stuck, often reaching thousands of steps per episode.

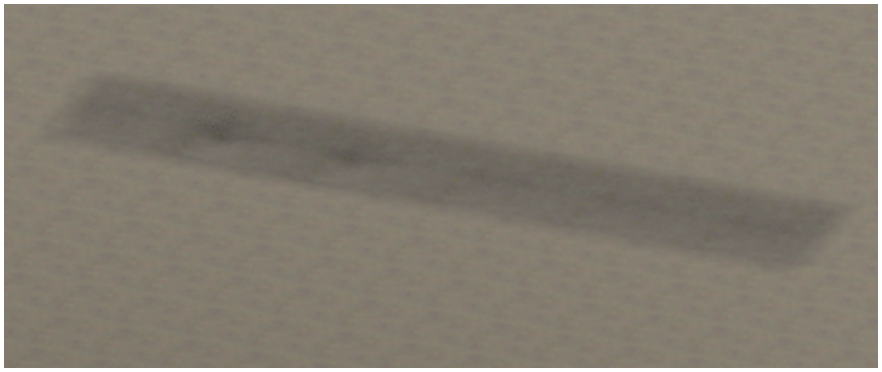

Fig. 17 Auto dig with persistent updates to the setpoint can produce smooth and shallow trenches. The depth of the trench decreases as the controller approaches the drum motor setpoint.

In addition to excavation efficiency, we also compare the cumulative volume excavated, shown in Figure 16. Though the learner excavates far more regolith over the course of our excavation cycle compared to the auto dig routine, it also consumes far more power, on average. This is indicated by the intersection of the upper and lower bounds of our efficiency measurements for the learner and auto dig, as shown in Figure 15. In our testing, the learned policy consumed 155.40% more battery charge over the same period of time, yet remained more efficient on average. During the execution of the learner and auto dig routines, the drums accumulated regolith at different rates on account of the asynchronous updates of the digging collider. This explains why the standard deviation gradually increases (generally) throughout the duration of an excavation cycle.

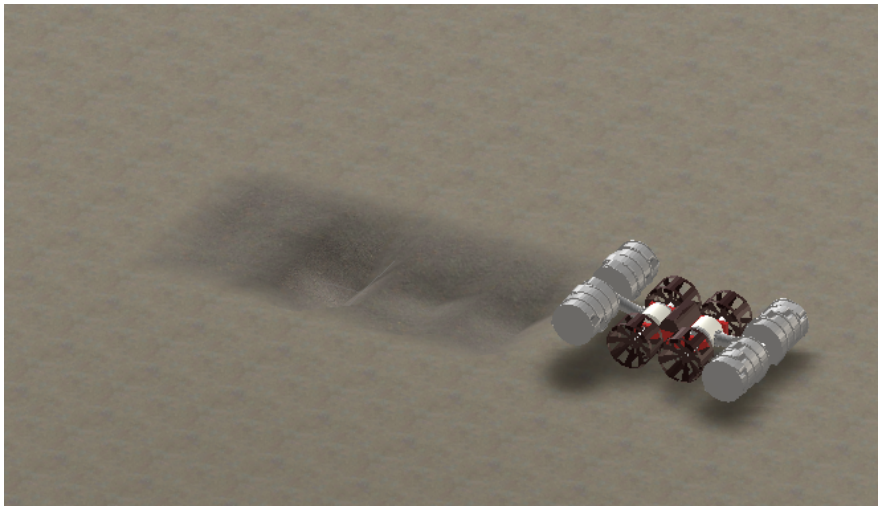

Fig. 18 During the second observable stage of learning, RASSOR's inconsistent force on the soil leads to an uneven trench.

Finally, in Figures 17, 18, and 19, we show results from the most recent iteration of our simulation. Figure 17 shows the results of the auto dig routine with relaxed controller setpoints. With this setting, the auto dig routine will grate the soil as it traverses the terrain. The default setpoints caused the arms to engage with more force during the beginning as the robot moved from left to right on the dig site, resulting in small indentations in the shape of the drums on the left side of the trench. Note that without manually incrementing the setpoint, RASSOR would slowly raise its arms over the course of excavation until the drums no longer engaged with the soil. As such, auto dig currently assumes active monitoring by an operator.

In Figure 18, we see the Deep Q-Learner after approximately 160 episodes of training. At this stage, the learner has largely overcome getting stuck, though must continue to optimize its behaviors. From the figure, we see more pronounced indentations from the drums and raised arms in the middle of excavation. This is because RASSOR has learned to bury drums in the soil, excavate some soil, then lift the drums so as to not get stuck when the robot is moving forward in the trench. This method is inefficient as the learner tends to prioritize one set of drums at a time. Within Figures 13 and 14, this would correspond to drum heights oscillating above the surface in increased frequency and amplitude. During the second stage of learning, RASSOR often terminates an episode early by exceeding slope angle limits by aggressively digging in place, in spite of temporal efficiency. We presume that as an effect of the slope angle penalty that RASSOR more frequently achieves grated excavation and successful episode termination by filling the drums.

In Figure 19, the results of the Deep Q-Learner are shown from a side view, similar to that used for the 2D simulation. Within this view, we see two cones in the corner of the site. These are the boundary markers for the dig site and there are four in total. We also note that the robotic platform starts slightly above the ground, though the episode does not begin until contact is reached. In the future, this simulation will be extended to support lunar topographical features using geological map data combined with noise to enhance small details. Dropping the platform is a simple solution to avoid instantiating the robot in intersection with the soil. Also worth noting in the visuals that the dusting effect produced during ground contact partially occludes the trench (nearest the drums). This does not affect the behavior of the learner, however. Figure 19 shows one of the smoothest executions of the policy and relatively balanced engagement of the arms to the soil. When driving, the policy tends to lift the leading drums so as to not collide with the topsoil.

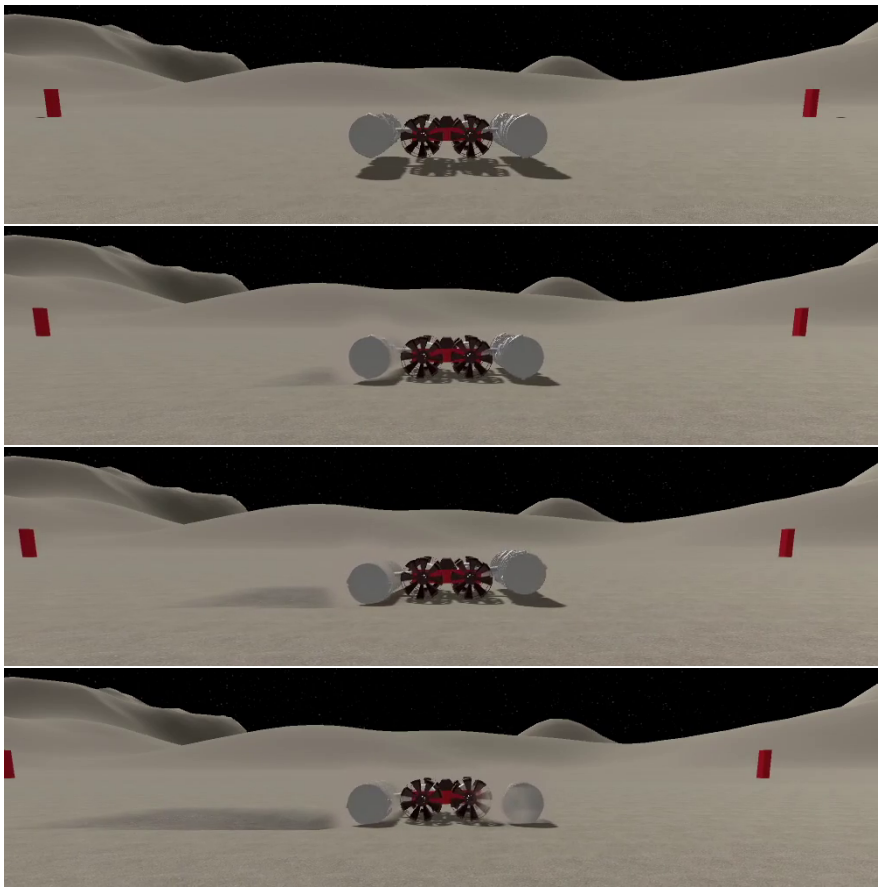

Fig. 19 Result of the excavation policy. The camera tracks the center of robot as it grates the terrain.

During inferencing of the learner routine, the control gains of the policy (governed by the learner's action) can be altered to decrease or increase the aggressiveness of the routine. This enables us to adapt the parameters of the policy online without changing the policy directly. By manipulating the velocity parameters of the actions, we can achieve desirable behavior based on the needs of higher level tasks, such as adapting gains based on the length of the trench desired.

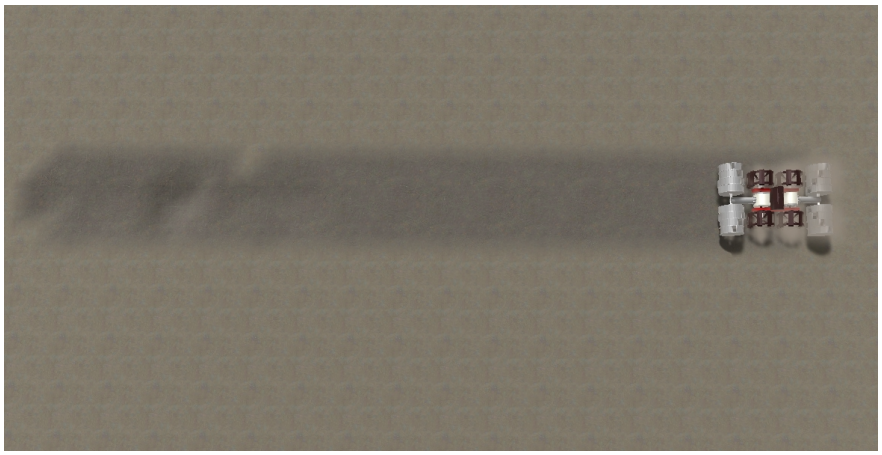

Fig. 20 Execution of the Deep Q-Learning policy yields results similarly smooth to the auto dig routine.

In Figure 20, we see the execution of the Deep Q-Learning policy after training is completed. Within the execution

of the dig cycle depicted, we see RASSOR gradually decrease the rate at which it ingests regolith while it balances the drums and moves forward (from left to right in the figure). The adaptation of control gains such as those in the excavation policy can be used within a hierarchical reinforcement learning framework to achieve more complex excavation behavior, and is just one example of the extensibility of the simulation and our framework for learning excavation, which we discuss further as part of future work.

VII. Conclusion

In this work, we developed 2D and 3D excavation simulations of a lunar environment. Within these simulations, we applied reinforcement learning algorithms to learn excavation policies using models based on the RASSOR excavator geometry and power consumption based on physical excavation data. Our learning system concluded with excavation policies outperforming autonomous digging routines in two efficiency metrics. Moreover, through the development of our simulation and reinforcement learning capabilities, we tuned our learning reward functions and state feedback such that we were able to identify novel sensor placement and needs of the RASSOR platform. Within this simulation learning framework, we are expanding the fidelity of the environment and its dynamics to more closely match our test and relevant environments. As such, there are several avenues of future work.

During the development of our 3D lunar simulation, several features were identified that would extend the capabilities of the tool as an ISRU simulation. These features include increased fidelity and resolution of the terrain and regolith properties as encountered during terrain traversal and manipulation, such as variance in surface topography and topsoil density. Using lunar surface data [30], accurate terrain details as found at proposed landing and excavation sites can also be incorporated. By increasing the correspondence between simulation and constrained real world scenarios, the efficacy of transferring knowledge learned in simulation to physical environments is also increased. This is in part achieved by improving the physics of the soil-drum interaction by refining data driven approaches and incorporating more sophisticated collision models. This remains the subject of ongoing work.

By increasing the visual fidelity of the environment, additional pseudo sensors can be implemented to imitate image and range-based sensing. The inclusion of vision sensors, for example, could be used for visual odometry and vision-based navigation within trenches. As the illumination during the lunar days consists of high contrast harsh lighting and shadows, methods such as active illumination [31] need to be considered to compensate the scene with enough light to extract usable data from images. The presence of electrostatically charged dust particles presents further challenges to vision-based sensors [32].

Within machine learning, transfer learning [33] remains a topic of importance. Where training robotic assets in the real world with uninitialized parameters remains challenging for a number of reasons, a simulation can be configured to automatically reset the environment after each episode, thus being much faster to learn in. Transfer learning for ISRU excavation policies is a topic of ongoing work. Kennedy Space Center's regolith test bin has been furnished with new motion capture capabilities to facilitate transfer learning and serve as ground truth for the development of dust-tolerant odometry methods.

Other topics of interest within machine learning include hierarchical learning. As part of this work, we focused on learning single end-to-end policies for straight trenching. End-to-end training can produce satisfying results, especially when desirable behavior is reached through elegant reward specification. However, it is often more practical (and explainable [34]) to decompose behaviors into distinct actions which can be trained and benchmarked independently. This comes at a cost of optimality [35], though low-level specification of actions such as control-level actions extend the flexibility of the model to new regimes since task specification is governed by the policy of a higher level [18].

VIII. Acknowledgements

This work is funded under the Intelligent Capabilities Enhanced RASSOR (ICE-RASSOR) project by the NASA Kennedy Space Center Independent Research and Technology Development (IR&TD) program.

References

[1] Sanders, G., Larson, W., Sacksteder, K., and Mclemore, C., "NASA in-situ resource utilization (ISRU) project: Development and implementation," *AIAA SPACE 2008 Conference & Exposition*, 2008, p. 7853.

[2] van Susante, P., and Gertsch, L., "Minerals from Space: Terrestrial and Extra-Terrestrial Perspectives," *Earth and Space 2018: Engineering for Extreme Environments*, American Society of Civil Engineers Reston, VA, 2018, pp. 390–400.

[3] Anand, M., Crawford, I. A., Balat-Pichelin, M., Abanades, S., Van Westrenen, W., Péraudeau, G., Jaumann, R., and Seboldt, W., "A brief review of chemical and mineralogical resources on the Moon and likely initial in situ resource utilization (ISRU) applications," *Planetary and Space Science*, Vol. 74, No. 1, 2012, pp. 42–48.

[4] Sanders, G. B., and Larson, W. E., "Integration of in-situ resource utilization into lunar/Mars exploration through field analogs," *Advances in Space Research*, Vol. 47, No. 1, 2011, pp. 20–29.

[5] Kelso, R. M., Romo, R., Andersen, C., Mueller, R. P., Lippitt, T., Gelino, N. J., Smith, J. D., Townsend, I. I., Schuler, J. M., Nugent, M. W., Nick, A. J., Zacny, K., and Hedlund, M., "Planetary Basalt Field Project: Construction of a Lunar Launch/Landing Pad, PISCES and NASA Kennedy Space Center Project Update," *Earth and Space 2016: Engineering for Extreme Environments*, American Society of Civil Engineers, 2016, pp. 653–667.

[6] Gelino, N. J., Mueller, R. P., Moses, R. W., Mantovani, J. G., Metzger, P. T., Buckles, B. C., and Sibille, L., "Off Earth Landing and Launch Pad Construction—A Critical Technology for Establishing a Long-Term Presence on Extraterrestrial Surfaces," *Earth and Space 2021*, 2020, pp. 855–869.

[7] Mueller, R. P., Smith, J. D., Schuler, J. M., Nick, A. J., Gelino, N. J., Leucht, K. W., Townsend, I. I., and Dokos, A. G., "Design of an excavation robot: regolith advanced surface systems operations robot (RASSOR) 2.0," *Earth and Space 2016: Engineering for Extreme Environments*, American Society of Civil Engineers Reston, VA, 2016, pp. 163–174.

[8] Buckles, B. C., Mueller, R. P., and Gelino, N. J., "Additive Construction Technology for Lunar Infrastructure," *LPI Contributions*, Vol. 2152, 2019, p. 5077.

[9] Mueller, R. P., Gelino, N. J., Smith, J. D., Buckles, B. C., Lippitt, T., Schuler, J. M., Nick, A. J., Nugent, M. W., and Townsend, I. I., "Zero Launch Mass Three-Dimensional Print Head," *Earth and Space 2018: Engineering for Extreme Environments*, American Society of Civil Engineers Reston, VA, 2018, pp. 219–232.

[10] Just, G., Smith, K., Joy, K., and Roy, M., "Parametric review of existing regolith excavation techniques for lunar In Situ Resource Utilisation (ISRU) and recommendations for future excavation experiments," *Planetary and Space Science*, Vol. 180, 2020, p. 104746.

[11] Sanders, G. B., and Larson, W. E., "Progress made in lunar in situ resource utilization under NASA's exploration technology and development program," *Earth and Space 2012: Engineering, Science, Construction, and Operations in Challenging Environments*, 2012, pp. 457–478.

[12] Burns, J. O., Mellinkoff, B., Spydell, M., Fong, T., Kring, D. A., Pratt, W. D., Cichan, T., and Edwards, C. M., "Science on the lunar surface facilitated by low latency telerobotics from a Lunar Orbital Platform-Gateway," *Acta Astronautica*, Vol. 154, 2019, pp. 195–203.

[13] Fong, T., Rochlis Zumbado, J., Currie, N., Mishkin, A., and Akin, D. L., "Space telerobotics: unique challenges to human–robot collaboration in space," *Reviews of Human Factors and Ergonomics*, Vol. 9, No. 1, 2013, pp. 6–56.

[14] Mueller, R. P., Cox, R. E., Ebert, T., Smith, J. D., Schuler, J. M., and Nick, A. J., "Regolith advanced surface systems operations robot (RASSOR)," *2013 IEEE Aerospace Conference*, IEEE, 2013, pp. 1–12.

[15] Sutton, R. S., and Barto, A. G., *Reinforcement learning: An introduction*, MIT press, 2018.

[16] Watkins, C. J., and Dayan, P., "Q-learning," *Machine learning*, Vol. 8, No. 3-4, 1992, pp. 279–292.

[17] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al., "Human-level control through deep reinforcement learning," *nature*, Vol. 518, No. 7540, 2015, pp. 529–533.

[18] Arulkumaran, K., Deisenroth, M. P., Brundage, M., and Bharath, A. A., "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, Vol. 34, No. 6, 2017, pp. 26–38.

[19] Allan, M., Wong, U., Furlong, P. M., Rogg, A., McMichael, S., Welsh, T., Chen, I., Peters, S., Gerkey, B., Quigley, M., et al., "Planetary rover simulation for lunar exploration missions," *2019 IEEE Aerospace Conference*, IEEE, 2019, pp. 1–19.

[20] EZRASSOR Team, "EZRASSOR: EZ Regolith Advanced Surface Systems Operations Robot," `https://github.com/FlaSpaceInst/EZ-RASSOR`, 2021. Commit: f034dc9fdf025e845d82f74d676f804e1c373b84.

[21] Walton, O. R., and Johnson, S. M., "DEM simulations of the effects of particle shape, interparticle cohesion, and gravity on rotating drum flows of lunar regolith," *Earth and Space 2010: Engineering, Science, Construction, and Operations in Challenging Environments*, 2010, pp. 36–41.

[22] Zeng, X., Burnoski, L., Agui, J., and Wilkinson, A., "Calculation of excavation force for ISRU on lunar surface," *45th AIAA aerospace sciences meeting and exhibit*, 2007, p. 1474.

[23] Koenig, N., and Howard, A., "Design and use paradigms for gazebo, an open-source multi-robot simulator," *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, Vol. 3, IEEE, 2004, pp. 2149–2154.

[24] Krüger, T., Schiele, A., and Hambuchen, K., "Exoskeleton control of the robonaut through rapid and ros," *Proceedings of the 12th Symposium on Advanced Space Technologies in Robotics and Automation, Noordwijk, Netherlands*, 2013.

[25] Bualat, M., Barlow, J., Fong, T., Provencher, C., and Smith, T., "Astrobee: Developing a free-flying robot for the international space station," *AIAA SPACE 2015 Conference and Exposition*, 2015, p. 4643.

[26] Thangavelautham, J., Law, K., Fu, T., El Samid, N. A., Smith, A. D., and D'Eleuterio, G. M., "Autonomous multirobot excavation for lunar applications," *Robotica*, Vol. 35, No. 12, 2017, pp. 2330–2362.

[27] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W., "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[28] Janmohamed, N. A., Cloud, J. M., Leucht, K. W., Bell, E. A., Buckles, B. C., and DuPuis, M. A., "Mass Inferencing Model Creation and Deployment to the RASSOR Lunar Excavation Robot," *AIAA ASCEND 2021*, 2021.

[29] Greene, M. L., DuPuis, M., Cloud, J., and Dixon, W. E., "Simultaneous Trajectory Tracking Control and Online Mass Estimation for a Regolith Excavating Robot via Integral Concurrent Learning," *AIAA Scitech 2021 Forum*, 2021, p. 1131.

[30] Smith, D. E., Zuber, M. T., Neumann, G. A., Lemoine, F. G., Mazarico, E., Torrence, M. H., McGarry, J. F., Rowlands, D. D., Head, J. W., Duxbury, T. H., et al., "Initial observations from the lunar orbiter laser altimeter (LOLA)," *Geophysical Research Letters*, Vol. 37, No. 18, 2010.

[31] Wong, U. Y., *Lumenhancement: Exploiting appearance for planetary modeling*, Doctoral dissertation, Carnegie Mellon University, 2012.

[32] Calle, C. I., "The electrostatic environments of Mars and the Moon," *Journal of Physics: Conference Series*, Vol. 301, IOP Publishing, 2011, p. 012006.

[33] Taylor, M. E., and Stone, P., "Transfer learning for reinforcement learning domains: A survey." *Journal of Machine Learning Research*, Vol. 10, No. 7, 2009.

[34] Puiutta, E., and Veith, E. M., "Explainable reinforcement learning: A survey," *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, Springer, 2020, pp. 77–95.

[35] Kaelbling, L. P., Littman, M. L., and Moore, A. W., "Reinforcement learning: A survey," *Journal of artificial intelligence research*, Vol. 4, 1996, pp. 237–285.