Crimson Publishers
Wings to the Research

Mini Review

Investigation of Tool-Soil Interaction and Autonomous Front-Loader Motion Planning

Lihi Kalakuda* and Amir Shapiro

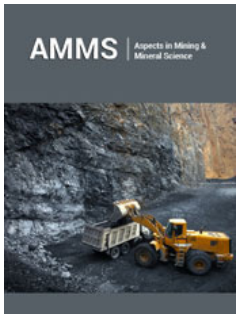
Department of Mechanical Engineering, Ben Gurion University, Israel

Abstract

The world still relies on human labor when it comes to operating heavy machinery. This dependence on human labor is expensive, time-intensive, and hazardous. This paper deals with an autonomous front-loader movement planning for a soil loading task. We used reinforcement learning algorithm trained by a Gazebo simulator integrated with ROS and OpenAI Gym framework. The continuum model was chosen as the soil-tool interaction model. Next, we trained outrobot to load soil utilizing Proximal Policy Optimization algorithms. Our results show that this algorithm yields high return with a moderate number of training steps.

Keywords: Autonomous; Material; Bucket filling; Soil-tool; Loader; Hydraulics

ISSN: 2578-0255



***Corresponding author:** Lihi Kalakuda, Department of Mechanical Engineering, Ben Gurion University, Israel

Submission: September 28, 2020

Published: October 22, 2020

Volume 5 - Issue 5

How to cite this article: Lihi Kalakuda, Amir Shapiro. Investigation of Tool-Soil Interaction and Autonomous Front-Loader Motion Planning. Aspects Min Miner Sci. 5(5). AMMS. 000621. 2020.
DOI: 10.31031/AMMS.2020.05.000621

Problem Definition

An autonomous earth moving machine is one that is able to navigate itself to target area, load and unload designated material, and repeat steps as needed. Fortunately, autonomous navigation has already been developed, but the problem of autonomous bucket filling step in the loading cycle remains open, despite three decades of research. To address the bucket filling problem, we aim to develop an algorithm for the most efficient loading of a front-loader's bucket. The absence of an accurate model of the material to be scooped prevents the use of optimization methods and therefore we decided to examine machine learning algorithms for model-free problems using reinforcement learning.

Related Work

Research aiming to automate earth-moving machines has a long history [1-4]. However, to the best of our knowledge and at the time of writing, a reliable or commercial system for autonomous earth-moving machines has not yet been introduced. Work has been done to include machine learning for bucket-filling motion-planning. Dadhich et al. [5] reveals that application of machine learning to automate the bucket filling process is feasible in principle and can lead to flexible solutions. Dadhich created a model for training with an appropriate dataset that can be adapted to a new machine, material, or environmental condition. In addition, Dadhich et al. [6] used a neural network ensemble to predict bucket filling control actions of an operator. The training data is recorded during a controlled experiment with an expert driver filling bucket. Hodel [7] used reinforcement learning algorithms to control an excavator and perform bucket leveling. His study, based on policy optimization by mimicking the human operation, showed that the reinforcement algorithms have excellent results. The methods presented are trained to mimic the actions of a human operator, not necessarily representing the best strategy for automated bucket filling.

Bucket Filling Algorithm

A reinforcement learning task is about training an agent which interacts with its environment. The agent arrives at different scenarios known as observations or states by performing actions. In Reinforcement Learning, the agent is the one who takes decisions based on the rewards and punishments. In our case, the Bobcat (mini front-loader model) is the agent. The environment contains all the necessary functionality to run an agent and allow it to learn. Gym environment provides a standardized interface for the reinforcement learning process (Figure 1). In order to build a custom environment for our Bobcat, we defined the

action space, which contains all the actions possible for the bobcat to perform, and similarly, the observation space which contains all the environment's data to be observed by the agent.

$$\overline{Observation} = \begin{pmatrix} x_{pos} \\ z_{pos} \end{pmatrix}, \quad \overline{action} = \bar{v}_t = \begin{pmatrix} v \\ \dot{\alpha} \\ \dot{\beta} \end{pmatrix} \qquad (1)$$

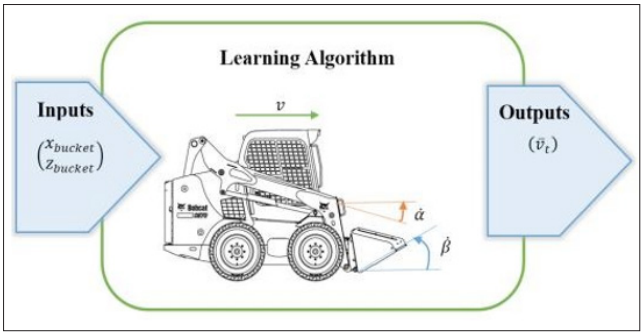


Figure 1: Skid-steer bucket velocities and algorithm planning method.

The algorithm input data is the existing condition of the bucket based on the data that the Bobcat receives from the environment through its sensors. In the future it will be possible, utilizing a front camera and image processing, to identify the pile slope and determine the type of soil and thus select the movement required to fill the bucket. The behavior of a learning agent is not programmed explicitly, but implicitly. The policy is optimized to maximize the accumulated returns from the reward function. The reward conditions were based on the definition of a desirable operation; shortest time, maximum soil loaded and minimum penetration height. Proximal Policy Optimization (PPO) algorithm is an on-policy algorithm which is based on policy gradient methods [8]. Policy Gradient methods have convergence problem which is referred by the natural policy gradient. In practice, natural policy gradient involves a second-order derivative matrix which makes it difficult to solve when it comes to large scale. PPO uses a different approach; it relies on specialized clipping in the objective function to remove incentives for the new policy to get far from the old policy. With clipped objective, we compute a ratio between the new policy and the old policy:

$$r_t(\theta) = \frac{\pi\theta(a_t/s_t)}{\pi\theta_{old}(a_t/s_t)} \qquad (2)$$

This ratio measures the difference between two policies, new and old. The objective function to clip the estimated advantage function if the new policy is far away from the old policy is:

$$L^{CLIP}(\theta) = \hat{E}_t\left[\min\left(r_t(\theta)\hat{A}_t, clip\left(r_t(\theta), 1-\varepsilon, 1+\varepsilon\right)\hat{A}_t\right)\right] \qquad (3)$$

Where ε is a hyperparameter which roughly says how far away the new policy can go from the old and $\hat{A}_t$ is an estimator of the advantage function. Based on "Deep Reinforcement Learning Hands-On Second Edition" [9], we maintain two policy networks (Figure 2). The first is the current policy that we want to refine called the Critic. The second is the policy that we last used to collect samples called the Actor. The actor and critic networks consist of two hidden layers with 64 units each. Training and testing our algorithm in the real world would consume many work hours, entail large sums of money, and be overall cumbersome. To streamline the process, we deployed a simulation of the environment, there we trained our algorithm.

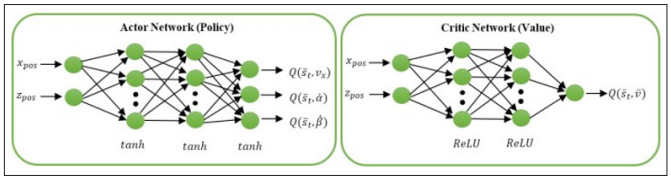


Figure 2: Networks architecture used in PPO algorithm.

Simulation Framework

The simulation hangs on two factors-soil-tool interaction and the learning algorithm. Robil system [10] was integrated with ROS and Gazebo. Robil system enables to switch between the Gazebo simulation and the real world. In order to use the same Bobcat model and to facilitate later the connection between the learned motion to the real-world Bobcat, we used Robil system as basis for our model agent. The soil forces plugin simulates the forces exerted by the soil on the tool that enters it. We chose the continuum model of soil which is an analytical model requiring relatively lower processing needs to achieve results (Figure 3).

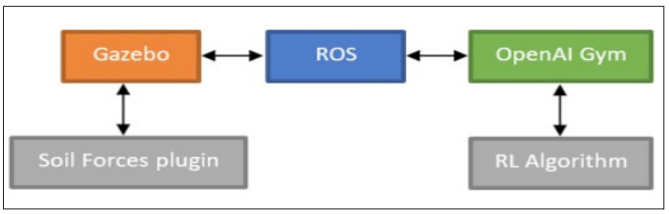


Figure 3: Simulation software architecture.

Simulation Results

The results obtained from the PPO simulation present in (Figure 4). From the graphs presented for the trainings process, we can assume that a successful learning test is achieved after 4M observations and 19 hours of training. The reward obtained after 1M observations and four hours of trainings was around 630 and doubled after 4M observations (reward around 1340). Even though the time required for training may seem long, it is much shorter and more economical than learning from real-world results. Since the study is performed from a simulator with a physical engine, the training time depends on the actual length of each episode. Furthermore, as part of the attempts to shorten the learning process, the number of possible steps in each episode was limited to 900 steps. Through this, the agent is trying to maximize the objective function for shorter episodes and will not preform long

attempts of steps that will get a low score due to their long-time operation.

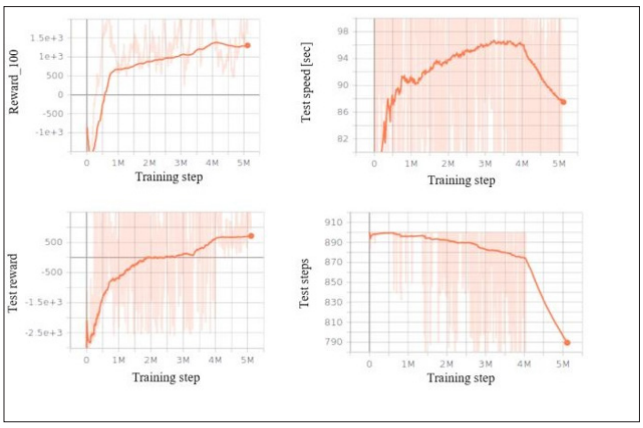


Figure 4: PPO algorithm training results.

Experiments and Conclusion

Figure 5: Bobcat in the experiment.

Experiments were conducted on real T190 BobCat Platform [10]. We used a distant site which enables remote control with LAN communication. This platform belongs to the Israel Aerospace Industries; therefore, experiment time was short and concentrated. Robil system designed with a GUI (Graphical User Interface) package that allows you to load a file for the bucket task (Figure 5). We planned a task for the Manipulator component but when checking the topic that is responsible for updating the movement speed of the main arm and the bucket (Low-Level Control), nothing happened. The link between the packages within the Robil system is cumbersome, so after several days of trying to link the GUI to the topic, it was decided to build an external publisher which would transfer the movement learned directly to the LLC. By updating the efforts, we controlled the joints efforts and published efforts to the hydraulics and the loader pistons engine. A wet sand pile was laid in front of the Bobcat and the robot actions were examined again. These experiments confirm that a set of orders can be planned in advance and executed by the robot to optimally move the bucket in the required trajectory.

References

1. Mikhirev PA (1983) Theory of the working cycle of automated rock-loading machines of periodic action. Soviet Mining 19(6): 515-522.
2. Hemami A (1995) Fundamental analysis of automatic excavation. Journal of Aerospace Engineering 8(4): 175-179.
3. Marshall, Alexander J (2001) Towards autonomous excavation of fragmented rock: Experiments, modelling, identification and control. Master of science: Engineering, Queen's University, Kingston, Canada.
4. Ahmad H, Hassani F (2009) An overview of autonomous loading of bulk material. 26th International Symposium on Automation and Robotics in Construction, USA.
5. Dadhich S, Bodin U, Sandin F, Andersson U (2016) Machine learning approach to automatic bucket loading. 2016 24th Mediterranean Conference on control and Automation (MED), IEEE, Greece.
6. Dadhich S, Sandin F, Bodin U (2018) Predicting bucket-filling control actions of a wheel-loader operator using a neural network ensemble. 2018 International Joint Conference on Neural Networks (IJCNN), IEEE, Brazil.
7. Hodel BJ (2018) Learning to operate an excavator via policy optimization. Procedia Computer Science 140: 376-382.
8. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O (2017) Proximal policy optimization algorithms. Cornell University, USA.
9. Lapan (2020) Deep reinforcement learning hands. Second Edition, Packt Publishing, UK.
10. Meltz D, Hugo G (2016) RobIL-Israeli program for research and development of autonomous UGV: Performance evaluation methodology. 2016 IEEE International Conference on the Science of Electrical Engineering (ICSEE), IEEE, Israel.