2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Towards RL-Based Hydraulic Excavator Automation

Pascal Egli[1] and Marco Hutter

***Abstract*— In this article we present a data-driven approach for automated arm control of a hydraulic excavator. Except for the link lengths of the excavator, our method does not require machine-specific knowledge nor gain tuning. Using data collected during operation of the excavator, we train a general purpose model to effectively represent the highly non-linear dynamics of the hydraulic actuation and joint linkage. Together with the link lengths a simulation is set up to train a neural network control policy for end-effector position tracking using reinforcement learning (RL). The control policy directly outputs the actuator commands that can be applied to the machine without unfounded filtering or modification. The proposed method is implemented and tested on a 12t hydraulic excavator, controlling its 4 main arm joints to track desired positions of the shovel in free-space. The results demonstrate the feasibility of directly applying control policies trained in simulation to the physical excavator for accurate and stable position tracking.**

I. Introduction

Hydraulic excavators are versatile machines, that are ubiquitous in construction, mining, agriculture or forestry. The automation of these machines can bring substantial economic and social benefits. The productivity can be improved by alleviating the problem of finding enough skilled and experienced human operators. Additionally, autonomous excavators can be used in hazardous environments and therefore, improve safety. Accurate control of hydraulic excavators can also give architects new design opportunities due to increased precision and the ability to create arbitrary shapes [1].

An important technology for the automation of hydraulic excavators is the accurate position control of the arm. Heavy hydraulic excavators exhibit strong non-linearities due to hydraulic coupling between the actuators, cylinder friction, control input dead-zones and delays [2]. Furthermore, machine properties vary a lot, not only in different operating and loading conditions, but also across different types of machines and even between machines of the same type due to manufacturing tolerances and wear. Also, cylinder attachment configurations are different depending on the machine. The corresponding linkage designs can be complex and proprietary to the manufacturer. Deriving the resulting joint angles is therefore not always trivial and requires physical access to precisely measure off the entire mechanism. This is in particular a problem for the Menzi Muck M545, which was used in this work to validate our approach (see fig. 1). It has a non-standard dipper joint configuration (see fig. 2) and the boom cylinder is inaccessible without dismantling parts of the machine because it is mounted underneath the cabin. These challenges motivate a data-driven control approach which inherently solves the aforementioned problems without requiring a precise analytical model of the system nor expert knowledge to tune control parameters. The proposed method is directly transferable and applicable to a broad range of machines.

This research was supported by the Swiss National Science Foundation through the National Centre of Competence in Digital Fabrication (NCCR dfab).

All Authors are with Robotic Systems Lab, ETH Zurich, Switzerland.
[1]pasegli@ethz.ch

Fig. 1. Menzi Muck M545, a 12t hydraulic walking excavator.

A. Related Work

Simple model-free PD controllers were proposed a long time ago for excavator control [3]. The problem of PD or PID controllers is that they are tuned for the nominal operating configuration and hence their performance deteriorates with increasing distance to the nominal configuration. Also, PID controllers are usually tuned for a particular engine speed and require the machine to be preheated. PID controllers were enhanced with feed-forward lookup tables [4]. Generating the lookup tables and tuning PID parameters is tedious and requires expert knowledge for the application on every machine.

Non-linear model-based control approaches have provided state-of-the-art performance for high-precision control of hydraulic manipulators [5]. These methods heavily rely on an analytical model of the system which is difficult and costly to obtain, hindering an easy transfer to another type of machine. Additionally, many of these methods were tested on systems with high-precision servo valves. Commercial hydraulic excavators however are usually equipped with a two-staged hydraulic system where only the proportional valves in the pilot stage can be controlled which introduces additional delays and bandwidth limitations. It is in general not feasible or desirable to retrofit a commercial excavator with servo valves due to the high cost of these valves.

To overcome the limitations of the above mentioned approaches, researchers have started looking into learning

methods that can deal with the complex dynamics of hydraulic excavators. In early research, Song and Koivo [6] proposed to learn the inverse dynamics of the excavator from operation data and to adapt the model online. The torque output of the model was then used as a feed-forward term for the PID controller. This approach, however, was only validated in simulation and neglects the fact that conventional excavators can usually not be torque controlled without expensive modifications to the hydraulic system [4], [7]. Cannon et al. [8] modeled the actuator dynamics with one neural network per cylinder. The actuator models were then used in conjunction with a soil-tool interaction model to select appropriate digging trajectories. Park et al. [9] proposed to learn an inverse model of the excavator online using echo-state networks (ESN) for position tracking. Their controller could adapt to varying operating conditions and improve its performance over time. However, the tracking error in the air remained large, even after a couple of repetitive motions. Dadhich et al. [10] trained a neural network based on data collected during expert operation of a wheel-loader for controlling the tilt and lift actuators during the bucket-filling phase. To refine the controller and adapt it to a different type of material they use online RL. Hodel [11] proposed to use RL for bucket leveling and compared the performance of different learning algorithms on a simplified excavator model in simulation, leaving the transfer to real-world an open problem.

B. Contribution

The main contribution of this work is a fully data-driven approach to synthesize a position tracking controller for a hydraulic excavator arm. Our approach only requires minimal machine-specific knowledge, namely the link lengths between the actuated joints, which facilitates the automation of any excavator. Machine-specific properties are incorporated into the actuator model which is trained based on measurements collected during operation of the physical machine. Using RL, we train a control policy which can deal with the non-linear dynamics of the system and directly outputs pilot stage valve current setpoints. We implement our method on a 12t hydraulic excavator controlling its 4 main arm joints to track a desired shovel contact position relative to the chassis (see fig. 3). We show that our controller successfully stabilizes the excavator even for large position reference steps. Our controller also shows good performance for tracking circular reference trajectories, even though it was not trained specifically for trajectory tracking. To the best of our knowledge, this is the first time that a control policy trained in simulation with RL is deployed on a full-size excavator.

II. System Description

Prior to explaining the details of our proposed method, we outline the system requirements of our approach. The following list contains the required modifications to an off-the-shelf excavator:

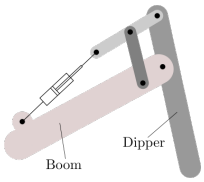

Fig. 2. Non-standard dipper joint linkage on the Menzi Muck M545.

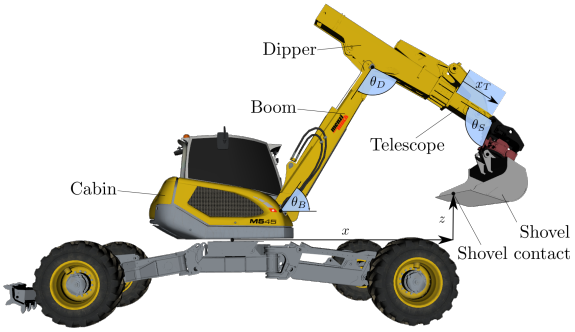

Fig. 3. Side view of the Menzi Muck M545 and its 4 main arm joints that are controlled to reach a desired shovel contact position relative to the cabin in task-space.

1) *Joint Angle/Displacement Measurement:* Our method relies on accurate measurements of the joint angles and positions for revolute and linear joints respectively. We obtain the cylinder displacements with a resolution of 0.01mm using Sick [12] draw wire sensors. The cylinder velocities are obtained by differentiating the position measurements. To obtain the joint states from the cylinder states we use the known geometry of the linkage mechanisms. It is important to note that joint states can also be directly obtained using aftermarket solutions like the Leica iCON iXE3 [13] that are simply mounted on the arm links such that knowledge about the geometric cylinder-joint relation is in general not required to apply our method.
2) *Link Lengths:* For position control in task-space we need in addition to the joint states also the distances between the joints, i.e. the link lengths. These can be directly measured on the machine. We control the position of the arm with respect to the forward kinematics computed based on link lengths and joint states. Therefore, absolute errors, e.g. due to link bending or play in the joints, are not considered.
3) *Electric Control Valves:* Electrically controllable pilot stage valves are needed for autonomous control of the excavator. Many modern excavators are equipped with steer-by-wire technology where no modification of the hydraulic system is required for controlling the valves remotely. The M545 used here does not feature steer-by-wire technology and is therefore retrofitted with Hawe [14] PMZ proportional pressure reducing valves in the pilot stage for autonomous and remote control.

III. Method

A. Overview

Fig. 4 shows an overview of our approach. First, an actuator model is trained based on input-output data collected

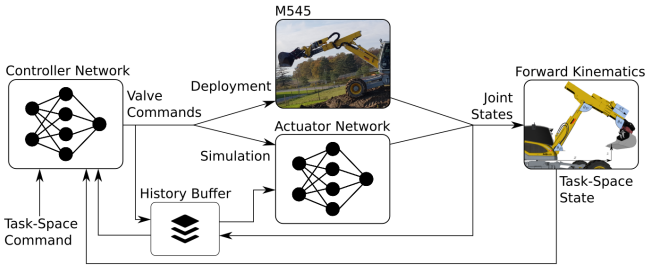

Fig. 4. Architecture overview. The actuator model is trained on input-output data collected on the machine. In simulation the actuator model replaces the physical machine to train a control policy with RL. The controller as well as the actuator model take a history of past states as inputs.

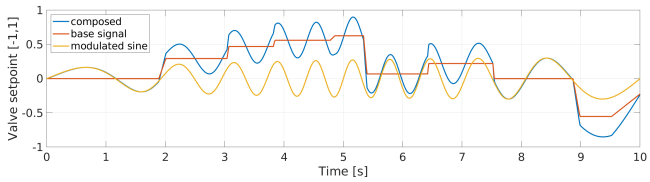

Fig. 5. Valve current setpoint command $u \in [-1, 1]$ (fully closed/open) for data collection (blue) consists of a base signal (red) that ensures coverage of the joint's motion range and a modulated sine (yellow, see eq. 1).

on the real machine in a supervised fashion. This actuator model is then used with fixed weights in simulation to synthesize a controller for task-space position tracking of the excavator's shovel contact position using RL. The final policy is then directly deployed on the physical excavator.

B. Data Collection

We collected input-output data during semi-autonomous operation of the machine through the electric pilot stage valves to model the dynamics of the actuators. Therefore, we designed a valve current setpoint signal that consists of two parts that are overlaid (see fig. 5): 1. a base signal, that ensures coverage of the whole motion range of the cylinders and 2. a frequency and amplitude modulated sine

$$s(t) = 0.3sin(2\pi 0.02t + \phi_1) \cdot sin(2\pi(t + 0.99sin(2\pi 0.1t) + \phi_2)) \quad (1)$$

with frequencies of up to approximately 2Hz, where ϕ_1 and ϕ_2 correspond to different phase shifts for each joint. The maximum frequency was chosen to be higher than the velocity control cut-off frequency which lies at 0.5Hz for the M545 [4]. To avoid collisions with the ground and self-collisions during data collection, the base signals for the boom and the dipper joints were commanded manually, using a gamepad joystick controller, whereas for the telescopic and shovel joints randomized ramp profiles were generated. We set the engine speed to a constant value and had the cabin of the excavator in a constant horizontal orientation. Data was collected during 2h at a rate of 100Hz.

C. Modeling the Hydraulic Actuation

We model the dynamics of the 4 hydraulic actuators with a simple multi-layer perceptron (MLP) with 3 hidden layers, each with 128 hidden nodes and ReLU non-linear activation

TABLE I

ACTUATOR MODEL INPUTS AND OUTPUTS.

Inputs	
Joint positions	q_t^j
Joint velocities	$\dot{q}_t^j, \dot{q}_{t-0.01s}^j, \dots, \dot{q}_{t-0.1s}^j$
Valve setpoints	$u_t^j, u_{t-0.03s}^j, \dots, u_{t-0.99s}^j$
Diesel engine rpm	R_t
Hydraulic oil temperature	T_t
Outputs	
Joint velocities	$\dot{q}_{t+0.01s}^j$

and a linear output layer. The actuator model predicts the joint velocities at the next time step, given the state at the current time step and a history of past states. Joint positions are obtained through forward integration in time.

Table I lists the inputs and outputs of the actuator model. We use one network to model all the 4 actuators to account for hydraulic coupling effects caused by all the 4 joints being supplied by the same accumulator and pump. In addition to the current velocities, we provide the model with a history of past joint velocities to allow the network to average over the effect of stiction friction and the discretization resulting from the differentiation of the joint position measurement. To account for input response delays, which depending on the actuator can be as high as 0.7s, we provide the actuator model with a history of the input commands over the past 0.99s [10], [15]. Since the commands do not change rapidly, we sample the history sparsely with a stride of 0.03s, to reduce the input dimensionality of the actuator model. Even though we set the engine speed to a constant value during data collection, we observe momentary drops in engine rpm under heavy load, i.e. when moving many actuators fast at the same time. Therefore, we also include the engine rpm as input to the actuator model. Since the behavior of the hydraulic actuators also varies depending on the temperature of the hydraulic oil, it is also included as an input. To speed up the training of the network, we normalize all the inputs to a mean of 0 and standard deviation of 1. We found, that the quality of the model improved a lot, when training for the difference in joint velocities between two time steps, rather than for the absolute velocities at the next time step. We train the actuator model in a supervised fashion using the Adam optimizer [16] and the mean-squared error (MSE) loss for 20'000 epochs with a single batch, which takes around 10h[1].

Since the actuators move relatively slowly and we use the current velocities as input to the model to predict the velocities at the next time step, there is a risk that the model would only learn an identity mapping of the current velocities. Table II shows the root-mean-squared errors (RSME) in joint velocities indicating that the model does indeed perform better than just an identity mapping. However, interpreting the quality of the model based on these numbers is difficult. In a simulated environment, which we intend to use to train a control policy, no ground truth is available and the model needs to be rolled forward in time based on its previous

[1]We use a PC with an AMD Ryzen9 3950x CPU (@4.0GHz), 32GB of RAM and a Nvidia RTX 2080s GPU.

TABLE II

ACTUATOR MODEL ERRORS.

RMSE $10^{-3}[\text{rad}, \text{m}]\text{s}^{-1}$	**Boom**	**Dipper**	**Telescope**	**Shovel**
Identity mapping	3.58	4.00	4.64	11.36
Training 90%	2.21	2.06	2.36	7.06
Validation 10%	2.28	2.10	2.44	7.96

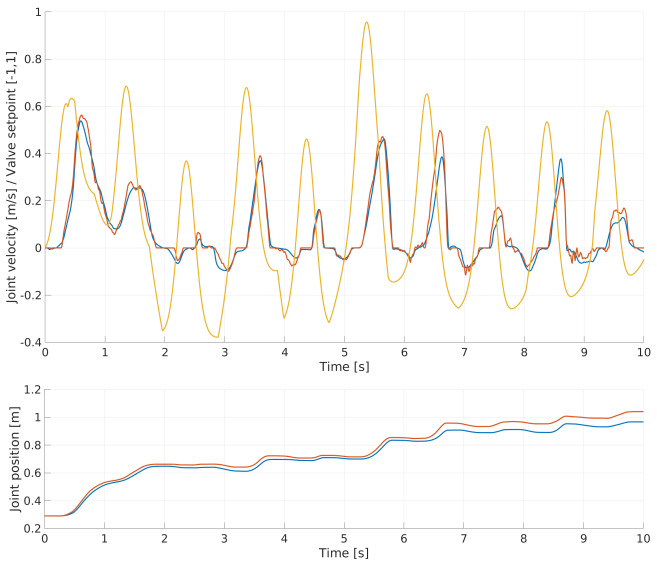

Fig. 6. Actuator model performance with joint velocities and inputs (top) and positions (bottom). Starting from the same state, the output of the model is compared to the measurements on the machine. The same inputs are applied to the model and the machine, the model however is rolled forward based on its own predicted states.

outputs. Therefore, we can only qualitatively compare model rollouts to the behavior measured on the machine by applying the same sequence of input commands. Fig. 6 shows the output of the model for the telescopic joint. The model output matches the measured velocities accurately and the accumulated error is small even over a large range of motion. By providing a sequence of past inputs and velocities, the model is able to capture input delays accurately.

D. Learning a Position Tracking Controller

To synthesize a task-space position tracking controller that is able to deal with the non-linear behavior of the actuators, we use RL. The problem basically consists of learning the inverse kinematics of the excavator arm and an inversion of the actuator model. Inverting the actuator model can not be done analytically due to the causality of the inputs to the actuator network, hence we use RL to learn the inversion.

TABLE III

CONTROLLER INPUTS AND OUTPUTS.

Inputs	
Joint positions	q_t^j
Joint velocities	$\dot{q}_t^j, \dot{q}_{t-0.1s}^j, \dot{q}_{t-0.2s}^j$
Valve setpoints	$u_{t-0.1s}^j, \ldots, u_{t-1.0s}^j$
Diesel engine rpm	R_t
Hydraulic oil temperature	T_t
Current shovel position	$\chi_t \in \mathbb{R}^2$
Current shovel velocity	$\dot{\chi}_t \in \mathbb{R}^2$
Desired shovel position	$\chi_t^* \in \mathbb{R}^2$
Outputs	
Valve setpoints	u_t^j

We model the problem as a discrete-time Markov Decision Process (MDP). At every time step t, the agent receives an observation $o_t \in \mathcal{O}$, takes an action $a_t \in \mathcal{A}$ and receives a scalar reward $r_t \in \mathcal{R} : \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$. The agent acts according to a stochastic policy $\pi(a_t|o_t, o_{t-1}, \ldots, o_{t-h})$, conditioned on the current and past observations. The objective of the agent is to find a policy that maximizes the cumulative discounted reward $\mathbb{E}\big[\sum_{k=t}^{\infty} \gamma_{t+k}^k\big]$ through interactions with the environment, where $\gamma \in (0, 1)$ is the discount factor, trading off between current and future rewards.

To train the control policy, we chose Trust Region Policy Optimization (TRPO) [17], an actor-critic RL algorithm, with Generalized Advantage Estimation (GAE) [18] using the default hyper-parameters. We use two separate neural networks for approximating the policy and value functions. Each network consists of an MLP with two layers with 128 hidden nodes with ReLU activation and a linear output layer. The observations and actions are listed in table III. The controller is conditioned on the command, which consists of a desired shovel contact position in task-space relative to the cabin. Since we only control the 4 main arm joints, the motion is restricted to two dimensions. Additionally, the controller inputs consist of the same inputs provided to the actuator model and the current end-effector position and velocity.

We train the control policy on samples from simulated episodes of 10s, initializing the excavator in a random configuration and sampling a valid random goal position. For every episode we sample a hydraulic oil temperature and an engine rpm uniformly from the range observed during data collection. An episode terminates, if joint limits are violated or the arm collides with itself or the ground and the agent receives -1.0 reward. The agent updates its actions at 10Hz and the actuator model is evaluated at 100Hz. We use a discount factor of 0.99 which corresponds of a half-life of 6.9s.

The reward function is defined as follows:

$$r_t = r_t^g + r_t^u + r_t^r, \tag{2}$$

where

$$\begin{aligned} r_t^g &= 0.005(0.5 + \exp\left(-k_{c,1}\|\chi_t^* - \chi_t\|_2\right)) \\ r_t^u &= (-0.0005)k_{c,2}\|u_t\|_2 \\ r_t^r &= (-0.025)k_{c,2}\|u_t - u_{t-1}\|_2. \end{aligned}$$

The first term (r_t^g) drives the agent to the goal position and the subsequent terms (r_t^u, r_t^r) penalize large controller inputs and large changes in control inputs between two time steps. Especially the last term (r_t^r) is essential for smooth control outputs. We found that adding a constant positive reward at all the time steps improved the learning performance, because it discourages the agent to terminate an episode early. $k_{c,1}$ and $k_{c,2}$ are curriculum factors which make the learning task easier at the beginning of the training to accelerate conversion [19]. They are increased linearly from 1 to 10 during 200 and 400 algorithm iterations respectively.

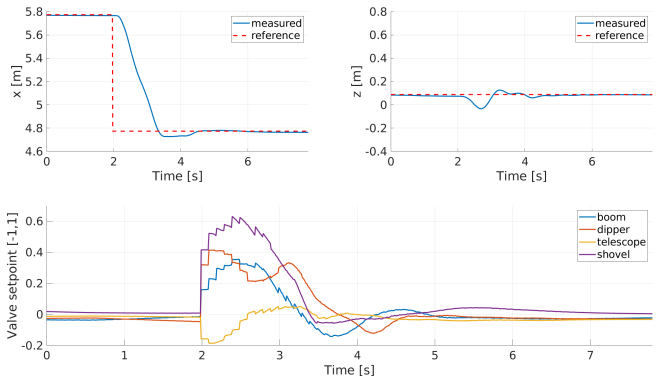

Fig. 7. Shovel position reference step in x direction (top) and corresponding valve setpoints (bottom) relative to the cabin.

TABLE IV

TRAJECTORY TRACKING PERFORMANCE IN CARTESIAN SPACE.

	$\|e\|_{avg}$[cm]	$\|e\|_{max}$[cm]	$\|v\|_{max}$[cm s^{-1}]	ρ[s]
Learning	7.8	13.5	22.3	0.61
IK	1.4	3.6	25.8	0.14

To account for noisy measurements on the machine and inaccuracies in the actuation, we add uniformly sampled white noise with mean 0 to actions and observations during training. The maximum amplitudes of the additive noise are $[0.04, 0.05, 0.034, 0.08]$ rad and m for position observations, $[0.04]_4$ rad s^{-1} and m s^{-1} for velocities, $[50]$ for engine rpm, $[0.02]_2$ m for shovel contact positions, $[0.1]_2$ m s^{-1} for shovel contact velocities and $[0.075]_4$ for inputs.

Training of the control policy takes only about 1h[1] using the *TensorFlow* [2] C/C++ API.

IV. EXPERIMENTAL RESULTS

We deploy the controller trained in simulation directly on the machine without modification of its outputs. The controller runs at 100Hz commanding directly electric pilot stage valve current setpoints. We found, that deploying the controller on the machine faster (100Hz) than during training (10Hz) increased the controller performance substantially. We assume that this is due to an increased difficulty of the task with lower control rate during training.

As a first validation of our controller we test it in the same scenario that was use during training in simulation, i.e. by providing position reference steps in free-space. Fig. 7 (top) shows the end-effector position during a position reference step of 1m in x direction relative to the cabin and the corresponding valve current setpoints (bottom).

Since we do not have a controller that can follow such large step references without going unstable, we compare it in a second set of experiments to a hierarchical optimization Inverse Kinematic (IK) trajectory controller [4]. The IK controller relies on manually tuned velocity PID controllers on cylinder level, including a lookup table containing feed-forward valve current setpoints given desired joint velocities and requires the geometric cylinder-joint conversion. We test the controllers on circular trajectories in different parts of

[2]https://github.com/leggedrobotics/tensorflow-cpp

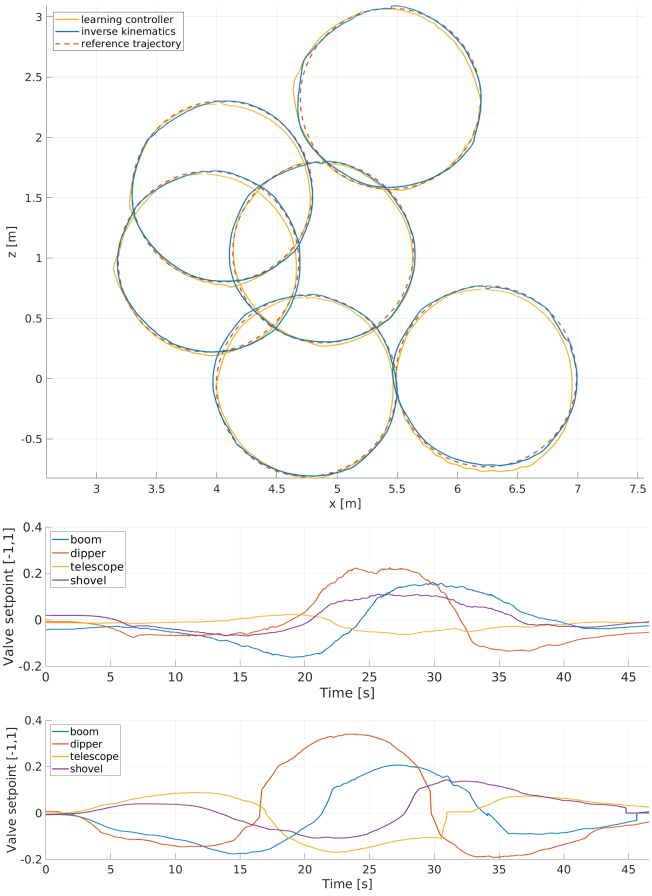

Fig. 8. Circular trajectory tracking in free-space (top). Distances are measured relative to the cabin position. Valve current setpoints generated by the learned controller (middle) and for the IK controller (bottom) for the circular trajectory starting at x=4.75m, z=0.69m.

the excavator's workspace (see fig. 8, top), which requires coordinated motions of all the 4 joints as well as cylinder velocity direction changes. We provide the learned controller with a position command that is updated at 100Hz. To ensure smooth motions, we use a fifth-order polynomial spline with 0 velocity and acceleration at start and end to scale the trajectories in time, an average shovel contact velocity of 0.1m s^{-1} and a maximum velocity of 0.1875 m s^{-1}. The IK controller was additionally provided with velocities extracted from the spline. Fig. 8 (middle, bottom) shows the valve current setpoints for the learned and the IK controller on one of the circular trajectories. The sharp step in the IK command for the telescopic joint at 31s is caused by arriving at the joint limit. The learned controller avoids joints limits because it was encouraged to do so by terminating training episodes when violating joint limits.

The trajectory tracking performance is summarized in table IV. The average position tracking error in Cartesian space is indicated by $|e|_{avg}$. To being able to compare the performance of our proposed method to other control approaches and set a benchmark for further developments in RL-based control of hydraulic manipulators we compute the normalizing performance indicator ρ [5], which consists of the ratio between the maximum tracking error $|e|_{max}$ and

the maximum velocity $|v|_{max}$ in Cartesian space, averaged over the 6 circular trajectories.

An excerpt of the data collection procedure as well as the position step reference and circle tracking experiments can be found in the supplementary video[3].

V. Discussion and Outlook

The results demonstrate the feasibility of directly applying a control policy trained purely in simulation to the physical excavator for accurate position tracking. The advantages of our approach are that no gain tuning on the machine is required and knowledge about the cylinder-joint geometry is in general not needed which facilitates the automation of any type of excavator. Our controller takes into account the non-linear actuator dynamics including delays, dead-zones, temperature and engine rpm dependencies through data collected on the machine and uses standard proportional valves in the pilot stage. Even for large position reference steps our controller successfully stabilizes the excavator. Even though our data-driven controller was not particularly trained for trajectory tracking it can successfully track arbitrary trajectories by continuously updating the position target. Compared to an IK trajectory tracking controller which is specifically optimized for this machine and task the learned controller performs expectedly worse, because it needs a certain offset to the commanded position in order to move as it does not take a reference velocity as command input.

Given that the presented results are achieved with only 2h of data collected on the physical machine, the performance of our controller can be enhanced, when more data is available. This can be done in a Dyna setup [20], by improving the actuator model with new data and continue training of the controller based on the updated actuator model. Or, another possibility is to update the control policy directly using RL as proposed by Dadhich [10].

In future work we will extend our controller to consider also reference velocities and we plan on including an orientation reference. Currently, we do not control the cabin turn actuator, limiting the arm motions to two dimensions. The actuator model can be extended and trained with data including the turn actuator. It is also possible to learn an individual model for the turn actuator. The required cabin turn position to reach an arbitrary position in three dimensions with the shovel is uniquely defined and therefore, the arm and the cabin turn can be commanded individually. Another limitation of our implementation is that we do assume the cabin to be in a horizontal orientation. This is a valid assumption for many excavators, however, in particular for the M545 which is intended to be used in rough and steep terrain, the assumption does not hold anymore. Since the load on the actuators due to gravity varies with the orientation of the cabin, the state spaces of the actuator model and the controller need to be augmented with the direction of the gravity, which can be easily measured with an Inertial Measurement Unit (IMU). In this work we only consider free-space motions without soil interaction. Our approach can be extended to also take into account soil interactions by collecting data during digging operation and measure the load on the actuators. Measuring the load can be achieved by installing two pressure sensors per cylinder. A digging controller could be trained in simulation with RL, using a simplified soil-tool interaction model that is computationally cheap. Soil-tool interaction models exist, but they are usually inaccurate (20-30% error) or are computationally expensive which hinders its usage for RL [21]. However, Hwangbo et al. [15] have shown that dynamics randomization in simulation during training can compensate for up to 20% modelling error.

[3] https://youtu.be/MDP96pqhYHc

References

[1] I. Hurkxkens, C. Girot, and M. Hutter, "Robotic landscapes. developing computational design tools towards autonomous terrain modeling," 2017.

[2] S. Dadhich, U. Bodin, and U. Andersson, "Key challenges in automation of earth-moving machines," *Automation in Construction*, vol. 68, 05 2016.

[3] A. Koivo, M. Thoma, E. Kocaoglan, and J. Andrade Cetto, "Modeling and control of excavator dynamics during digging operation," *Journal of Aerospace Engineering*, vol. 9, pp. 10–18, 01 1996.

[4] D. Jud, P. Leemann, S. Kerscher, and M. Hutter, "Autonomous free-form trenching using a walking excavator," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3208–3215, Oct 2019.

[5] J. Mattila, J. Koivumäki, D. G. Caldwell, and C. Semini, "A survey on control of hydraulic robotic manipulators with projection to future trends," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 2, pp. 669–680, 2017.

[6] Bumjin Song and A. J. Koivo, "Neural adaptive control of excavators," in *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, vol. 1, Aug 1995, pp. 162–167 vol.1.

[7] Quang Ha, M. Santos, Quang Nguyen, D. Rye, and H. Durrant-Whyte, "Robotic excavation in construction automation," *IEEE Robotics Automation Magazine*, vol. 9, no. 1, pp. 20–28, March 2002.

[8] H. Cannon and S. Singh, "Models for automated earthmoving," *Experimental Robotics VI, Lecture Notes in Control and Information Science*, 09 1999.

[9] J. Park, B. Lee, S. Kang, P. Y. Kim, and H. J. Kim, "Online learning control of hydraulic excavators based on echo-state networks," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 1, pp. 249–259, Jan 2017.

[10] S. Dadhich, "Automation of wheel-loaders," *PhD dissertation, Luleå*, 2018.

[11] B. J. Hodel, "Learning to operate an excavator via policy optimization," in *Procedia Computer Science 140*, 2018, pp. 376–382.

[12] "SICK Sensor Intelligence," https://www.sick.com/, accessed: 2020-02-01.

[13] "Leica Geosystems," https://leica-geosystems.com/products/machine-control-systems/excavator/leica-icon-ixe3—3d-system, accessed: 2020-02-01.

[14] "HAWE Hydraulik," https://www.hawe.com/, accessed: 2020-02-01.

[15] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, 2019.

[16] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.

[17] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," *CoRR*, vol. abs/1502.05477, 2015.

[18] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 06 2015.

[19] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," vol. 60, 01 2009, p. 6.

[20] R. S. Sutton, "Dyna, an integrated architecture for learning, planning, and reacting," *SIGART Bull.*, vol. 2, no. 4, pp. 160–163, Jul. 1991.

[21] V. Chacko, H. Yu, S. Cang, and L. Vladareanu, "State of the art in excavators [this project does not form a part of fusion]," 08 2014.