frontiers in Robotics and AI

ORIGINAL RESEARCH
published: 17 November 2016
doi: 10.3389/frobt.2016.00067

An Organic Computing Approach to Self-Organizing Robot Ensembles

Sebastian von Mammen[1], Sven Tomforde[2] and Jörg Hähner[1]*

[1] Organic Computing, Faculty of Applied Informatics, Institute of Computer Science, University of Augsburg, Augsburg, Germany, [2] Intelligent Embedded Systems Group, University of Kassel, Kassel, Germany

Similar to the Autonomous Computing initiative, which has mainly been advancing techniques for self-optimization focusing on computing systems and infrastructures, Organic Computing (OC) has been driving the development of system design concepts and algorithms for self-adaptive systems at large. Examples of application domains include, for instance, traffic management and control, cloud services, communication protocols, and robotic systems. Such an OC system typically consists of a potentially large set of autonomous and self-managed entities, where each entity acts with a local decision horizon. By means of cooperation of the individual entities, the behavior of the entire ensemble system is derived. In this article, we present our work on how autonomous, adaptive robot ensembles can benefit from OC technology. Our elaborations are aligned with the different layers of an observer/controller framework, which provides the foundation for the individuals' adaptivity at system design-level. Relying on an extended Learning Classifier System (XCS) in combination with adequate simulation techniques, this basic system design empowers robot individuals to improve their individual and collaborative performances, e.g., by means of adapting to changing goals and conditions. Not only for the sake of generalizability but also because of its enormous transformative potential, we stage our research in the domain of robot ensembles that are typically comprised of several quad-rotors and that organize themselves to fulfill spatial tasks such as maintenance of building facades or the collaborative search for mobile targets. Our elaborations detail the architectural concept, provide examples of individual self-optimization as well as of the optimization of collaborative efforts, and we show how the user can control the ensembles at multiple levels of abstraction. We conclude with a summary of our approach and an outlook on possible future steps.

Keywords: organic computing, adaptive systems, observer/controller architecture, robot ensembles, evolutionary robotics, classifier systems

OPEN ACCESS

Edited by:
Aleš Zamuda, University of Maribor, Slovenia

Reviewed by:
Yara Khaluf, Universiteit Gent, Belgium
Phil Ayres, Centre for Information Technology and Architecture – CITA/KADK, Denmark

****Correspondence:***
Sebastian von Mammen
sebastian.von.mammen@informatik.uni-augsburg.de

Specialty section:
This article was submitted to Computational Intelligence, a section of the journal Frontiers in Robotics and AI

Received: *25 May 2016*
Accepted: *21 October 2016*
Published: *17 November 2016*

Citation:
von Mammen S, Tomforde S and Hähner J (2016) An Organic Computing Approach to Self-Organizing Robot Ensembles. Front. Robot. AI 3:67. doi: 10.3389/frobt.2016.00067

In order to benefit from an ever more complex technical environment, its behavioral autonomy needs to increase appropriately as well. Only then may it serve its users without requiring overwhelming amounts of attention. At the same time, a technical system is expected to offer appropriate access for controlling its individual components as well as its global goals. The control of robotic ensembles lends itself well to elucidate this challenge: ideally, the user would communicate his goals to the ensemble as a whole, without the need of micromanaging each of the individuals' parameters and interaction relationships. For instance, the user might navigate a flock of flight-enabled robotic units toward a building and make them work on facade maintenance, e.g., scrapping off paint, cleaning windows, or trimming greenery. For this to work, a line of command has to be established that links several levels of the system's design concept – the

user needs to communicate target and task to the ensemble, and the individuals communicate to coordinate their efforts. In addition, each individual needs to learn how it can contribute to the newly posed, global goals, and how it can maximize its contribution.

The field of Organic Computing (OC) aims at translating well-evolved principles of biological systems to engineering complex system design (Müller-Schloer et al., 2011). It provides the theoretical underpinnings to quantitatively capture system attributes such as their autonomy and robustness, or processes of emergence based on measures of entropy. It also promotes complex system design by means of a universal, observer/controller-based concept for adaptive, self-organizing behavior (Tomforde et al., 2011). With respect to robotics, OC research initially focused on failure tolerant and robust hardware concepts, mainly applied to multi-legged walking machines. The most prominent example is the *Organic Robot Control Architecture* (ORCA), see Brockmann et al. (2005) and Mösch et al. (2006). In ORCA, two kinds of behavioral modules are discerned. *Basic Control Units* (BCUs) implement the core behavior of the robot, rendering it fully functional with respect to the range of possible tasks. In addition, *Organic Control Units* (OCUs) observe and modify the BCUs' configuration during runtime (Hestermeyer et al., 2004). The separation between a system's basic and its extended functionality has proven itself numerous times – the sympathetic and the parasympathetic division of the human autonomous nervous system may serve as a famous biological example.

Similar to ORCA, we follow an OC approach to self-organizing robotic systems. In our approach, each agent in a robotic ensemble implements a multilayered observer/controller (O/C) system design concept (Tomforde and Müller-Schloer, 2014; Tomforde et al., 2016) that allows for local, and in unison, global optimization of the ensemble's behavior. The user interface is explicitly included as one layer that accepts modifications of the global and local goals. The main contributions of this article are as follows: (1) we present a novel design concept originating in the OC domain to allow for self-adaptive and self-optimizing robot behavior at runtime, (2) demonstrate the applicability of this concept in real-world applications, and (3) present a user-oriented interaction mechanism to control robot ensemble and their individuals in an intuitive manner.

We present the details of the multilayered O/C design concept of a single robot individual, and we explain how it works in organizing ensembles (Section 1). In Section 2, we provide examples of the reactive, self-regulatory capacity of the O/C concepts. Section 3 highlights the longer-term evolution of collaborative behavior, and Section 4 demonstrates the workings of the user interfacing layer of the design concept. We provide links to related works in the respective sections, and we conclude with a brief summary and an outlook on future work.

1. SEMINAL PRECEDING WORKS

As mentioned in the introduction, our approach relies on an architectural setup similar to ORCA. Therefore, we first reinforce the link between our approach and ORCA and related works. Next, we build on these analogs to preceding works to detail our approach – from the perspective of a generic design concept as well as of its concrete implementation.

1.1. From Individuals to Ensembles

In ORCA, the Organic Control Units change the system under observation and control (SuOC) based on periodically issued *health signals*, i.e., messages from the Basic Control Units indicating their functional working state. In contrast, our approach observes all kinds of available data about the SuOC. An according *observation model* specifies exactly, which input data, configuration parameters, or internally computed results of the SuOC are passed on to the observer/controller layer. ORCA's restrictive policy of data retrieval matches its fairly conservative array of options for changing the system. Few choices, however, drastically limit the system's configuration space and thus promote ORCA's primary design goals of (a) unearthing an optimal learning guideline for adaptation ("the law of adaptation") and of (b) protecting acquired knowledge against corruption and maintaining its validity and consistency (Brockmann et al., 2011).

The ORCA approach is further limited to single, isolated robots – information exchange with other robots or collaborative efforts among robotic teams were not envisaged in the original design concept. Yet, it has been shown that observer/controller-driven robots can increase their learning speed imitating each other (Jungmann et al., 2011). Local communication between robots allows for establishing real teams that collaboratively perform tasks such as the exploration of unknown terrain, and that assign each other subtasks in a fair manner – decentralized, without the need for global control (Brandes et al., 2011; Kempkes and Meyer auf der Heide, 2011). In addition, recent work shifted the focus toward task allocation strategies for swarm robotics systems characterized by soft deadlines; these self-organized task allocation schemes aim at minimizing the costs associated with missing the task deadlines, see Khaluf and Rammig (2013) and Khaluf et al. (2014).

1.2. Observer/Controller Architecture

As suggested above, the OCbotics approach is founded in a multi-level observer/controller design concept. An according diagram is presented in **Figure 1**. It shows four interwoven architectural levels. Level 0 denotes the system under observation and control (SuOC), the base of the design concept located at the bottom of the figure. Immediately above, level 1 retrieves and evaluates data about the SuOC's performance. Based on these data, it changes the SuOC's configuration in order to optimize its performance, to adapt it to varying conditions and needs. In particular, the SuOC's parameters/behaviors are adapted according to observed success that is calculated with respect to a predefined goal (introduced by level 3). As a consequence, the best possible configuration set, or behavior, known to level 1 is exhibited by level 0 at any given situation. True innovation is realized by level 2, one step above in the multi-level design concept. Here, completely new behavioral options are generated, simulated, and optimized in a sandboxed simulation environment. Only if the new model specifications satisfy all safety constraints considered as part of the simulation process, are they eventually fed into level 1. The general design concept as illustrated by **Figure 1** is

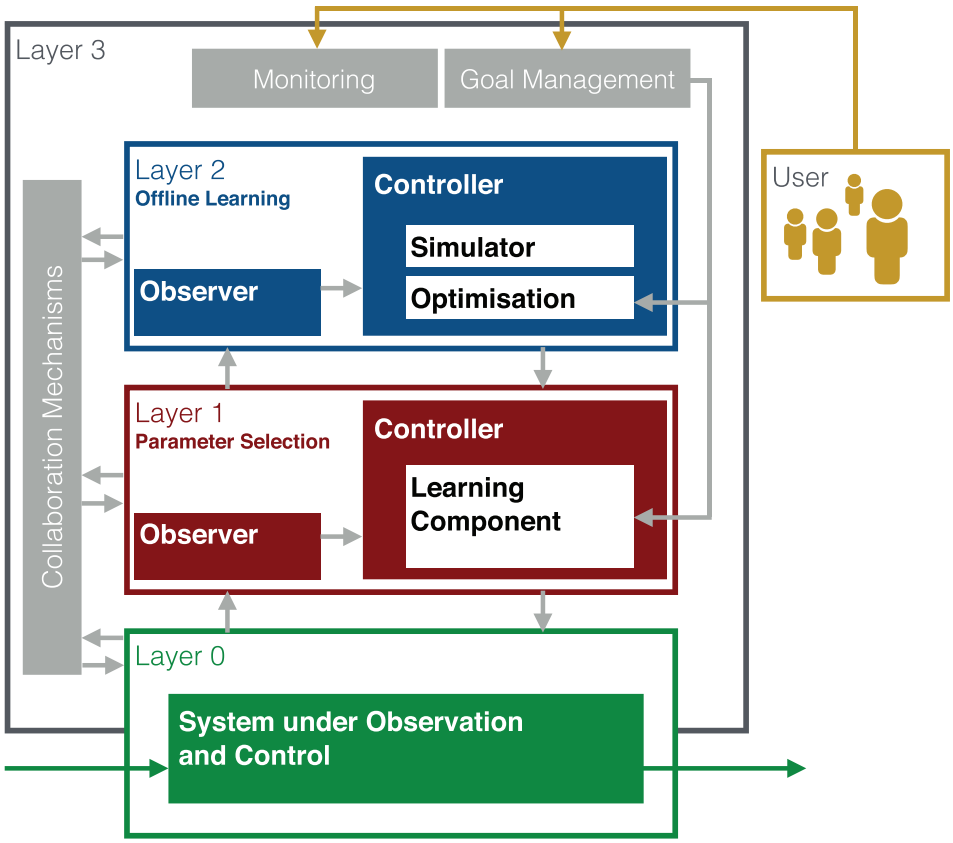

FIGURE 1 | Multi-level observer/controller architecture. The System under Observation and Control (SuOC) at the bottom is observed and adjusted by the O/C layers above. Their responsibilities are (in this order): reinforcement of existing behavior, innovating new behavior, and interfacing local behavior with (a) user-set targets and (b) cooperative units' goals. Figure adapted from Müller-Schloer et al. (2011).

explained in detail in Tomforde et al. (2011). Besides robotics, it has been successfully applied to domains such as control of urban traffic lights, see Prothmann et al. (2011), adaptation of data communication protocols, see Tomforde et al. (2009, 2010), or cloud computing environments, see Sommer et al. (2016). However, dealing with robots and robot ensembles opens a new range of challenges (Tomforde et al., 2014), mainly concerned with the human–ensemble interaction mechanisms – which are a major contribution of this article.

1.3. Learning Classifier Systems

Several studies in Organic Computing have emphasized the adequacy of Learning Classifier Systems (LCS) as a comprehensive framework to support the self-adaptation process based on the observer/controller design concept, see, for example, Richter (2009) and Tomforde et al. (2011). Holland (1975) presented the first LCS that combined basic rule-based reactive behavior with an evolutionary component to evolve and improve the rule base. With the introduction of accuracy-based reinforcement of classifiers, LCS research reached an important milestone in Wilson (1995) [for an overview of LCS research and ongoing research topics see, for instance, Urbanowicz and Moore (2009)]. In the context of safety-critical Organic Computing applications, the latter extension to LCS, also referred to as eXtended Classifier System (XCS), was further modified to suit the multilayer O/C design concept outlined above. In particular, three modifications were implemented: (1) the use of continuous value ranges as promoted in Wilson (2000), (2) the generalization of the closest fitting existing rule in layer 1 instead of the generation of a new rule, in case that a given situation is not covered by the existing rule set ("widening" covering mechanism), and (3) "sandboxed" offline learning in layer 2 to ensure safety and maturity of new rules/behaviors. In addition, the user can track the impact of those triggered rules effecting changes identical to the newly generated rule and, thereby, building up trust in new rules before they are considered by themselves. Current developments of LCS research in the context of Organic Computing can be found in Stein et al. (2016).

1.4. OCbotics Interfaces

At layer 3, an OC system interfaces with the user. Here, he can specify and alter goals and inspect the system's current efforts and states. Next to specifying goals at a rather high level of abstraction, the interplay of the user with the system is situated here. In particular, Human–Swarm Interaction (HSI) methodologies (Vasile et al., 2011; Nagi et al., 2012) are used to access either

the system as a whole or individual decentralized, autonomous components in particular. Besides taking heart in the general user interface design guidelines, which all aim at minimizing the cognitive load of the user (Foley et al., 1984; Preim and Dachselt, 2015), an HSI interface should support an effective utilization of the machines' intelligences, promote local interference rather than global ones, and be scalable (Bashyal and Venayagamoorthy, 2008). Multi-user interaction with the swarm is also desirable, as is the interaction with subsets of swarm individuals. Ideally, an HSI interface should also be applicable to a multitude of application scenarios and not only one very specific one. An according exemplary interface was already presented by McLurkin et al. (2006). Here, robotic ground units coordinate wirelessly, whereas one of them, the gateway robot, communicates with the user and disseminates new instructions to the swarm. The user may also take direct control of individual units of the swarm, thereby influencing its dynamics indirectly. A recent survey on HSI research can be found in Kolling et al. (2016).

In the remainder of this article, we show examples of the OCbotics approach each of which works at one or two different levels of the presented design concept. In particular, we show examples of the reactive behavior of two different systems under observation and control (layer 0), which has been, in parts, presented in von Mammen et al. (2014), and we elaborate on their integration with layer 1 (Section 2). Instances of optimized behavior (layer 2) in collaborative robotics ensembles are presented in Section 3, whereas more detailed results are presented about a collaborative facade cleaning scenario, which was previously presented in von Mammen et al. (2016). The communication among groups of agents as well as the interface mechanism with the user of the system, i.e., layer 3, is explained in Section 4.

2. CONSTRUCTION, LEVITATED, AND SELF-ORGANIZED

Tensile structures play an important role in postmodern architecture [see, e.g., Lewis (2003)], and they promise to become increasingly important still, considering their unique versatility and flexibility in combination with advances in technologies in built material and construction methods (Schock, 1997). They have also been subject to Aerial Robotic Construction (ARC) research (Willmann et al., 2012; Augugliaro et al., 2013) due to their light mass, load-carrying ability, and their ability of connecting large distances. Quad-rotors have been identified as vehicles apt for aerial manipulation mainly due to their robust flight behavior and their hovering capability (Mahony et al., 2012). In Augugliaro et al. (2013), prototypic building primitives such as single and multi-round turn hitches, knob and elbow knots as well the trajectories resulting from their concatenation have been discussed. Different from pre-calculating trajectories, we have been working on a self-organizing approach to building tensile structures. We detail our approach below, followed by elaborations on its OCbotic-specific features.

Typically, a spider weaves its web by itself (von Frisch, 1974; Hansell, 2005). Complex web constructions, however, may require collaborative entanglement and tightening of ropes. This can, for instance, be achieved by synchronized flight through pre-calculated control points to cross the ARC quad-rotors' trajectories. Alternatively, the swarm individuals may coordinate themselves relying on local stimuli, like social insects do (Bonabeau et al., 1999; Camazine et al., 2003). In this section, we present an accordingly motivated ARC experiment.[1] It features an autonomous quad-rotor that tightens a rope around a tent pole's four suspension lines, see **Figures 2** and **3**.

For our lab-experiments, we employ the AR.Drone Parrot 2.0 quad-rotor system. It is connected to a standard PC via WLAN. In order to emulate performant autonomic control of the drone, a PC retrieves the sensory data of the quad-rotor and issues the according navigational instructions. We make use of the quad-rotor's VGA camera that has a 90° field of vision, built-in image-processing capabilities such as recognition of QR markers,

[1] Please find an accompanying video at https://youtu.be/Lt8Von2kFK8.

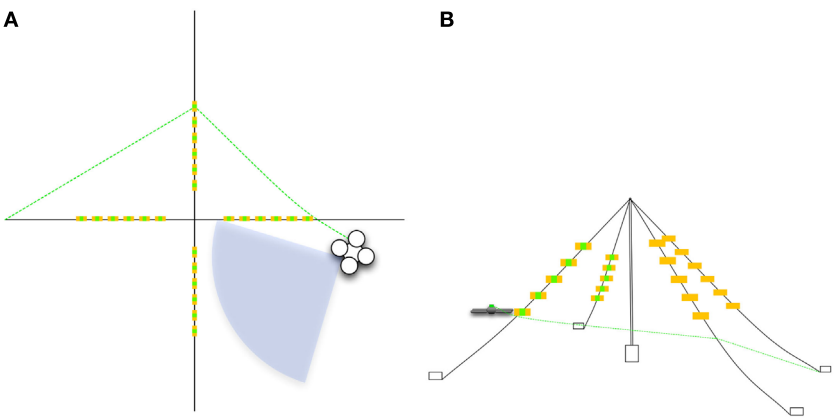

FIGURE 2 | **(A)** The quad-rotor hovers clock-wise around a pole that is suspended by four lines. It tightens a rope (green, dashed) along the suspension lines. **(B)** A schematic side-view extracted from a photograph, highlighting the orientation of the markers pinned to the suspension lines. Both figures © 2014 IEEE. Reprinted, with permission, from von Mammen et al. (2014).

and the estimates of its ultrasonic distance sensor. As this sensor and a downward directed camera are used by the quad-rotor to stabilize its flight, we attached a coil at the top of the vehicle and unwind the cord through an eye at its back. We interface with the quad-rotor relying on Nodecoper.js and the node-ar-drone module (Childers, 2014).

The quad-rotor behaves only based on locally available sensory information. In particular, it implements the reactive behavior schematically summarized in **Figure 4A**: after taking off, it searches for an orange–green–orange marker, which is one of the designs that the vehicle is programed to recognize automatically. It keeps spinning right until it eventually finds one. If the distance to the marker is less than a certain threshold (1 m worked quite well), it drifts left. As a consequence, the detected marker moves outside of its field of view. At this point, the quad-rotor has surpassed the previous marker and looks for the next one, which is attached to the next suspension line (also consider **Figure 2**). The distance to the next marker along the circumference of the pole is greater than the given threshold. The quad-rotor can go straight ahead, if the tag is within the right-hand side of its view (this condition is labeled "tag in area" in **Figure 4A**). Otherwise, it needs to shift a bit to the left.

FIGURE 3 | Illustration of a spider drone at work (screenshot from a video): the AR.Drone Parrot 2.0 weaves a net surrounding the triangle installed in our lab. The corresponding video is available at https://www.youtube.com/watch?v=Lt8Von2kFK8 (last access: 13/09/2016).

Programmatically, the quad-rotor's behavior (**Figure 4A**) is represented as a set of simple *if-then* rules. As such, they can be easily subjected to standard LCS implementations and its extensions such as XCS (Section 1.3). Hereby, those rules with the best prediction accuracy in terms of marker detection may be reinforced to gain the greatest fitness over time, yielding the best possible behavior. In this way, the quad-rotor of the ARC example would learn to query the proper sensors at the right times to react in the best possible way, if the behavioral rule set was enriched with according alternatives. At the interface of level 0 (the SuOC) and level 1 (the reinforcement learner), the measure of success can typically be calculated based on locally available information such as the distance flown or the number of recognized tags. For good learning results, the parametrization of the behavior should be realized at a rather high level, focusing on the selection of queries and operations and only cover small ranges of variability. Potential benefits of level 1 learning would not only be optimization of one particular learning pattern but also behavioral rules that adapt to hardware particularities such as deviating sensory intake or imbalanced motor control.

3. COLLABORATIVE SPATIAL WORK

At level 2 of the multilayer O/C design concept, behaviors can be created by means of generative model building approaches such as evolutionary algorithms and be optimized for deployment by means of simulations. As a first OCbotics prototype of offline level 2 generation and optimization, we have evolved quad-rotor behavior for collaborative surface maintenance. In this section,

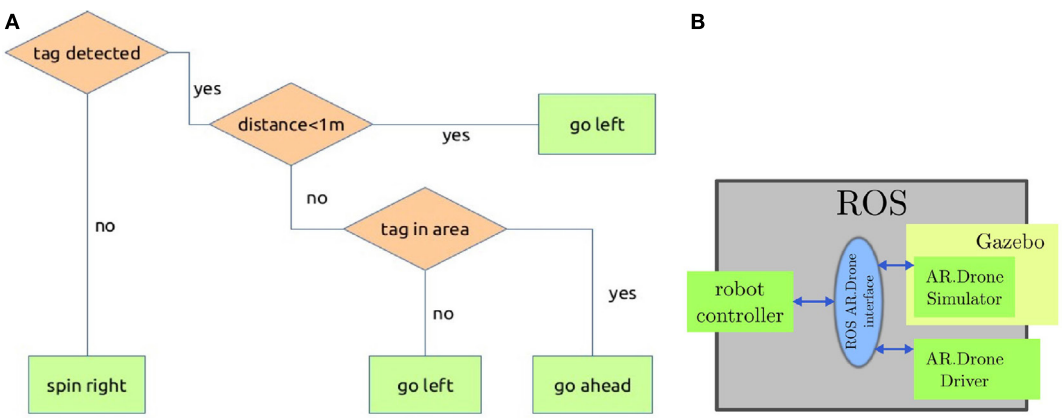

FIGURE 4 | (A) Weaving behavior of a self-organizing ARC Quad-rotor. It circles clock-wise around a pole, tightening its thread around suspension lines tagged with directionally oriented markers. © 2014 IEEE. Reprinted, with permission, from von Mammen et al. (2014). **(B)** Interwoven Simulation Modules. The Robot Operating System integrates various software components to simulate quad-rotor swarms. In particular, we control the robot using a common ROS-based instruction set that is understood by both the Gazebo simulation software and its quad-rotor simulation plug-in as well as a ROS driver that steers the actual quad-rotor hardware. © von Mammen et al. (2016) CC BY-NC-SA 2.0.

we introduce the challenge of optimizing collaborative surface maintenance. We detail the technical setup we relied on for both simulation and optimization, and we describe the behavioral options of each swarm individual. Afterward, we draw a very rough picture of the evolutionary experiments that we have run, and we discuss the interactions between layers 2 and 3 for propagating successfully bred behaviors that require synchronization between the individuals in an OCbotics swarm.

Consider the facades of large office buildings as examples of vertical surfaces: they are subject to cleaning (Bohme et al., 1998; Elkmann et al., 2002), trimming greenery (Pérez et al., 2011), and other maintenance tasks. As in the previous example, these tasks might benefit more from collaborative efforts than only in terms of efficiency. For instance, fast growing greenery might require one machine to bend, the other one to cut a branch. Equally, during cleaning, several hovering robots might have to join to build up sufficient pressure to remove persistent dirt. Of course, the respective operations might also be split into several procedures performed by individually optimized machines. In this example, however, we only consider the most modest objective, namely collaborative efficiency.

The technical setup of our level 2 experiment comprises (a) a simulation environment to calculate aviation and robotic mechanics and (b) a machine learning environment with a generative model component and an optimization component. **Figure 4B** depicts the software modules that we have used in order to simulate collaborative quad-rotor swarms. The Robot Operating System (ROS) acts as a hub for these modules. It provides a high-level software interface for programing and communicating with different kinds of robots (Quigley et al., 2009). Gazebo is a simulation engine that natively integrates with ROS, offering 3D rendering, robot-specific functionality, and physics calculations (Koenig and Howard, 2004). Thanks to a ROS driver for the AR.Drone Parrot quad-rotor (Hamer et al., 2014), and thanks to a Gazebo plug-in that simulates the quad-rotor's behavior based on the very same ROS-based instruction set (Huang and Sturm, 2014), any of the generated behaviors immediately work *in silico* and *in vivo*. For the generation of novel behaviors as well as for their evolution, we decided to use the Evolving Objects framework (EO) by Keijzer et al. (2002). EO is an open framework for evolutionary computation featuring an extensible, object-oriented design concept, and turnkey implementations of genetic algorithms, particle swarm optimization, and genetic programing.

Our approach to collaborative facade maintenance is inspired by nest construction of social insects (Bonabeau et al., 1999). Each individual works on a small part of the construction proportional in size to the insects' physique. Accordingly, each simulated quad-rotor divides the target surface in a grid, each cell measuring 2 m × 2 m, its field of view covering six cells, two rows of three (**Figure 5**). This partitioning scheme is a result of the size of the quad-rotor itself and its perceived area from a vantage point close to the surface. Without loss of generality, a dirtiness value is assigned to each cell that indicates whether it needs to be worked on or not. The quad-rotor's internal state, i.e., its remaining battery life, as well as the configuration of dirty and clean cells that reveals itself in front of it trigger specific actions. The quad-rotor may return to the base station to recharge. It may fly to one of the cells in its field of view and clean it. Alternatively, it may move to one of the four neighboring vantage points to inspect the respective neighboring batches of cells.

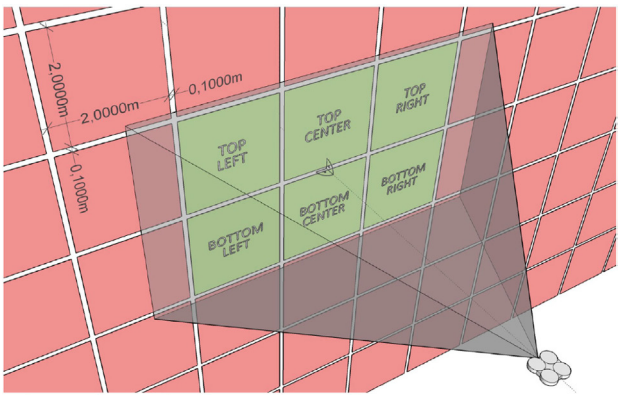

FIGURE 5 | Cell grid for surface maintenance. The quad-rotor divides the building facade in a grid of cells. The individual cells represent the immediate target areas to work on. Their states of cleanliness also provide the local cues for decision-making, i.e., for approaching an individual cell or to moving to another vantage point.

3.1. Evolving Collaborative Behavior

Figure 6 captures the behavioral options of a quad-rotor in the context of facade maintenance. Any activity is initiated by the decision-making component; subsequent events guide the quad-rotor back into the decision-making process. Again, the behaviors can easily be written as *if-then* rules that ensure the coherence and simplicity of interfacing across the layers of the O/C design concept. Notice that in this model, quad-rotors cannot stop working. Instead, the whole simulation is terminated after a given amount of time. During this period of time, the decision-making component determines the success of the simulated swarm. We generate an according program tree using Genetic Programing (Poli et al., 2008). In this article, we refrain from presenting the evolutionary approach in all detail, but we want to convey its basic mechanisms and how it ties into the OCbotics approach. The generated decision program may conditionally deploy the operations outlined in **Figure 6A** and introduce references or primitive values as their parameters. The resultant behavior trees are considered the individuals in an evolutionary optimization cycle, and thus their fitnesses (penalty, i.e., for remaining amount of dirt) are calculated in according simulation runs, and they serve as an important criterion for selecting ancestors for subsequent generations of individuals (deterministic tournaments).

We ran experiments featuring two or four quad-rotors, or "aerial robotic units" (ARUs), working in parallel for 900–1500 simulated seconds. Their individual base stations were distributed randomly in rectangular area sharing two sides with the target surfaces as seen in **Figure 6B**. Population sizes varied from 30 to 100 individuals, the generational cycle was repeated between 10 and 50 times – depending on the work load of an individual simulation, which was mainly determined by the number of interacting

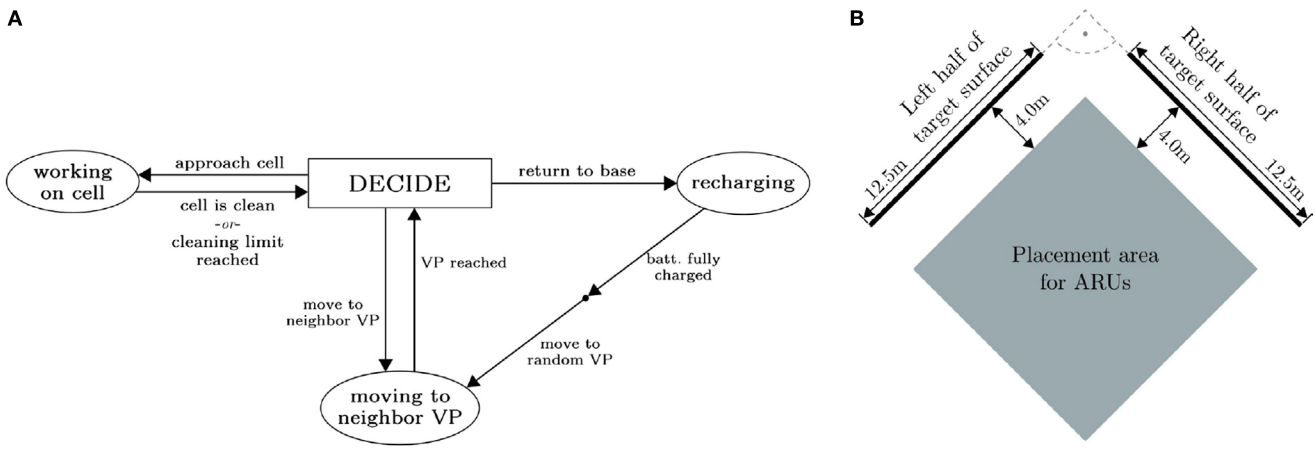

FIGURE 6 | **(A)** Options of activity of a facade maintaining quad-rotor agent. Any activity – working on one of the cells ahead, flying to the base station to recharge, or moving to a neighboring vantage point – is initiated by the decision component, which considers the agent's battery state and the surface configuration in its field of view. **(B)** Simulated environment for collaborative surface maintenance evaluation. Two flat surfaces are presented to the quad-rotor swarms as target area, which needs to be cleaned. The base stations of the swarm individuals are randomly placed in the rectangular area between the surfaces. Both figures © von Mammen et al. (2016) CC BY-NC-SA 2.0.

agents and the simulated time. One of the best individuals in an experiment that started from a set of previously evolved specimen worked as follows: having arrived at a random cell of the target surface, work through single rows of vantage points from right to left. If the border of the target surface is reached, return to the base station and approach the target area again. It turns out that this behavior proved significantly faster than two decision programs we manually designed before running the evolutionary process: one of them stochastically selecting dirty cells and considering the remainder of the battery before taking action (low batteries are also penalized by the fitness calculation), the other one letting the quad-rotor follow the dirt gradient exhibited in the perceived 3 by 2 cell matrix.

Figure 7 provides a glimpse at the progression and the success of the evolutionary optimization runs. In addition to 20 randomly generated individuals, we fed the best results of a first experiment that aimed at rigorous preselection of solutions into a second phase of experiments that was more directed, introducing some restrictions, as well as simplifications to speed up the simulations. For instance, an extended simulation time limit of 1500 s forces the quad-rotors to land at least once, an activity they could avoid during the first phase of experiments, as their batteries support up to 1200 s of flight. **Figure 7A** shows the merged results of both phases: for comparison, ten previously evolved seed individuals (green) and two manually created decision functions (blue) are also shown. The seed genotypes `0aabf834` and `131cfa1e` did not finish any of the simulations. In order to provide a basis for comparison, the resulting statistics are extended to include the ten seed individuals from the first phase and manually created decision functions, all of which are reevaluated in the second simulation scenario. In **Figure 7B**, we see the (non-averaged) penalty value calculated during the second phase of evolutionary experiments vs. the associated genotype's syntax tree depth. We plotted the penalty value against the individuals' syntax tree depth, not averaging multiple evaluations of the same genotype but showing them as multiple data points. The lower boundary of the scattered points indicates that trees below a depth of five do not perform well. The best individuals are located in the range of depths five to eight, whereas the individuals' penalties do not rise until a tree depth of 11 (from about 9600 to 9800). A substantially steeper penalty increase follows from depths 14 to 16, stabilizing at about 11,300. This time the scattered points aggregate along two horizontal lines, one at a penalty value of around 11,400, the other one at about 16,200. These aggregations emerge due to genotypes that perform neither particularly effectively nor particularly poorly. The duality of the recovered baseline arises from one strong scheme injected with the seeded individuals from phase one and from a dominant scheme that evolved from random initializations in phase two.

Level 2 is capable of generating and evolving collaborative behavior such as the one described above. Initially, the novel behavior does not have any impact on the system under observation and control. One may say the innovation process is encapsulated in a sandbox and runs completely separated process, offline. At the same time, collaborative behavior needs to be communicated, if it is required to be performed by all individuals of a swarm in order to function in a coordinated way. The observation of Layer 2 by Layer 3 has to detect such impending necessary changes and broadcast it to all the other members of the swarm. Similar to an auction in multi-agent systems (Wooldridge, 2009), the best broadcast solution, i.e., decision program and fitness value, would be implemented. As an extension, any population-based simulation and optimization approaches could be distributed among the OCbotics individuals and their evolution be concerted across the whole swarm [for distributed population-based optimization, see, for instance, Sarpe et al. (2010) and Jacob et al. (2011)]. Especially

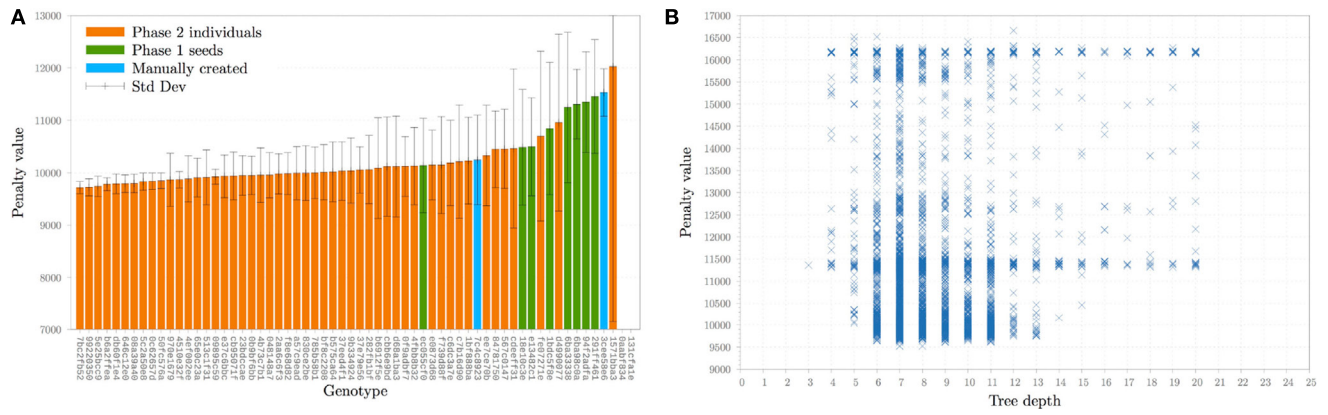

FIGURE 7 | **(A)** Penalty ranking of evolutionary experiments. **(B)** Penalty values vs. syntax tree depth in the second phase of evolutionary experiments. Both figures

in situations with imbalanced computational loads across the swarm, following a smart distributed optimization strategy could yield an important advantage.

3.2. Other Coordinated Spatial Tasks

Distributing the lawnmower problem and finding feasible real-world parameters is an important challenge for coordinated robotic ensembles. In addition, we have worked toward models that allow for simulating and optimizing the coordinated behaviors of ARUs that (a) optimize their flight paths in context of both static and dynamic obstacles, (b) search for moving targets, and (c) scan building interiors. In the following paragraphs, we illustrate and briefly summarize our findings regarding these respective models.

3.2.1. Digital Pheromones for Path Finding and Planning

Social insects drop chemical cues, so-called pheromones, on their way from a food source back to their nest. Reinforcement of the pheromone trails ensures that an increasing number of foraging insects follow the trails and exploit the food source, whereas the evaporation of the chemical signals ensures that new opportunities are found when the food source runs low (Sumpter and Beekman, 2003). In order to translate this concept to the coordination of ARUs (or less generic "unmanned aerial vehicles," UAVs), technical solutions for leaving digital pheromones need to be found. One way to do so lies in utilizing locally distributed smart transceivers that store and disseminate information of passers-through, as in Parunak et al. (2002). Alternatively, passive RFID chips could be distributed that convey the respective information when queried (Mamei and Zambonelli, 2005). Based on such a hardware infrastructure, the parameters for concerting one's behaviors need to be optimized for dynamic path planning and path finding in different kinds of environments. While the approach proved robust in our experiments in general, choices as to when pheromones should be placed and which initial directions the individuals should head into impacted the target performance (**Figure 8**).

3.2.2. Search for Mobile Targets

Coordinated search by ensembles of ARUs is an important task. There have been efforts in this direction that yield the minimal number of units needed to find (mobile) targets within a given time and with a certain probability (Vincent and Rubin, 2004) or adapt concise trajectory plans for sets of units based on their individual properties (Yang et al., 2007). A general taxonomy for cooperative search is provided by El-Abd and Kamel (2005). The simulation model that we have devised for level 2 of the O/C design concept allows one to determine the ideal individual velocities and binding forces between elements of loosely coupled chains of ARUs (**Figure 9**). These chains are established in a self-organizing fashion, linking the units with their neighbors and agreeing on local leaders. Their advantage over alternative topologies lies in sweeping large, connected areas, thus minimizing the odds of the target slipping through the search grid. Similar to Vincent and Rubin (2004), we can optimize the couplings to minimize the number of deployed units and to conduct a comprehensive search within a given time [as in Yang et al. (2007)]. Considering inconsistencies in the ARUs' flight behaviors, the impact of the environment, and possible heterogeneities of the deployed units, the tandem of level 1 learnings and level 2 offline adjustments become all the more relevant.

3.2.3. Scanning Interior Spaces

Different from the line-formation approach to coordinated search, in which the ensemble proactively establishes a topology for covering connected areas, the environment itself may be topologically organized, i.e., divided in locally connected space partitions such as rooms in a building. Giese (2013) exemplarily demonstrated the usefulness of drones for surveying large buildings when digitizing the cathedral of the German world-heritage city of Bamberg. Ballarin et al. (2013) deployed drones to survey inaccessible buildings, whereas Shen et al. (2011) and Muhleisen and Dentler (2012), e.g., showed the feasibility of drones exploring a building autonomously. Through our according model, the scanning progress of individual units is shared in a local *ad hoc* network (**Figure 10**). Cells marked as scanned in a discretized

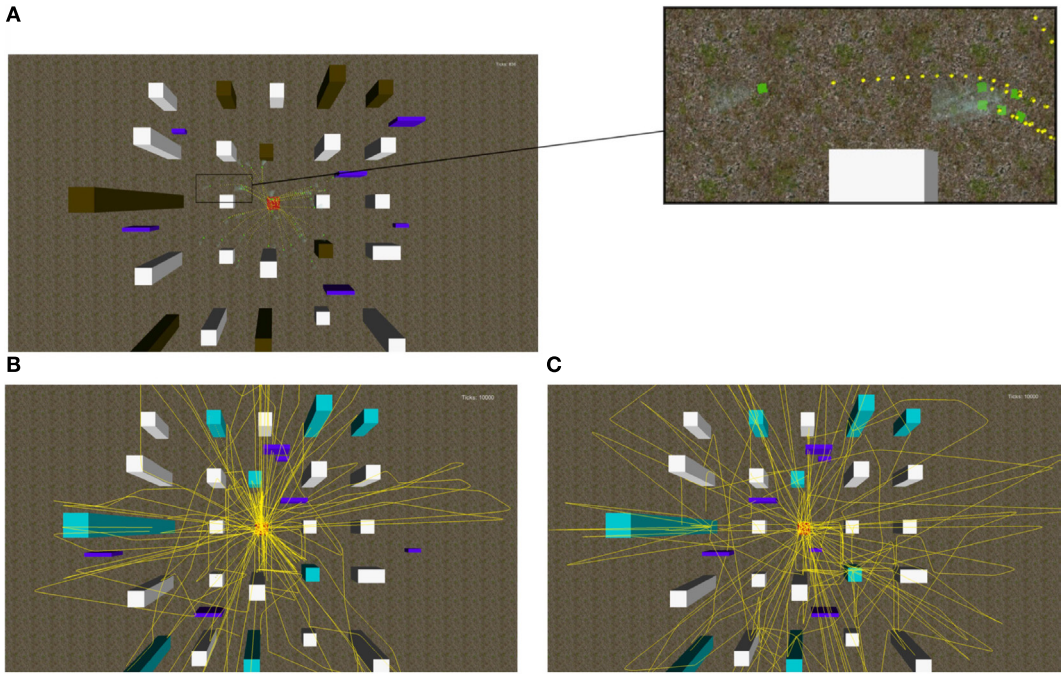

FIGURE 8 | **(A)** Placing path pheromones (in yellow) from the start may ensure that several individuals (in green) define new trails in a common direction. Competition among these trails will determine the most efficient path through an area populated with static and mobile obstacles, brown/white and purple boxes, respectively. **(B)** Omnidirectional sensitivity for pheromones results in high density trails at the base station. **(C)** Furthermore, scattered sensitivity ensures a wider coverage of the target area.

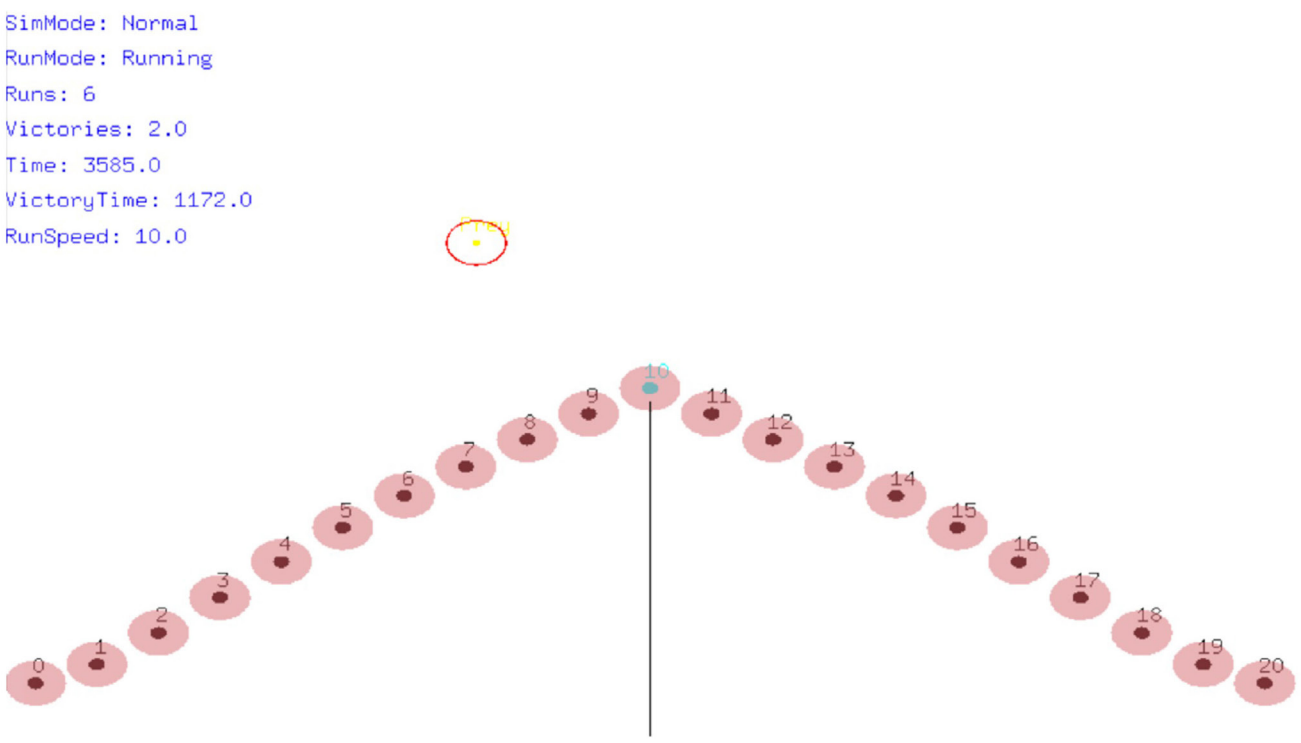

FIGURE 9 | **Loosely coupled chains of ARUs (filled circles) are formed to cooperatively search large, connected areas for mobile targets (hollow circle)**.

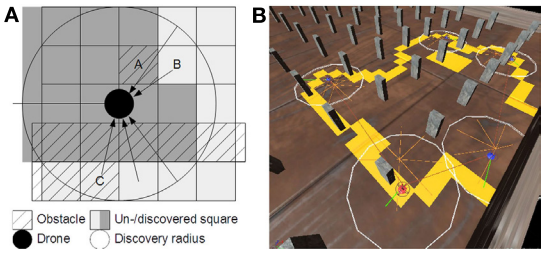

FIGURE 10 | **(A)** The drone marks cells of a discretized environment grid to indicate scanned areas and determine its next steps. The done can mark unobstructed cells and unobstructed obstacles such as A but not occluded cells such as B and C. **(B)** The drones (blue boxes) are moving in different directions, maximizing their shared knowledge gain. Moving along they might get out of reach to synchronize. In this case, they need to move back to the "root" of the communication tree, i.e., the drone shown as the red box.

lattice space determine the paths taken by the individuals. In our model, it is a strong real-world restriction to maintain network connectivity with the remainder of the ensemble. Therefore, a loss in connectivity would result in individuals approximating the "root" unit of the current communication tree.

4. INTERACTIVE SELF-ORGANIZATION

In our last example, we demonstrate an early prototype of the user interfacing component of layer 3 of the multilayered O/C design concept that drives the OCbotics approach. As hinted at in **Figure 1**, layer 3 mediates the user's goals vertically to all system layers below and horizontally to all OCbotics individuals of the system.

The preceding examples of web-weaving quad-rotors in Section 2 and collaborative swarms in Section 3 implement spatial operations. To some extent, they all culminate in the tandem of local cues and resultant trajectories. As a consequence, defining spatial targets for arbitrary subsets of a swarm deems to be an adequate task generalization for a first prototype of a level 3 user interface. A "human-in-the-loop" system design forces one to clearly define the level of influence a user may exercise versus the level of autonomy the system may keep (Narayanan and Rothrock, 2011). Therefore, we elaborate on the different levels of access implemented by our prototype right after we outline its technical foundation.

Focusing on interactivity, we decided to utilize the turnkey infrastructure of one of the comprehensive game and simulation engines. In particular, we decided to use Unity as it provides a very shallow learning curve (compared to its competitors) while still providing a powerful coding infrastructure that allows to write custom plug-ins in C# and which offers a wide range of third-party plug-ins in a dedicated asset store, see Unity Technologies (2014). Aiming at the implementation of a high-level interface, we tapped into these resources as much as possible and bought, for instance, commercial code bases for simulating flocking behaviors (Different Methods, 2014) and automated path finding in three-dimensional environments (Allebi, 2014). We further built on Unity demos and plug-ins that support current hardware solutions such as the Oculus Rift head-mounted display (Oculus VR Inc, 2014) and the Razer Hydra motion controller (Razer Inc, 2014). In combination, these hardware solutions allow us to emulate an augmented reality scenario for controlling an OCbotics swarm. **Figure 11** shows the model of an OCbotics swarm being setup in Unity3D. The light green circles depict waypoints computed by the path finding algorithm, the dual-view perspective at the bottom-left corner of the screen indicates the current view of the attached head-mounted display. The bottom-right window displays the library of components used for modeling the scene, the list at the right-hand side of the screen shows the components that already constitute the scene.

Our user interface prototype immerses the user into a virtual reality shared with the OCbotics swarm. In the long run, the simulated swarm is meant to make way for a real one, and the virtual reality for an augmented reality. Already, the user can observe the whole swarm or a subset tracking it with a virtual camera that follows in a distance and which aims at the center of the set of selected individuals. The user can exercise control on any subset of the swarm, hence he may direct flocks of individuals or single individuals at a time. The interface provides all kinds of state information about the selected individuals, such as (averaged and variance of) remaining battery life, current target, current trajectory, and currently perceived neighbors. The user may switch between individuals and greater subsets of the swarm by simply selecting them. Next, he may change the target of flight or even individual control points along the way. Of course, he may also change the parameters of the selected individuals such as their urge for alignment. In our prototype, the user is immersed into the scene of the simulated swarm (see **Figure 12A**) so he can easily trace its activity, understand its relationship to the current target and to obstacles, and to rectify it, whenever necessary.

The presented simulated prototype for immersive swarm control shows how high-level goals such as setting a new target of the swarm can be communicated in an intuitive way. Differentiated selection of swarm individuals as well as setting local attributes, such as local targets or local waypoints, are simple yet clear examples of moving from abstract, high-level goal descriptions (target/swarm) to specific low-level commands (trajectory waypoints/individual). For a swarm and an individual to reach the specified targets or waypoints, complex calculations have to be performed. In the given example, the need to avoid obstacles and to find optimal paths as well as the coordination among swarm individuals on their way are outsourced to third-party plug-ins (Allebi, 2014; Different Methods, 2014). In the general case, also considering other tasks communicated on layer 3, the necessary behaviors could evolve in sandboxed simulations (layer 2) and be optimized based on local performance feedback (layer 1), see. We are currently working on furthering the accessibility of human–swarm interfaces by designing and evaluating game-like test scenarios in virtual reality (**Figure 12B**). In addition, we derive mutual influences among distributed robot societies at runtime to further improve the cooperation of individuals (Rudolph et al., 2015a,b, 2016).

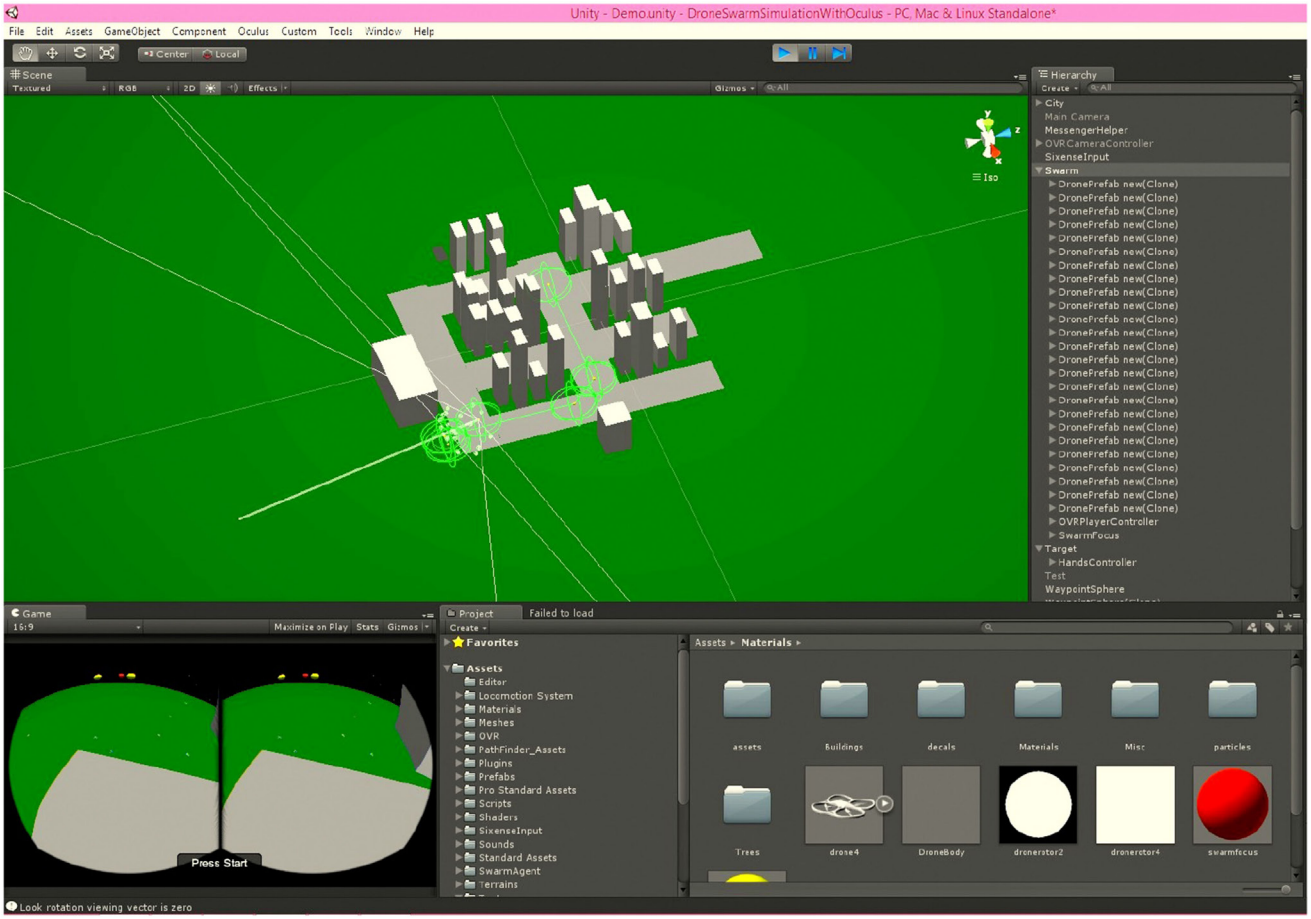

FIGURE 11 | **OCbotics swarm modeled in Unity3D**. The Unity3D environment allows us to integrate complex simulation models and immersive user interaction hardware such as motion-based input controllers and head-mounted displays. © 2014 IEEE. Reprinted, with permission, from von Mammen et al. (2014).

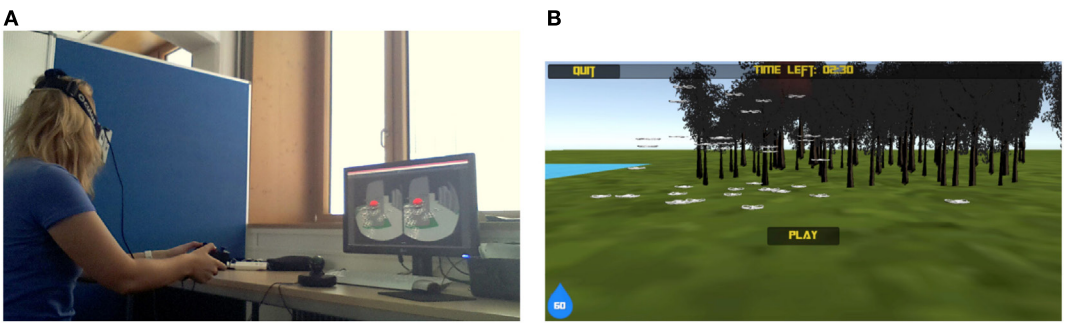

FIGURE 12 | **(A)** The user guides the swarm through movement of her head and two 3D joysticks. The pair of controllers empowers her to draw new spatial relations between the simulated objects, e.g., to set new targets for subsets of the swarm. © 2014 IEEE. Reprinted, with permission, from von Mammen et al. (2014). **(B)** We are working on testing usability and user experience of different interfaces and input modalities [also based on indirect touch (Voelker et al., 2013) and hand-gestures (Hackenberg et al., 2011)] in the context of this fire-fighting game.

5. CONCLUSION

In this article, we have introduced OCbotics as a comprehensive approach to designing self-organizing aerial robotic ensembles. OCbotics is driven by a multilayered observer/controller design concept that allows to optimize and adapt an adaptable system. Adaptation is required in order to maintain or increase the performance exhibited by the system under observation and

control – either by optimizing or extending existing behaviors, or by innovating, i.e., generating, simulating, and optimizing novel behaviors. The performance, in turn, is measured in terms of user-defined goals that may also change over time. In comparison to concepts from the state-of-the-art [see, e.g., Augugliaro et al. (2013) where trajectories are pre-calculated], the OCbotics approach provides a large step toward applicability in non-lab and unstructured conditions, since it establishes a self-adaptation loop that considers safety constraint and continuously self-optimizes the robot's behavior.

We have presented three different projects that operate at different levels of the discussed design concept: web-weaving quad-rotors with an emphasis on optimized local reactive behavior, evolution of collaborative behavior to efficiently work on surfaces, and an immersive user interface for setting and changing user-defined goals. While the three examples slightly vary regarding their applications, they are connected through the common themes of self-organization, rule-based behavior, and adaptation, and of course, the O/C design concept to host them all. With the pieces of the puzzle at hand, the next obvious step is to put them into place, to forge the software components into one (if heterogeneous) code base, to connect the layers of the design concept, to develop a repertoire of recombinable goal definitions, and to transfer the partially still virtual implementations of all levels onto an actual OCbotics infrastructure.

AUTHOR CONTRIBUTIONS

SM: main author, main project supervisor. ST and JH: coauthors, secondary project supervisors.

REFERENCES

Allebi, F. (2014). *Easy Path Finding System*. Available at: http://u3d.as/content/allebi/easy-path-finding-system/4a7

Augugliaro, F., Mirjan, A., Gramazio, F., Kohler, M., and D'Andrea, R. (2013). "Building tensile structures with flying machines," in *International Conference on Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ* (Tokyo, JP: IEEE), 3487–3492.

Ballarin, M., Buttolo, V., Guerra, F., and Vernier, P. (2013). Integrated surveying techniques for sensitive areas: San felice sul panaro. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inform. Sci.* 5, W1.

Bashyal, S., and Venayagamoorthy, G. K. (2008). "Human swarm interaction for radiation source search and localization," in *Swarm Intelligence Symposium, 2008. SIS 2008. IEEE* (St. Louis, MO: IEEE), 1–8.

Bohme, T., Schmucker, U., Elkmann, N., and Sack, M. (1998). "Service robots for facade cleaning," in *Industrial Electronics Society, 1998. IECON'98. Proceedings of the 24th Annual Conference of the IEEE*, Vol. 2 (Aachen: IEEE), 1204–1207.

Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. New York: Santa Fe Institute Studies in the Sciences of Complexity, Oxford University Press.

Brandes, P., Degener, B., Kempkes, B., and Meyer auf der Heide, F. (2011). "Energy-efficient strategies for building short chains of mobile robots locally," in *SIROCCO'11: Proc. of the 18th International Colloquium on Structural Information and Communication Complexity* (Gdansk: Springer), 138–149.

Brockmann, W., Maehle, E., and Mösch, F. (2005). "Organic fault-tolerant control architecture for robotic applications," in *IARP/IEEE-RAS/EURON Workshop on Dependable Robots in Human Environments* (Nagoya: IEEE).

Brockmann, W., Rosemann, N., and Maehle, E. (2011). "A framework for controlled self-optimisation in modular system architectures," in *Organic Computing – A Paradigm Shift for Complex Systems*, eds C. Müller-Schloer, H. Schmeck, and T. Ungerer (Basel, CH: Birkhäuser Verlag), 281–294.

Camazine, S., Deneubourg, J.-L., Franks, N. R., Sneyd, J., Theraulaz, G., and Bonabeau, E. (2003). *Self-Organization in Biological Systems. Princeton Studies in Complexity*. Princeton: Princeton University Press.

Childers, B. (2014). Hacking the parrot ar drone. *Linux J.* 2014, 1. Available at: https://de.scribd.com/document/234999646/Linux-Journal-May (accessed November 08, 2016).

Different Methods. (2014). *Swarm Agent*. Available at: http://u3d.as/content/different-methods/swarm-agent/608

El-Abd, M., and Kamel, M. (2005). "A taxonomy of cooperative search algorithms," in *Proceedings of the International Workshop on Hybrid Metaheuristics* (Barcelona: Springer Berlin Heidelberg), 32–41.

Elkmann, N., Felsch, T., Sack, M., Saenz, J., and Hortig, J. (2002). "Innovative service robot systems for facade cleaning of difficult-to-access areas," in *International Conference on Intelligent Robots and Systems, 2002. IEEE/RSJ*, Vol. 1 (Lausanne, CH: IEEE), 756–762.

Foley, J. D., Wallace, V. L., and Chan, P. (1984). The human factors of computer graphics interaction techniques. *Comput. Graphics Appl. IEEE* 4, 13–48. doi:10.1109/MCG.1984.6429355

Giese, J. (2013). "'Technologiemix im praxistest: Baudokumentation am bamberger dom," in *Dokumentation und Innovation bei der Erfassung von Kulturgütern II*, ed. E. I. Faulstich Würzburg: Bundesverband freiberuflicher Kulturwissenschaftler e.V., Otto-Friedrich-Universität Bamberg.

Hackenberg, G., McCall, R., and Broll, W. (2011). "Lightweight palm and finger tracking for real-time 3d gesture control," in *Virtual Reality Conference (VR), 2011 IEEE* (Singapore: IEEE), 19–26.

Hamer, M., Engel, J., Parekh, S., Brindle, R., and Bogert, K. (2014). *Ardrone Autonomy: A Ros Driver for Parrot Ar-Drone Quadrocopter*. Available at: https://github.com/AutonomyLab/ardrone_autonomy

Hansell, M. (2005). *Animal Architecture*. New York, NY: Oxford University Press.

Hestermeyer, T., Oberschelp, O., and Giese, H. (2004). "Structured information processing for self-optimizing mechatronic systems," in *Proceedings of the 1st International Conference on Informatics in Control, Automation and Robotics (ICINCO 2004), Setubal, Portugal*, eds H. Araujo, A. Vieira, J. Braz, B. Encarnacao, and M. Carvalho (Setubal: INSTICC Press), 230–237.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Oxford, UK: U Michigan Press.

Huang, H., and Sturm, J. (2014). *Tum Simulator*. Available at: http://wiki.ros.org/tum_simulator

Jacob, C., Sarpe, V., Gingras, C., and Feyt, R. P. (2011). "Swarm-based simulations for immunobiology," in *Information Processing and Biological Systems* (Berlin: Springer Berlin Heidelberg), 29–64.

Jungmann, A., Kleinjohann, B., and Richert, W. (2011). "Increasing learning speed by imitation in multi-robot societies," in *Organic Computing – A Paradigm Shift for Complex Systems* (Basel, CH: Birkhäuser Verlag), 295–307.

Keijzer, M., Merelo, J. J., Romero, G., and Schoenauer, M. (2002). "Evolving objects: a general purpose evolutionary computation library," in *Artificial Evolution*, eds P. Collet, C. Fonlupt, J.-K. Hao, E. Lutton, and M. Schoenauer (Heidelberg, DE: Springer), 231–242.

Kempkes, B., and Meyer auf der Heide, F. (2011). "Local, self-organizing strategies for robotic formation problems," in *ALGOSENSORS, Volume 7111 of Lecture Notes in Computer Science*, ed. T. Erlebach (Heidelberg, DE: Springer), 4–12.

Khaluf, Y., Birattari, M., and Hamann, H. (2014). "A swarm robotics approach to task allocation under soft deadlines and negligible switching costs," in *Proceedings Animals to Animats 13 – 13th International Conference on Simulation of Adaptive Behavior, SAB 2014, Castellón, Spain, July 22-25, 2014*, 270–279.

Khaluf, Y., and Rammig, F. J. (2013). "Task allocation strategy for time-constrained tasks in robot swarms," in *Proceedings of the Twelfth European Conference on the Synthesis and Simulation of Living Systems: Advances in Artificial Life, ECAL 2013, Taormina, Italy, September 2-6, 2013*, 737–744.

Koenig, N., and Howard, A. (2004). "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proceedings of the Intelligent Robots and Systems, 2004 (IROS 2004). International Conference on 2004 IEEE/RSJ*, Vol. 3 (IEEE), 2149–2154.

Kolling, A., Walker, P., Chakraborty, N., Sycara, K., and Lewis, M. (2016). Human interaction with robot swarms: a survey. *IEEE Trans. Human-Mach. Syst.* 46, 9–26. doi:10.1109/THMS.2015.2480801

Lewis, W. (2003). *Tension Structures: Form and Behaviour*. London, UK: Thomas Telford.

Mahony, R., Kumar, V., and Corke, P. (2012). Multirotor aerial vehicles: modeling, estimation, and control of quadrotor. *Rob. Autom. Mag. IEEE* 19, 20–32. doi:10.1109/MRA.2012.2206474

Mamei, M., and Zambonelli, F. (2005). "Physical deployment of digital pheromones through rfid technology," in *Proceedings of the Swarm Intelligence Symposium, SIS 2005* (Pesedina: IEEE), 281–288.

McLurkin, J., Smith, J., Frankel, J., Sotkowitz, D., Blau, D., and Schmidt, B. (2006). "Speaking swarmish: human-robot interface design for large swarms of autonomous mobile robots," in *AAAI Spring Symposium: To Boldly Go Where No Human-Robot Team Has Gone Before* (Palo Alto: AAAI Press), 72–75.

Mösch, F., Litza, M., Auf, A. E. S., Maehle, E., Großpietsch, K.-E., and Brockmann, W. (2006). "Orca – towards an organic robotic control architecture," in *Proc. of First International Workshop on Self-Organizing Systems, IWSOS/EuroNGI* (Passau: Springer Berlin Heidelberg), 251–253.

Muhleisen, H., and Dentler, K. (2012). Large-scale storage and reasoning for semantic data using swarms. *Comput. Intell. Mag. IEEE* 7, 32–44. doi:10.1109/MCI.2012.2188586

Müller-Schloer, C., Schmeck, H., and Ungerer, T. (eds) (2011). *Organic Computing – A Paradigm Shift for Complex Systems. Autonomic Systems*. Basel, CH: Birkhäuser Verlag.

Nagi, J., Ngo, H., Giusti, A., Gambardella, L. M., Schmidhuber, J., and Di Caro, G. A. (2012). "Incremental learning using partial feedback for gesture-based human-swarm interaction," in *Proceedings of the 21st IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)* (Paris: IEEE), 898–905.

Narayanan, S., and Rothrock, L. (2011). *Human-in-the-loop Simulations: Methods and Practice*. London, UK: Springer.

Oculus VR Inc. (2014). *Oculus Rift: Next Gen Virtual Reality*. Available at: http://www.oculusvr.com/rift/

Parunak, H. V. D., Purcell, L. M., SIX, F. C. S., and O'Connell, M. R. (2002). "*Digital Pheromones for Autonomous Coordination of Swarming uav's*," in *Proceedings of the 1st UAV Conference* (Portsmouth: American Institute of Aeronautics and Astronautics), 1–9.

Pérez, G., Rincón, L., Vila, A., González, J. M., and Cabeza, L. F. (2011). Green vertical systems for buildings as passive systems for energy savings. *Appl. Energy* 88, 4854–4859. doi:10.1016/j.apenergy.2011.06.032

Poli, R., Langdon, W. B., McPhee, N. F., and Koza, J. R. (2008). *A Field Guide to Genetic Programming*. Raleigh: Lulu Press, Inc.

Preim, B., and Dachselt, R. (2015). *Interaktive Systeme: Band 2: User Interface Engineering, 3D-Interaktion, Natural User Interfaces*, 2nd Edn. Berlin: Springer-Verlag.

Prothmann, H., Tomforde, S., Branke, J., Hähner, J., Müller-Schloer, C., and Schmeck, H. (2011). "Organic traffic control," in *Organic Computing – A Paradigm Shift for Complex Systems, Volume 1 of Autonomic Systems*, eds C. Müller-Schloer, H. Schmeck, and T. Ungerer (Basel, CH: Springer Basel), 431–446.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., et al. (2009). "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, Vol. 3. (Kobe: IEEE Press), 5.

Razer Inc. (2014). *Razer Hydra Portal 2 Bundle*. Available at: http://www.razerzone.com/de-de/gaming-controllers/razer-hydra-portal-2-bundle/

Richter, U. M. (2009). *Controlled Self-Organisation Using Learning Classifier Systems*. Ph.D. thesis, Universität Karlsruhe (TH), Fakultät für Wirtschaftswissenschaften, Karlsruhe, DE.

Rudolph, S., Tomforde, S., and Hähner, J. (2016). "A mutual influence-based learning algorithm," in *Proceedings of the 8th International Conference on Agents and Artificial Intelligence (ICAART16)* (Rome: INSTICC, ScitePress), 181–189.

Rudolph, S., Tomforde, S., Sick, B., and Hähner, J. (2015a). "A mutual influence detection algorithm for systems with local performance measurement," in *Proceedings of the 9th IEEE International Conference on Self-adapting and Self-organising Systems (SASO15)* (Boston, MA: IEEE Press), 144–150.

Rudolph, S., Tomforde, S., Sick, B., Heck, H., Wacker, A., and Hähner, J. (2015b). "An online influence detection algorithm for organic computing systems," in *Proceedings of the 28th GI/ITG International Conference on Architecture of Computing Systems* (Porto: VDE Verlag), 1–8.

Sarpe, V., Esmaeili, A., Yazdanbod, I., Kubik, T., Richter, M., and Jacob, C. (2010). "Parametric evolution of a bacterial signalling system formalized by membrane computing," in *CEC 2010, IEEE Congress on Evolutionary Computation* (Barcelona, Spain: IEEE Press), 1–8.

Schock, H.-J. (1997). *Soft Shells: Design and Technology of Tensile Architecture*. Basel, CH: Birkhäuser.

Shen, S., Michael, N., and Kumar, V. (2011). "Autonomous multi-floor indoor navigation with a computationally constrained mav," in *International Conference on Robotics and automation (ICRA), 2011 IEEE* (Shanghai: IEEE), 20–25.

Sommer, M., Tomforde, S., and Hähner, J. (2016). "Predictive load balancing in cloud computing environments based on ensemble forecasting," in *Proceedings of the 13th IEEE International Conference on Autonomic Computing (ICAC)* (Würzburg: IEEE), 300–307.

Stein, A., Rauh, D., Tomforde, S., and Hähner, J. (2016). "Augmentic the algorithmic structure of XCS by means of interpolation," in *Proceedings of the 29th GI/ITG International Conference on Architecture of Computing Systems (ARCS)* (Nuremberg: VDE Verlag), 348–360.

Sumpter, D. J., and Beekman, M. (2003). From nonlinearity to optimality: pheromone trail foraging by ants. *Anim. Behav.* 66, 273–280. doi:10.1006/anbe.2003.2224

Tomforde, S., Cakar, E., and Hähner, J. (2009). "Dynamic control of network protocols – a new vision for future self-organised networks," in *Proceedings of the 6th International Conference on Informatics in Control, Automation, and Robotics (ICINCO'09), Held in Milan, Italy (2 – 5 July, 2009)*, eds J. Filipe, J. A. Cetto, and J.-L. Ferrier (Milan: INSTICC), 285–290.

Tomforde, S., Hähner, J., and Sick, B. (2014). Interwoven systems. *Informatik-Spektrum* 37, 483–487. doi:10.1007/s00287-014-0827-z

Tomforde, S., Hurling, B., and Hähner, J. (2010). "Dynamic control of mobile ad-hoc networks – network protocol parameter adaptation using organic network control," in *Proc. of the 7th Int. Conf. on Informatics in Control, Automation, and Robotics (ICINCO'10), Held in Funchal, Portugal (June 15 – 18, 2010)* (Setubal: INSTICC), 28–35.

Tomforde, S., and Müller-Schloer, C. (2014). Incremental design of adaptive systems. *J. Ambient Intell. Smart Environ.* 6, 179–198. doi:10.3233/AIS-140252

Tomforde, S., Prothmann, H., Branke, J., Hähner, J., Mnif, M., Müller-Schloer, C., et al. (2011). "Observation and control of organic systems," in *Organic Computing – A Paradigm Shift for Complex Systems, Autonomic Systems*, eds C. Müller-Schloer, H. Schmeck, and T. Ungerer (Basel, CH: Birkhäuser Verlag), 325–338.

Tomforde, S., Rudolph, S., Bellman, K., and Würtz, R. (2016). "An organic computing perspective on self-improving system interweaving at runtime," in *Proceedings of the 13th IEEE International Conference on Autonomic Computing (ICAC)* (Würzburg: IEEE Press), 276–284.

Unity Technologies. (2014). *Unity – Game Engine*. Available at: http://unity3d.com/

Urbanowicz, R. J., and Moore, J. H. (2009). Learning classifier systems: a complete introduction, review, and roadmap. *J. Artif. Evol. Appl.* 2009, 1. doi:10.1155/2009/736398

Vasile, C., Pavel, A., and Buiu, C. (2011). "Integrating human swarm interaction in a distributed robotic control system," in *Proceedings of the IEEE Conference on Automation Science and Engineering (CASE)* (Trieste: IEEE), 743–748.

Vincent, P., and Rubin, I. (2004). "A framework and analysis for cooperative search using uav swarms," in *Proceedings of the ACM Symposium on Applied Computing (SAC)* (Nicosia: ACM Press), 79–86.

Voelker, S., Wacharamanotham, C., and Borchers, J. (2013). "An evaluation of state switching methods for indirect touch systems," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris: ACM), 745–754.

von Frisch, K. (1974). *Animal Architecture*. New York: Harcout Brace Jovanovich.

von Mammen, S., Lehner, P., and Tomforde, S. (2016). "Evolving a facade-servicing quadrotor ensamble," in *Proceedings of COGNITIVE 2016 the Eighth International Conference on Advanced Cognitive Technologies and Applications (COGNITIVE)* (Rome, Italy: IARIA, ThinkMind), 16–21.

von Mammen, S., Tomforde, S., Hähner, J., Lehner, P., Förschner, L., Hiemer, A., et al. (2014). "Ocbotics: an organic computing approach to collaborative robotic swarms," in *IEEE Symposium on Swarm Intelligence (SIS), 2014* (Orlando: IEEE Press), 1–8.

Willmann, J., Augugliaro, F., Cadalbert, T., D'Andrea, R., Gramazio, F., and Kohler, M. (2012). Aerial robotic construction towards a new field of architectural research. *Int. J. Arch. Comput.* 10, 439–460. doi:10.1260/1478-0771.10.3.439

Wilson, S. (2000). "Get real! XCS with continuous-valued inputs," in *Learning Classifier Systems, Volume 1813 of Lecture Notes in Computer Science*, eds P. Lanzi, W. Stolzmann, and S. Wilson (Berlin, Heidelberg: Springer), 209–219.

Wilson, S. W. (1995). Classifier fitness based on accuracy. *Evol. Comput.* 3, 149–175. doi:10.1162/evco.1995.3.2.149

Wooldridge, M. J. (2009). *An Introduction to Multiagent Systems*, 2nd Edn. West Sussex, UK: John Wiley and Sons Ltd.

Yang, Y., Polycarpou, M. M., and Minai, A. A. (2007). Multi-uav cooperative search using an opportunistic learning method. *J. Dyn. Syst. Meas. Control* 129, 716–728. doi:10.1115/1.2764515

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.