Machine Learning for Process Automation of Agricultural Machines in Field Applications

Karam Daaboul[1], Simon Becker[1], Kevin Daiß[1], Karl Kurzer[1], Marcus Geimer[1] and J. Marius Zöllner[1,2] [1]

Abstract

Modern machines for agricultural field applications, such as tractors with different implements, offer a vast variety of static as well as dynamic configuration parameters that must be managed by the operator during field operation. To operate these machines efficiently and achieve the desired results, experience is required. In order to facilitate the operation of these complex machines for less experienced users, the capabilities of machine learning are being researched to control the machine and hence increase its level of automation, easing the cognitive load for the operator. The use of model-free reinforcement learning (RL) enables us to find the optimal parameter configuration for a given machine state, with respect to a operator defined performance metric. The use of RL enables us to be learn directly from measurement data without requiring a detailed model of this complex system. An initialization of the system is learned from behavior that has previously been recorded (Imitation Learning), based on this initialization the system improves itself by exploring untried actions in the real world. The evaluation is conducted on a real tractor during field operation.

1. Introduction

Tractors are used to handle all kind of agricultural tasks. In combination with implements a high number of tasks from transportation to soil tillage can be accomplished. Every implement offers different settings and parameters that need to be adjusted according to environmental conditions and agricultural needs. Together with the capabilities and limitations of the tractor most agricultural tasks can result in complex systems.

This complexity makes it difficult for drivers to operate the machine optimally and makes it hard for engineers to model the tasks and thereby to control them. Since modern agricultural machines are equipped with a variety of different sensors, a data driven approach for the optimal control problem can be used avoiding the model complexity, leading to more accurate and faster decision making.

Machine learning (ML) is the study of computer algorithms that allow computer programs to automatically improve through experience[1]. In the last years ML is being applied to more and more fields including, for example, medicine, finance, robotics and agriculture [2]. In contrast to supervised learning, primarily used for classification and regression tasks, and unsupervised learning, used for tasks such as clustering and anomaly detection, reinforcement learning has emerged as the go to method to solve sequential decision making problems e.g. optimal control. In the past years RL algorithms reached major breakthroughs in the area of game playing [3, 4] and robotic control[5, 6].

Our main contribution is a machine learning framework that leverages model free RL to optimize for agricultural tasks. The evalautin of our framework shows its ability to optimize either the efficiency or the performance (reduction of fuel consumption, or reduction of time) of the ploughing task. To our knowledge, this is the first work apply an RL method to the task of optimal control for tasks such as ploughing.

2. Related Work

Previous approaches focused on modelling the complex dynamics of the tractor. The simulation based project Organic Computing in Off-highway machine [7] used a simulation model to analyze the machine state and Organic Computing to derive actions that define the new machine state.
A concept for the optimization of the rotary harrow is presented in [8]. The concept is based on recognizing the quality of the work results with a stereo camera. A point cloud is generated and the roughness profile is determined on the basis of this cloud. The process is controlled on the basis of this roughness profile. The controller adjusts driving and power takeoff (PTO) speed.
Reinforcement learning (RL) has successfully trained computer programs to play games at a higher level than the world's best human players [9]. Applications of RL in high-dimensional control problems, like robotics and driverless cars, are the subject of current research [6, 10]. In

3. Approach

In this section we describe the sytem that we try to optimize and define the problem in reinforcement learning terms and present the technical solutions we adopted.

3.1. System to be Optimized

Ploughing serves as an exemplary task. On the one hand, this process is characterized by the fact that it demands a lot of experience from the driver in order to find suitable process parameters. On the other hand, ploughing requires a high tractive effort, which results in a large fuel saving potential.
A Fendt 516 Vario tractor with continuous variable transmission and a Lemken Juwel 8 5+1 furrows plough with 1.60 m to 2.70 m variable working width are

Figure 1: Machine combination used

used for the research. The tractor is solely equipped with series production sensors. No modifications were made to the plough. The computer on which the control system runs is located on the tractor. Communication between tractor and computer is illustrated in figure 2. The computer communicates with the tractor via a CAN interface. Machine parameters can be read and written via the CAN interface.
The machine combination as described above has two kinds of parameters. There are parameters that have to be retrieved only once for the specific tractor implement combination, e.g. the front furrow width, the pulling point and the height of stabilizer wheel. These parameters are usually adjusted once before the work starts. So dynamically changing these parameters is not of interest. The other group of parameters are parameters that have to be adjusted continuously during field operation. For the process considered here these parameters are the tractor velocity over ground as well as rear linkage height. The tractor velocity's on the one hand directly corresponds to its working speed, on the other hand it influences the tractor's slippage between tires and ground, and hence efficiency. The parameter rear linkage height is prespecified by the plants, need but it can be adjusted locally at a certain level in situations where this is necessary. These situations occur in regions of the field with rapidly changing conditions, e. g. because they are very wet. Here the tractor could get stuck if the rear linkage height would not be adjusted. Thus, these parameters have to be adjusted continuously during work as conditions change. These conditions depend on machine parameters and on environment parameters like weather and soil quality. Especially the soil quality can change many times even on small fields. Here an automatic control system can react with a much higher frequency than a human, managing all the different conditions and parameters.

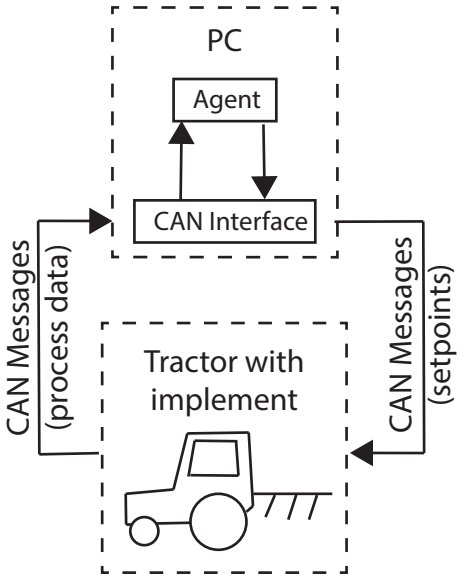

Figure 2: Machine setup

3.2. Problem Formulation

The problem of finding the optimal parameters for the tractor is formulated as a Markov Decision Problem (MDP) [11]. For this specific problem the MDP is modeled with:

- $\mathcal{S}$ is the state space of the tractor.
- $\mathcal{A}$ is the action space of the tractor.
- $p : S \times A \to P(S)$ is a transition function, which for every pair $(s, a) \in S \times A$ assigns a probability distribution $p(s_{t+1}|s_t, a_t)$ representing the probability of entering a state s_{t+1} from state s_t using action a_t.
- $R : S \times S \times A \to R$ is a reward function, which describes the reward $R(s_{t+1}, s_t, a_t)$ associated with entering state s_{t+1} from state s_t using action a_t.
- $\gamma \in [0, 1]$ is a discount rate parameter that controls the effect of future rewards on the current state.

3.2.1. Reinforcement Learning

The goal of reinforcement learning is to discover an optimal policy $\pi^*(s)$ that maps states to actions so as to maximize the expected reward R. Q-learning is an off-policy model-free reinforcement learning algorithm that can find $\pi^*(s)$ by approximating the Action-Value Function $Q^\pi(s, a)$. $Q^\pi(s, a)$ gives the expected return if you start in state s, take an action a, and then forever

after act according to policy π. The optimal policy can directly obtained if we have Q^*,

$$\pi^*(s) = \arg\max_a Q^*(s, a) \tag{1}$$

In the deep Q network algorithm (DQN) a neural network was used to approximate $Q^\pi(s, a)$ [12].

3.2.2. State Space

The key for defining the state space is the definition of the observations O_t that the algorithm receives at each time step. The state space consists of parameters read from the tractor's CAN and machine characteristics (such as the working width of the plough). The following parameters are used:

- engine speed n (min^{-1}): engine ECU
- engine torque T (N m): engine ECU
- gear ratio i: tractor ECU
- fuel consumption per time C (L ha): engine ECU
- rear linkage forces F (N): load cells via tractor ECU
- velocity v: ($\mathrm{km\,h}^{-1}$): radar sensor via tractor ECU
- wheel velocity v_w ($\mathrm{km\,h}^{-1}$): calculated with engine speed and gear ratio via tractor ECU
- slip ratio s: calculated using velocity and desired velocity

$$s = \frac{|v_w - v|}{v_w \cdot \eta} \tag{2}$$

  where η is the wheel correction

- working width of plough w (m): implement specific
- rear linkage height h: tractor ECU

Since the measurements had different frequencies, we implemented synchronization to synchronize them.

3.2.3. Action Space

The set of actions which can be used to control the tractor are: the wheel velocity, the rear linkage height and the engine speed. We chose the wheel velocity to control the system, because for tractors with continuous variable transmission the wheel velocity is the parameter that the driver has to adjust while driving. Using one of three discrete actions, decelerate, no change in velocity and accelerate the system can find the ideal velocity for a given state. The action is applied for a constant time step Δt. Since the acceleration of

Decelerate	No change in velocity	Accelerate
$\Delta v = -0.2\,\mathrm{km\,h^{-1}}$	$\Delta v = 0\,\mathrm{km\,h^{-1}}$	$\Delta v = 0.4\,\mathrm{km\,h^{-1}}$

Table 1: Action space used by the DQN to control the tractor

a tractor with implement is a harder action than the deceleration, we chose different values for the actions deceleration and acceleration, see table 1.

For safety reasons the system was only allowed to change its velocity in a range $\mathcal{V}$ predefined by an expert (hard constraint).

3.2.4. Reward Function

The reward function rewards the agent for its actions. So it defines the optimization goal for the system. Therefore the definition of a reward function is a critical task. We trained our agent for two different reward functions "efficient" and "performant". The first one was to drive efficiently with the goal to minimize fuel consumption per area

$$R_{\text{efficient}}(s_{t+1}, s_t, a_t) = -\omega_1 \cdot \frac{C}{v \cdot w} \tag{3}$$

and the second one was to optimize the time to complete the task

$$R_{\text{performant}}(s_{t+1}, s_t, a_t) = \omega_2 \cdot v \cdot w \tag{4}$$

where ω_1 and ω_2 are the reward weights.
To improve the robustness of the system, we also used a soft constraint to train the policy to change the velocity in a range $\mathcal{V}$ predefined by an expert.

$$R_{\text{safety}}(s_{t+1}, s_t, a_t) = -2 \quad if \quad s_{t+1} \notin \mathcal{V} \tag{5}$$

3.2.5. Network

To approximate the Action-Value Function $Q^{\pi}(s, a)$ we used a feedforward neural network. The input layer of our model consists of ten units, which is equal to the dimension of the state space, it also has three hidden layers, with 128, 128 and 64 units, and an output layer with three units, which corresponds to the size of the action space.

3.2.6. Training

Starting training directly from an initial policy that knows nothing about the task and the environment will damage the tractor because the policy has to explore many undesirable actions. Therfore we first trained the policy offline to imitate a human driver. In offline training the system can only learn from the actions that are in the recorded data and can not explore new actions.
The next step was the online training of the policy on the task. So the policy controlled the machine and received immediate rewards as feedback for the actions taken. To optimize the policy, we gave it the ability to explore the parameter space by testing new actions that had not been used in similar states

before.
We trained the policy online several times and evaluated the resulting policy between training sessions. If the collected rewards from the new policy were better or as good as the collected rewards from the old policy, then we saved the new policy, if not, we discarded it and continued training with the old policy again.

4. Experiments

Experiments were conducted both for training, evaluation, as well as testing. As it is important to eliminate undesired and unrecorded distractions as far as possible, mechanical adjustments of the plough are only done once before starting the tests. The upper link is hooked into the long hole in order to prevent tensions between plough and tractor when driving on crooked ground. As usual in ploughing the tractor steers itself in the plough furrow.
As a non-deterministic algorithm under development controls a potentially dangerous machine the following safety precautions were enforced:

- The agent is only allowed to adjust the machine parameters in a predefined range.
- There is always a driver on board that is able to control the machine if necessary.
- The agent is only able to write parameters on the machine after it was given dedicated write access. Write access has to be given over both a hardware and a software button. Write access can always be withdrawn over a hardware button.
- If all software safety features fail, the machine can always be stopped using its mechanical brake and clutch.

4.1. Framework

The necessary software is implemented using the ROS framework [13]. One ROS program converts CAN to ROS messages. The sub programs of the agent are also implemented in ROS and communicate with each other using ROS messages. This modular implementation makes testing different algorithms easy. We borrowed from OpenAi Gym [14] to define our agent, see figure 2. We used Tensorflow [15] to implement our network and OpenAi Baselines [16] to applay the Q-learning algorithm.

4.2. Scenarios

The system is only active during ploughing itself. This means the system starts controlling the tractor when the plough is lowered and stops controlling the tractor when the plough is raised. The system is inactive while the machine is turning between the rows.

5. Evaluation

In this chapter the function of the algorithm is evaluated by conducting a test run. Models with the two different reward functions "efficient" and "performant" are evaluated. Additionally a row driven by a human is used for comparison.

For initialization the agents were trained offline using approximately 2 hectare of ploughing. Then each agent was trained online using four rows of 100m length on the test field. Figure 3 shows the velocity over distance for three rows from the evaluation field. The "Reference" row was driven conventionally, the two other rows were driven by the system.
It is noticeable that both agents start at the same speed and behave very similarly in the first thirty meters, although they were trained with different reward functions. This is because the system takes time to set the appropriate speed. After this adjustment phase, however, the agent optimized for high speed reaches a higher speed than the human driver. This can also be seen in table 2, where the right column shows the average speed in the range after the first thirty meters. A similar behavior can be seen in figure 4 for the fuel consumption per area.

Since the tractor has started its work without speed and the guideline could only accelerate it with a maximum acceleration of 0.4 at each timestep, the tractor needs about thirty meters to reach the omptimal machine setting.

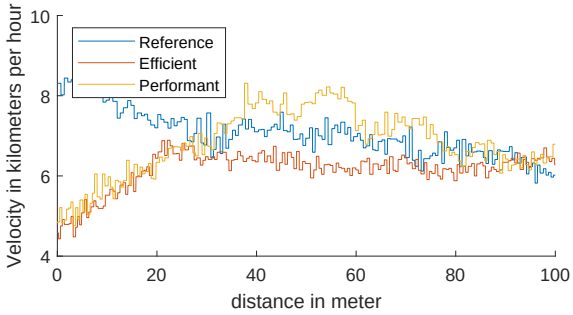

Figure 3: Velocity over distance

	Complete Distance	Begin after 30 meters
Reference	7.03	6.77
Efficient	6.11	6.29
Performant	6.72	7.12

Table 2: Mean velocity in kilometers per hour

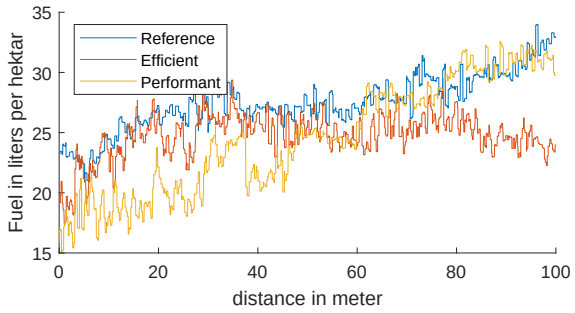

Figure 4: Fuel consumption over distance

	Complete Distance	Begin after 30 meters
Reference	27.69	28.60
Efficient	24.68	25.29
Performant	24.34	27.10

Table 3: Mean fuel consumption per area in liters per hektar

6. Conclusions

We introduced a new framework based on a model-free RL algorithm that is suitable for learning a policy to optimize a task of a tractor with an implement, either by minimizing time or by minimizing fuel consumption per area. As can be seen in the evaluation section, the policy is able to achieve the goals defined in the reward function. Our framework trained first offline to imitate a human driver, then online to further optimize itself. Hard and soft constraints are used to increase the safety of the system during training and testing. The next step would be to apply policies with continuous action spaces to avoid the time required to achieve the optimal machine setting and to use other methods from the safe reinforcement learning research area to improve the safety of the system during the training.

7. Acknowledgements

We wish to thank the Association for the Promotion of Teaching an Research of Mobile Machines (MOBIMA e.V.) for funding the project Trainable control systems for mobile machines within which the research leading to this contribution was conducted. Furthermore we wish to thank AGCO/Fendt for providing us the test machine and Lemken for supporting us with the test plough. For precise navigation we wish to thank geo-concept and the Landesamt fuer Geoinformation und Landentwicklung Baden-Wuerttemberg.

[1] T. Mitchell, *Machine Learning*, ser. McGraw-Hill International Editions. McGraw-Hill, 1997. [Online]. Available: https://books.google.de/books?id=EoYBngEACAAJ

[2] K. Liakos, P. Busato, D. Moshou, S. Pearson, and D. D. Bochtis, "Machine learning in agriculture: A review," in *Sensors*, 2018.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," dec 2013. [Online]. Available: http://arxiv.org/abs/1312.5602

[4] D. Ha and J. Schmidhuber, "World models," *CoRR*, vol. abs/1803.10122, 2018. [Online]. Available: http://arxiv.org/abs/1803.10122

[5] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. W. Pachocki, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, "Learning dexterous in-hand manipulation," *CoRR*, vol. abs/1808.00177, 2018. [Online]. Available: http://arxiv.org/abs/1808.00177

[6] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *CoRR*, vol. abs/1603.02199, 2016. [Online]. Available: http://arxiv.org/abs/1603.02199

[7] T. Kautzmann, M. Geimer, M. Wünsche, and S. Mostaghim, "Holistic Optimization of Tractor Management Organic Computing in Off-highway Machines," 2010.

[8] P. R. Nurscher, J. Karner, J. Huber, G. Moitzi, H. Wagentristl, M. Hofinger, and H. Prankl, "A system for online control of a rotary harrow using soil roughness detection based on stereo vision," 2017.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, feb 2015. [Online]. Available: http://www.nature.com/doifinder/10.1038/nature14236

[10] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah, "Learning to drive in a day," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2019.

[11] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013. [Online]. Available: http://arxiv.org/abs/1312.5602

[13] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[14] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[15] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[16] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov, "Openai baselines," https://github.com/openai/baselines, 2017.