Evolving a Facade-Servicing Quadrotor Ensamble

Sebastian von Mammen, Patrick Lehner and Sven Tomforde

Organic Computing, University of Augsburg, Germany
Email: sebastian.von.mammen@informatik.uni-augsburg.de
patrick.lehner@student.uni-augsburg.de
sven.tomforde@informatik.uni-augsburg.de

***Abstract*—The fast evolution of quadrotors paves the way for the application of robotic maintenance on walls and facades. In this work, we present an application-oriented modelling and simulation effort to evolve self-organising behaviour of a set of quadrotors. In particular, we rely on Genetic Programming (GP) to optimise control programs to collaboratively direct the route of autonomous quadrotors across a (vertical) rectangular surface relying on local knowledge only. In order to account for various real-world constraints, we made use of a three-dimensional, physical model that considers battery consumption, collisions, and rudimentary avionics. The evolved control programs that link sensory information to actuation could be executed on the Robot Operating System (ROS) available for numerous robotic systems. Our results show that very simple evolved control programs (moving left until the battery is drained, starting from a random location) perform better than those that we had previously engineered ourselves.**

Keywords–Quadrotors; ensambles; genetic programming; lawnmower problem; facades.

I. INTRODUCTION

Performing repetitive tasks across a large surface is an apt target for automation. Accordingly, several generations of semi-autonomous vacuum cleaners and lawnmowers have already entered the consumer market [1], [2]. Fast technological advances in quadrotors [3] promise versatile task automation on surfaces also in three dimensions, such as cleaning building facades [4].

Inspired by the efficient and robust collaboration of social insects [5], for instance in building their nests, we especially consider the case of numerous quadrotors working on a facade concurrently. To a great extent, social insects coordinate their efforts by means of indirect communication through the environment, or stigmergy [3]. In this fashion, all the members of the colony can work based on local needs, which assures that all the local actions are taken to meet global goals and that they can be executed in parallel.

It is a challenge to find the best possible behaviour for each colony member to make such a self-organised setup work. We have developed a model for facade maintenance by a quadrotor ensamble and evolved behaviours for the homogeneous individuals in physical simulations. After giving credit to related works in the context of GP and the Lawnmower Problem, we outline our simulation model in Section III. We provide details and results of differently staged evolutionary experiments in Section IV, which are discussed subsequently in Section V. We conclude this paper with a summary and an outlook on potential future work.

II. RELATED WORK

Our contribution builds on preceding works from the fields of GP and Evolutionary and Swarm Robotics. A recent survey of Evolutionary Robotics stresses the challenges of modular and soft robotics, evolvability of a system, self-organisation, and the gap between evolved models and their applicability to reality [6]. We take the latter challenge into consideration by providing a comprehensive simulation approach based on the physics-enabled robotics simulator Gazebo [7] and ROS [8]. It will be outlined in detail in the next section.

In terms of self-organisation, several works have influenced our design. Lerman and Galstyan [9] have introduced a method for macroscopic analysis of the behaviour of a robotic swarm's members. In their scenario, a homogeneous group of robots must perform two distinct but similar tasks in one target area. The individuals autonomously switch between the two tasks solely based on local information. Based on a limited memory they can estimate the state of the global task and derive local decisions. Jones and Matarić have investigated the effect of the memory capacity in such a collaborative setup [10]. In order to speed up work across a large area, Schneider-Fontán and Matarić split the overall surface into discrete segments and assigned the segments to each robot [11].

The task that our robot ensamble is evolved to address is similar to the Lawnmower Problem, introduced by Koza in 1994 [12]. The challenge here is to efficiently traverse a discretised rectangular surface moving along cardinal directions. Alongside the problem, Koza presented first solutions based on GP techniques [12]. GP is an evolutionary approach especially suited to generate new programming code or behaviours, working on according (syntax-)tree structures. In general, evolutionary computing approaches are often used when novel behaviours need to be generated and optimised at the same time. Random sets of candidate solutions to a problem, or *populations of individuals*, are created at the beginning of an evolutionary algorithm and slightly modified and recombined over several generations to evolve their performances.

After Koza's work, the Lawnmower problem has repeatedly been used as a measure of reference for evaluating the performance of Evolutionary Computing approaches, examples are found in [13], [14], [15].

Extrapolation to arbitrary polygons have also been considered [16]. Nevertheless, we focus on rectangular surfaces with the extension of considering the physicality of our interacting agents and several agents working concurrently.

 ISBN: 978-1-61208-462-6

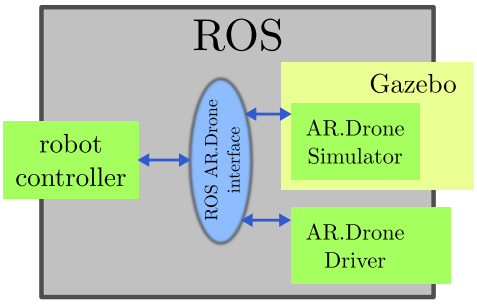

Figure 1. Interwoven software modules for efficient, accurate simulation.

III. Physical and Behavioural Model

In order to establish a physical simulation model of a quadrotor ensamble, we rely on Gazebo [7], which uses the *Open Dynamics Engine* (ODE) for physics simulation [17] and which is supported by an extensive library of tools and sensor implementations. The utilisation of an established physics engine ensured that efficient and accurate collision detection routines were in place to guide the quadrotor agents and also to automatically detect if they crashed. As only very basic Newton-Euler equations for integrating the drones' movements and rigid-body collision detection was needed in the scope of our simulations, most other efficient 3D physics engines would have worked for us as well (e.g., PhysX or Bullet).

By means of a plugin, Gazebo simulates the flight behaviour of the Parrot AR.Drone 2.0 quadrotor, an affordable off-the-shelf consumer product. In addition, Gazebo integrates natively with the ROS [8], a software distribution providing a unifying infrastructure for programming robot controllers. Among numerous hardware designs, ROS also supports the AR.Drone 2.0 hardware. As a result of the tight integration of Gazebo and ROS, the same set of control commands and telemetry reports is used for the simulation model as in reality. Figure 1 schematically shows the relationships of the involved software components to provide for a physics-enabled, agent-based simulation of quadrotor ensambles based on the Parrot AR.Drone 2.0 technology. The setup allows for in-place switching between simulated and real hardware.

The concrete task of the quadrotors is visualised in Figure 2: (a) shows how perpendicular facades border a ground surface on two sides. The quadrotors need to hover from randomly chosen points on the ground surface (wherever they are placed) to the respective facades and clean individual cells, before returning to their origins for a battery recharge. Figure 2 (b) shows the arrangement of cells to work on. A cell's colour reflects its dirtiness (from white/clean to red/dirty). The blue normals identify the front of the cells The quadrotors are randomly placed on the ground between two perpendicular facades. A grid is superimposed on the facades that divides it into cells with dimensions similar to one quadrotor. In our model, we assume that the quadrotor can determine the degree of dirtiness of each cell that it can see from a small distance. Figure 3 shows the perceived and augmented view of the quadrotor agents on the facades. In Figure 3(a), the gray translucent pyramid shows the agent's field of view. The six green, labeled cells are considered visible, whereas the red ones are not. Figure 3 (b) depicts a section of the grid of vantage points for the quadrotor agents. From these points, the

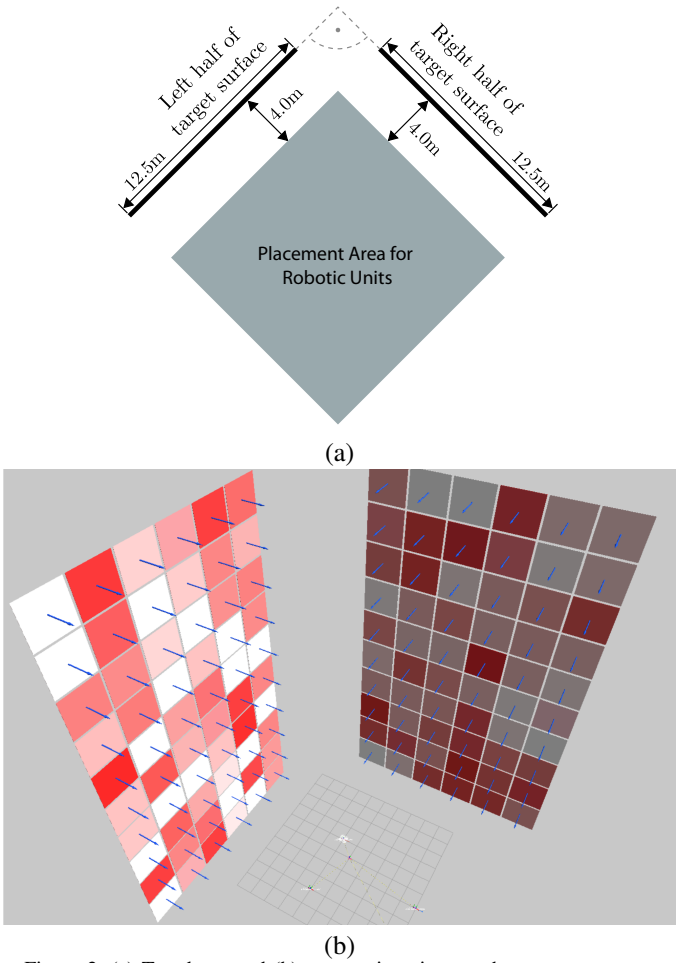

Figure 2. (a) Top-down and (b) perspective view on the assay setup.

agents determine what to do next. The green spheres represent the vantage points, the red arrows illustrate neighbour relations between them. For two exemplary vantage points, the view frustums are also included.

The quadrotor is heading towards according *vantage points* in front of the facade to perceive a number of cells and ensuite to determine which course of action it should pursue, i.e., to work on one of the cells it sees, to move to a neighbouring vantage point, or to return to its origin on the ground to recharge. These states and activities are summarised in Figure 4. Here, the edge labels describe the conditions triggering state transitions. Elliptical outlines denote longer-term states, while the square outline marks a transient decision-making state.

IV. Evolutionary Experiments

Based on the model outlined in the previous section, we bred behaviours for a homogeneous ensamble of quadrotors that result in efficient collaborative cleaning behaviours by means of the *Evolving Objects* framework [18]. In a first phase of evolutionary experiments, randomised populations filter the vast search space of valid configurations, or *genotypes*, for viable individuals. In a second phase, we use the best individuals from the first phase to seed the populations.

Each genetic experiment follows the evolution cycle depicted in Figure 5. The diagram shown is loosely based

 ISBN: 978-1-61208-462-6

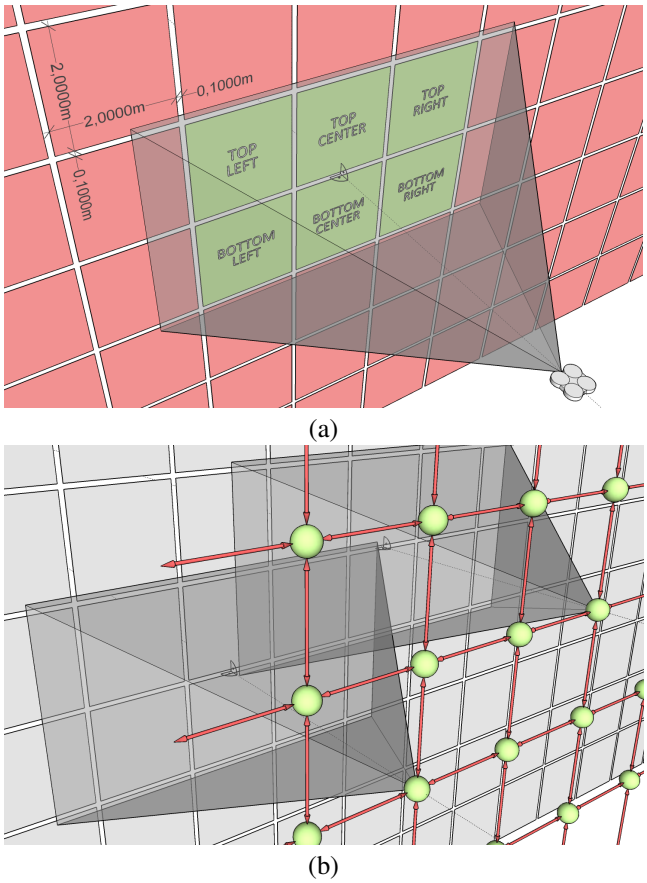

Figure 3. The view of a quadrotor agent (a) in relation to the projected facade grid and (b) considering the grid of flight positions.

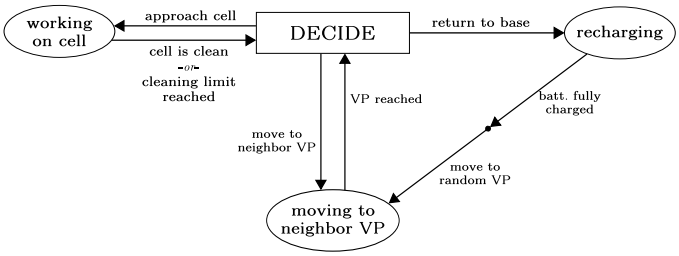

Figure 4. High-level state diagram of an agent.

on [19]: Elliptical nodes represent populations, rectangular outlines denote GP operators. The transition arrows represent the specimen flow between operators. Edge labels denote the groups' semantics and sizes in relation to the total population size N_P. The evolution cycle breeds a population of size N_P for a maximum of G_{max} generations. The individuals represent homogeneous flocks of N_R quadrotors, the number of facade cells N_C is proportional to the size of the flock. At the beginning of each simulation, a total amount of dirt of $\frac{1}{2}N_C$ is distributed randomly across the target surface so that the cells have an average dirt value of 0.5. Each simulation has an upper time limit t_{limit} of simulated time. Once the simulation finishes, the flock's penalty value is calculated by

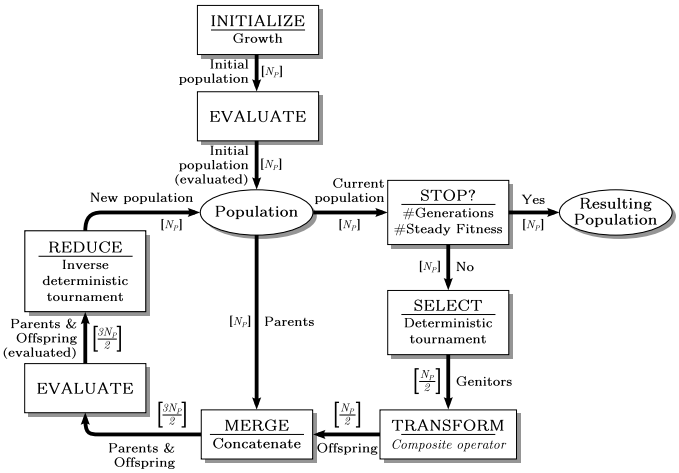

Figure 5. GP cycle used in our experiments.

means of the following equation.

$$h(i) = D_{remain} \cdot 50 + \sum_{j=1}^{N_R} h_{r_j}(i)$$

where $D_{remain} \in [0, \frac{1}{2}N_C]$ is the total amount of dirt remaining on the target surface and each quadrotor r_j contributes a penalty share $h_{r_j}(i)$ defined as:

$$h_{r_j}(i) = t_c + E_c \cdot 100 + b_{limit} \cdot 500 + b_{static} \cdot 500 + b_{drained} \cdot 2000 + n_i \cdot 300$$

where t_c and E_c denote the time and, respectively, energy until completion, the booleans b_{limit}, b_{static} and $b_{drained}$ indicate whether the time limit was reached, the quadrotor never moved, or its battery got fully drained, and n_i denotes the number invalid action selections. The coefficients reflect the weightings we deemed appropriate considering the factors' contributions to the overall performance.

In order to minimise the penalty values, GP operators modify the genetic encodings of the flocks, i.e., the decision function of the quadrotors encoded in strongly-typed tree-structured programs. These trees connect non-terminal nodes for *control flow*, *boolean* and *arithmetic* operators, and *actions* that would move the quadrotors into a new state (see Figure 4). Terminal nodes of the trees can comprise closed instructions, such as returning to the base, or information about the system state, such as the current distance to the base station, the remaining battery life, status information about the perceived cells of the facade, or arbitrary constants. In order to narrow down the search space, we ensured to only consider syntactically correct genotypes with tree-depths of at most 30 that include instructions to return to the base, to move to a neighbouring vantage point and to approach a cell of the facade—without these three basic instructions, the quadrotor could not possibly succeed in its task. We further provide support functions to let the quadrotor move to neighbouring vantage points, fly back to the base and recharge, and to let it test the presence and return cells in its field of view within a certain dirt range.

 ISBN: 978-1-61208-462-6

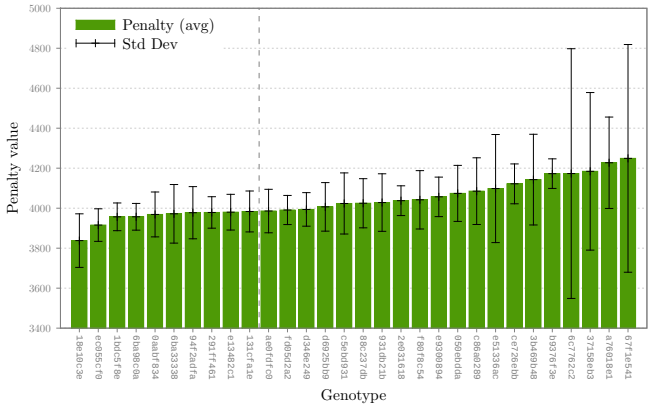

Figure 6. The average penalty and standard deviation across 10 simulations for each genotype.

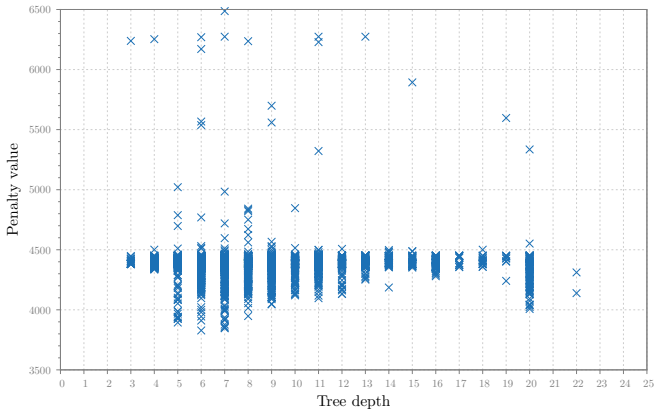

Figure 7. Each data point shows the (non-averaged) penalty value vs. syntax tree depth of a genotype.

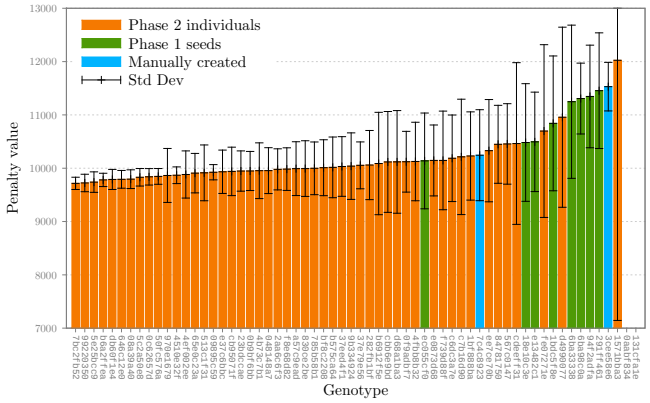

Figure 8. Penalty ranking of the second phase of evolutionary experiments.

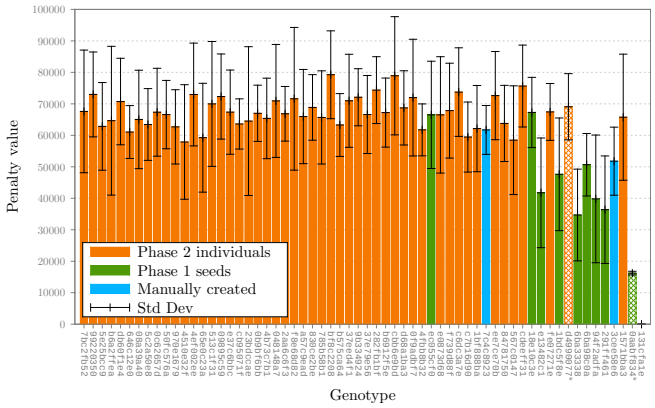

Figure 9. Re-evaluation of the best 50 bred individuals.

A. *Preselection*

In a first relatively unconstrained phase of experiments, we were looking for a diverse set of viable solutions. Therefore, we setup three trials to generate rather large populations comprising 50 and 100 individuals, breeding them only for 20 and 10 generations, respectively. Although our experiments ran on a distributed infrastructure, the heavy computational burden of the runs lasting 900 simulated seconds did not allow us to consider more than two quadrotors in this first phase of evolutionary trials.

In order to identify the best solutions of the first phase, we merged the penalty value for all genotypes into one preliminary ranking. Subsequently, we re-evaluated the best 30 individuals ten more times in order to validate the statistical significance of the ranking. Figure 6 shows the according ranking based on the genotypes' average penalty value. To the left of the vertical gray dashed line are the ten individuals we consider the best solutions of the first phase. We observe strong similarities in the performances of the best individuals, achieving a penalty value within a small margin around 4000. Upon inspecting the structure of these individuals, we found that several of them are members of the same lineages, having syntax trees of similar structure, differing only in minor branches.

Figure 7 shows the penalty values of the individuals on the vertical axis in relation to the according syntax trees' depths on the horizontal axis, with each data point representing one evaluation. The lower boundary of the data points in the figure is particularly interesting, as it indicates that very small syntax tree depths of three and four do not yield good performances. The respective specimen do not achieve penalty value lower than 4300, but on average, most data points at these depths are below 4500, which is not a poor performance considering the overall results. We found the best individuals in the narrow range from depths five to eight. The relatively large number of evaluations in this range reflects a prevalence of genotypes with these depth values in the observed populations. In the depth range from nine to 19, we see only average performance. Note that the depth values 17 through 19 show relatively few data points, especially compared with the much greater number of evaluations and the renewed performance peak at depth 20. Overall, frequent evaluations for individuals that achieved a penalty value of about 4400 are displayed. This penalty baseline represents individuals, which are not particularly effective in terms of the cleaning task but that successfully avoid physical damage.

B. *Refinement*

In addition to 20 randomly generated individuals, we fed the best results of the first phase into the second phase

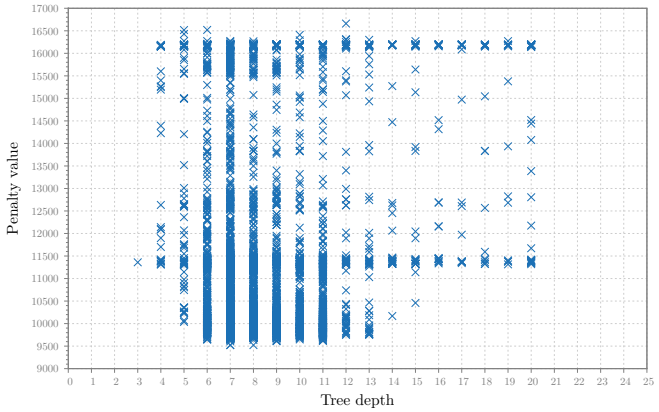

Figure 10. Penalty values vs. syntax tree depth in the second phase of evolutionary experiments.

of experiments. This phase was more directed, introducing some restrictions, as well as simplifications to speed up the simulations. At the same time, to increase the pressure towards effective collaboration, we increased the number of simulated quadrotors from two to four, also imposing a duplication of facade cells (previously 60, now 120). An extended time limit of $t_{limit} = 1500s$ forces the quadrotors to land at least once, an activity they could avoid during the first phase of experiments, as their batteries support up to $1200s$ of flight. The GP parameters have been tweaked to a reduced population size of $N_P = 30$, maintaining a maximum number of $G_{max} = 50$ generations. Figure 8 shows the merged results of the second phase: 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 re-evaluated in the second simulation scenario. In Figure 9 subsequent re-evaluations of the best 50 individuals are shown (the transparent bars indicate that the respective specimens failed to complete all evaluation runs). With an increase of the maximally simulated time from $t_{limit} = 1500$ to $20000s$, most of the best bred individuals improve their performance. However, although most individuals further reduce their penalty value, the previous ranking cannot be maintained (compare with Fig.8).

In Figure 10, we see the (non-averaged) penalty value calculated in the second phase of evolutionary experiments vs. the associated genotype's syntax tree depth. Again, 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. In analogy to the results from the first phase, the best individuals are still located in the range of depths five to eight. However, different from the results of the first phase, where a steady increase in penalty from depths nine to 13 can be seen (from about 4100 to 4300), 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, stabilising at about 11300. This time the scattered points aggregate along two horizontal lines, one at a penalty value of around 11400, the other one at about 16200. Again, they emerge due to genotypes that are not particularly effective but not particularly bad either. 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 initialisations in phase two.

V. Discussion

In the previous section, we compared the evolved quadrotor behaviours to two manually engineered genotypes with IDs $7c4c8923$ and $3cee58e6$. The first would return to the base station to recharge, if necessary (less than 10% battery life remaining). Next, it would choose to work on a dirty cell in its field of view. It gives priority to cells with high degrees of dirt (equal or above 0.8). In the absence of heavily dirtied cells, a cell with value between 0.3 and 0.8 is chosen with 50% chance. If no cell is chosen, the quadrotor flies to the next available vantage point to its right or below. Note that cells with values below 0.3 are not considered. As a result, after a some time, the quadrotor moves from one vantage point to the next without doing any actual work, see Figure 9.

The other preconceived genotype, ID $3cee58e6$, again starts out with conditional recharging. Next, depending on their degree of dirt, it may work on one of the two cells at the centre of its field of view. Alternatively, it returns to the base station, recharging the batteries, and to approach a new, arbitrarily chosen vantage point afterwards. Approaching a random vantage point after recharge is also exploited by well-performing specimen bred throughout our genetic experiments.

The best genotype that emerged from our evolutionary experiments carries the ID $7bc2fb52$. If the upper-left cell in its field of view is clean, it moves to the vantage point to the left, if available, and to the base station, otherwise. If the upper-left cell is dirty, it either starts cleaning this cell or any other cell that has accumulated even more dirt. This process is repeated until the upper-left cell is finally addressed and the quadrotor moves to the next vantage point (possibly diverting past the base station). As a result, the quadrotor works through single rows of vantage points, moving to the left whenever the top left cell of their field of vision is clean and returning to their base station when it reachs the left border of the target surface. This behaviour is only more efficient than our engineered specimen, given an overall high number of dirty cells. With a decline of dirty cells over time, its performance drops, as can be seen in the results of the longer, second experimental runs (Figure 8).

In the further prolonged re-evaluation runs summarised in Figure 9, ID $6ba33338$, evolved within the first set of experiments, performed best. This specimen flies to the base station, if the lower-left cell is clean – unless the upper-left cell is also clean, in which case it moves to the left-hand vantage point, if available. Otherwise, it starts cleaning the (dirty) lower-left cell or any other dirtier cell. However, the probability that a dirtier cell is selected is directly proportional to the remaining battery life. This implies that given less energy, it is better to not start working on rather dirty cells, as this will take longer and use more battery.

Due to the performance requirements of the prolonged simulation scenario, it was not eligible for evaluation within an evolutionary setup. It proved useful, however, for the purpose

 ISBN: 978-1-61208-462-6

of testing the scalability of the bred solutions. For instance, it clearly showed that our refinement runs suffered from overfitting. That is the best specimen in the second experiment phase were bred to remove as much dirt as possible within the first 1500 simulated seconds, not addressing the need to find leftover dirty spots on the facade. This insight stresses an important weakness in our approach: Instead of a single, if partially randomised, simulation scenario, another study has to be conducted emphasising variation in order to prevent overfitting.

VI. Conclusion and Future Work

We presented an approach to self-organised quadrotor ensambles to perform homogeneous tasks on large surfaces. We detailed the physical simulation model, as well as the individuals' behavioural representation. Our results show GP experiments that led to self-organising behaviours better than manually engineered ones. Yet, as pointed out in the discussion, more robust and more generic behaviours have to be bred. This might be achieved by an extension of the training set, i.e., by a larger pool of experiment scenarios. However, as the simulation is the performance bottleneck of our approach, a related goal is to speed up the robot simulation while preserving its accuracy. Furthermore, our preliminary investigations were limited to syntax trees with a depth of 20 or lower. The statistical results of our first evolutionary trials suggested that larger syntax trees might perform as well as or even better than those observed. Hence, another future endeavour might be a more strategic examination of the solution space.

References

[1] J. Forlizzi and C. DiSalvo, "Service robots in the domestic environment: a study of the roomba vacuum in the home," in Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction. ACM, 2006, pp. 258–265.

[2] H. Sahin and L. Guvenc, "Household robotics: autonomous devices for vacuuming and lawn mowing [applications of control]," Control Systems, IEEE, vol. 27, no. 2, 2007, pp. 20–96.

[3] S. Gupte, P. I. T. Mohandas, and J. M. Conrad, "A survey of quadrotor unmanned aerial vehicles," in Southeastcon, 2012 Proceedings of IEEE. IEEE, 2012, pp. 1–6.

[4] A. Albers et al., "Semi-autonomous flying robot for physical interaction with environment," in Robotics Automation and Mechatronics (RAM), 2010 IEEE Conference on. IEEE, 2010, pp. 441–446.

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

[6] J. C. Bongard, "Evolutionary Robotics," Communications of the ACM, vol. 56, no. 8, 2013, pp. 74–83.

[7] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on, vol. 3. IEEE, 2004, pp. 2149–2154.

[8] M. Quigley et al., "ROS: an open-source Robot Operating System," in ICRA workshop on open source software. IEEE, 2009, pp. 1–6.

[9] K. Lerman and A. Galstyan, "Macroscopic analysis of adaptive task allocation in robots," in Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on, vol. 2. IEEE, 2003, pp. 1951–1956.

[10] C. Jones and M. J. Mataric, "Adaptive division of labor in large-scale minimalist multi-robot systems," in Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on, vol. 2. IEEE, 2003, pp. 1969–1974.

[11] M. Schneider-Fontan and M. J. Mataric, "Territorial multi-robot task division," Robotics and Automation, IEEE Transactions on, vol. 14, no. 5, 1998, pp. 815–822.

[12] J. R. Koza, Genetic programming II: automatic discovery of reusable programs. MIT press, 1994.

[13] W. S. Bruce, "The lawnmower problem revisited: Stack-based genetic programming and automatically defined functions," in Genetic Programming 1997: Proceedings of the Second Annual Conference. Morgan Kaufmann, 1997, pp. 52–57.

[14] K. E. Kinnear, L. Spector, and P. J. Angeline, Advances in genetic programming. MIT press, 1999, vol. 3.

[15] J. A. Walker and J. F. Miller, "Embedded cartesian genetic programming and the lawnmower and hierarchical-if-and-only-if problems," in Proceedings of the 8th annual conference on Genetic and evolutionary computation. ACM, 2006, pp. 911–918.

[16] E. M. Arkin, S. P. Fekete, and J. S. Mitchell, "Approximation algorithms for lawn mowing and milling," Computational Geometry, vol. 17, no. 1, 2000, pp. 25–50.

[17] E. Drumwright, J. Hsu, N. Koenig, and D. Shell, "Extending open dynamics engine for robotics simulation," in Simulation, Modeling, and Programming for Autonomous Robots. Springer, 2010, pp. 38–50.

[18] M. Keijzer, J. J. Merelo, G. Romero, and M. Schoenauer, "Evolving Objects: A general purpose evolutionary computation library," Artificial Evolution, vol. 2310, 2002, pp. 829–888.

[19] M. Schönauer, "EO tutorial," http://eodev.sourceforge.net/eo/tutorial/html/eoTutorial.html, retrieved January 2016.

 ISBN: 978-1-61208-462-6