agricultural engineering.eu 80(1), 2025, 73–103

DOI:10.15150/ae.2025.3331

agricultural
engineering.eu

Organic Computing for Intelligent Agricultural Technology: Perspective and Case Study

Anthony Stein, Jonas Boysen

Organic Computing (OC) refers to a systems engineering paradigm for designing intelligent technical systems which are deployed in complex real-world environments. With a focus on nature-inspired mechanisms, OC aims at the transfer of principles observable in nature to complex technical systems to endow them with lifelike qualities such as intelligence, flexibility, robustness, and the ability to self-organize. Agricultural engineering develops highly automated and autonomous technology for sustainable agricultural production which must work reliably in dynamic real-world environments. Higher degrees of systems' autonomy and intelligence require an increasing number of information technology components to interact over various system levels what in turn increases system complexity. In this article, we cast light on the suitability of the OC paradigm for designing Intelligent Agricultural Technology systems. In a case study, we adopt an OC approach for the optimization of secondary tillage and report on promising empirical results from integrating contemporary artificial intelligence methods into a multi-level system architecture to achieve an intelligent tractor control.

Keywords
Digital Agriculture, Intelligent Systems, Observer/Controller Architecture, Secondary Tillage

The scientific discipline of agricultural engineering is the study and application of engineering principles dedicated to agriculture. It involves the design and continual optimization of agricultural machinery and processes as well as the general development of technological advances to sustainably produce food, feed, fiber, fuels and energy. Examples comprise the automation of plant production processes, the conversion of bio-based resources to processable or directly marketable products, as well as modern technological concepts for animal husbandry. Past and current developments, starting from mechanization, over computer-aided automation to the rapidly progressing digital transformation, have each led to substantial technological advancements in agricultural production, and further leaps through e.g., autonomous agricultural machinery and robots, decision support systems and assistants leveraging latest breakthroughs in artificial intelligence (AI) are expected (Fountas et al. 2024). Such a rapid development as currently observable in the digitalization of agriculture, however, often unfolds as double-edged sword where new possibilities have to be weighed against co-occurring challenges. In order to obtain broad acceptance of emerging deeply digitalized solutions, it must be systematically dealt with concerns introduced by integrating state of the art agricultural engineering practice with latest information and communication technology (ICT) approaches pervading several technological scales. The resulting complex agricultural technology systems thus will not only manifest in the form of machinery performing agricultural tasks in the fields or barns anymore. Future

received 31 July 2024 | accepted 16 January 2025 | published 12 February 2025

agricultural technology will increasingly also take effect on superordinate technological levels in the digital agriculture system, for instance at the individual farm's, inter-farm and purely virtual or data space levels (Bökle et al. 2022) for a discussion on such a framework). In our article, we refer to such technical systems affecting one or several technological levels through the incorporation of digital technologies broadly as *AgTech systems*. Due to the increasing complexity of modern AgTech systems, we argue that systems thinking and new approaches of system engineering are needed. To this end, intelligent system architectures that assure resilient operations across all technological scales are deemed key.

Organic Computing (OC) refers to a systems engineering paradigm for building 'intelligent', i. e., adaptive, robust and flexible technical systems which are deployed in complex real-world environments. Therefore, OC heavily draws inspiration from biological systems and how they deal with the complexity and continual change as observed in nature. Aiming at the transfer to complex technical systems, OC sets a strong methodological focus on bio-inspired mechanisms to endow the systems with 'lifelike' qualities such as intelligence, self-organization or resilience. As a systems engineering discipline, OC provides principled methodologies and frameworks for building such complex systems which are subject to unanticipated disturbances and continual change when deployed in real-world environments.

In this article, we will cast light on how an AgTech system can be understood and modelled as an OC system. We therefore first provide an overview of the OC idea and recapitulate the well-known observer/controller-architecture. Afterwards, we review previous work on applying OC to agricultural settings. We then shift our perspective on digital agriculture settings and introduce our vision of intelligent agricultural technology systems. Subsequently, we touch upon a current case study where we demonstrate the utilization of a multi-level variant of the generic observer/controller system model to render the complex technical system of a tractor-implement combination for secondary tillage into such an intelligent system. Our system under observation and control comprises a power harrow attached to a tractor as well as a sensing system that observes the state of the soil and the machine. We discuss how this 'organic system' implements a feedback control loop and how intelligent machine adaptation behavior can be achieved by means of model-based (offline) reinforcement learning and discuss our findings.

The contribution of our work is the elaboration of a computer science perspective on the design of intelligent agricultural technology systems. Drawing from the knowledge available in the overarching discipline of self-adaptive and self-organizing systems research and their numerous approaches manifesting in initiatives such as autonomic computing (Kephart and Chess 2003), self-aware computing systems (Kounev et al. 2017), lifelike computing systems (Stein et al. 2021), in this work we specifically focus on an Organic Computing approach. We investigate the hypothesis that "Organic Computing provides a viable approach to systematically design and implement intelligent agricultural technology systems". By reporting on a conducted case study investigating the optimization potential within the process of secondary tillage, we provide initial empirical evidence that supports our hypothesis.

The motivation and scientific significance of our work results from the following aspects: A continuing rise in operation complexity of agricultural machines can be observed which is driven by the increasing integration of digital tools and resulting additional steps in the process workflows. Numerous decisions must be made by the farmers and machine operators which need to be properly trained

to deal with the emerging complexity due to the newest advances in the machinery and encompassing digitized processes. Taking a systems engineering approach and drawing from the recent advances in artificial intelligence (AI) technology, we attempt to advance the interdisciplinary research field of mastering the complexity of agricultural processes following the OC paradigm. To this end, our proposed agent architecture (see section "System Model"), contextualized to the domain of agricultural engineering, can serve as a reference model to fuel the exchange between the agricultural engineering and AI communities originating from the disciplines of mechanical engineering and computer science. Furthermore, our work is intended to revive the research work on Organic Computing applied to agricultural engineering challenges. For that, we provide an overview of past work in that intersection, classify it into the overarching domain of self-adaptive and self-organizing systems research and report on an exemplary case study that advances the field by incorporating contemporary AI techniques into an OC control loop.

Background

Organic Computing

Building technical systems with information processing inspired by the self-organization capability of natural systems lies at the heart of OC. It defines itself as a systems engineering paradigm rooting in the computational engineering discipline. Thereby, OC is strongly intersecting with many other branches of computer science these days, most prominently with artificial intelligence, software engineering and embedded systems. OC's inception dates back to the early 2000s, where the conceptual idea was presented in a research seminar and published in a position paper (VDE and GI e.V. 2003). A comprehensive discussion on what OC is and different perspectives for how to define the term can be found in the most recent textbook of Müller-Schloer and Tomforde (2017d). In Tomforde et al. (2017) a concise definition for an OC system is provided, which we adopt for this paper:

"*An OC system is a technical system equipped with sensors to perceive its environment and actuators to manipulate it. It adapts autonomously and dynamically to changing conditions in its environment. This adaptation process influences the system's utility, which is continually optimized by the OC system itself. To allow for such an adaptive behavior, it employs so-called self-x mechanisms.*"

OC research quickly attracted high attention leading to the establishment of a priority program within the German Research Foundation (SPP1183) running from 2005 to 2011. Since that time, tremendous progress has been sparked by the OC community, what led to a sort of OC toolbox comprising methods and principles for designing complex and trustworthy technical systems (Müller-Schloer et al. 2011, Reif et al. 2016). Applications of OC technology (Müller-Schloer and Tomforde 2017a) manifested in several research projects and technical systems embedded into real-world domains such as urban traffic control (Prothmann et al. 2011, Stein et al. 2016), self-adaptive network protocols, e. g., (Tomforde et al. 2011b), self-learning system-on-chip architectures, e. g., (Bernauer et al. 2011, Zeppenfeld and Herkersdorf 2011), an organic robot control architecture (Hartmann et al. 2013), smart camera control systems (Müller-Schloer and Tomforde 2017c, Rudolph et al. 2019), or decentralized energy and smart grid management (Allerding et al. 2011, Mauser et al. 2015, Reif et al. 2016). The research conducted in OC and related initiatives dealing with the design of future computing systems mutually fertilized each other. Autonomic computing, self-aware computing,

and general research on self-adaptive and self-organizing systems are the most prominent examples (Müller-Schloer and Tomforde 2017e). Out of the developed toolbox, one particular artifact is the generic observer-controller architecture, as schematically depicted in Figure 1.

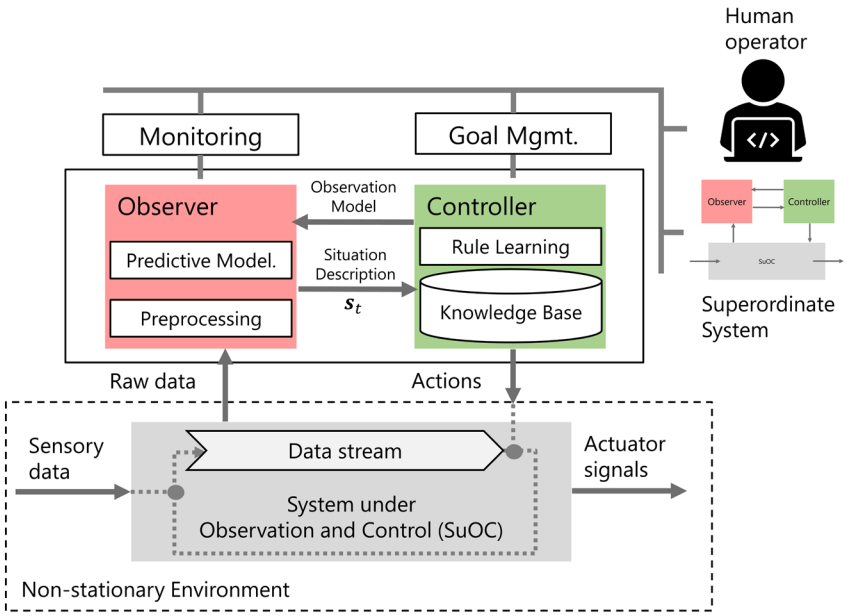

Figure 1: Generic Observer/Controller architecture of Organic Computing

This architecture can also be interpreted as an agent blueprint for building an AI system. Thus, we use the terms O/C architecture and OC agent architecture interchangeably in the remainder of this paper. A generic OC agent comprises at least two layers: The bottom layer, or often denoted layer 0, denotes the so-called *system under observation and control* (SuOC). That is the productive system which is typically situated within a non-stationary environment. The system uses sensors to perceive its surrounding environment conditions and uses the sensory data as input to be further processed. After the data processing, control actions are selected and from that actuator signals are derived. These control actions eventually lead to modifications of the productive environment and in turn influence the input signals observable by the sensors. Over the SuOC's lifetime, this results in a continual data stream which can be observed and controlled by upper system layers. This basically forms a closed-loop control system which allows the adaptation of the SuOC's behavior in a way to optimize its utility. The SuOC can consist of only one technical system, such as a production plant or a mobile robot, but can also be a collection of semi-autonomous entities, e. g., a robot swarm or another kind of distributed computing system (e. g., a grid or peer-to-peer system or a smart camera network). This in turn already implies the formation of OC agent hierarchies, which we will not further discuss here. The interested reader is referred to Tomforde et al. (2011c) for further details. We still note that the further elaboration is indeed worthwhile considering the pervasion of digital technology across different technological scales in agriculture, i. e., from the machine, over the farm- and inter-farm level, to the virtual cloud level (Bökle et al. 2022). However, because this would far exceed the scope of this paper, we leave this discussion for future work.

An OC agent sets at least one layer – usually denoted Layer 1 – on top of the SuOC. This layer comprises an *observer* as well as a *controller* component. The observer gathers the raw data from the data stream of the SuOC. It has subcomponents to preprocess, filter, and further analyze the system parameters observed through sensor readings. For instance, if the SuOC comprises many (semi-) autonomous entities, the observer could analyze the emergence of desired or unwanted patterns to derive important system properties to be actively controlled (Fisch et al. 2010). In other scenarios, such as urban traffic control, the observer might use this data to learn a predictive model forecasting the upcoming traffic demands at intersections (Sommer et al. 2016). The observer can also detect patterns or conduct clustering to find typical, abnormal or recurring system states, which can be learned to be avoided or maintained by the system's controller.

The controller component is then responsible for learning specific actions in order to react to observed system states. It receives an abstracted situation description from the observer (often called system state at time). Based on this system state, it consults a knowledge base which can be fixed and predetermined by the system designer or learned by the system itself during runtime (i. e., online). The knowledge base consists of IF(system state)-THEN(action) rules which determine the most suitable reaction to particular system states. These actions are then translated into control signals to be realized by the SuOC's effectors. Most often reinforcement learning based on *Learning Classifier Systems* (LCS) is applied here to learn an interpretable rule base which can be analyzed and enriched manually by human operators (Stein 2017). Thus, interpretable machine learning has been central to OC research from the beginning. Another task of the controller is to determine the observer's observation model, which can for instance influence the frequency of state observations, as well as the granularity of abstraction. It can thus also affect the features included in the abstracted situation description what has in turn to be coped with in the controller's rule learning or any other decision-making component (Stein and Tomforde 2021).

It should be noted that OC agents not per default strive for full system autonomy. Development of such systems usually follows an incremental design process that also considers the Human-in-the-Loop (Tomforde and Müller-Schloer 2014). Accordingly, the layers on top of the SuOC are commonly encapsulated by a collaboration layer, which provides interfaces for monitoring and goal management. The first interface allows the system operators (but also super-ordinate OC agents in multi-agent system hierarchies) to inspect or supervise the system's behavior. The latter interface allows for modifying the system's goals through adjusting certain evaluation criteria that parametrize the utility function of the OC system. The controller has to account for such goal changes. This can be done by e. g., adapting its current control policy, replacing it from a set of predefined ones, or by learning entirely new ones. The extent of this self-adaptation capability determines the flexibility of an OC systems.

We have delineated the description of this generic OC architecture by concentrating only on one adaptation layer set on top of the SuOC. However, most applications have extended this basic model and implemented a specific variant of it – the so-called *multi-layer observer/controller architecture* (MLOC) (Müller-Schloer and Tomforde 2017b). The most distinguishing point to mention rather shortly here, is that these variants employ further layers on top of the adaptation layer 1 as described before. Such layers serve the purpose of further system cognition, but introspectively directed, i. e., observing the system's performance and intervene accordingly by triggering self-adaptations, if for instance performance declines are detected. Such layers are thus often called 'reflection' or 'cognition'

layer. In our case study as presented below, we present a specific example of such an MLOC instance for the agricultural use case of intelligent tractor control for secondary tillage.

Today, OC is still understood as a paradigm that aims to make the ever-increasing complexity of current and future interconnected computing systems manageable. Therefore, current OC research centers around developing principles, methods and architectures for the flexible design and robust operation of technical systems embedded in various dynamic real-world contexts. As mentioned before, resulting OC methods are strongly nature-inspired manifest the underlying principles in algorithmic counterparts. The idea is that due to employing adaptivity and (controlled) self-organization into technical systems, these become able to function robustly, re-configure flexibly and act adaptively in real world environments (Schmeck et al. 2010). This is achieved by implementing so-called *self-x properties* that let these systems act in a 'lifelike' manner. That means that future technical systems will be enabled to shift decisions that traditionally had to be made by engineers at the system's design time to the runtime and, thus, the responsibility of the technical systems themselves. To this end, methods from the broad field of classical and modern artificial intelligence (AI) and artificial life play a central role. For example, machine learning and metaheuristic optimization (e. g. evolutionary algorithms) are utilized to render OC systems continually *self-learning* and *self-optimizing* (e. g., (Tomforde et al. 2011a). Concepts of distributed systems and swarm intelligence are being introduced to address the ever-increasing degree of distribution in open systems and thereby increase system's resilience and dependability through implementing *self-organization* and *self-healing* properties. For instance, novel computational mechanisms inspired by the natural hormone system and DNA have been proposed to increase system dependability (e. g., (Pacher and Brinkschulte 2020). A comprehensive overview of algorithmic concepts and methods developed for OC can be found in Müller-Schloer et al. (2011).

Prior Work on Organic Computing in Agricultural Engineering

Besides the various real word applications as exemplarily mentioned above, OC has also been considered in the domain of agriculture. Primarily the optimization of tractor control and management has been subject of research. The project was called OCOM for Organic Computing for Off-Highway Machines and applied the generic O/C architecture to optimize fuel consumption of a tractor-implement combination in simulation (Kautzmann et al. 2010, Wünsche et al. 2010, Mostaghim et al. 2011, Kautzmann et al. 2012). We consider a similar problem setting in our work but extend the previously done investigations by optimizing a multi-criteria utility function taking next to fuel consumption and area output also the observable work quality and engine utilization into account. Our approach makes use of an *environment model* (as explained in the next section) recently proposed in Boysen et al. (2023) which is calibrated and trained on real data collected from field experiments. This allows our proposed system architecture to leverage model-based machine learning allowing individual components for continual optimization. Such an approach unfolds its potential when the system exhibits the *self-reflection* property. We discuss this in more detail in the case study section below.

A second scenario where OC technology was applied to an agricultural problem setting is presented by Mnif et al. (2007). The authors addressed the severe problem of cannibalistic poultry behavior in livestock farming. In an agent-based simulation study, the observable cannibalistic behavior of hen, i. e., attacking injured birds by flocking around them and picking them to death, has been modeled. An observer component was employed to detect this self-organization behavior and calculate a so-called emergence fingerprint. Each time this emergence value surpasses a predefined threshold,

the controller could infer that an intervention is necessary. The controller could then emit a disturbing signal followed by the dispersion of the attacking bird flock what overall led to an improved survival rate of injured hen.

Recently, another work has been published in the context of autonomous field vegetable monitoring (Lüling et al. 2022). An O/C architecture with a self-reflection layer was presented which allowed for the online detection and elimination of knowledge gaps in a robotic monitoring system. The monitoring task was the instance segmentation of cabbage which yields important information of the field performance and expectable yield. Each time the system delivered a poor segmentation of low confidence, the reflection layer detected this fault and triggered an active learning loop to rectify the segmentation model in order to obtain an improved prediction the next time a similar situation occurs.

In Modak and Stein (2024), the authors present a methodology to generate artificial training data for intelligent weed control systems. Their approach utilizes the SAM foundation model combined with a stable diffusion model to synthesize images of weed-infested crop areas. It is shown conceptually, how such a generative AI pipeline can be integrated into an Organic Computing architecture in order to enable proactive self-improvement in the context of automated weed control.

Looking beyond the concrete Organic Computing scope, Boubin et al. (2019) discusses challenges on applying autonomic computing principles to fully autonomous precision agriculture. Based on an autonomic computing approach, they demonstrated in simulation that the sampling rate of crop fields can be reduced to 40 % and less for predicting accurate yield maps, what in turn has the potential to decrease labor cost and energy demand in UAV-supported precision agriculture settings.

As can be seen, initial concepts and experimental studies already exist that acknowledge the suitability and potential of OC and autonomic computing techniques when adopted to complex technical systems in the domain of agricultural engineering. With this article, we want to take a further step. Firstly, by abstracting from particular agricultural use cases in the following section, we intend to elaborate on the general suitability of OC for designing future intelligent agricultural technology. Afterwards in a case study considering the secondary tillage process, we build upon these conceptual thoughts and propose and formalize an OC-based system model for optimizing tractor control incorporating modern tools of AI and demonstrate the viability of our approach by reporting on empirical results from an initial experimental analysis.

OC Perspective on Intelligent Agricultural Technology

The fourth industrial revolution is driven by digital technology. The so-called digital transformation impacts all sectors of our economy and thus also primary production. The digitalization of agriculture promises increase in efficiency, transparency, and productivity by offering the possibility to extract and make use of relevant information hidden in the vastly collected data of production processes – even across the entire agri-food value chain (Krupitzer and Stein 2024).

The fundamental principles of gathering data and using information to support decision making and optimize farming processes date back to the 1990s, when the concept of *precision agriculture* was introduced. With sensors and computing units utilized becoming increasingly smart and powerful in the early 2000s, higher automation of the farming processes through in-situ real-time analysis of the collected data have become possible, what is often referred to as *smart farming*. Digitalization of agriculture or *digital agriculture* expands the methods used to new digital technologies such as IoT, edge and cloud computing, intelligent big data analysis, and artificial intelligence but, importantly,

still shares the overarching goals already formulated for precision agriculture in the 1990s and was revised by the International Society of Precision Agriculture (ISPA 2024) just in 2024:

"Precision Agriculture is a management strategy that gathers, processes and analyzes temporal, spatial and individual plant and animal data and combines it with other information to support management decisions according to estimated variability for improved resource use efficiency, productivity, quality, profitability and sustainability of agricultural production."

The reader is referred to (CEMA 2017, DLG e.V. 2018, Paraforos and Griepentrog 2021) for a more detailed discussion on these terms and related concepts. Concrete examples of technical systems belonging to manifestations of the sometimes interchangeably used terms 'smart' and 'digital' agriculture comprise: Camera-steered weeding machines, either in form of tractor implements or self-propelled mobile robots; smart implements that adjust their controllable parameters (e.g., working depth or application rate) based on real-time sensory; multi-vehicle systems that automatically navigate fields (e.g., a fleet of machines following a leader) or detect a trailer's fill status to automatically control the off-loading from forage harvester; or automated steering systems that not only use GNSS-RTK signals but also sensory information to obtain a fault-tolerant row navigation. We only give some examples here and restrict ourselves to technical systems in form of complex machinery, thereby neglecting purely software-based digital solutions (such as smartphone apps detecting health status of crops). These technologies and others are often termed 'smart' sometimes even 'intelligent', for the latter of which in the opinion of the authors the criteria are usually not satisfied (see definition of Intelligent Agricultural Technology below).

A current observable trend however is the increasing appearance of mobile robots for automating certain steps of agricultural production processes or performing processes in a fully autonomous manner (Vougioukas 2019). Enhancing these machines' degree of autonomy increasingly demands for the utilization of AI in order to maintain reasonable levels of robustness and reliability (Bechar and Vigneault 2016). Being able to fully automate entire agricultural production processes by means of autonomously operating machines can contribute to alleviating present challenges in agriculture such as labor shortage or the strong need for adaptivity due to volatile environmental conditions.

Today's commercially available robotic applications for crop production concentrate on weed control, seeding, unmanned aerial vehicles (UAVs) for e.g., monitoring and mapping plant stand conditions, as well as driverless automation of field navigation (Vougioukas 2019, Fountas et al. 2020). Highly autonomous indoor farming is another application area of robotic solutions. Looking in the livestock farming domain, milking robots, feed pushers, barn cleaners, belong to the most available solutions. Since a more detailed discussion of these topics would fall beyond the scope of this paper, we refer the interested reader to recent surveys on the state of automation and digital technologies in livestock and indoor farming applications (Groher et al. 2020, Ragaveena et al. 2021).

Research on the other side, proposes tremendously more applications for autonomous robots in agriculture. To name only a few, potential applications ranging from fully autonomous robotic systems for field and green-house vegetable farming, hybrid swarms of UAVs and small ground robots for detecting and controlling weed, to field swarms comprising heavy-weight interoperable and modular entities (driverless tractors) that collectively navigate and manage fields (Herlitzius et al. 2021, Albiero et al. 2022). The question to what extent currently available solutions and concepts from the

scientific literature possess intelligence, however, is not sufficiently discussed yet. Intelligence is often associated with cognitive capabilities of naturally intelligent beings such as humans (Boden 2018, Russell et al. 2022). Adopting these for the definition of artificial technical systems is however not always straightforward.

For the purpose of this paper, we introduce the concept of *Intelligent Agricultural Technology* (IAT). The term IAT is not established yet. We therefore provide a first attempt for its definition along with some classification criteria which we deem necessary to qualify agricultural technology as 'intelligent':

"IAT describes technical systems designed for the purpose of performing and optimizing agricultural measures within agricultural processes without the immediate and permanent need for human intervention, but the interfaces to allow for farmer monitoring and control at any time. IAT systems are thus autonomous systems, which embed one or more constituent AI programs (agents) via computational units with mechatronic (hardware) components. The embedded AI agents enable the resulting cyber-physical system to continually perceive its environments (e.g., soil conditions, vegetation status) through sensors and other information sources (e.g., weather forecasts), to self-reflect on the current system performance, and to react through immediate self-adaptations and planning of the best next actions to optimize the system's utility (e.g., area output, energy efficiency, field efficiency) what captures the goals as set by human operators or superordinate systems. Actions can be either internal adaptations of machine parameters (e.g., working speed, engine speed), signals to control effectors that modify the environment (e.g., harrows, hoes, grippers for fruit harvest), or recommendations to the operators."

Following our definition of IAT above, we qualify autonomous agricultural systems, such as mobile robots, only as 'intelligent' if they have the capability to decide for and plan necessary actions to take by themselves. Furthermore, IAT systems have to show a certain degree of reflective capacity about their own performance (or system utility) what allows them to detect disturbances during the process and autonomously trigger measures to recover from deteriorated system states (*robustness*). Despite spatio-temporal variability in their environment (e.g., different soil conditions, nutrient availability, vegetation stages), IAT can deal with complex, contradicting and multiple performance criteria and through their *adaptivity* can continually maintain or even self-optimize toward a high-utility machine behavior, e.g., in terms of working quality, resource efficiency, as well as other economic or ecological objectives. Furthermore, due to their continual self-improvement, IAT can deal with abruptly or gradually changing goals as set by humans or superordinate systems (*flexibility*).

As an example, we would qualify a mobile agricultural robot as IAT, if it fully autonomously navigating fields, planning and performing its assigned agricultural tasks, and, in the process being capable of adapting to local conditions in the surrounding environment or circumventing unexpected disturbing objects by using AI technology for perception, manipulation and reflection. On the other hand, an unmanned aerial vehicle (UAV) system flying a preplanned route over a field and using an isolated deep learning-based AI model to detect weeds, would, according to our definition above, not be deemed an IAT system – but nevertheless a highly important tool of smart agriculture.

By means of putting an explicit emphasis on the intelligence qualities (e.g., reflective capacity, self-learning and decision-making capabilities) we demand of such systems to realize the claimed properties of adaptivity, robustness, and flexibility, we attempt to delineate our proposed concept of

IAT from other terms in the literature as discussed before. Additionally, we note that a new journal entitled 'Smart Agricultural Technology' had its first issue in 2021 and covers applications of smart and digital technology, including artificial intelligence use, to all agricultural sectors. Despite of having similar names, we argue that the delineation also applies analogously here, but still feel that the journal's broader scope clearly encompasses future works on IAT.

Suitability of OC for IAT

Having defined the terms OC and IAT in more detail, we now briefly recapitulate on their common challenges. In Table 1 we depict a brief comparison of OC's envisioned evolution of modern and soon observable computing systems in the left column (cf. also the 'Complexity challenge', as elaborated in Müller-Schloer and Tomforde (2017d)) and developments and challenges faced by modern agricultural engineering. Please note that this juxtaposition is intended to provide a sense how the developments envisioned by OC and the current and soon increasingly observable trends in smart agricultural technology align with each other. The aim is not to be fully comprehensive or surveying the latest advancements. For more insights on this topic, the interested reader is referred to e. g., (CEMA 2017, Paraforos and Griepentrog 2021, Bökle et al. 2022, Fountas et al. 2024).

Table 1: Comparison of OC's insights on future computing systems (left) to current and future agricultural engineering challenges (right).

Organic Computing postulates…	**Agricultural Engineering copes with…**
Rising complexity through ▪ decreasing size of computing devices ▪ increasing connectivity ▪ increasing computing capacity (Moore's law)	IoT devices and IT connected from Edge to Cloud ▪ Smart sensors and connected real-time capable computing units on machines ▪ Digital software services as part of emerging digital ecosystems (platforms, federated services in data spaces) ▪ Telemetry and telematics systems, mobile and ad-hoc wireless networks (e. g., 5G, LoRaWAN) ▪ AI-capable rugged computers on land machines, mart-phones, tablet computers, farm-level edge servers, access to powerful cloud resources
Impossibility of central administration ▪ Open, interconnected distributed systems ▪ Vast amounts and velocity of both structured and unstructured data ▪ Demand for data security and privacy	Digitalized farms being decentralized IT systems ▪ Various proprietary systems by OEMs ▪ Limited interoperability due to lack of normed interfaces for secure and sovereign data exchange ▪ Digital pervasion through several technological scales from machine, farm- and inter-farm to virtual cloud level
Challenge of open, dynamic environments ▪ Real world environments often are complex adaptive systems ▪ Competing objectives from social, economic and ecological dimensions ▪ Non-stationarity due to adaptation and interaction of changing system entities and subsystems as well as natural stochasticity ▪ Results in high requirements for dependability and trustworthiness of autonomous behavior of self-organizing and self-adaptive systems	Highly complex agricultural domain ▪ Both, agricultural objects (crops, animals) and environment (fields, barns) are dynamic and hardly predictable ▪ Agriculture can be understood as a highly decentralized, social-ecological and social-technical system ▪ Spatio-temporal variability and heterogeneity (e. g., plants / soil) ▪ Enormous complexity for autonomous machines and robots which need to comply with machine regulations and strict functional safety requirements ▪ Lack of trust in new digital technologies, especially in AI, and uncertainty about reliability of such systems

The above table leads to the hypothesis that future agricultural technology will evolve into systems of systems that manifest on different technological scales which all make strong use of modern information technology and AI in order to deal with the increasing system complexity. Accordingly, IAT systems built by leveraging modern systems engineering approaches which address the already persistent overlap of classical machine engineering, electronics and computer sciences constitute a key requirement for keeping pace with the tremendous developments fueled by AI and other new digital technologies as expected in the upcoming decades.

We postulate that OC constitutes a viable approach for designing IAT systems, allowing them to master the increasing system complexity by building upon the innate 'self-x properties by design' principle of OC. To make a first step to substantiate our claim, we proceed with adopting an OC approach to design an IAT system for a particular use case in crop production – the optimization of a secondary tillage process (Köller and Hensel 2019). Our case study comprises the definition of a concrete system model as well as a formalization into mathematical descriptions which in turn allow for the application and evaluation of AI methods. We demonstrate how the targeted IAT system can be framed using a multi-layer variant of the generic O/C system architecture from the OC toolbox. To this end, we flesh out most relevant components with concrete machine learning techniques. In this study we focus on deep reinforcement learning (Mnih et al. 2015) to allow for reactive machine parameter adaptation combined with planning through integrating a deep learning-based environment model. Lastly, we evaluate our IAT system approach in silico based on data collected from in situ field trials and compare our AI-based solution against a common practice baseline.

Case Study: OC-Optimized Secondary Tillage

Primary tillage describes the process of mixing and/or burying organic residues into the soil of agricultural fields while also removing soil compaction (Guérif et al. 2001). It is followed by secondary tillage to prepare the seedbed for seeding. In practice, secondary tillage is often combined with seeding by utilizing, e. g., in our case a power harrow-seeding combination. The preparation of the seedbed and the resulting aggregate sizes are essential for crop establishment which includes the germination and early growth stages (Braunack and Dexter 1989). Secondary tillage is performed by mechanically treating the soil to crush soil clods, provide good germination conditions and level the soil surface. The way secondary tillage is performed has an impact on the soil aggregate distribution and therefore the seedbed quality (Adam and Erbach 1992). While the seedbed quality is often measured by the *mean weighted diameter* (Sandri et al. 1998, Riegler-Nurscher et al. 2020), this measurement method requires manual effort for soil sampling and sieving and is thus not useable for online system control. Riegler-Nurscher et al. (2020) estimated the seedbed quality to control their machine by calculating the *roughness coefficient* utilizing height measurements of the soil surface. They found a correlation of the roughness coefficient to the mean weighted diameter which describes the distribution of soil aggregates. Following their findings, the roughness coefficient is used in this case study to quantify the tillage quality.

System Model

Besides the *seedbed quality (SQ)* (in mm) that we measure by the roughness coefficient, other performance metrics observable during secondary tillage process are the *fuel consumption (FC)* (in l/ha) and the *area output (AO)* (in ha/h). During secondary tillage, the operator of the machine needs

to observe their working environment by assessing current soil conditions in front and in the back of the machine, i. e., after the tillage operation. To allow for a continual parameter adaptation, these assessments need to be put in relation to the current machine configuration (e. g., working speed), the machines overall capabilities and the targets of the tillage process (i. e., the performance metrics as introduced above). The interdependent nature of the performance metrics renders the performance optimization an intricate problem. In our case, the soil conditions are sensed by two stereo cameras (SceneScan Pro-system of nerian vision technologies) and an Emlid Reach RS+ each attached with aluminum profiles in the front and in the back of the machine with a distance of 10.69 m between them. In Figure 2, we show the setup consisting of a Kverneland e-drill compact which is attached to a Claas Arion 660 while performing secondary tillage in the field.

Figure 2: Kverneland e-drill compact attached to a Claas Arion 660 with a computer mounted in the cabin of the tractor and stereo cameras as well as GNSS antennas in the front and in the back of the machine (© Stein and Boysen)

These cameras provide both RGB as well as depth information of the soil surface in the front img_f and in the back img_b displaying a square of about 0.5 m × 0.5 m with a spatial resolution of about 1 mm per pixel. The depth information is used to calculate the according two soil surface roughness coefficients rc_f, rc_b. Further information of the current state of the environment is gathered by reading telemetry data from the CAN- and ISOBUS of the tractor. This includes information on the current working speed v_w (in m/s), engine speed n_e (in n/min), PTO (power take off) speed n_{PTO} (in n/min), engine torque utilization M_e (in %) and engine fuel rate fr_e (in l/h). Furthermore, GNSS-RTK antennas are used to spatially match the positions of images taken in front of and behind the machine (Figure 3), to calculate the slope of the field (in °) and to calculate GNSS-corrected working speed to take slip into account. For each image and position taken in front of the tractor, images from the back of the tractor are matched by using the constant distance of 10.69 m between the front and the back cameras. This is possible with the constraint that the tractor is always driving in a straight line and datapoints at the beginning and at the end of a lane are discarded. Additionally, the manually set depth of the power harrow ph_d is included in the current state of the machine. All information is streamed to a computing unit which is mounted in the cabin of the tractor. The computing unit comprises an Intel Skylake i5-6500TE, an NVIDIA GeForce GTX 1050TI and 32 GB RAM.

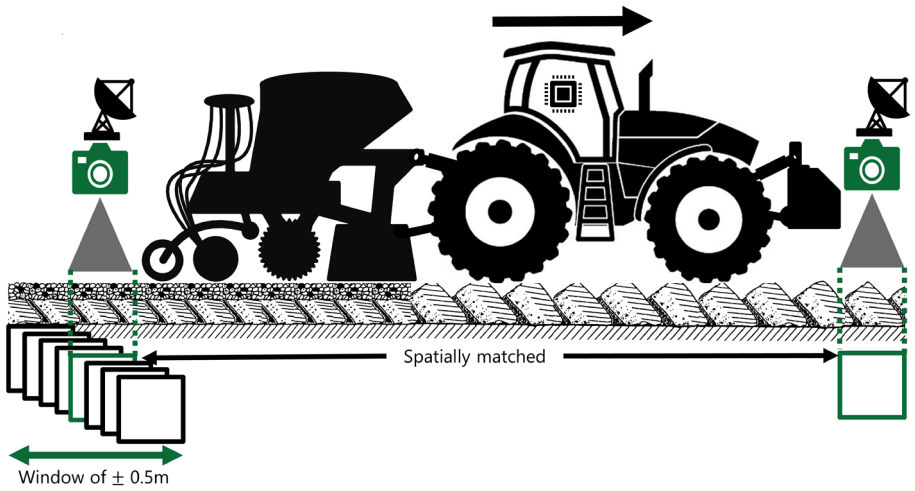

Figure 3: Schematic of the tractor-implement with the positions of the stereo cameras in the front and the back of the machine. An GNSS-RTK antenna is mounted on top of each camera adapted from Boysen et al. (2023)

Based on the generic observer/controller architecture as discussed before, we present a multi-level variant of this model (Müller-Schloer and Tomforde 2017b) dedicated for controlling the secondary tillage process in Figure 4. The technical system of a tractor seeding combination determines the SuOC. By means of wrapping it with the MLOC architecture as depicted in Figure 4, we render our technical system an intelligent system, more precisely an IAT. This system design allows the embedding of an intelligent control agent, i. e., AI software, to intelligently choose actions for system adaptation and also to self-reflect on the system performance.

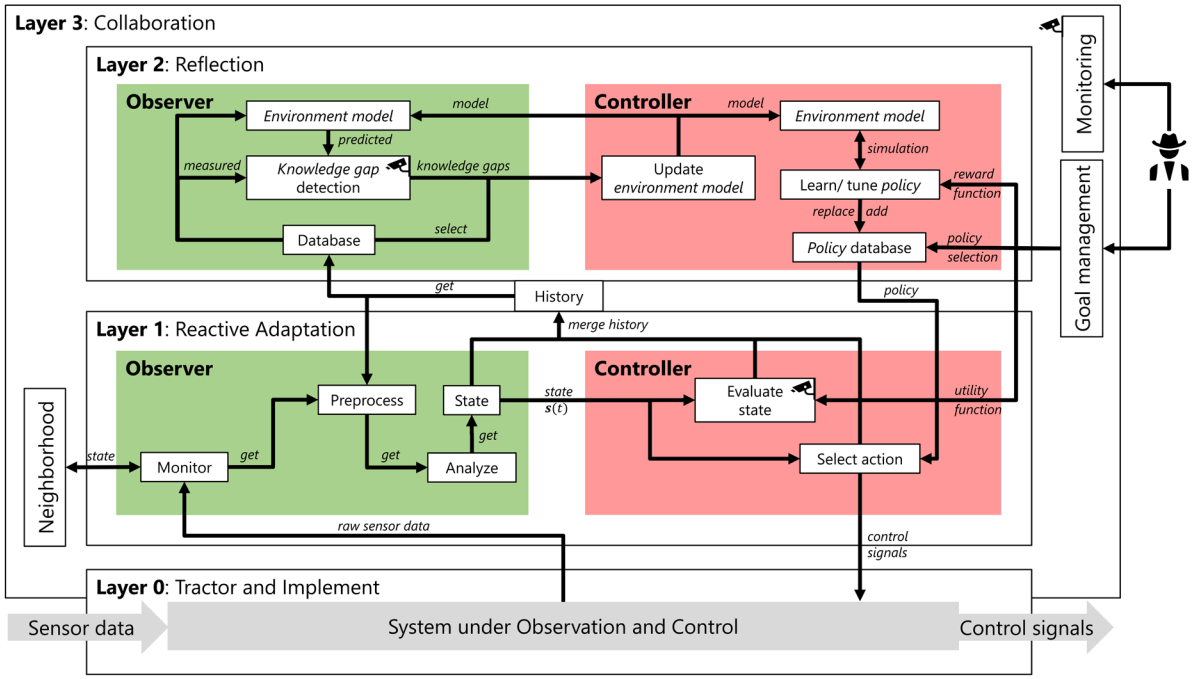

Figure 4: Multi-level Observer/ Controller (MLOC) architecture for secondary tillage control adapted from Müller-Schloer and Tomforde (2017b); arrows between the components indicate the data flow and functional relationships which are derived from the originating components and used in the receiving components

The first layer of the architecture is for establishing a reactive adaptation loop. The observer part of this layer is responsible to monitor raw sensor data from the tractor-implement combination (SuOC at Layer 0). Our architecture includes an interface to receive further information from the neighborhood, which could be comprised of machines working at the same time on the same field, or even from previous process steps, e. g., a tractor performing primary tillage and measuring soil conditions. The observer then preprocesses the incoming sensor data and further information utilizing the ROS2 middleware (Macenski et al. 2022). Therefore, the CAN- and ISOBUS of the tractor are connected via CAN-Interface for USB (PEAK-CAN) to an onboard rugged computing unit mounted in the cabin of the tractor. The preprocessing step includes parsing selected information (v_w, n_e, n_{PTO}, M_e, fr_e) which is important for the later defined state vector from CAN- and ISOBUS using the messages defined in J1939 and ISO 11783. Additionally, this step includes calculation of GNSS-speed and slope, detection of low-quality stereo images, georeferencing the captured images as described above and computing averages for data with high variance (e. g., image information for roughness coefficient calculation over the displayed window in Figure 3). During this step, information from past system observations (history) is supplied. Next, the preprocessed data is further analyzed to derive important information for building a situation representation describing the current system state $\boldsymbol{s}$. This includes the calculation of the soil surface roughness coefficients and the current working state of the machine.

After this analysis, the complete state is available, passed to the controller of the reactive adaptation layer and stored in the history. The state vector $\boldsymbol{s}$ is defined according to equation 1:

$$\boldsymbol{s} = \left(img_f, rc_f, rc_b, v_w, n_e, n_{PTO}, ph_d, slope, M_e, fr_e\right) \in S, \quad \text{(Eq. 1)}$$

where S determines the system's state space. Within the controller of the reactive layer, the state is evaluated by a *policy* (or adaptation strategy) that determines the next action to take. The selected action serves as a control signal to the tractor. In our case-study, the action determines a necessary adaptation of the machine's working speed, since it has been found to be the most important machine configuration of the tractor during secondary tillage as it has a large impact on all performance metrics (Varani et al. 2023). Depending on the learning problem formalization there are several options to define the action space . The choice of the action space has an important impact on the selection of appropriate machine learning methods. Options include a discrete action space with predefined working speeds or a continuous action space defined by an interval. In our case, we decided to define a discrete action space A with action values $\boldsymbol{a}$ specifying to increase ($+\Delta v_w$) or decrease ($-\Delta v_w$) the working speed by a specified delta (Δv_w) or to keep (+0) the current working speed:

$$\boldsymbol{a} \in A \coloneqq \{\Delta v_w, -\Delta v_w, 0\} \quad \text{(Eq. 2)}$$

It is assumed that the machine should operate between a minimum working speed $v_{w_{min}}$ and a maximum working speed $v_{w_{max}}$. To keep the machine parameter within a viable range, we restrict the adaptation by applying the following rule:

$$v_w \leftarrow \begin{cases} v_{w_{min}} & if\ (v_w + \boldsymbol{a}) < v_{w_{min}} \\ v_{w_{max}} & if\ (v_w + \boldsymbol{a}) > v_{w_{max}} \\ v_w + \boldsymbol{a}\ otherwise \end{cases} \quad \text{(Eq. 3)}$$

The roughness coefficient in the back of the machine rc_b, the engine torque utilization M_e and the engine fuel rate fr_e will be predicted by a model of the environment which allows modification of the machine configuration including v_w (Boysen et al. 2023). Other components of the state vector are either read from the dataset during model training or will be perceived from the environment during application (img_f, r_{cf}, n_e, n_{PTO}, ph_d, *slope*). To allow the tractor to have sufficient time to accelerate to the targeted working speed, the working speed delta Δv_w and the interval of control signals λ_a (in s) needs to be appropriately set to allow the tractor to adapt regularly. For this study Δv_w and λ_a are fixated, but an adaptation of these parameters could be handled by an adaptive observation model.

Besides selecting an action, the controller also evaluates the current system state $\boldsymbol{s}$ to retrieve a measure of system utility $U(\boldsymbol{s})$. This evaluation includes the calculation of the performance metrics *AO, FC* and *SQ. SQ* is directly accessible from the state vector through the soil surface roughness coefficient as measured behind the machine (rc_b). In our case the target was to minimize rc_b since the experimental fields were ploughed and not further tilled afterwards. In other scenarios a target rc_b as in Riegler-Nurscher et al. (2020) can be used instead. The area output and fuel consumption are calculated through the working speed, the working width (3 m) and the engine fuel rate of the machine. Accordingly, we define:

$$SQ = rc_b; \quad AO = v_w * 3\,m * 3600\frac{s}{h} * \frac{ha}{10{,}000\,m^2}; \quad FC = \frac{fr_e}{AO} \qquad \text{(Eq. 4)}$$

These metrics are then normalized in the controller component, merged with the system state and stored in the history buffer and thereby become accessible to the operator via the monitoring interface in the MLOC's collaboration layer. The minimum and maximum values required for the normalization are in case of the area output calculated based on $v_{w_{min}}$ and $v_{w_{max}}$. And for the roughness coefficient rc_b and the engine fuel rate fr_e the normalization is empirically based on the 1st and 99th percentiles of an already recorded dataset (Boysen et al. 2023). The normalized performance metrics are denoted by a hat above the symbol. The resulting system utility of a current state, $U(\boldsymbol{s}) \in [0,1]$, can be calculated by the weighted sum of the normalized performance metrics:

$$U(\boldsymbol{s}) = \beta_1 * \left(1 - \widehat{SQ}(\boldsymbol{s})\right) + \beta_2 * \left(1 - \widehat{FC}(\boldsymbol{s})\right) + \beta_3 * \widehat{AO}(\boldsymbol{s}) \;\; with \;\; \sum \beta_i = 1 \qquad \text{(Eq. 5)}$$

Since in our approach we want to maximize overall system utility, the normalized performance metrics $\widehat{SQ}$ and $\widehat{FC}$ need to be inverted in the above formula. We note that other methods to approach such multi-criteria optimization problems apart from this linear combination of evaluation criteria can be applied. Research on this fall beyond the scope of this work and is thus left for future research efforts.

The reflection layer (layer 2) of the MLOC allows the agent to self-reflect on the system's performance. In our approach, the layer 2 observer has access to past experiences and to an environment model, allowing the system to continually improve the adaptation behavior in the reactive adaptation layer's controller. Such an environment model has been developed in Boysen et al. (2023) and it constitutes a deep learning-based predictive model and is an essential component of this layer in our architecture. It is capable of predicting the response of the soil-machine interaction, i.e., fr_e, M_e and rc_b, for various context information (img_f, rc_f, *slope*) as well as machine configurations (v_w,

n_e, n_{PTO}, ph_d). The model was trained on a massive data set (n > 170,000) collected during in situ measurements across various real fields. For more details, we refer the reader to Boysen et al. (2023). The environment model is used in both the observer and controller part of the reflection layer. In the layer 2 controller, it is used to learn and continually improve a policy offline (i. e., without directly affecting the layer 1 reactive adaptation strategy) in a model-based reinforcement learning approach. This policy will map an action $\boldsymbol{a}$ to each possible state $\boldsymbol{s}$. In our case study, we will exemplary elaborate on how to learn such a policy and show simulation results in the following sections. Since the environment model only approximates the soil-machine response based on previously seen data, it is expectable that during the intelligent system's runtime completely different, so far unseen situations occur. If the system has no or hardly any experience with such an observed situation, we speak of *knowledge gaps* (Stein et al. 2018). The observer part of the reflection layer is responsible to detect such knowledge gaps by continuously comparing the predicted model response with the actually measured values that can only be obtained after the adaptation decision as done in layer 1. Identified knowledge gaps can then be counteracted by further fine-tuning the environment model specifically to the observed local situations and subsequently improving the agent's control policies. The detection of the knowledge gaps and the fine-tuning of the environment model are beyond the scope and are part of future work.

Furthermore, the aforementioned comparison of the agent's predictions with the actually observed real-world conditions can be used to derive a confidence score of the agent. Such a measure of uncertainty can be used to switch between different models of the environment, detect the aforementioned knowledge gaps or as a value source of information for the farmer which can be monitored through the human-machine-interfaces in the collaboration layer. Layer 3 also provides the farmer with the possibility to adjust goals by e. g., modifying the importance weights in the utility function or directly choose predefined policies based on a priori and tacit knowledge. Additionally, over the generically defined neighborhood interface, the agent can exchange data and information with other agents (Oroojlooy and Hajinezhad 2023). In our scenario, additional information could be either exchanged with other machines operating simultaneously on large fields (cf. the concept of field swarm technology in Herlitzius et al. (2021)). Or across process steps, i. e., sensor information such as draught force measurements as an indicator of soil compactness measured by machines during the primary tillage activity.

Markov Decision Process for Secondary Tillage Optimization

Based on the detailed system model, we proceed by formalizing the learning problem following the notional approaches of Sutton and Barto (2018) and Moerland et al. (2023). Performing secondary tillage on a field can be formalized as a sequential decision problem when concentrating only on the in-lane phase. In our system model, each future state is only dependent on the present state and therefore the Markov property is assumed to be fulfilled as well as full observability of the system state. Based on our defined state space S of the system, the action space A and the utility function $U(\boldsymbol{s})$, we formulate the system's learning problem as follows:

Find a policy π that for any observed system state $\boldsymbol{s}$ (soil roughness, machine parameters, etc.) selects the best action $\boldsymbol{a}$ (here working speed adaptations) to maximize the overall system utility $U(\boldsymbol{s})$ (here optimizing area output, seedbed quality and fuel consumption) by maximizing estimated discounted future rewards.

The sequential decision problem is formalized as a *Markov decision process* (MDP) $M = \{S, A, T, R, \gamma\}$. An MDP consists of a state space S, an action space A, a *transition function* $T : S \times A \rightarrow S$, $T(\boldsymbol{s}, \boldsymbol{a}) \mapsto \boldsymbol{s'}$ which returns the succeeding state $\boldsymbol{s'}$ for action $\boldsymbol{a}$ taken in state $\boldsymbol{s}$, and the *reward function* $R(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s'})$ which returns the scalar reward $r \in \mathbb{R}$ for the transition from state $\boldsymbol{s}$ performing action $\boldsymbol{a}$ and reaching the succeeding state $\boldsymbol{s'}$. The parameter $\gamma \in [0,1]$ sets the discount for future rewards. In reinforcement learning, the target is to learn a policy $\pi(\boldsymbol{a}|\boldsymbol{s})$ to maximize the expected discounted reward while performing a *trace* of length K through the so-called environment. In this work, we define a trace as one lane in which secondary tillage is performed. Thus, an entire field constitutes many traces of different lengths and therefore different K. For value-based reinforcement learning approaches for control problems, $Q^{\pi}(\boldsymbol{s},\boldsymbol{a})$ is the *state-action value* for the expected cumulative discounted reward following policy π under the transition function T at time t (Sutton and Barto 2018):

$$Q^{\pi}(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}_{\pi,T}\left[\sum_{k=0}^{K} \gamma^{k} \cdot r_{t+k} | \, S_t = \boldsymbol{s}, A_t = \boldsymbol{a}\right] \quad \text{(Eq. 6)}$$

The state-action value of state $\boldsymbol{s}$ while performing action $\boldsymbol{a}$ given the transition function T is recursively defined by the sum of the current reward and the discounted expected state-action value of the next state $\boldsymbol{s'}$ taking the next action $\boldsymbol{a'}$ due to following policy π. This relationship between state-action values can be expressed through the Bellman equation (Moerland et al. 2023). The target is to find an optimal policy that maximizes the expected state-action value (Moerland et al. 2023):

$$\pi^{*} = \arg\max_{\pi} Q^{\pi}(\boldsymbol{s}, \boldsymbol{a}) \quad \text{(Eq. 7)}$$

One way to find the optimal policy π^{*} is by estimating the Q-values through *temporal difference learning* where the intelligent agent updates the Q-estimates after each state transition using the reward to calculate a weighted average between the new and the old estimate. One approach is called Q-learning (Watkins and Dayan 1992), where the temporal difference error TD can be calculated as follows (Sutton and Barto 2018):

$$TD = R(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s'}) + \gamma \max_{\boldsymbol{a'}} Q(\boldsymbol{s'}, \boldsymbol{a'}) - Q(\boldsymbol{s}, \boldsymbol{a}) \quad \text{(Eq. 8)}$$

In *offline* reinforcement learning algorithms, a static dataset of transitions is collected in the target environment $D_{env} = \{(\boldsymbol{s}, \boldsymbol{a}, r, \boldsymbol{s'})\}$ and used to train the policy π. Following our model as introduced above, in this work the environment is considered to be the field and soil conditions plus the machines working conditions. Without a model of the environment, in offline reinforcement learning the training of the agent is limited to the actions actually performed and recorded during data collection, in our case in real field trials. In *model-based* reinforcement learning the dataset is further utilized to train a parameterized transition function $\hat{T}_{\Psi_{env}}$ of the simulated MDP $\widehat{M} = \{S, A, \hat{T}, R, \gamma\}$ which allows the agent to learn from a broader set of system states that can differ from the behavior as recorded in the real world (and used for offline training). In the learnt MDP, $\hat{T}_{\Psi_{env}}$ can be used to derive the next state and the reward of performed transitions (Yu et al. 2020). The rewards gained from $\widehat{M}$ should be pessimistic to avoid the policy purely exploiting the $\hat{T}_{\Psi_{env}}$ itself being a learned model. To

account for this, a penalty can be added to the reward function for the magnitude of uncertainty in transitions (Kidambi et al. 2020). This can be done by either adding an uncertainty penalty in the form of a reward scaling factor (Yu et al. 2020) or by using an uncertainty threshold and applying a penalty when exceeding this threshold (Kidambi et al. 2020). In many cases, it is useful to learn an ensemble of transition functions to infer the certainty of the predictions by the variance of the ensemble predictions. An ensemble of environment models is created by training multiple soil-machine response models (Boysen et al. 2023) on different subsets of training data with different random initializations. This also allows to draw the next state from a distribution of states to accommodate for the stochasticity of e.g. real-world scenarios (Nix and Weigend 1994, Lakshminarayanan et al. 2017). The transition to the next state is drawn by sampling from a normal distribution generated by the mean value and the variance of the ensemble predictions scaled by the hyperparameter κ_s, i.e., $\boldsymbol{s}' \sim N(\mu_{\boldsymbol{s}'}, \kappa_s \sigma_{\boldsymbol{s}'})$. The uncertainty measure $u(\boldsymbol{s},\boldsymbol{a})$ of the prediction is determined by the standard deviation of the ensemble predictions multiplied by a scaling factor κ_u.

$$\mu_{\boldsymbol{s}'} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{s}_i' ; \qquad u(\boldsymbol{s},\boldsymbol{a}) = \kappa_u \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{s}_i' - \mu_{\boldsymbol{s}'}\right)^2} \qquad \text{(Eq. 9)}$$

In our case, we trained an ensemble of $N = 10$ environment models Ψ_{env} on our dataset D_{env} with about 172,000 datapoints using supervised learning. These environment models serve the purpose of a transition function for the part of the state vector that would change as a consequence of an adaptation in the machine configuration. The training of the environment model ensemble uses different training seeds and therefore different splits of training and validation data as well as (partly) different initial parameters in the neural network architecture for each model. For more details of the training process we refer to our recent previous work on the soil-machine response model (Boysen et al. 2023). More specifically, the environment models each predict the resulting engine fuel rate fr_e, engine torque utilization M_e and the soil surface roughness in the back of the machine rc_b for the changed machine configuration as adapted by the reinforcement learning agent during training. By following this model-based approach, we allow the agent to explore much more state possibilities during simulation as are available in the data for offline training. The rest of the state, in our case, the working speed v_w in the resulting state $\boldsymbol{s'}$ is directly determined by $\boldsymbol{a}$ and $\boldsymbol{s}$ when $M_e \leq M_{e_{max}}$, otherwise it is reduced until $M_e \leq M_{e_{max}}$. Most of the other parts of the state vector (img_f, rc_f, n_e, n_{PTO}, ph_d, $slope$) are loaded from the dataset during the model-based training and simulation.

In addition to that, the model needs to identify and penalize invalid actions. In our scenario, invalid actions could occur e.g., due to exceeding maximum engine torque utilization or moving out of speed boundaries. We use a negative penalty scalar Φ for that. Therefore, we formalize our reward as follows: In case all predefined boundaries are met, the normalized utility of the current state is returned as payoff. Otherwise, the penalty scalar is returned in case of exceeding an engine torque utilization threshold $M_{e_{max}}$ or resulting working speeds outside the valid interval $[v_{w_{min}}, v_{w_{max}}]$. In both cases, the returned quantities are scaled using the truncated inverse of the uncertainty factor as defined above. More formally the reward function is defined according to equation 10:

$$r = \max\big(0, 1 - u(\boldsymbol{s}, \boldsymbol{a})\big) * \begin{cases} U(\boldsymbol{s})\ if\ valid \\ \Phi\ else \end{cases} \quad \text{(Eq. 10)}$$

For continuous state spaces (as in our real-world tillage use case) classical Q-Learning is not feasible due to its tabular mapping of state-action values. Therefore, we use Deep Q Networks (DQN) to approximate the Q-values of state-action pairs (Mnih et al. 2015). During training the agent will gather experiences by rolling out traces of the modelled MDP. In our setting, these traces comprise one tilled row. These experiences are stored in a *replay memory* buffer D_{rm} of size N_{rm}. Mini-batches of size N_b are sampled from this experience buffer to update the deep neural network-based policy models. In this approach, two deep neural networks are used: (1) a policy network (denoted θ) that is used during the loss optimization through backpropagation and (2) a target network θ^- that is initialized with the same weight parameters as the policy network (Mnih et al. 2015) and receives a soft update (weighted by τ) after each step as in Lillicrap et al. (2015). Both extensions to the standard Q-learning approach allow for a smoother and non-diverging learning process – for more detail of the mathematical reasons, the reader is referred to the original source (Lillicrap et al. 2015, Mnih et al. 2015). The policy network is updated to minimize the Huber loss *HL* (Huber 1992) which is less sensitive to outliers compared to standard squared error (or L2) loss and also calculated over the temporal difference *TD* (see TD calculation above) obtained through the target network and the received reward (Mnih et al. 2015):

$$L(\theta) = \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a},\boldsymbol{r},\boldsymbol{s}')\sim D_{rm}}\left[HL\left(r + \gamma \max_{\boldsymbol{a}'}(Q(\boldsymbol{s}', \boldsymbol{a}'; \theta^-)) - Q(\boldsymbol{s}, \boldsymbol{a}; \theta)\right)\right] \quad \text{(Eq. 11)}$$

By minimizing this loss function, we obtain a deep neural network-based estimator for the *Q*-values of all possible actions from our discrete action space. Subsequent to training completion a greedy policy then maximizes over the *Q*-value estimates, i.e., $\pi = \max_a Q(\boldsymbol{s}, \boldsymbol{a}; \theta)$. During training, an exploring learning scheme such as ϵ-greedy is chosen that chooses random actions with a probability of ϵ in every update step (potentially decaying this ϵ probability in the course of learning).

Integrating this learning approach into our MLOC system architecture for secondary tillage, such trained policies are added to the policy database on layer 2 and used in the action selector component in the layer 1 controller. Through the use of the environment model Ψ_{env}, the training of better policies can continue on the reflection layer 2. This can happen either reactively as a consequence of a detected knowledge gap, or rather proactively by continually simulating new situations which are not well covered in the system state history.

Simulation experiments

In the following case study, we want to demonstrate the most crucial part of our IAT system architecture which utilizes the soil-machine response environment model as introduced in Boysen et al. (2023) for training a reinforcement learning agent to control the working speed v_w of the tractor-im-

plement combination. We conduct the following study using in silico experiments based on a vast amount of real data collected in numerous field trials.

Accordingly, we act in the reflection layer of the proposed MLOC architecture where the RL-based control agent is trained. For the simulation experiments of our case study, the dataset as introduced in Boysen et al. (2023) consists of approximately 172,000 datapoints including 15 fields and a test field that has neither been used for the training of the environment model nor will be seen during the training of the DQN agent. This hold out field will be used for the validation part of this study. The simulated controller at layer 1 utilizing a DQN agent is asked to choose an action in each step of the simulation while the environment model and the dataset will serve to realistically replay state transition to the next state according to the action and previous state.

In our experiments, the optimization of the policy network is conducted after each step (state transition) followed by a soft update of the target network with factor τ = 0.005. During training, the actions are selected using ϵ-greedy policy with decay, i.e., either greedily by the policy network or at random with a chance of ϵ. During training we repeat every field in the dataset for ten times (excluding the holdout test field). ϵ decreases by $\epsilon = \epsilon_{start} * e^{-2*e*\frac{steps}{total\ steps}}$ with ϵ_{start} = 0.9 ; ϵ_{end} = 0.05 to reach ϵ_{end} at around halfway through training. Each field is divided into the single lanes and each of the lanes is denoted as a trace with different lengths (i.e., number of steps). Preliminary experiments determined three most important hyperparameters for which we conducted hyperparameter optimization using grid search. The search space includes $\alpha \in \{10^{-3}; 10^{-4}; 10^{-5}\}$, $\gamma \in \{0.1; 0.5; 0.9\}$ and $\alpha_{decay} \in \{0.2; 0.5; 1.0\}$. The weights in the combined utility function are set to β_1 = 0.5, β_2 = 0.25, β_3 = 0.25. This reflects the observations in our data that during secondary tillage both fuel consumption and area output can be improved through an increased working speed as also found by Kautzmann et al. (2012). Therefore, the seedbed quality evaluation criterion is weighted with 0.5 while the other targets are weighted with 0.25 each. Since the reward is a normalized value in the interval [0,1], the penalty is set to −0.5 to make sure the agent learns to perform valid actions. Further hyperparameters are set according to literature (N_b = 32 (Mnih et at. 2015) or if not applicable by preliminary studies as follows: κ_u = 1.0; κ_s = 0.1; N_{rm} = 1024.

The shared architecture of the policy network and the target network consists of four linear fully connected layers with batch normalization and ReLU activation function. Only the output layer which is predicting the Q-values for each action is neither normalized nor activated and inhabits three neurons according to the magnitude of the action space. The input size of the first layer corresponds to the size of a state vector $\boldsymbol{s}$ and is 2057.This flattened layer input size results from the length of the encoded input image representation from the environment model (size 2048) and the other nine components of the state vector $\boldsymbol{s}$ (size 9). The output size of the first layer is 2054 and the input and output size of the following two layers is 1024 while the last layer has an output size of three, i.e., the magnitude of the action space in our setting.

During the evaluation, the agent selects an action in an interval λ_a of three seconds (λ_a = 3*s)* to simulate a realistic frequency in which the tractor can be controlled assuming usual working speeds between 1 and 3 m/s. The policy network is fed with a batch of states collected over the interval λ_a to receive the Q-values of the batch. To accommodate for equal weighting of the states, the softmax function is applied over the Q-values of the actions of each state in the batch. The final action is then determined by maximizing over the mean Q-values activated by the softmax function. This has the advantage that more states can be included to determine the action during large intervals, e.g.,

λ_a = 3*s*. Alternatively, the interval of state observation could be adjusted to exactly match λ_a which would result in a single state mapped directly to a single action through the usual maximization step over the Q-values output by the policy network. In the real world, this approach is constrained by a time window in which the action needs to be calculated and target speeds needs to be reached before the soil surface captured on the RGB-D image in front of the tractor will reach the power harrow in the back of the machine.

In Figure 5, the learning progress of the DQN agents' training is depicted. We trained 10 DQN agents offline, which are initialized with different random seeds (i.e., we conducted 10 experiment repetitions) and let each agent learn for 10 epochs over the training dataset. All agents are trained with the best hyperparameters as found by the hyperparameter optimization experiment. The four plots show the development of the constituent average reward, as well as of the reward components area output, fuel consumption and roughness coefficient over the ten epochs through the training data. The average is calculated for each reward and a single agent. For each epoch, box plots indicate the distribution among the ten repetitions of agent training, with the orange line depicting the median as usual. Mean values with standard deviation for the different hyperparameter configurations are reported in Table 2 below. As formalized above, the agent's learnable parameters (i.e., the weights of the policy and target networks θ and θ^-) are optimized by gradient descent to minimize the temporal difference error (here in the form of Huber Loss) what in turn leads to agents maximizing the discounted cumulative reward after training. Figure 5 shows different learning curves of the trained agent. The upper left corner of Figure 5, depicts the distribution resulting from the 10 repetitions of the average reward received by the agent (y-axis) over each of the 10 training epochs (x-axis). The average reward ranges between 0.4 and 0.55 within the theoretical range of [0,1]. Over the ten training data epochs, the average reward increases steadily until epoch 6 and then converges. Overall, this demonstrates a successful learning progress. The other three plots show the isolated performance metrics of the tractor-implement control system: The average area output *AO* (upper right) is directly affected by the working speed and increases from around 2 ha/h to around 2.25 ha/h over the course of the training. The average fuel consumption *FC* depicted in the bottom left decreased from around 17.75 to 16.5 l/ha at the end of the training. The progression of fuel consumption and area output reflects the aforementioned observations in our data that during secondary tillage both fuel consumption and area output can be improved through an increased working speed. Remarkably, as can be seen in the lower right learning curve, the agent also decreased the average roughness coefficient during training from around 12.68 mm to around 12.62 mm. This observation shows that on average the agent learns to obtain higher reward by increasing the working speed during training on the simulated fields while still achieving a reduced roughness coefficient. These observations confirm the general suitability of the designed reward function. The distribution and outliers displayed in Figure 5 are expected due to the stochasticity in the agent training process.

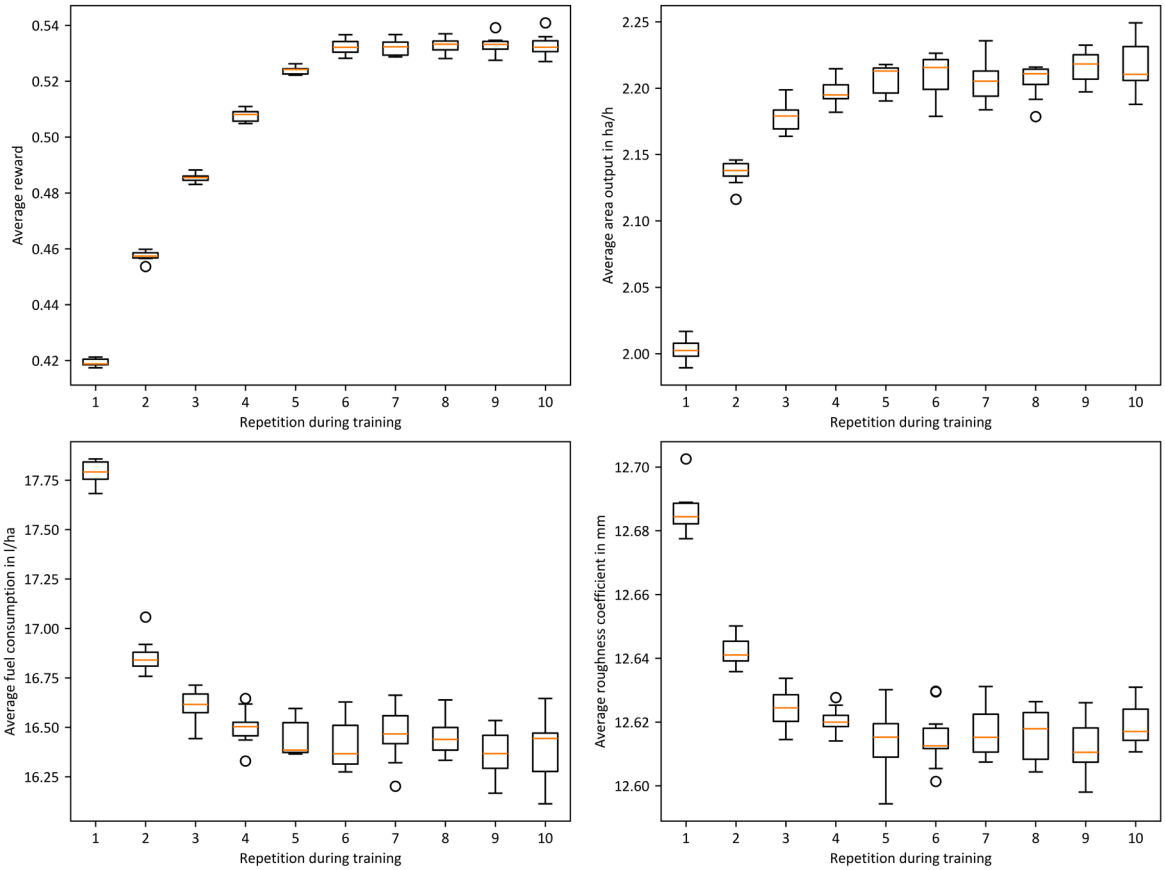

Figure 5: The training progress over ten epochs through the training data and for each epoch presented as box plots showing the distribution across agent training repetitions with different random seeds. Hyperparameters with highest overall evaluation utility have been chosen (Table 2). The agent training targets to maximize the average reward (top left) by increasing the average area output (top right) and at the same time decreasing average fuel consumption (bottom left) and roughness coefficient (bottom right).

Table 2 depicts the results of the evaluation of the best five hyperparameter sets in terms of average utility over ten experiment repetitions to also evaluate model stability. Each row displays the hyperparameters used for the training of the experiment, the average *FC*, *AO*, *SQ* and $U(\boldsymbol{s}, \boldsymbol{a})$. Reported are the means along with the standard deviation for each of the performance metrics. In the last row of the table, a simulated baseline is displayed for performance comparison with the different DQN agents. This baseline is set to accelerate to and keep a constant working speed of the tractor at 4.5 what represents common practice when drivers make use of cruise control. The evaluation is performed on the data of a 'holdout field' that was neither part of the DQN training data nor of the data used for training the environment model. The holdout field is the same as described in more detail in Boysen et al. (2023). It comprises positive and negative slopes as well as changing soil conditions. Bold entries indicate the best values in terms of the performance metrics for each column. The baseline with a constant speed of 4.5 km/h reached an average *FC* of 23.04 l/ha. The average values of five different agents trained with the five best found hyperparameters range from 22.46 to 23.3 l/ha showing high standard deviations up to 4.98 l/ha. Similarly, the average area output comes with a high standard deviation of up to 0.45 ha/h with average values ranging from 1.23 to 1.5 ha/h. In contrast, the baseline achieves an average of 1.34 ha/h. The average *SQ* is determined by the roughness

coefficient. Over all of the five agents we could observe a high variance in terms of achieved average *FC* and *AO* over the ten experiment repetitions. The average roughness coefficient lies between 13.37 and 13.43 mm with standard deviations up to 0.08 mm. The baseline results in an average roughness coefficient of 13.44 mm. Accordingly, it turns out that the approaches with higher average *AO* and similar *FC* could still achieve slightly lower roughness coefficients on average. In contrast to the baseline controller driving at constant speed, our agent has the possibility to continually adapt the working speed to the perceived state which includes the RGB and depth image of the soil surface. Our agent with the best found hyperparameter configuration (first row in Table 2) outperforms the baseline in two of the three constituent performance metrics (*FC* and *AO*) and achieves a comparable level of *SQ*. This is also reflected in the average utility (weighted sum of the normalized performance metrics) used as reward signal during training. With one exception, the other agents show similar performance compared to the baseline. The average utility of these four models ranges from 0.461 to 0.478 compared to the baseline of 0.451. Looking at the standard deviations, it turns out that the model training seems to be sensitive to the initialization of the agents (i.e., the weight initialization) as well as to the environment conditions (i.e., the sampled lane sequence from the overall dataset) randomized over the ten experiment repetitions. Accordingly, to draw a first conclusion of the achievable performance improvements using our DQN-based OC agent we now inspect the performance of selected policies among the repetitions which scored best in our experiments.

Table 2: Performance of the best DQN agents trained on the five best hyperparameter configurations found. Each set of hyperparameters is used to train an individual agent. The training is repeated for 10 times using 10 different random seeds. The agent results are compared to a baseline (4.5 km/h) reflecting common practice.

Hyperparameters	Average *FC* in l/ha [1)]	Average *AO* in ha/h [1)]	Average *SQ* in mm [1)]	Average $U(s, a)$ [1)]
$\alpha = 10^{-4}$ $\gamma = 0.5$ $\alpha_{decay} = 1.0$	**22.46** ± 3.34	**1.50** ± 0.33	13.43 ± 0.08	**0.478** ± 0.066
$\alpha = 10^{-3}$ $\gamma = 0.9$ $\alpha_{decay} = 0.2$	23.11 ± 2.49	1.41 ± 0.24	13.41 ± 0.04	0.462 ± 0.050
$\alpha = 10^{-3}$ $\gamma = 0.9$ $\alpha_{decay} = 0.5$	23.09 ± 4.98	**1.50** ± 0.45	13.43 ± 0.07	0.472 ± 0.095
$\alpha = 10^{-3}$ $\gamma = 0.9$ $\alpha_{decay} = 1.0$	25.97 ± 3.46	1.23 ± 0.30	13.40 ± 0.05	0.415 ± 0.064
$\alpha = 10^{-3}$ $\gamma = 0.5$ $\alpha_{decay} = 0.5$	23.30 ± 3.22	1.41 ± 0.25	**13.37** ± 0.04	0.461 ± 0.058
Baseline with constant speed (4.5 km/h)	23.04	1.34	13.44	0.451

[1)] ± 1SD standard deviation (n = 10).

In Table 3, we report the results from the selected top-3 agents from the overall 50 trained agents for a more detailed evaluation. These are again compared against the constant baseline. As before, bold values indicate the best scores. This analysis shows that selected, best performing agents are

indeed capable of generalizing to unseen data (i.e., fields) and can substantially increase the performance in the simulated environment. The best three agents achieve a *FC* ranging from 17.39 to 18.77 l/ha compared to the *FC* of 23.04 of the baseline. Thereby decreasing *FC* by values ranging from 18.5 % to 24.5 %. The *AO* is increased between 35.8 % to 50 % compared to the baseline. This results in values deviating from the baseline (1.34) with values from 1.82 to 2.05 ha/h for the best three agents. The average *SQ* evaluates to a range from 13.42 to 13.54 mm for the learning agents. In contrast to the baseline performance of 13.44 mm, this results in a difference from −0.2 to +0.6 % where smaller values are better. Eventually, we observed an increased overall utility compared to the baseline of 0.451. The DQN agents achieved utility values in the range of 0.583 to 0.590 increasing this score by 29.3 and 30.8 %, respectively. In summary, we can conclude that the best agents are capable of increasing *AO* while decreasing *FC* and still keep a similar seedbed quality in our simulated environment. Therefore, the overall utility can be substantially increased while the environment model keeps the machine configuration in realistic bounds according to simulated engine torque utilization.

Table 3: Performance of the best three models with the baseline in terms of fuel consumption *FC*, area output *AO*, seedbed quality *SQ* and utility *U*(***s***, ***a***)

Model	***FC*** **in l/ha**	***AO*** **in ha/h**	***SQ*** **in mm**	***U*(*s*, *a*)**
Top 1	17.46	2.01	13.51	**0.590**
Top 2	18.77	1.82	**13.42**	0.589
Top 3	**17.39**	**2.05**	13.54	0.583
Baseline with constant speed (4.5 km/h)	23.04	1.34	13.44	0.451

Discussion

Agricultural machines are increasingly equipped with sensors and computing devices and interconnected with telematics systems and cloud services to increase their degrees of operational autonomy and operator usability. At the same time, contemporary AI technology is now integrated into these systems to allow for even higher autonomy in performing agricultural processes. This in turn leads to increasing system complexity spanning across multiple technological scales beginning from the machine and field level (IoT), to the farm and inter-farm level (edge to fog), up to an even more global cyber-sphere level (cloud) – for a more detailed discussion on such scales in digital farming systems the reader is referred to Bökle et al. (2022). Accordingly, the (semi-)autonomous machines become part of an overarching system comprising more digital technology components such as other machines operating on the same fields, edge- to cloud-based IT-services and farm management information systems. It thus has to be acknowledged, that IAT and other digital agriculture solutions, more and more finding their way into practice, must be designed by following viable systems and software engineering principles that stress the importance of attaining controlled system autonomy and self-organization as well as technical resilience. We have claimed in this work that OC offers exactly such a systems thinking and design paradigm and thus constitutes a viable candidate for engineering future IAT systems. We corroborate our claim by the insights from a case study, whose results we further discuss in the following:

The design of an MLOC-based IAT architecture for our case study of an intelligent secondary tillage control system allows it to self-reflect on its current knowledge. By comparing the predicted outcomes with the actually observed state shortly after performing the selected actions, current knowledge gaps can be detected and handled. This allows for a robust system design able to counter detected knowledge gaps for instance by means of reactive finetuning of the various machine learning models employed over the reactive and reflection layers and eventually leads to continual agent self-improvement (Lüling et al. 2022). Regarding this possible self-improvement process a notable advantage of our approach is that no external data annotation by an expert is necessary for further finetuning of the involved machine learning models. Neither for the action selection policy at layer 1, nor for the environment model at layer 2. On the other hand, these models are machine-specific and thus require new training for a different setup. Fortunately, this problem can again be alleviated by exploiting the automatic annotation capability (ground-truth is automatically sensed while navigating the fields) of our approach. This allows to collect sufficient amounts of data in a short period of time without the need for manual preprocessing and annotation. But it still remains unclear how much data would actually be needed to sufficiently train a system for a specific tractor-implement combination and if data of similar combinations could be used to pretrain models.

The promising results obtained by training an intelligent agent for automatic tillage control in our in silico experiments yet need to be validated in field experiments. The leap in performance that is to be expected when reinforcement learning agents are transferred from simulation to the real world needs to be analyzed. As a first estimate of the achievable real-world performance, we can state that the environment model's performance on test data, which is based on observations under real field conditions, was found to have a relative root mean squared error (rRMSE) of between 12.5 to 13.6 % for the prediction of resulting roughness coefficient, engine torque utilization and engine fuel rate (Boysen et al. 2023). This error quantity evaluates lower than the performance increases as observed for the best agents in our study, when compared to the baseline which represents common practice in agriculture (Table 3). This suggests that the performance gain might be higher than the possible error of the simulation which is based on the current state of the environment model. Furthermore, the current observation of sensitivity to agent policy network initializations, found by analyzing repeated agent trainings with different random seeds, can be alleviated as follows: After training and continual fine tuning of several agents, policy selection (as done manually in our analysis in Table 3) from a policy database can be done automatically by the controller of the reflection layer, e. g., based on the utility of the selected goals or uncertainty estimates. Nevertheless, the analyzed sensitivity to policy network initializations should be tackled in future works.

While the use of a model-based reinforcement learning agent utilizing the soil-machine environment model comes to the cost of additional offline training time, it has the notable advantage of allowing to learn a global solution, i. e., a policy, suitable for mixed and continuous state spaces as well as different forms of action spaces. In the case of a DQN-agent as applied in this work, the focus was set on discrete action spaces. Other reinforcement learning algorithms such as e. g., Proximal Policy Optimization (Schulman et al. 2017) are capable of generating global solutions for continuous action spaces and can be straightforwardly integrated in our MLOC-based system architecture. A potential alternative design choice could be the immediate use of the environment model for conducting *planning* by calculating the predicted utility values for all possible next actions and selecting the highest scoring action. This approach thus yields a so-called local solution which is always only valid for

the current state and discarded afterwards (Moerland et al. 2023). According to Sutton and Barto (2018), this can be classified as *decision-time planning*, i.e., conducting a one-step-look-ahead, where the environment model is used to predict and evaluate possible next states for available actions. A similar approach from control theory called *model predictive control* is closely related to decision-time planning. This approach for the design of control systems has already been applied to various use cases in agriculture including irrigation systems, agricultural machinery, agricultural production and product processing as well as controlled environment agriculture (Ding et al. 2018). For instance, one related work by Backman et al. (2012) is proposing to use model predictive control on agricultural machinery for navigation of a tractor with a trailed seed drill. However, for larger action spaces, the computational demand increases since the outcome of each possible next action needs to be predicted, but for reasonable small action spaces, decision-time planning could be a suitable choice for the action selector in layer 1 of our system architecture. Recently, conventional control designs have been used in the domain of tillage. Riegler-Nurscher et al. (2020) use the roughness coefficient as feedback for a proportional control system in a closed loop controller to reach a target roughness coefficient during secondary tillage. Closely related, Mohammadi et al. (2022) control the soil shield of a tiller to reach a roughness coefficient within a certain target range. In contrast to this work, both used the roughness coefficient as their only target for the controller. Our work increases the complexity by also adding fuel consumption and area output as targets for optimization thus generating the demand for more complex control design choices. Further alternative design choices comprise the implementation of a fuzzy controller as done e.g., in the work of Heiss et al. (2022) for variable rate nitrogen application or the application of rule-based learning systems such as the XCS classifier system as done e.g., in the context of urban traffic control by different authors (Prothmann et al. 2011).

Our study so far focuses on the adoption of an OC approach to render a complex agricultural machine for secondary tillage an IAT. The set scope is therefore on the technological level of machines on the fields which we deem important to be addressed as a first step. We proclaim that OC fits the extended scope on higher technological levels (farm, inter-farm, etc.) as well, but the discussion would fall beyond the scope of this work. Accordingly, we let this elaboration be part of our future work.

Conclusions

Past and current developments, starting from mechanization, over computer-aided and sensor-based automation to the rapidly progressing digital transformation, have each led to substantial technological advancements in agricultural production. The most recent advancements manifest in connected highly automated agricultural machines and robots performing agricultural processes (semi-)autonomously by leveraging latest breakthroughs in AI. By moving more and more decision autonomy, self-adaptivity and self-organization capabilities to these systems, we arrive at a new technological level, which we define as Intelligent Agricultural Technology (IAT) in this paper. To obtain broad acceptance of emerging deeply digitized solutions, it must be systematically dealt with both perceived concerns and technical issues occurring through the integration of current agricultural engineering practice with latest information technology approaches at several technological scales. We argued that today's AgTech systems and future IAT systems thus are becoming increasingly complex, what demands for new ways of system design and architectures that accommodate for resilient operations at the machine and at overarching superordinate system levels such as the farm, inter-farm up to entirely virtual levels digitally connecting many actors and their systems. As a systems engineering

discipline, OC provides principled methodologies and frameworks for building such complex systems deployed in real-world environments. OC's methodological repertoire strongly draws from natural computation and AI, utilizing for instance swarm intelligence algorithms to approach dynamic and multi-objective optimization problems, or machine learning to enable such systems to continually learn and self-adapt.

Building on this methodological repertoire, in a case study we designed a new multi-level system architecture derived from the generic observer/controller model and integrated a modern deep reinforcement learning method for application to the technical system of a tractor-implement combination performing secondary tillage. Based on in silico experiments, we showcased that based on rigorous formal modeling that allows the integration of suitable learning components into an OC system, it is generally possible to embed higher levels of intelligence in complex agricultural machines. This opens possibilities for future autonomous control of complex agricultural machinery what can, next to enhancing process efficiency, also strongly relieve machine operators by improving the overall operability. Yet our findings serve as starting point for commencing further interdisciplinary research for paving the way to future-proof IAT. Further research is needed to generalize our initial findings to even more complex system levels, incorporating for instance multiple cooperating machines or systems connected across spatial and technological scales stressing the importance of the self-organization property of OC systems. Furthermore, analyzing the extent of the simulation to reality gap, its impact on the transfer of RL-based controllers to various field conditions, as well as a comparison to more traditional approaches to building control systems such as model predictive control will be investigated in future work. In addition, concretizing and developing mechanisms on the reflective layer of our proposed architecture to proactively alleviate expectable performance declines constitutes a top priority on our research agenda.

References

Adam, K.M.; Erbach, D.C. (1992): Secondary Tillage Tool Effect on Soil Aggregation. Transactions of the ASAE 35(6), pp. 1771–1776, https://doi.org/10.13031/2013.28796

Albiero, D.; Pontin Garcia, A.; Kiyoshi Umezu, C.; Leme de Paulo, R. (2022): Swarm robots in mechanized agricultural operations: A review about challenges for research. Computers and Electronics in Agriculture 193, p. 106608, https://doi.org/10.1016/j.compag.2021.106608

Allerding, F.; Becker, B.; Schmeck, H. (2011): Decentralised Energy Management for Smart Homes. In: Organic Computing – A Paradigm Shift for Complex Systems. Eds. Müller-Schloer, C.; Schmeck, H. et al., Basel, Birkhäuser, pp. 605–607

Backman, J.; Oksanen, T.; Visala, A. (2012): Navigation system for agricultural machines: Nonlinear model predictive path tracking. Computers and Electronics in Agriculture 82, pp.32-43, https://doi.org/10.1016/j.compag.2011.12.009

Bechar, A.; Vigneault, C. (2016): Agricultural robots for field operations: Concepts and components. Biosystems Engineering 149, pp. 94–111, https://doi.org/10.1016/j.biosystemseng.2016.06.014

Bernauer, A.; Zeppenfeld, J.; Bringmann, O.; Herkersdorf, A.; Rosenstiel, W. (2011): Combining Software and Hardware LCS for Lightweight On-chip Learning. In: Organic Computing – A Paradigm Shift for Complex Systems. Eds. Müller-Schloer, C.; Schmeck, H. et al., Basel, Birkhäuser, pp. 253–265

Boden, M.A. (2018): Artificial intelligence. A very short introduction, Oxford, United Kingdom, Oxford University Press

Bökle, S.; Paraforos, D.S.; Reiser, D.; Griepentrog, H.W. (2022): Conceptual framework of a decentral digital farming system for resilient and safe data management. Smart Agricultural Technology 2, p. 100039, https://doi.org/10.1016/j.atech.2022.100039

Boubin, J.; Chumley, J.; Stewart, C.; Khanal, S. (2019): Autonomic Computing Challenges in Fully Autonomous Precision Agriculture. In: 2019 IEEE International Conference on Autonomic Computing (ICAC), 16.06.2019–20.06.2019, Umea, Sweden, IEEE, pp. 11–17

Boysen, J.; Zender, L.; Stein, A. (2023): Modeling the soil-machine response of secondary tillage: A deep learning approach. Smart Agricultural Technology 6, p. 100363, https://doi.org/10.1016/j.atech.2023.100363

Braunack, M.V.; Dexter, A.R. (1989): Soil aggregation in the seedbed: A review. I. Properties of aggregates and beds of aggregates. Soil and Tillage Research 14(3), pp. 259–279, https://doi.org/10.1016/0167-1987(89)90013-5

CEMA (2017): Digital Farming: what does it really mean? And what is the vision of Europe's farm machinery industry for Digital Farming? https://www.cema-agri.org/images/publications/position-papers/CEMA_Digital_Farming_-_Agriculture_4.0__13_02_2017_0.pdf, accessed on 10 July 2024

Ding, Y.; Wang, L.; Li, Y.; Li, D. (2018): Model predictive control and its application in agriculture: A review. Computers and Electronics in Agriculture 151, pp.104-117, https://doi.org/10.1016/j.compag.2018.06.004

DLG e.V. (2018): Digitale Landwirtschaft. Ein Positionspapier der DLG https://www.dlg.org/fileadmin/downloads/landwirtschaft/themen/ausschuesse_facharbeit/DLG_Position_Digitalisierung.pdf, accessed on 10 July 2024

Fisch, D.; Janicke, M.; Sick, B.; Muller-Schloer, C. (2010): Quantitative Emergence—A Refined Approach Based on Divergence Measures. In: 2010 4th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO), 27.09.2010 - 01.10.2010, Budapest, Hungary, IEEE, pp. 94–103

Fountas, S.; Espejo-García, B.; Kasimati, A.; Gemtou, M.; Panoutsopoulos, H.; Anastasiou, E. (2024): Agriculture 5.0: Cutting-Edge Technologies, Trends, and Challenges. IT Professional 26(1), pp. 40–47, https://doi.org/10.1109/MITP.2024.3358972

Fountas, S.; Mylonas, N.; Malounas, I.; Rodias, E.; Hellmann Santos, C.; Pekkeriet, E. (2020): Agricultural Robotics for Field Operations. Sensors (Basel, Switzerland) 20(9), https://doi.org/10.3390/s20092672

Groher, T.; Heitkämper, K.; Umstätter, C. (2020): Digital technology adoption in livestock production with a special focus on ruminant farming. Animal 14(11), pp. 2404-2413, https://doi.org/10.1017/S1751731120001391

Guérif, J.; Richard, G.; Dürr, C.; Machet, J.; Recous, S.; Roger-Estrade, J. (2001): A review of tillage effects on crop residue management, seedbed conditions and seedling establishment. Soil and Tillage Research 61(1-2), pp. 13–32, https://doi.org/10.1016/S0167-1987(01)00187-8

Hartmann, J.; Stechele, W.; Maehle, E. (2013): Self-adaptation for Mobile Robot Algorithms Using Organic Computing Principles. In: Architecture of Computing Systems – ARCS 2013. Eds. Hutchison, D.; Kanade, T. et al., Berlin, Heidelberg, Springer Berlin Heidelberg, pp. 232–243

Heiß, A.; Paraforos, D.S.; Sharipov, G.M.; Griepentrog, H.W. (2022): Real-time control for multi-parametric data fusion and dynamic offset optimization in sensor-based variable rate nitrogen application. Computers and Electronics in Agriculture 196, p.106893, https://doi.org/10.1016/j.compag.2022.106893

Herlitzius, T.; Hengst, M.; Grosa, A.; Fichtl, H. (2021): Fieldswarm technology for tillage and crop care. at – Automatisierungstechnik 69(4), pp. 316–324, https://doi.org/10.1515/auto-2020-0127

Huber, P.J. (1992): Robust Estimation of a Location Parameter. In: Breakthroughs in Statistics. Eds. Kotz, S.; Johnson, N.L., New York, NY, Springer New York, pp. 492–518

ISPA (2024): Precision Ag Definition, International Society of Precision Agriculture https://www.ispag.org/about/definition (last accessed 17.12.2024).

Kautzmann, T.; Geimer, M.; Wünsche, M.; Mostaghim, S.; Schmeck, H. (2012): Organic Computing lernt nie aus. Mobile Maschinen 4, pp. 36–38, https://doi.org/10.5445/IR/1000029516

Kautzmann, T.; Wünsche, M.; Bliesener, M.; Mostaghim, S.; Geimer, M.; Schmeck, H. (2010): Selbstadaptierendes und lernfähiges Management für mobile Arbeitsmaschinen. agricultural engineering.eu 65(4), pp. 240–243, https://doi.org/10.15150/lt.2010.491

Kephart, J.O.; Chess, D.M. (2003): The vision of autonomic computing. Computer 36(1), pp. 41–50, https://doi.org/10.1109/MC.2003.1160055

Kidambi, R.; Rajeswaran, A.; Netrapalli, P.; Joachims, T. (2020): MOReL: Model-Based Offline Reinforcement Learning. In: NIPS '20: Proceedings of the 34th International Conference on Neural Information Processing Systems, 06.–12.12.2020, Vancouver BC Canada, Curran Associates Inc, pp. 21810–21823

Köller, K.; Hensel, O. (2019): Verfahrenstechnik in der Pflanzenproduktion. Stuttgart, Deutschland, utb GmbH

Kounev, S.; Kephart, J.O.; Milenkoski, A.; Zhu, X. (Eds.) (2017): Self-Aware Computing Systems. Cham, Springer International Publishing

Krupitzer, C.; Stein, A. (2024): Unleashing the Potential of Digitalization in the Agri-Food Chain for Integrated Food Systems. Annual review of food science and technology 15(1), pp. 307–328, https://doi.org/10.1146/annurev-food-012422-024649

Lakshminarayanan, B.; Pritzel, A.; Blundell, C. (2017): Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In: NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems, 4.-9.12.17, Long Beach California USA, Curran Associates, Inc, pp. 6405–6416

Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. (2015): Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971, https://doi.org/10.48550/arXiv.1509.02971

Lüling, N.; Boysen, J.; Kuper, H.; Stein, A. (2022): A Context Aware and Self-improving Monitoring System for Field Vegetables. In: Architecture of Computing Systems. ARCS 2022, 13.–15.09.2022, Heidelberg, Germany, Springer International Publishing, pp. 226–240

Macenski, S.; Foote, T.;Gerkey, B.; Lalancette, C.; Woodall, W. (2022): Robot Operating System 2: Design, architecture, and uses in the wild. Science Robotics 7, https://doi.org/10.1126/scirobotics.abm6074

Mauser, I.; Hirsch, C.; Kochanneck, S.; Schmeck, H. (2015): Organic Architecture for Energy Management and Smart Grids. In: 2015 IEEE International Conference on Autonomic Computing (ICAC), 07.–10.07.2015, Grenoble, France, IEEE, pp. 101–108

Mnif, M.; Richter, U.; Branke, J.; Schmeck, H.; Müller-Schloer, C. (2007): Measurement and Control of Self-organised Behaviour in Robot Swarms. In: Architecture of Computing Systems – ARCS 2007, 12.–15.03.2007, Zurich, Switzerland, Springer, pp. 209–223

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; Hassabis, D. (2015): Human-level control through deep reinforcement learning. Nature 518(7540), pp. 529–533, https://doi.org/10.1038/nature14236

Modak, S.; Stein, A. (2024): Synthesizing Training Data for Intelligent Weed Control Systems Using Generative AI. In: Architecture of Computing Systems – ARCS 2024, Springer International Publishing, 14.-16.05.2024, Potsdam, Germany, Springer, https://doi.org/10.1007/978-3-031-66146-4_8

Moerland, T.M.; Broekens, J.; Plaat, A.; Jonker, C.M. (2023): Model-based Reinforcement Learning: A Survey. Foundations and Trends® in Machine Learning 16(1), pp. 1–118, https://doi.org/10.1561/2200000086

Mohammadi, F.; Maleki, M.R.; Khodaei, J. (2022): Control of variable rate system of a rotary tiller based on real-time measurement of soil surface roughness. Soil and Tillage Research 215, p.105216, https://doi.org/10.1016/j.still.2021.105216

Mostaghim, S.; Schmeck, H.; Wünsche, M.; Geimer, M.; Kautzmann, T. (2011): Organic Computing in Off-highway Machines. In: Organic Computing – A Paradigm Shift for Complex Systems. Eds. Müller-Schloer, C.; Schmeck, H. et al., Basel, Birkhäuser, pp. 601–603

Müller-Schloer, C.; Schmeck, H.; Ungerer, T. (Eds.) (2011): Organic Computing – A Paradigm Shift for Complex Systems. Basel, Birkhäuser

Müller-Schloer, C.; Tomforde, S. (2017a): Applications. In: Organic Computing – Technical Systems for Survival in the Real World, Cham, Birkhäuser, pp. 429–547

Müller-Schloer, C.; Tomforde, S. (2017b): Building Organic Computing Systems. In: Organic Computing - Technical Systems for Survival in the Real World, Cham, Birkhäuser, pp. 171–258

Müller-Schloer, C.; Tomforde, S. (2017c): Distributed Smart Cameras. In: Organic Computing - Technical Systems for Survival in the Real World, Cham, Birkhäuser, pp. 486–496

Müller-Schloer, C.; Tomforde, S. (2017d): Organic Computing - Technical Systems for Survival in the Real World. Cham, Birkhäuser

Müller-Schloer, C.; Tomforde, S. (2017e): The Major Context. In: Organic Computing - Technical Systems for Survival in the Real World, Cham, Birkhäuser, pp. 549–572

Nix, D.A.; Weigend, A.S. (1994): Estimating the mean and variance of the target probability distribution. In: Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94), 28.06.–02.07.1994, Orlando, FL, USA, IEEE, 55–60 vol.1

Oroojlooy, A.; Hajinezhad, D. (2023): A review of cooperative multi-agent deep reinforcement learning. Applied Intelligence 53(11), pp. 13677–13722, https://doi.org/10.1007/s10489-022-04105-y

Pacher, M.; Brinkschulte, U. (2020): Evaluation of the dependability of an artificial DNA in a dynamic automotive environment. Journal of Systems Architecture 107, p. 101736, https://doi.org/10.1016/j.sysarc.2020.101736

Paraforos, D.S.; Griepentrog, H.W. (2021): Digital Farming and Field Robotics: Internet of Things, Cloud Computing, and Big Data. In: Fundamentals of Agricultural and Field Robotics. Eds. Karkee, M.; Zhang, Q., Cham, Springer International Publishing, pp. 365–385

Prothmann, H.; Tomforde, S.; Branke, J.; Hähner, J.; Müller-Schloer, C.; Schmeck, H. (2011): Organic Traffic Control. In: Organic Computing – A Paradigm Shift for Complex Systems. Eds. Müller-Schloer, C.; Schmeck, H. et al., Basel, Birkhäuser, pp. 431–446

Ragaveena, S.; Shirly Edward, A.; Surendran, U. (2021): Smart controlled environment agriculture methods: a holistic review. Reviews in Environmental Science and Bio/Technology 20, pp. 887–913, https://doi.org/10.1007/s11157-021-09591-z

Reif, W.; Anders, G.; Seebach, H.; Steghöfer, J.-P.; André, E.; Hähner, J.; Müller-Schloer, C.; Ungerer, T. (2016): Trustworthy Open Self-Organising Systems. Cham, Springer International Publishing

Riegler-Nurscher, P.; Moitzi, G.; Prankl, J.; Huber, J.; Karner, J.; Wagentristl, H.; Vincze, M. (2020): Machine vision for soil roughness measurement and control of tillage machines during seedbed preparation. Soil and Tillage Research 196, p. 104351, https://doi.org/10.1016/j.still.2019.104351

Rudolph, S.; Tomforde, S.; Hähner, J. (2019): Mutual Influence-aware Runtime Learning of Self-adaptation Behavior. ACM Transactions on Autonomous and Adaptive Systems 14(1), pp. 1–37, https://doi.org/10.1145/3345319

Russell, S.J.; Norvig, P.; Chang, M.-W. (op. 2022): Artificial intelligence. A modern approach. Fourth edition, Harlow, Pearson Education

Sandri, R.; Anken, T.; Hilfiker, T.; Sartori, L.; Bollhalder, H. (1998): Comparison of methods for determining cloddiness in seedbed preparation. Soil and Tillage Research 45(1), pp. 75–90, https://doi.org/10.1016/S0167-1987(97)00071-8

Schmeck, H.; Müller-Schloer, C.; Çakar, E.; Mnif, M.; Richter, U. (2010): Adaptivity and self-organization in organic computing systems. ACM Transactions on Autonomous and Adaptive Systems 5(3), pp. 1–32, https://doi.org/10.1145/1837909.1837911

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. (2017): Proximal Policy Optimization Algorithms. arXiv preprint arXiv:1707.06347, https://doi.org/10.48550/arXiv.1707.06347

Sommer, M.; Tomforde, S.; Hähner, J. (2016): An Organic Computing Approach to Resilient Traffic Management. In: Autonomic Road Transport Support Systems. Eds. McCluskey, T.L.; Kotsialos, A. et al., Cham, Springer International Publishing, pp. 113–130

Stein, A. (2017): Reaction Learning. In: Organic Computing – Technical Systems for Survival in the Real World, in Chapter 7 Basic Methods, Cham, Birkhäuser, pp. 287–328, https://link.springer.com/chapter/10.1007/978-3-319-68477-2_7

Stein, A.; Tomforde, S. (2021): Reflective Learning Classifier Systems for Self-Adaptive and Self-Organising Agents. In: 2021 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C), 27.09–01.10.2021, DC, USA, IEEE, pp. 139–145

Stein, A.; Tomforde, S.; Botev, J.; Lewis, P.R. (2021): Lifelike Computing Systems. In: Lifelike Computing Systems Workshop 2021, 19.07.2021, Prague, Czech Republic (held entirely online), CEUR Workshop Proceedings

Stein, A.; Tomforde, S.; Diaconescu, A.; Hähner, J.; Müller-Schloer, C. (2018): A Concept for Proactive Knowledge Construction in Self-Learning Autonomous Systems. In: 2018 IEEE 3rd International Workshops on Foundations and Applications of Self* Systems (FAS*W), 03.–07.09.2018, Trento, Italy, IEEE, pp. 204–213

Stein, A.; Tomforde, S.; Rauh, D.; Hähner, J. (2016): Dealing with Unforeseen Situations in the Context of Self-Adaptive Urban Traffic Control: How to Bridge the Gapp. In: 2016 IEEE International Conference on Autonomic Computing (ICAC), 17.–22.07.2016, Würzburg, IEEE, pp. 167–172

Sutton, R.S.; Barto, A.G. (2018): Reinforcement learning: An introduction. MIT press

Tomforde, S.; Brameshuber, A.; Hähner, J.; Muller-Schloer, C. (2011a): Restricted on-line learning in real-world systems. In: 2011 IEEE Congress on Evolutionary Computation (CEC), 05.–08.06.2011, New Orleans, LA, USA, IEEE, pp. 1628–1635

Tomforde, S.; Hurling, B.; Hähner, J. (2011b): Distributed Network Protocol Parameter Adaptation in Mobile Ad-Hoc Networks. In: Informatics in Control, Automation and Robotics. Eds. Cetto, J.A.; Ferrier, J.-L. et al., Berlin, Heidelberg, Springer Berlin Heidelberg, pp. 91–104

Tomforde, S.; Müller-Schloer, C. (2014): Incremental design of adaptive systems. Journal of Ambient Intelligence and Smart Environments 6(2), pp. 179–198, https://doi.org/10.3233/AIS-140252

Tomforde, S.; Prothmann, H.; Branke, J.; Hähner, J.; Mnif, M.; Müller-Schloer, C.; Richter, U.; Schmeck, H. (2011c): Observation and Control of Organic Systems. In: Organic Computing – A Paradigm Shift for Complex Systems. Eds. Müller-Schloer, C.; Schmeck, H. et al., Basel, Birkhäuser, pp. 325–338

Tomforde, S.; Sick, B.; Müller-Schloer, C. (2017): Organic Computing in the Spotlight. arXiv preprint arXiv:1701.08125, https://doi.org/10.48550/arXiv.1701.08125

Varani, M.; Mattetti, M.; Molari, G.; Biglia, A.; Comba, L. (2023): Correlation between power harrow energy demand and tilled soil aggregate dimensions. Biosystems Engineering 225, pp. 54–68, https://doi.org/10.1016/j.biosystemseng.2022.11.008

VDE; GI e.V. (2003): Organic Computing. Computer- und Systemarchitektur im Jahr 2003. Positionspapier https://fg-oc.gi.de/fileadmin/FG/OC/Publications/VDE-ITG-GI-Positionspapier_Organic_Computing.pdf, accessed on 10 July 2024

Vougioukas, S.G. (2019): Agricultural Robotics. Annual Review of Control, Robotics, and Autonomous Systems 2(1), pp. 365–392, https://doi.org/10.1146/annurev-control-053018-023617

Watkins, C.J.; Dayan, P. (1992): Q-learning. Machine learning 8, pp.279-292, https://doi.org/10.1007/BF00992698

Wünsche, M.; Mostaghim, S.; Schmeck, H.; Kautzmann, T.; Geimer, M. (2010): Organic Computing in Off-highway Machines. In: ICAC '10: 7th International Conference on Autonomic Computing, 07.06.2010, Washington DC USA, Association for Computing Machinery, pp. 51–58

Yu, T.; Thomas, G.; Yu, L.; Ermon, S.; Zou, J.Y.; Levine, S.; Finn, C.; Ma, T. (2020): MOPO: Model-based Offline Policy Optimization. In: NIPS'20: 34th International Conference on Neural Information Processing Systems, 06-12.12.2020, Vancouver BC Canada, Curran Associates, Inc, pp. 14129–14142

Zeppenfeld, J.; Herkersdorf, A. (2011): Applying autonomic principles for workload management in multi-core systems on chip. In: ICAC '11: 8th International Conference on Autonomic Computing, 14.–18.06.2011, Karlsruhe Germany, ACM, p. 3–10

Authors

Jun.-Prof. Dr. Anthony Stein is Head of Department of Artificial Intelligence in Agricultural Engineering and **Jonas Boysen, M.Sc.**, is a Research Assistant at the Department of Artificial Intelligence in Agricultural Engineering, Institute of Agricultural Engineering, University of Hohenheim, Garbenstr. 9, 70599 Stuttgart. E-mail: anthony.stein@uni-hohenheim.de

Notes and Acknowledgements

Parts of the research presented in this work is supported by funds of the Federal Ministry of Food and Agriculture (BMEL) based on a decision of the Parliament of the Federal Republic of Germany. The Federal Office for Agriculture and Food (BLE) provides coordinating support for artificial intelligence (AI) in agriculture as funding organization, grant number 28DK109A20.