IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 16, 2020, accepted November 15, 2020, date of publication November 24, 2020,
date of current version December 10, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3040246

Automated Excavator Based on Reinforcement Learning and Multibody System Dynamics

ILYA KURINOV[1], GRZEGORZ ORZECHOWSKI[1,2], PERTTU HÄMÄLÄINEN[3], AND AKI MIKKOLA[1]

[1]Department of Mechanical Engineering, LUT University, 53850 Lappeenranta, Finland
[2]Mevea Ltd., 53850 Lappeenranta, Finland
[3]Department of Computer Science, Aalto University, 02150 Espoo, Finland

Corresponding author: Ilya Kurinov (ilya.kurinov@lut.fi)

This work was supported in part by the European Union's Horizon 2020 Research and Innovation Programme through the Marie Sklodowska-Curie Project under Grant 845600 (RealFlex) in part with Academy of Finland #316106.

ABSTRACT Fully autonomous earth-moving heavy equipment able to operate without human intervention can be seen as the primary goal of automated earth construction. To achieve this objective requires that the machines have the ability to adapt autonomously to complex and changing environments. Recent developments in automation have focused on the application of different machine learning approaches, of which the use of reinforcement learning algorithms is considered the most promising. The key advantage of reinforcement learning is the ability of the system to learn, adapt and work independently in a dynamic environment. This article investigates an application of reinforcement learning algorithm for heavy mining machinery automation. To this end, the training associated with reinforcement learning is done using the multibody approach. The procedure used combines a multibody approach and proximal policy optimization with a covariance matrix adaptation learning algorithm to simulate an autonomous excavator. The multibody model includes a representation of the hydraulic system, multiple sensors observing the state of the excavator and deformable ground. The task of loading a hopper with soil taken from a chosen point on the ground is simulated. The excavator is trained to load the hopper effectively within a given time while avoiding collisions with the ground and the hopper. The proposed system demonstrates the desired behavior after short training times.

INDEX TERMS Autonomous agents, discrete event dynamic automation systems, learning and adaptive systems, real-time simulation, multibody system dynamics, reinforcement learning, PPO-CMA.

I. INTRODUCTION

Mobile earth-moving machinery is widely used in the construction, forestry and mining industries. The operation of such machinery is challenging and often hazardous, because the machines are used in dangerous environments like open-pit mines and large infrastructure projects. From the performance perspective, machine operators are often the weakest link, as they introduce delays in the form of statutory rest periods, transportation to and from the worksite, and fatigue-induced errors. Consequently, many manufacturers of heavy machinery have begun to explore greater automation of machine operations. Tele-remote operation [1] is nowadays possible, which eliminates the aspect of the dangerous environment, but other issues remain. To fully address the challenges associated with heavy earth-moving operations, there is a need for fully automated and operator-independent machines.

Use of autonomous earth-moving machinery can increase safety and productivity, but production and development of such equipment is a formidable task. The main challenge facing earth-moving machinery automation is the complexity of the working environment. In mining and other earth-moving operations, the environment is characterized by diversity, variable geometry, the effects of the machine or other machines on the environment itself, the properties of the soil or media to be excavated, and external factors such as ambient temperature and climatic conditions. Control of machine operations using conventional methods is therefore a challenging enterprise [2]. The autonomous machine should be able to adapt to the sum of external factors and make decisions based on the task and state of the environment.

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Zhang.

In view of the complexity of the task and environment, utilization of machine learning can be considered a compelling option [3]. Reinforcement learning (RL) is a machine learning method that can be adopted for efficient learning of interactions with a complicated environment. RL uses the concept of an agent – a program capable of performing actions in the environment [4] – to determine actions with the aim of maximizing a pre-defined expected cumulative future reward. The learning process in RL consists of constant exploration of possible actions, acquiring information about the environment from observations, and receiving an evaluation of an action by a reward [4]. The observation and reward functions allow programming of the behavior of the model to adapt to changes in the environment: a positive reward is provided for the right behavior and a negative reward given after incorrect behavior. In addition, the agent can be easily retrained for different behavior patterns, which provides opportunities for enhancement of the automated machine.

Machine automation can be addressed in multiple ways, as reported in previous studies. In [5], the automation was based on training a fuzzy logic algorithm to mimic the behavior of a skilled human operator. Another study [6] utilized laser rangefinders mounted onto the excavator for automation of digging and loading procedures. In [7], a tele-operated assistant system controlled a real excavator via a wireless local area network. A multibody simulation and behavior-based excavation control approach for application with excavators performing landscaping tasks was introduced in [8].

In recent years, numerous publications have focused on the topic of artificial intelligence in automation. An example from the area of automation of heavy machinery is a study examining the application of Proximal Policy Optimization to teach an agent to operate an excavator [9]. The main goal of the study was to train the neural network to accomplish a leveling task known as bucket dragging. The research used a Dynasty simulation engine (propriety software of Caterpillar Inc.) in combination with the Open AI Gym framework [9]. Another study applied a machine learning approach to automation of a wheel loader [10]. Experimental data were collected on an expert driver filling the bucket of a Volvo L110G wheel loader and a regression model was then optimized with a reinforcement learning algorithm in terms of bucket filling and fuel efficiency [10]. It should be noted that utilization of reinforcement learning for heavy machinery is not limited to controlling the movements of the machine. For example, reinforcement learning has been used in development of a real-time energy management system of a hybrid excavator to control energy flow and optimize efficiency [11].

The approach of mimicking the behavior of a human operator can have negative effects on efficiency, because the operator might not operate the machine optimally. Methods based on training with a virtual simulation model use simplistic representations of the environment and machine, which may result in difficulties when adopted for implementation on real machines.

There is increasing evidence that neural network control policies trained in simulation can also work in the real world if the training scheme is designed correctly [12]–[14]. However, conducting real-world tests on heavy machinery such as an excavator is expensive, and simulation experiments are first needed to lay the groundwork. This article extends the only prior study on neural network excavator control [9] by learning to control a more detailed multibody excavator model in a more complex task, demonstrating the scalability of the approach. We also describe the multibody simulation in detail, whereas [9] treats the simulator as a black box. A multibody model is a computer-based simulation model replicating the physical structure and behavior of a machine with interconnected rigid or flexible bodies. Use of a multibody machine model is beneficial for training, because it eliminates risks of damaging the real machine and objects on the construction site and endangering personnel.

This study applies five methods for achieving training of the autonomous agent. The reinforcement learning method applies Proximal Policy Optimization with Covariance Matrix Adaptation (PPO-CMA) algorithm [15]. The agent is operating the excavator model that was based on the use of the multibody system dynamics. The current study employs the semi-recursive multibody approach, in which relative joint coordinates are used to minimize the number of differential equations [16]–[19]. The excavator is actuated by hydraulics. In this study, the hydraulic systems of the excavator are modeled using lumped fluid theory, in which the hydraulic system is discretized to volumes [20], [21]. To simulate real-life tasks of the excavator, deformable ground model based on a combination of the cellular automata and particle method is implemented [22]–[25]. The agent and the multibody solver are connected by ZeroMQ [26] asynchronous messaging library, where reinforcement learning algorithm is the server side and simulation environment is the client side.

The objective of this study is to computationally combine efficient multibody simulations with the reinforcement learning algorithm. This combination, in turn, enables the creation and training of an autonomous agent. In this way autonomous agents can be designed effectively without endangering personnel or machinery.

II. METHODS

This study combines five elements. The first element is reinforcement learning implemented using the PPO-CMA algorithm [15]. The second element is a multibody system dynamics solver that uses a semi-recursive method [16]–[19]. The third method is the hydraulics simulation using lumped fluid theory [20], [21]. The fourth method is a deformable ground implemented in the multibody simulation by combining cellular automata and particle methods [22]–[25]. The last element focuses on connecting the RL algorithm with the multibody simulation via a custom application program interface (API) in Python based on ZeroMQ [26].

A. REINFORCEMENT LEARNING AND PROXIMAL POLICY OPTIMIZATION WITH COVARIANCE MATRIX ADAPTATION ALGORITHM

Reinforcement learning is a machine learning paradigm in which learning is realized by a trial and error process. Unlike the other two main machine learning techniques, supervised learning and unsupervised learning, RL does not require a predefined dataset and uses the environment directly for learning. The environment and agent are usually simulation models that enable observation – a vector of data about the environment and reward – which is the score of a performed action. The agent is the part of the simulation capable of taking action. The action is a set of inputs for interaction with the environment. The agent interacts with an environment by providing an action vector, getting an observation vector and receiving evaluation via a reward function.

Reinforcement learning algorithms can be divided into model-based and model-free algorithms [27]. The difference between model-based and model-free approaches is the knowledge of the agent about the reward function and transition matrix which maps observations to actions. In model-based approaches, the transition matrix and reward function define the model of the world. On the other hand, in model-free methods, the agent does not have knowledge about the transition matrix and reward function and, thus, the agent must estimate state-action pairs according to previous experience.

This research focuses on application of a model-free policy optimization method. The policy defines the behavior of the agent. In reinforcement learning, the policy is parametrized, e.g., by weights and biases of the neural network, which can be adjusted to change the behavior of the agent. A common policy optimization method is Proximal Policy Optimization (PPO), which has been shown to be able to operate in complex environments [28]. The basic idea of PPO is to optimize the policy with a gradient and limit changes to the area of sampled actions [28]. Nevertheless, PPO has a tendency to slow down or get stuck at local optima, because of premature shrinkage of the exploration area [15]. The algorithm called Proximal Policy Optimization with Covariance Matrix Adaptation was designed to alleviate this problem [15]. The PPO-CMA algorithm is presented in the PPO-CMA study [1].

B. MULTIBODY SYSTEM DYNAMICS

Semi-recursive formulation is an efficient multibody-based method for simulating complex mechanical systems [16]–[19]. It enables dynamic responses to be solved in real-time and sometimes even faster than real-time [16]. Next, the formulation is briefly introduced.

Consider two rigid bodies n and $n-1$ with reference frames located at their center of mass as depicted in Fig.1. The bodies produce an open-chain system with $N_b = 2$ bodies connected by a joint. The locations of the joint in body $n-1$ and n are denoted Q and P respectively. The relative displacement

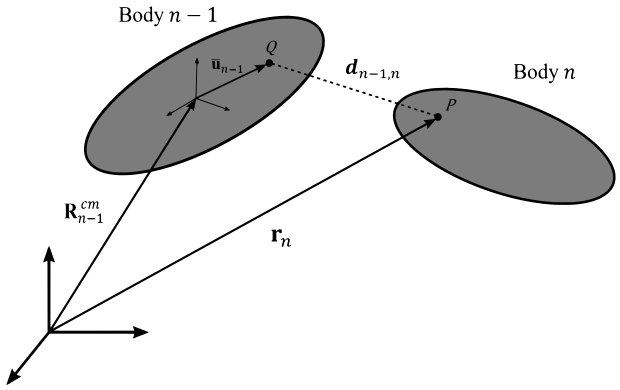

FIGURE 1. Multibody system of two rigid bodies.

vector between points Q and P can be described using vector $\boldsymbol{d}_{n-1,n}$.

The position of the joint fixture point P in the body n can be expressed as [16]:

$$\mathbf{r}_n = \mathbf{R}_{n-1}^{cm} + \mathbf{A}_{n-1}\bar{\mathbf{u}}_{n-1} + \boldsymbol{d}_{n-1,n} \tag{1}$$

where $\mathbf{R}_{n-1}^{cm}$ is the position vector that describes the center of mass of body $n-1$, $\mathbf{A}_{n-1}$ is the rotation matrix of body $n-1$, and $\bar{\mathbf{u}}_{n-1}$ is the location of point Q with respect to the body frame of reference $n-1$. The rotation matrix $\mathbf{A}_n$ of body n can be expressed using a concept of successive rotations as [16]:

$$\mathbf{A}_n = \mathbf{A}_{n-1}\mathbf{A}_{n-1,n} \tag{2}$$

where $\mathbf{A}_{n-1,n}$ is the relative rotation matrix between two neighboring bodies. In this study, the rotation of the bodies is expressed by Euler parameters. Velocity of point P can be expressed as [16]:

$$\dot{\mathbf{r}}_n = \dot{\mathbf{R}}_{n-1}^{cm} + \boldsymbol{\omega}_{n-1} \times \mathbf{A}_{n-1}\bar{\mathbf{u}}_{n-1} + \dot{\boldsymbol{d}}_{n-1,n} \tag{3}$$

where $\dot{\mathbf{R}}_{n-1}^{cm}$ is the velocity vector of the center mass of body $n-1$ and $\boldsymbol{\omega}_{n-1}$ is the angular velocity of body $n-1$. Angular velocity of body n can be expressed as [16]:

$$\boldsymbol{\omega}_n = \boldsymbol{\omega}_{n-1} + \boldsymbol{\omega}_{n-1,n} \tag{4}$$

where $\boldsymbol{\omega}_{n-1,n}$ is the relative angular velocity between body $n-1$ and body n.

By employing the principle of virtual work, equations of motion for a multibody system can be expressed [16]:

$$\delta W = \delta\dot{\mathbf{q}}^T (\mathbf{M}\ddot{\mathbf{q}} + \mathbf{C} - \mathbf{Q}) \tag{5}$$

where δW is the virtual work of multibody system, $\delta\dot{\mathbf{q}}$ is the vector of $6N_b$ generalized virtual velocities, $\mathbf{M}$ is the $6N_b \times 6N_b$ mass matrix, $\ddot{\mathbf{q}}$ is the vector of acceleration of generalized coordinates, $\mathbf{C}$ is the quadratic velocity vector, and $\mathbf{Q}$ is the vector of generalized external forces. Vectors $\ddot{\mathbf{q}}$, $\mathbf{C}$ and $\mathbf{Q}$ have $6N_b$ dimensions and have the form [16]:

$$\ddot{\mathbf{q}} = \begin{bmatrix} \ddot{\mathbf{q}}_1^T & \ddot{\mathbf{q}}_2^T & \dots & \ddot{\mathbf{q}}_{N_b}^T \end{bmatrix}^T \tag{6}$$

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_1^T & \mathbf{C}_2^T & \dots & \mathbf{C}_{N_b}^T \end{bmatrix} \tag{7}$$

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_1^T & \mathbf{Q}_2^T & \dots & \mathbf{Q}_{N_b}^T \end{bmatrix} \tag{8}$$

where $\ddot{\mathbf{q}}_n = \begin{bmatrix} \ddot{\mathbf{R}}_n^{cm^T} & \dot{\boldsymbol{\omega}}_n^T \end{bmatrix}^T$. The vector of virtual velocity $\delta\dot{\mathbf{q}}$ can be related to virtual joint velocities $\delta\dot{\mathbf{z}}$ as [16]:

$$\delta\dot{\mathbf{q}} = \mathbf{R}\delta\dot{\mathbf{z}} \tag{9}$$

where $\mathbf{R}$ is the velocity transformation matrix. For open-loop systems, the joint coordinates represents a set of minimal coordinates. By taking the derivative of (9), generalized accelerations can be expressed as [16]:

$$\ddot{\mathbf{q}} = \mathbf{R}\ddot{\mathbf{z}} + \dot{\mathbf{R}}\dot{\mathbf{z}} \tag{10}$$

Using (10) and (11), the equations of motion can be rewritten as [16]:

$$\delta\dot{\mathbf{z}}^T\mathbf{R}^T\left[\mathbf{M}\left(\mathbf{R}\ddot{\mathbf{z}} + \dot{\mathbf{R}}\dot{\mathbf{z}}\right) + \mathbf{C} - \mathbf{Q}\right] = 0 \tag{11}$$

(11) also holds for independent virtual velocities $\delta\dot{\mathbf{z}}$; therefore, virtual velocities can be eliminated [16]:

$$\mathbf{R}^T\mathbf{M}\mathbf{R}\ddot{\mathbf{z}} = \mathbf{R}^T\left(\mathbf{Q} - \mathbf{C}\right) - \mathbf{R}^T\mathbf{M}\dot{\mathbf{R}}\dot{\mathbf{z}} \tag{12}$$

The velocity transformation matrix $\mathbf{R}$ plays an important role in the formulation as it directly affects the efficiency of the method. One of the most efficient ways to compute the velocity transformation matrix $\mathbf{R}$ is application of the element-by-element technique [17]:

$$\mathbf{R}_{N_b} = [\mathbf{R}_{N_b}^1 \quad \mathbf{R}_{N_b}^2 \quad \dots \quad \mathbf{R}_{N_b}^{P_b}] \tag{13}$$

Closed-loop systems can be computed by expressing the system in the form of an open-loop system and then closing it be employing constraints [16]. Accordingly, the equation of motion takes the form [17]:

$$\begin{aligned}(\mathbf{R}^T\mathbf{M}\mathbf{R} + \alpha\boldsymbol{\Phi}_{\mathbf{z}}^T\boldsymbol{\Phi}_{\mathbf{z}})\ddot{\mathbf{z}} = \mathbf{R}^T\left(\mathbf{Q} - \mathbf{C}\right) - \mathbf{R}^T\mathbf{M}\dot{\mathbf{R}}\dot{\mathbf{z}} \\ - \alpha\boldsymbol{\Phi}_{\mathbf{z}}^T\left(\beta^2\boldsymbol{\Phi} + \mu\dot{\boldsymbol{\Phi}} + \dot{\boldsymbol{\Phi}}_{\mathbf{z}}\dot{\mathbf{z}} + \dot{\boldsymbol{\Phi}}_t\right)\end{aligned} \tag{14}$$

where α is a constant penalty factor, β^2 and μ are constants representing natural frequency and damping ratio respectively, $\boldsymbol{\Phi}$ is the loop closure constraint equations, $\boldsymbol{\Phi}_{\mathbf{z}}$ is a Jacobian matrix of the constraints, and $\dot{\boldsymbol{\Phi}}$ and $\ddot{\boldsymbol{\Phi}}$ are the first and second order derivatives of the constraint equations.

C. HYDRAULICS

In this study, the hydraulic system is modelled using the lumped fluid theory [20]. The approach is based on partitioning the system into discrete volumes in which the pressure is assumed to be equally distributed. The pressure in one volume can be computed as:

$$\dot{p_s} = \frac{B_{e_s}}{V_s}\sum_{k=1}^{n_c} Q_{sk} \tag{15}$$

where V_s is a volume, Q_{sk} is the sum of the flows going in or out of the volume, nc is the number of hydraulic components belonging to the volume, and B_{e_s} is the effective bulk modulus corresponding to the volume. In this study, hydraulic valves are modelled by semi-empirical approach [20]. In this approach, flow through a throttle valve can be described as:

$$Q = \begin{cases} C_v\sqrt{|dp|}, & dp > 0 \\ 0, & dp = 0 \\ -C_v\sqrt{|dp|}, & dp < 0 \end{cases} \tag{16}$$

where C_v is the flow rate constant of the valve, and dp is pressure difference between the input and output. Flow through a directional valve can be described as [21]:

$$Q = \begin{cases} C_vU\sqrt{|dp|}, & dp > 0 \\ 0, & dp = 0 \\ -C_vU\sqrt{|dp|}, & dp < 0 \end{cases} \tag{17}$$

where U is the relative spool position, which is described [21]:

$$\dot{U} = \frac{U_{ref} - U}{\tau} \tag{18}$$

where U_{ref} is a control signal and τ is a time constant describing the spool dynamics of the valve.

D. DEFORMABLE GROUND

Together with the multibody approach and hydraulics simulation, the deformable ground forms the basis for development of the autonomous agents. The deformable ground provides a representation of soil behavior and allows simulation of working scenarios of the machine. In this study, the environment of the excavator is described by employing a combination of the cellular automata-based [22], [23] and particle-based methods, as in [24], [25]. This combination of techniques defines an accurate deformable ground and enables ground material to be moved from one location to another as shown in Fig.2.

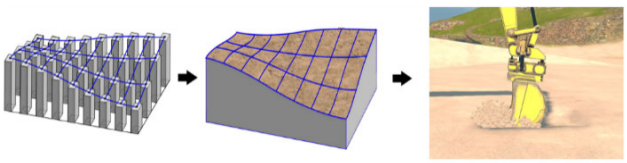

FIGURE 2. Deformable ground working principle [25].

In the cellular automata-based method [22], [23], the soil is described as cells in a grid (heightfield) which allows the formation of the landscape, as shown in Fig. 2. The cells are vertically divided into piles of blocks. The pressure over the base of one cell is obtained as a combination of the pressure received from their own block and the pressure from a finite number of nearby blocks above the base cell.

In this model, the displacement of the soil can occur for two reasons: 1) A large difference in the heightfield of two neighboring cells, resulting in avalanches; and 2) A large difference between the vertical forces applied to the neighboring cells, resulting in soil compression and displacement. In this study, the deformation of the heightfield depends on the interaction between the ground and external objects, such as an excavator

bucket or tracks. Depending on the amount of force induced on the soil, the cellular automata redefine the heightfield of the cells, i.e. adding the vertical deformation.

The particle-based method allows one to simulate the behavior of the soil when it is sheared with an external force [25]. In this method, particles are generated when the horizontal component of the applied force exceeds the shear impulse limit. During this event, the portion of the corresponding heightfield is substituted with spherical shaped particles, which have six degrees of freedom. The dynamics of the particles can be expressed by methods discussed in the multibody system dynamics subchapter [25]. Particle swarm has a void factor, which is responsible for the space between generated particles, i.e. soil compaction [24]. The soil compaction allows accurate dynamics representation of the soil compared with nature [24]. When the particles reach an equilibrium state, they merge back to the heightfield with a volume update [24].

E. DESCRIPTION OF APLICATION PROGRAMMING INTERFACE

RL training and simulation are performed by separate programs. Therefore, a connection between the simulation models and the reinforcement learning algorithm was developed using a custom Application Programming Interface (API). In this study, the API was implemented using the ZeroMQ asynchronous messaging library to send data between the applications. The client-server messaging pattern, which allows multiple clients to be connected to the server [26], was used as the base of the API. The RL side was used as a server and the simulation side as a client. The server side is shown in Algorithm 1 and the simulation side is shown in Algorithm 2. This structure allows simultaneous use of multiple simulations for training as depicted in Fig.3.

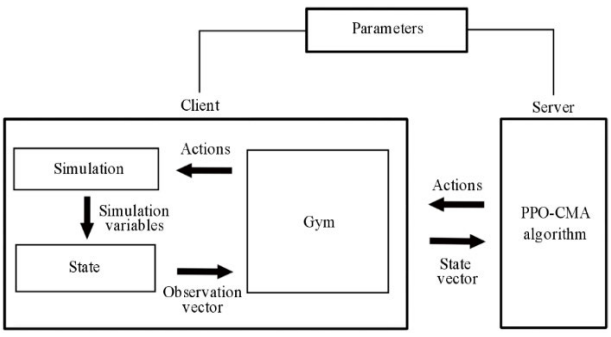

FIGURE 3. Structure of the API based on Gym, State and Parameters classes.

The API consists of the three main classes: Gym, Parameters and State. The Gym class is used to control the simulation and handle the connection to the algorithm. The class consists of the make, step and reset methods. The make method is used for initialization of the variables in the simulation. The step method is called for each time step of the simulation. It sends the observation vector, receives the action vector and assigns it to the corresponding inputs. The reset method is used to restart the simulation when the simulation budget is exceeded or the termination state has been reached. The Parameters class is used for transfer of simulation-related parameters to the RL algorithm. The State class obtains the observation vector from the simulation and calculates the reward value of the provided action.

Algorithm 1 PPO-CMA (Server Side)

1: **call function** make:
2 **for** iteration = 1, 2, ... do
3: **while** iteration simulation budget N not exceeded do
4: **call function** reset
5: **for** T timesteps run agent on current policy or until a terminal state
6: **call function** step
7: **end while**
8: Train critic network and policy
9: Save (serialize) agent
10: **end for**

Algorithm 2 Step Function in Simulation Environment (Client Side)

function step (simulation parameters list):
1: Receive action vector from agent (server)
2: Set actions to simulation model inputs
3: **for** action
4: Set action value to the hydraulic system input
5: **end for**
6: **for** observation parameter
7: Get simulation variables from Solver
8: Compose observation vector
9: Calculate reward
10: Encode reward and observation to a single message
11: **end for**
12: Send encoded observation vector, reward and done to the agent
end function

III. PROBLEM STATEMENT

The above procedure was applied to an excavator model with a complicated environment with deformable ground. Fig.4 shows the structure of the model. The model is a system with closed loops and has 4 Degrees of Freedom (DOF). The model consists of a total of nine rigid bodies interconnected by 10 joints.

The model is equipped with the hydraulic circuit shown in Fig.5. The excavator model has 4 inputs controlling inputs rotation of the upper carriage, boom lift, boom tilt and bucket. The resulting action vector takes the form:

$$\mathbf{a} = [u_{uc} \quad u_{lb} \quad u_{da} \quad u_b]^T \tag{19}$$

where u_{uc} is the upper carriage input, u_{lb} the lift boom input, u_{da} the dipper arm input and u_b the bucket rotation input. The input u_{uc} controls the hydraulic motor slew, which rotates the

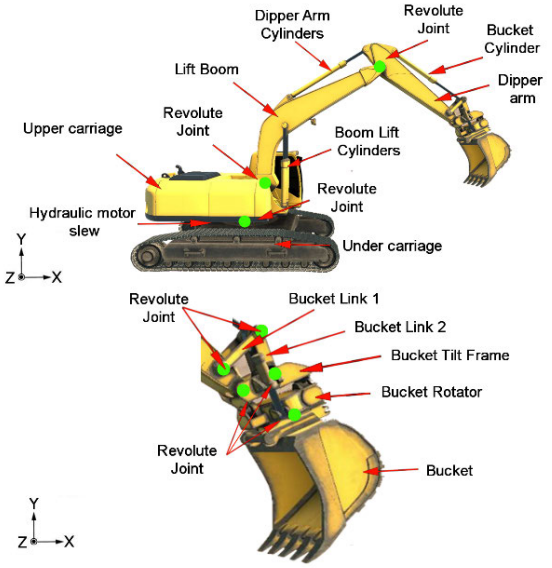

FIGURE 4. Topological structure of the excavator model with hydraulic actuators.

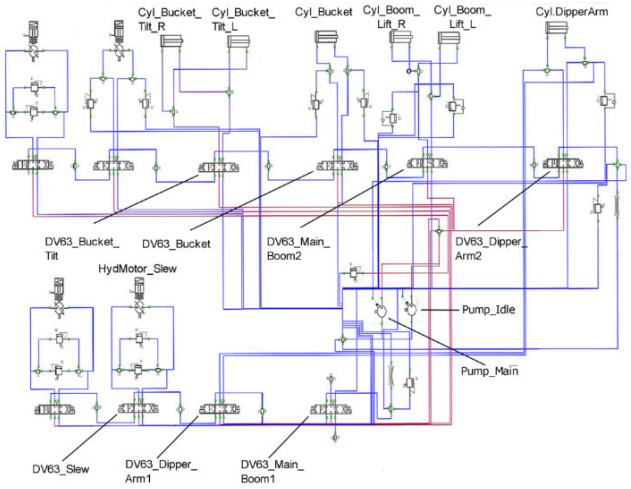

FIGURE 5. Hydraulic circuit schematics of the excavator.

upper carriage around Y-axis relative to the under carriage. The lift boom input u_{lb} controls the boom lift hydraulic cylinders, which rotates the lift boom around Z-axis relative to the upper carriage. The dipper arm input u_{da} controls the dipper arm hydraulic cylinder, which rotates the dipper arm around Z-axis relative to the lift boom. The bucket input u_b controls the hydraulic cylinder, which rotates the bucket around Z-axis relative to the dipper arm, please see Figure 4.

For evaluation of the performed actions, the agent must access the observation vector and reward function. The observation vector of the excavator model is different for each reward function type but contains the same parameters groups, which are set as bodies, task and goal position. The first group consists of the global coordinates of the bodies in meters and the rotations of the bodies in Euler parameters. In the case of an excavator, this group includes the upper carriage, lift boom, tilt boom and bucket bodies. Task-related observations are represented by values from sensors. Normalized mass in the hopper and bucket, a number of collisions of the bucket with the hopper are used as the goal group. Combination of these groups results to the following observation vector:

$$\mathbf{s} = \begin{bmatrix} t & m_h & m_b & l_{t-1} & l_t & \Delta l & \mathbf{r}^T & \mathbf{r}_g^T & n_c & n_g \end{bmatrix}^T \quad (20)$$

where, t is elapsed time, m_h is normalized mass in the hopper, m_b is normalized mass in the bucket, l_{t-1} is the distance to current goal on the previous time step, l_t is the distance to the goal, Δl is the difference between l_t and l_{t-1}, $\mathbf{r}$ is position vector of bodies, $\mathbf{r}_g$ is position vector of goals, n_c is the number of the collisions during one episode and n_g is the number of the goals already reached. The $\mathbf{r}$ vector has a size of 28, which is equal to the number of positions and rotations of the 4 main moving bodies. It consists of:

$$\mathbf{r} = \begin{bmatrix} \mathbf{r}_{b1}^T & \mathbf{r}_{b2}^T & \mathbf{r}_{b3}^T & \mathbf{r}_{b4}^T \end{bmatrix}^T \quad (21)$$

where, $\mathbf{r}_{bn}$ is the vector of the body n positions. The $\mathbf{r}_{bn}$ vector has the following form:

$$\mathbf{r}_{bn} = \begin{bmatrix} x & y & z & e_0 & e_1 & e_2 & e_3 \end{bmatrix}^T \quad (22)$$

where x, y and z are XYZ-axes positions of the body, e_0, e_1, e_2 and e_3 are the Euler parameters of the body. In $\mathbf{r}_{bn}$ vectors stored locations and rotations of the Upper Carriage, Lift Boom, Dipper Arm and Bucket bodies. The $\mathbf{r}_g$ stores the positions of the goals and have a size of 6. It has the following form:

$$\mathbf{r}_g = \begin{bmatrix} x_{g1} & y_{g1} & z_{g1} & x_{g2} & y_{g2} & z_{g2} \end{bmatrix}^T \quad (23)$$

where x_{g1}, y_{g1}, z_{g1} are positions of Goal To Soil and x_{g2}, y_{g2}, z_{g2} are positions of Goal To Hopper markers. In total, the observation vector consists of 42 parameters.

Before development of the reward function, it is important to understand the task given to the agent. Multiple excavator operations exist in real life. For instance, in construction of underground pipelines, the excavator must create a trench, which requires so-called bucket leveling. On the other hand, mining work requires movement of soil from the ground into a truck without hitting the truck. Clearly, each operation requires a different approach, but they all involve movement of soil from one point to another. Therefore, movement of soil was chosen as the basis of the experiment.

The selected task consists of three steps: loading the bucket near the loading point, moving to the unloading point, and unloading the bucket. In the first step, the agent moves the bucket to the loading position and grabs the soil. During this step, it is important to move the bucket quickly and to grab the maximum amount of soil. After loading of the bucket, the soil must be moved to the unloading point over the hopper. This step requires the agent to move the bucket with little or no loss of soil. In the final step, the agent must unload the maximum amount of soil without hitting the hopper.

The setup of the simulation is shown in Fig.6. The environment consists of the soil model, points of interest for the task,

FIGURE 6. Simulation setup. The excavator is located uphill of the hopper and the unloading hopper is positioned to the right of the excavator.

and a hopper. The soil model allows collection and transfer of soil, which is essential information for training. Points of interest provide the agent with information about soil grabbing and unloading positions. The hopper is equipped with two sensors. The first sensor collects data about the mass in the hopper, and the second sensor tracks the number of collisions with the bucket. Using this setup, it is possible to track all the variables needed for formulation of a reward function.

The reward function of the excavator model utilizes three components. The first component is a reward for reaching the point where the bucket should be loaded. This point is marked as a blue dot in Fig.6. The second component is a reward for the mass collected in the bucket during loading. For preventing abuse of soil mass reward, it is inverse proportional to episode time. The third component utilizes the mass in the hopper, which is penalized by the number of collisions between the hopper and bucket. Thus, the reward function can be written as:

$$r(t) = e^{-l} + km_{bn}t_n + \frac{1}{n_c}m_{hn} \tag{24}$$

where $r(t)$ is a reward at the time t, k is the mass reward coefficient, l is the distance to the current point of interest, m_{bn} is the normalized mass in the bucket, m_{hn} is the normalized mass in the hopper, and n_c is the number of collisions with the hopper.

IV. RESULTS

The excavator model is highly nonlinear and includes mechanics, hydraulics, sensor measurements, a changing environment and process visualization. Consequently, the training process is challenging. In the simulation, the agent learnt acceptable performance after 10^5 steps, which took around two days of continuous training. Computing efficiency was affected by the deformable ground submodel, which could not be updated without reloading the simulation. Therefore, a significant amount of training time was spent on reloading the simulation file.

FIGURE 7. Result of the training process.

Fig.7 shows the result of the training process. As can be seen from the figure, the excavator grabs soil at the desired loading position. At the start of the training, during the loading stage, the excavator used a slightly higher angle of attack to the soil compared to the middle of the training, which resulted in a small inclination of the undercarriage when applying of the dipper arm force. Within the scope of this work, this behavior of the model is not dangerous, but in real-life applications, it can damage the machine. Therefore, in subsequent work, the reward function should penalize changes in the undercarriage rotation using data from an inclinometer. Interesting behavior of the agent was also noticed prior to the addition of collision sensor data. Without the collision penalty, the agent tends to push with force on the hopper to achieve a higher reward. Complications with the reward for soil in the bucket are also worth mentioning. The agent had a tendency to abuse the reward for soil in the bucket and stop moving after filling the bucket. Therefore, the coefficient k with value of 0.2 was introduced to reduce the effect of the soil-based reward.

An agent with a reward function, presented in (24), was able to grab soil and move it to the hopper, as shown in the plot of the average mass in the hopper in Fig.8. It can be clearly observed that the agent started moving ground at around 2×10^4 steps. The bucket is able to hold around 1200 kg of soil. As can be seen from Fig.9, on average the agent learnt to move 67% of the initial mass to the hopper. The maximum value of soil in the hopper was 1168 kg. Therefore, at maximum, the agent was able to deliver 97% of the initial mass in the bucket. In the Fig.8 and Fig.9 can be observed high standard deviation caused by the soil transfer

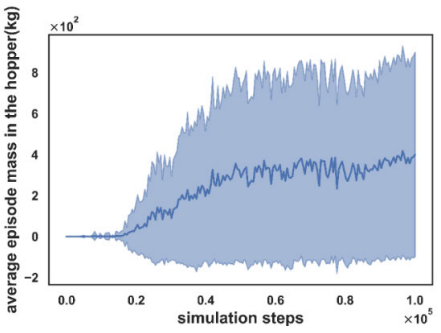

FIGURE 8. Average and standard deviation of episode mass in the hopper.

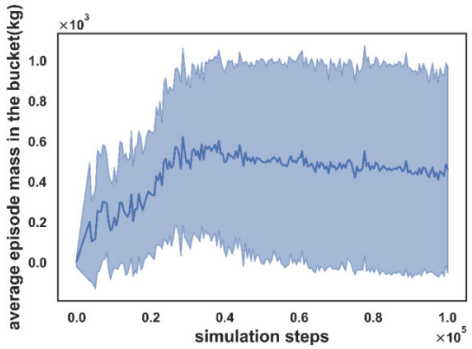

FIGURE 9. Average and standard deviation of episode mass in the bucket.

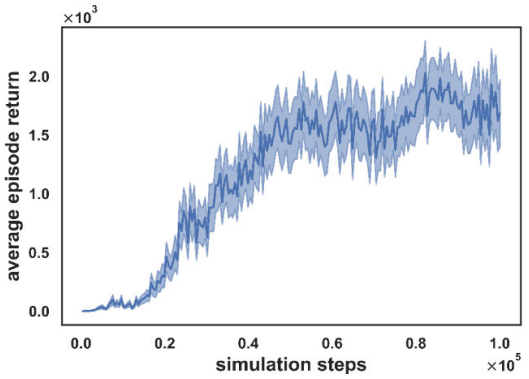

FIGURE 10. Average and standard deviation of episode return.

during simulation cycles, such as loading and unloading of the bucket.

As shown in Fig.10, the average episode return increased over the course of the training, indicating that the agent was learning the policy correctly. Nevertheless, the high amplitude of the average reward shows that the agent did not reached the convergence.

V. CONCLUSION

This article introduced a method for development of autonomous machines based on reinforcement learning and a semi-recursive multibody method. The approach was tested on an excavator model with hydraulics and a deformable ground. The research concentrated on the learning agent used to operate the excavator model and move ground from a predefined point to the unloading point. The study used the PPO-CMA model-free algorithm. The algorithm and environment were connected via an API built based on the ZeroMQ library.

The agent was able to learn a policy for working with the simulation models. The agent of the excavator was trained to load and unload the excavator with satisfactory accuracy. The agent was able to move 67-97% of the maximum mass of the bucket. Nevertheless, as was seen from the learning curves, the reward function still fluctuated. Thus, it can be concluded that increasing the number of episodes will likely result in enhanced excavator performance.

The setup presented extends opportunities for development of automated machines using reinforcement learning. The multibody system dynamics allows the creation of machinery model and its environment with a little effort. This makes possible training of agents on the wide variety of machines. Therefore, the method allows highly customized simulation models to be created which can be used for training machine learning algorithms or neural networks. Hence, it will be possible to generate agents for existing machines effectively without endangering the machine or personnel. Furthermore, the approach provides a solid foundation for extension of RL learning to new types of machines, which can be easily reprogrammed by downloading agents to the machine model.

REFERENCES

[1] S. Dadhich, U. Bodin, F. Sandin, and U. Andersson, "From tele-remote operation to semi-automated wheel-loader," *Int. J. Electr. Electron. Eng. Telecommun.*, vol. 7, no. 4, pp. 178–182, 2018, doi: 10.18178/ijeetc.7.4.178-182.

[2] S. Dadhich, U. Bodin, and U. Andersson, "Key challenges in automation of Earth-moving machines," *Autom. Construct.*, vol. 68, pp. 212–222, Aug. 2016, doi: 10.1016/j.autcon.2016.05.009.

[3] T. Hester, *TEXPLORE: Temporal Difference Reinforcement Learning for Robots and Time-Constrained Domains*, vol. 503. Cham, Switzerland: Springer, 2013, p. 164, doi: 10.1007/978-3-319-01168-4.

[4] R. Sutton, F. Bach and A. Barto, *Reinforcement Learning*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018, pp. 1–4.

[5] X. Shi, P. J. A. Lever, and F.-Y. Wang, "Experimental robotic excavation with fuzzy logic and neural networks," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 1996, pp. 957–962, doi: 10.1109/robot.1996.503896.

[6] A. Stentz, J. Bares, S. Singh, and P. Rowe, "Robotic excavator for autonomous truck loading," *Auto. Robots*, vol. 7, no. 2, pp. 175–186, 1999, doi: 10.1023/A:1008914201877.

[7] H. Shao, H. Yamamoto, Y. Sakaida, T. Yamaguchi, Y. Yanagisawa, and A. Nozue, "Automatic excavation planning of hydraulic excavator," in *Intelligent Robotics and Applications* (Lecture Notes in Computer Science), vol. 5315. Berlin, Germany: Springer, 2008, pp. 1201–1211, doi: 10.1007/978-3-540-88518-4_128.

[8] D. Schmidt, M. Proetzsch, and K. Berns, "Simulation and control of an autonomous bucket excavator for landscaping tasks," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2010, pp. 5108–5113, doi: 10.1109/ROBOT.2010.5509546.

[9] B. J. Hodel, "Learning to operate an excavator via policy optimization," *Procedia Comput. Sci.*, vol. 140, pp. 376–382, Jan. 2018, doi: 10.1016/j.procs.2018.10.301.

[10] S. Dadhich, U. Bodin, F. Sandin, and U. Andersson, "Machine learning approach to automatic bucket loading," in *Proc. 24th Medit. Conf. Control Autom. (MED)*, Jun. 2016, pp. 1260–1265, doi: 10.1109/MED.2016.7535925.

[11] Q. Zhu and Q.-F. Wang, "Real-time energy management controller design for a hybrid excavator using reinforcement learning," *J. Zhejiang Univ.-Sci. A*, vol. 18, no. 11, pp. 855–870, Nov. 2017, doi: 10.1631/jzus.A1600650.

[12] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2017. Accessed: Nov. 25, 2020, doi: 10.1109/iros.2017.8202133.

[13] J. Xu, T. Du, M. Foshey, B. Li, B. Zhu, A. Schulz, and W. Matusik, "Learning to fly?: Computational controller design for hybrid UAVs with reinforcement learning learning to fly?: Computational controller design for hybrid UAVs with reinforcement learning," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 1–12, Jul. 2019, doi: 10.1145/3306346.3322940.

[14] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. Van De Panne, "Iterative reinforcement learning based design of dynamic locomotion skills for cassie," 2019, *arXiv:1903.09537*. [Online]. Available: https://arxiv.org/abs/1903.09537

[15] P. Hämäläinen, A. Babadi, X. Ma, and J. Lehtinen, "PPO-CMA: Proximal policy optimization with covariance matrix adaptation," 2018, *arXiv:1810.02541*. [Online]. Available: http://arxiv.org/abs/1810.02541

[16] J. de García Jalón, E. Álvarez, F. A. de Ribera, I. Rodríguez, and F. J. Funes, "A fast and simple semi-recursive formulation for multi-rigid-body systems," in *Advances in Computational Multibody Systems*. Dordrecht, The Netherlands: Springer, 2005, pp. 1–23.

[17] A. Avello, J. M. Jiménez, E. Bayo, and J. G. de Jalón, "A simple and highly parallelizable method for real-time dynamic simulation based on velocity transformations," *Comput. Methods Appl. Mech. Eng.*, vol. 107, no. 3, pp. 313–339, Aug. 1993, doi: 10.1016/0045-7825(93)90072-6.
[18] J. G. de Jalón and E. Bayo, *Kinematic and Dynamic Simulation of Multibody Systems: The Real-Time Challenge*. New York, NY, USA: Springer, 1993.
[19] Y. Pan, W. Dai, Y. Xiong, S. Xiang, and A. Mikkola, "Tree-topology-oriented modeling for the real-time simulation of sedan vehicle dynamics using independent coordinates and the rod-removal technique," *Mechanism Mach. Theory*, vol. 143, Jan. 2020, Art. no. 103626, doi: 10.1016/j.mechmachtheory.2019.103626.
[20] J. Watton, *Fluid Power Systems?: Modeling, Simulation, Analog and Microcomputer Control*. Upper Saddle River, NJ, USA: Prentice-Hall, 1989.
[21] H. M. Handroos and M. J. Vilenius, "Flexible semi-empirical models for hydraulic flow control valves," *J. Mech. Design*, vol. 113, no. 3, pp. 232–238, Sep. 1991, doi: 10.1115/1.2912774.
[22] M. Pla-Castells, I. García, and R. J. Martínez, "Approximation of continuous media models for granular systems using cellular automata," in *Cellular Automata* (Lecture Notes in Computer Science), vol. 3305. Berlin, Germany: Springer, 2004, pp. 230–237, doi: 10.1007/978-3-540-30479-1_24.
[23] M. Pla-Castells, I. García-Fernández, and R. J. Martínez, "Interactive terrain simulation and force distribution models in sand piles," in *Cellular Automata*, vol. 4173. Berlin, Germany: Springer, 2006, pp. 392–401, doi: 10.1007/11861201_46.
[24] D. Holz, T. Beer, and T. Kuhlen, "Soil deformation models for real-time simulation: A hybrid approach," in *Proc. Vriphys 6th Workshop Virtual Real. Interact. Phys. Simulations*, Jan. 2009, pp. 21–30, doi: 10.2312/PE/vriphys/vriphys09/021-030.
[25] S. Jaiswal, P. Korkealaakso, R. Aman, J. Sopanen, and A. Mikkola, "Deformable terrain model for the real-time multibody simulation of a tractor with a hydraulically driven front-loader," *IEEE Access*, vol. 7, pp. 172694–172708, 2019, doi: 10.1109/ACCESS.2019.2956164.
[26] P. Hintjens, "ZeroMQ," O'Reilly Media, Sebastopol, CA, USA, Tech. Rep., 2013, vol. 53, no. 9.
[27] *Part 2: Kinds of RL Algorithms—Spinning Up documentation*. Accessed: May 29, 2020. [Online]. Available: https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html
[28] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*. [Online]. Available: http://arxiv.org/abs/1707.06347

ILYA KURINOV received the B.S. degree in mechanical engineering from the Saimaa University of Applied Sciences and the M.S. degree in mechatronics from the Lappeenranta University of Technology.

Since 2018, he has been working as a Ph.D. Researcher with the Machine Design Laboratory, LUT University. His research interests include application of multibody system dynamics and machine learning algorithms in automation of mechatronic machines.

GRZEGORZ ORZECHOWSKI received the B.S. and M.S. degrees and the Ph.D. degree in automation and robotics from the Warsaw University of Technology, Warsaw, Poland, in 2007 and 2012.

From 2012 to 2017, he was an Assistant Professor with the Division of Theory of Machines and Robots, Warsaw University of Technology. Since 2017, he has been a Postdoctoral Researcher with the Laboratory of Machine Design, LUT University, Lappeenranta, Finland. His research interests include the computational methods in mechanics, studies of the deformable models in challenging dynamical excitations, and reinforcement learning techniques in application to machine control.

PERTTU HÄMÄLÄINEN received the M.Sc. (Tech.) degree from the Helsinki University of Technology, in 2001, the M.A. degree in new media from the University of Art and Design, Helsinki, in 2002, and the Ph.D. degree in computer science from the Helsinki University of Technology, in 2007.

He is currently an Associate Professor with Aalto University. He has published widely on human–computer interaction, computer animation and movement synthesis, and game research.

AKI MIKKOLA received the Ph.D. degree in the field of machine design in 1997.

Since 2002, he has been working as a Professor with the Department of Mechanical Engineering, Lappeenranta University of Technology, Lappeenranta, Finland. He is currently leading the Research Team of the Laboratory of Machine Design, Lappeenranta University of Technology. His research interests include machine dynamics and vibration, multibody system dynamics, and bio-mechanics. He has been awarded five patents and has contributed to more than 90 peer-reviewed journal articles.