Comparison of the Behavior of Swarm Robots with their Computer Simulations Applying Target-Searching Algorithms

Vivienne Jia Zhong, Rolf Dornberger and Thomas Hanne
Institute for Information Systems, University of Applied Sciences and Arts Northwestern Switzerland, Basel/Olten, Switzerland
Email: {viviennejia.zhong, rolf.dornberger, thomas.hanne}@fhnw.ch

***Abstract*—This paper investigates the functionality and quality of the implementation of a search- and target-surrounding swarm robotic algorithm using physical swarm robots named Kilobots. The implementation was developed and tested in the simulator V-REP, then transferred onto the actually running Kilobots: Ten Kilobots were used for the experiment, where one Kilobot acts as the target and nine Kilobots act as the searchers. The algorithm allows the searchers to swarm out to find the target while avoiding collisions with other searchers, to orbit around other searchers, which are closer to the target, and finally to surround the target once it is found. The results of the implementation using the physical Kilobots are compared with the results of two adjusted computer simulations. Differences between the simulations and the real robot implementation are investigated: Discrepancies regarding the locomotion and the communication capabilities are identified and discussed.**

***Index Terms* — search-surrounding swarm algorithm, target-surrounding swarm algorithm, collective behavior, swarm robotics, swarm intelligence, Kilobot**

I. Introduction

This article is extending the conference paper of Zhong et al. [1] applying an additional experimentation. Compared to the conference paper, the article extends, investigates and discusses the implementation of a search- and target-surrounding swarm robotic algorithm using a set of ten real swarm robots, the so-called Kilobots [2]. A Kilobot is a robotic system, which was developed by the Self-Organizing Systems Research Group of Harvard University [2]. A Kilobot robot is equipped with vibration-based differential drive locomotion, on-board computation power and neighbor-to-neighbor communication with a distance sensing capability of up to 10 cm using infrared light.

In their introduction paper, Rubenstein et al. have demonstrated some Kilobot capabilities, such as orbiting where a Kilobot moves along a circular path with the assistance of a stationary robot by computing the distance to this stationary robot [2]. In another work, Rubenstein et al. demonstrated a self-assembly algorithm consisting of three collective behaviors: edge-following, gradient formation and localization [3]. In the algorithm for gradient formation, every Kilobot sends a gradient value message to its surrounding Kilobots, where this value is transmitted from one Kilobot to another.

Magsino et al. reported the use of Kilobots to develop a set of collective algorithms for target-surrounding and search-and-rescue problems [4]. They validated their algorithms in the simulator V-REP. The goal of the target-surrounding algorithm is to surround a target Kilobot (target) with a group of other Kilobots (searchers). The algorithm uses the gradient value to determine the distance towards the target. Magsino et al. proposed three search-and-rescue algorithms. One of them is the dispersal search algorithm: The setup consists of a group of searchers, a target and a base. Both the searchers and the target are moving arbitrarily. Only the base is stationary. Once a searcher finds the target, the specific searcher becomes the leader and the target becomes the first follower. Other searchers join the group once they receive a message from the leader. The group returns to the base once the group is complete [4].

II. Background and Related Work

During the last ten years, the study of multi-robot systems attracted considerable interest in research. Swarm robotic systems came into focus, which do not use a central or hierarchical control of the whole system, but are based on individual robots' behavior including sensor and communication features (e.g. Şahin and Winfieldq [5]). It is generally assumed and often observed that such systems lead to a sufficiently good coordination of the behavior of the robots. Patterns of movement or other forms of collective behavior are often similar to observations among animals that form swarms or show other types of more or less coordinated group behavior. Even with rather simple movement, communication and control capabilities, it is said that the resulting collective behavior might be rather complex and suitable to solve particular problems such as mapping the environment or finding particular locations. Similar as for animals and for more abstract models and algorithms, it is frequently referred to as swarm intelligence [6]. For the respective robots, it is mostly assumed that they can perform local interactions with their environment or other robots and that the applied control logic or rules are rather simple. The robots are

Manuscript received June 1, 2018; revised August 1, 2018.

considered to be autonomous, but may achieve common decisions or a cooperative solution of tasks by simple means of communication which usually do not allow for communicating with all other robots, but, for instance, only with those in the neighborhood.

Besides theoretical studies, such swarm robotic systems can be analyzed by experiments. The experiments can be performed with real robots or by using a model of the robots. The models can be microscopic or macroscopic [6]. In particular, for microscopic models, simulation approaches can be employed, which represent the individual behavior of robots (including the sensing of the environment, communication, and control characteristics) and the environment they interact with. Currently, quite a number of different robot simulators exist with different levels of detail, accuracy, usability, and maturity. For instance, in Vaughan [7] seven simulation tools are briefly examined. The survey by Craighead et al. [8] discusses 14 tools, which also include simulators not related to robotics such as a flight simulator. In Castillo-Pizarro et al. [9], 14 software packages for mobile robot simulation are mentioned. 12 simulators are considered in Ivaldi et al. [10] who also show some evaluation results based on different criteria. The survey by Cook, Vardy, and Lewis [11] discusses eight simulation tools with a focus on autonomous vehicles.

Robot simulators are a convenient means for analyzing robot behavior, the emergent behavior of a group of robots or the effectiveness or efficiency of solving given problems. In particular, the acquisition of real, physical robots is not required and the effort to set up real experiments, conduct them, and analyze the results can be significantly reduced. For instance, for the Stage simulator, it is reported [7] that it runs 1000 times faster than the real system and is able to simulate systems of up to 100000 robots, which would hardly be possible in a real experiment. A possible disadvantage of robot simulation is the question of how good the simulations are and how accurately they reflect the behavior of real robots.

In most cases, the representation of the control logic of a real robot in a respective simulator should not be a problem (especially when non-automatic programming approaches are applied, see Biggs and MacDonald [12]). Instead, the main problem in robot simulation is frequently the accuracy of the robot geometry and kinematics, the modelling of the sensors, the environment model and the model of robot interaction with this environment, including inter-robot communication. On the one hand, there may be deviations of the respective models from reality, just because of the model complexity or granularity, the insufficient tuning of the models or deviations of individual robots from the models due to manufacturing deviations, and the noise in any physical experiment. On the other hand, basic problems of the modelling approaches such as the step size applied in simulations can be relevant.

For single robot simulation, there are comparatively many results, often showing good accuracy, even under adverse conditions such as noise, e.g. in Balaguer, Carpin, and Balakirsky [13]. However, with multi-robot simulations, fewer results are available and the accuracy can be more severely affected by collective effects of communication or robot collisions. In Castillo-Pizarro et al. [9], three robot simulators (Carmen, Gazebo, and ODE) are compared and errors regarding the deviation in movements (end-point position errors) are investigated. As a result, these deviations are considered significant (21-29 cm for relatively small distances).The errors result as cumulative simulation errors because of fluctuations in the range of movements, and differences in relation to the direction of movement and rotation angle deviations between model and reality. In the paper by Carpin et al. [14], a generally high accuracy is reported for the USARSim tool, which offers good rendering and physical simulation. Relevant problems in accuracy are only mentioned for legged robots. Specific problems with contacts or collisions within humanoid robot simulation are discussed in Ivaldi et al. [10]. Kudelski, Gambardella, and Di Caro [15] analyze the effect of the communication model on the simulation accuracy. Moreover, the effect of the simulation step size is studied. Good results are found for very exact, but computationally costly model variants, whereas less detailed models lead to less accurate results.

III. Problem Description

The problems addressed by this paper are twofold: First, as discussed above, it is (almost) impossible to fully accurate model the movements, sensations and communications of the robots with others in computer-based simulation, compared to the situation in a real world experiment. Due to this discrepancy, the algorithm might not operate the same way as expected in the real robotic collective environment. Moreover, the simulation may hide scaling issues within the algorithm that can be discovered only when the algorithm is operated on a large collection of robots [2].

Secondly, the collision avoidance of robots is an important issue in the real world, because this could greatly affect the localization of robots. However, this was considered neither in Magsino et al.'s work [4] nor in the algorithms developed by Self-Organizing Systems Research Group [16]–[18].

In this paper, we aim to implement a search- and target-surrounding algorithm for Kilobot robots that incorporates collision avoidance. We test the mechanism both in a simulator and on real Kilobots. Then we discuss the results of both the actual implementation and the simulation.

IV. Algorithms

The dispersal and orbiting algorithm [16]–[18] served as the starting point for the development of our search- and target-surrounding algorithm. The algorithm consists of three parts: searching, approaching the target and orbiting a fellow.

A. *Searching*

The goal of this part is to allow a searcher to move randomly and avoid collisions with other searchers by detecting its neighbors. At start, each of the searchers is initialized with a gradient value of 0. It broadcasts its gradient value and listens to the environment (infrared light) roughly twice in a second.

The dispersal algorithm enables a random walk. The randomness is determined by dicing: A searcher can walk straight, turn left or right. Each direction has a one-third probability of being chosen. While walking, the searcher turns around when the collision distance is reached.

B. Approaching the Target

As soon as a searcher detects the target, it approaches until a desired distance from the target is reached. Then it stops moving. It sets its gradient value to 1 and broadcasts it indicating that the target has been found.

C. Orbiting a Fellow

To enable an effective search for the target, a searcher compares its gradient value with that of its neighbor. Two scenarios apply: In the first scenario, a searcher that has the initial gradient value orbits its neighbor if the neighbor has a gradient value that is different from 0. This helps the searcher to stay in touch with another searcher.

In scenario 2, a searcher that has an initial gradient value that is different from 0 orbits its neighbor if the neighbor has a lower gradient value (i.e. closer to the target) than it does. The searcher adopts a new gradient value by incrementing its neighbor's gradient value by one.

V. Experiments

The algorithms were first implemented in the programming language Lua and tested in the simulator V-REP. For the actual implementation, the algorithms were then translated in the programming language C.

Three experiments were tested in total, while two of the experiments were tested in the simulator using a different desired distance for reaching. In Simulation I, the desired distance is 4 cm, whereas 6 cm is the desired distance in Simulation II and the actual implementation. This can be used to gain a better understanding of the effect of the desired distance on the target to be found.

A. Settings of the Experiments

A set of ten Kilobots are used for both the actual implementation and the simulations where one of the Kilobots has the target role and the nine other Kilobots are the searchers. The target is stationary. It sends messages to its environment, waiting to be found and surrounded by the searchers. A searcher is looking for the target. While searching, the searcher communicates with its environment by sending and listening to messages. It then reacts (i.e. either searches, orbits its fellow or approaches the target) according to the message it receives.

In the starting position, the searchers are placed around the target in a circle. The distance between the target and the searchers ranges from 21 to 23 cm, far beyond the 10-cm sensing range of a Kilobot. All searchers face towards the target (i.e. charging tab faces to the target). The distance between a searcher and its neighbor is between 13 and 15 cm. The same overhead controller collectively controls the Kilobots.

Table I shows the configurations used for the actual implementation and simulations. During a test run, a Kilobot might show different colors indicating its state as described in Table II.

In the actual implementation, the target and the searchers are placed on a desk with a smooth, flat, glossy surface. The overhead controller hangs 45 cm above the Kilobots.

In both simulations, the default time step of 50 ms is used. Each test run stops automatically after it has been run for one minute.

The following indicators are of interest when discussing the results of the actual implementation and simulation, because they reflect the results under the same conditions.

- Number of searchers close to the target when a test run ends.
- Number of searchers that reach the target when a test run ends.
- Time needed to find the target and to reach it for the first time.
- Time needed to find the target and for the target to be reached by a further searcher.
- Dispersal of the searchers during search.

B. Calibration of the Kilobots for the Actual Implementation

According to the documentation of the Kilobot developers [19], there are manufacturing differences. Therefore, the Kilobots need to be calibrated to achieve good forward and rotary motions. Table III depicts the calibration value of the searchers.

TABLE I. Configurations of the Experiment

Configuration	Value
Desired distance for reaching in Simulation I	4 cm
Desired distance for reaching in Simulation II and actual implementation	6 cm
Collision distance	5 cm
Frequency of sending message to environment	Twice every second
Duration of a test run	1 minute
Number of test runs	10 test runs

TABLE II. Description of the Color

Color	State
Magenta	Target blinking in magenta
Green	Target found
Yellow	Approaching target
Blue	Orbiting fellow
White	Other searchers detected, keep searching
Red	Distance to neighbor too close, avoid collision
Cyan	No incoming message received

TABLE III. Calibration Value of the Searchers

Searcher No.	Left	Right	Straight (Left/Right)
1	65	65	60/60
2	66	68	60/62
3	58	58	51/51
4	65	64	59/61
5	61	66	55/57
6	66	63	59/56
7	55	65	50/58
8	58	56	48/49
9	61	54	54/48

VI. Results

A. Actual Implementation

The results of the actual implementation were recorded using a smartphone camera. Table IV depicts the results.

Fig. 1 illustrates the test runs and highlights the searchers that were relatively close to the target in the respective test run. The color used for highlighting corresponds to the ones described in Table II.

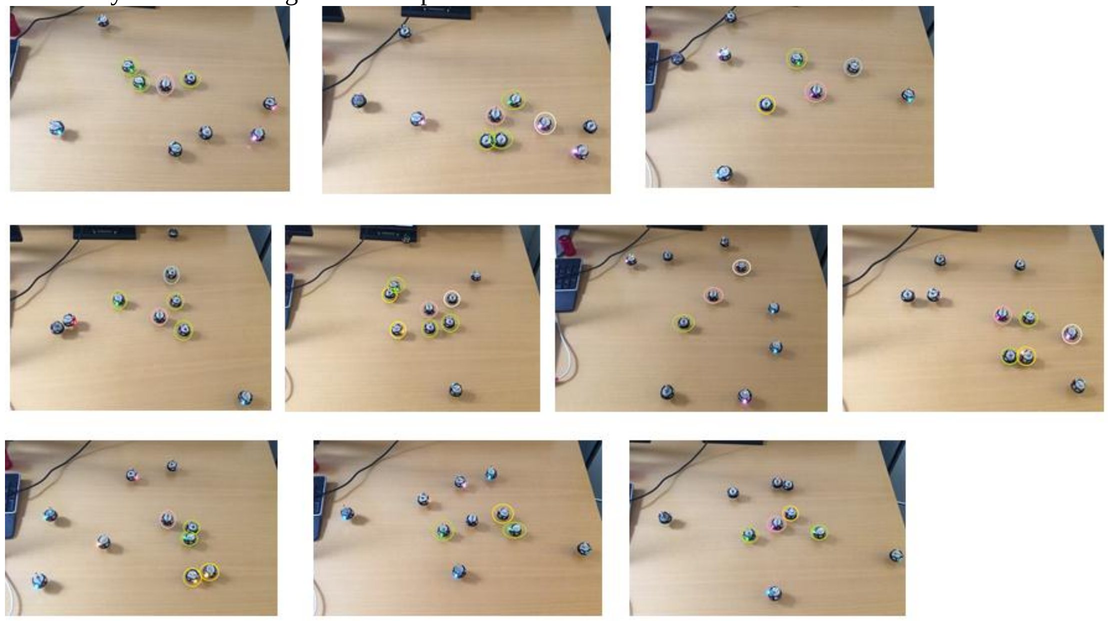

Figure 1. Results of the actual implementation

Overall, there was an average of three searchers close to the target at the end of a test run. In all test runs, the target was successfully found by at least one searcher and on average by two searchers. In four test runs, three searchers successfully reached a target. In the test runs No. 3 and No. 6, only one searcher reached the target.

Across all test runs, the time required to find and reach the target for the first time, ranged from 16 seconds to 58 seconds, whereas one half of the test runs needed more than 21 seconds but less than 25 seconds (see Fig. 2).

The time required for the target to be found and reached by a further searcher was between 23 to 52 seconds, whereas in the majority of the test runs at least 48 seconds were required for the target to be found and reached by a further searcher.

TABLE IV. NUMBER OF SEARCHERS CLOSE TO THE TARGET IN THE ACTUAL IMPLEMENTATION

Test run	Number of searchers close to the target				
	Green: target found	*Yellow: approa-ching target*	*White: other searchers detected, still searching*	*Cyan: no message received, keep searching*	*Total*
1	3				3
2	3		1		4
3	1	1		1	3
4	3			1	4
5	3	2	1		6
6	1				1
7	2	1			3
8	2				2
9	2				2
10	2	1			3
Aver age	2,2				3,1
Med ian	2				3

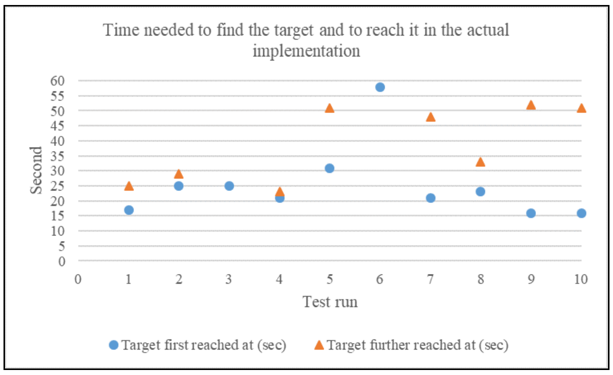

Figure 1. Time needed to find the target and to reach it in the actual implementation

The time difference between the first successful reaching and the second successful reaching seemed arbitrary.

In all test runs, the searchers detected messages from other searchers right at the beginning. In most test runs, they spread across the area to find the target in the course of the search.

B. Simulation in V-REP

The results of both simulations were recorded using the integrated recording function of V-REP.

Simulation I: Table V shows the number of searchers close to the target of Simulation I. Fig. 3 illustrates the test runs.

In Simulation I with the desired distance of 4 cm, at least one searcher could reach the target across all test runs, whereas an average of 3.3 searchers reached the target. In two test runs, a searcher was tipped over due to a collision with others. In fact, while running the test, it could be observed several times that some searchers came closer to others and that searchers that were approaching the target pushed away the searchers that had reached the target successfully.

Regarding the time required for the target to be found and reached for the first time (see Fig. 5), the median time required was on average 34.5 seconds respectively 35.8 seconds. After the target was found, a second searcher would reach the target between 3 to 12 seconds.

As to the spreading of the searchers, in all test runs, the searchers first walked towards the target. The spreading behavior emerged after the test had been run for 10 seconds. The interaction with other searchers followed soon. The change of color (i.e. state) of a searcher could be observed well as soon as the searchers came closer to each other and to the target.

Compared to other test runs, many searchers in the test run No. 3 were located far away from the target at the end of the test run. However, the video of the test run No. 3 does not show that the searchers had spread differently. It seems that the searchers in the test run No. 3 failed to find the target by chance.

Simulation II: Table VI shows the number of searchers close to the target of Simulation I. Fig. 4 illustrates the test runs.

An average of five searchers were close to the target. Across all test runs, the target was found and reached by at least two searchers. In seven out of ten test runs, at least half of the searchers found and reached the target. In test run No. 5, a searcher was tipped over due to a collision with another searcher.

A closer look at the cyan colored searchers revealed that these searchers were actually orbiting their fellow or avoiding collision with their fellow. Because of the asynchronous communication, the message from the fellow was not received because the simulation stopped.

Concerning the time required for the target to be found and reached for the first time (see Fig. 6), the data conspicuously concentrated on the time span between 27 and 31 seconds. To be found and reached by a further searcher, the required time span ranged between 29 to 42 seconds. In six out of ten test runs, the time difference between the first and the second finding of the target was less than five seconds.

TABLE V. NUMBER OF SEARCHERS CLOSE TO THE TARGET IN SIMULATION I

Test run	Number of Searchers close to the Target				
	Green	*Yellow*	*White*	*Cyan*	*Total*
1	3			2	5
2	4	1		1	6
3	1				2
4	5				5
5	3			2	5
6	4			2	6
7	4			1	5
8	4			1	5
9	3			1	4
10	2			1	3
Average	3,3				4,6
Median	3,5				5

TABLE VI. NUMBER OF SEARCHERS CLOSE TO THE TARGET IN SIMULATION II

Test run	Number of Searchers close to the Target				
	Green	*Yellow*	*White*	*Cyan*	*Total*
1	2			1	3
2	6			1	7
3	5				5
4	3				3
5	5			1	6
6	7				7
7	5				5
8	3				3
9	5				5
10	5			1	6
Average	4,6				5
Median	5				4,5

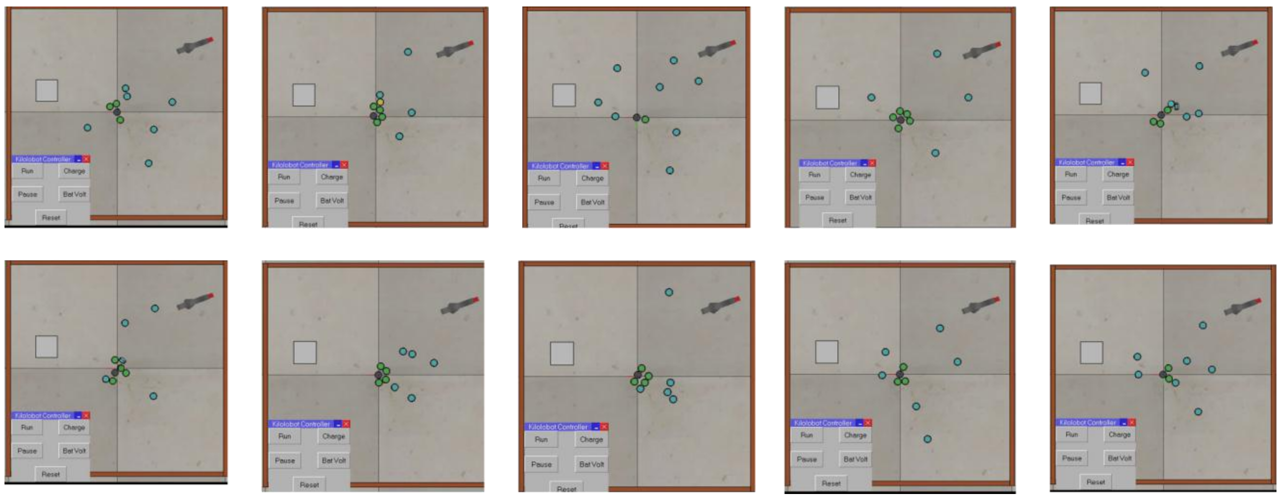

Figure 3. Results of the Simulation I

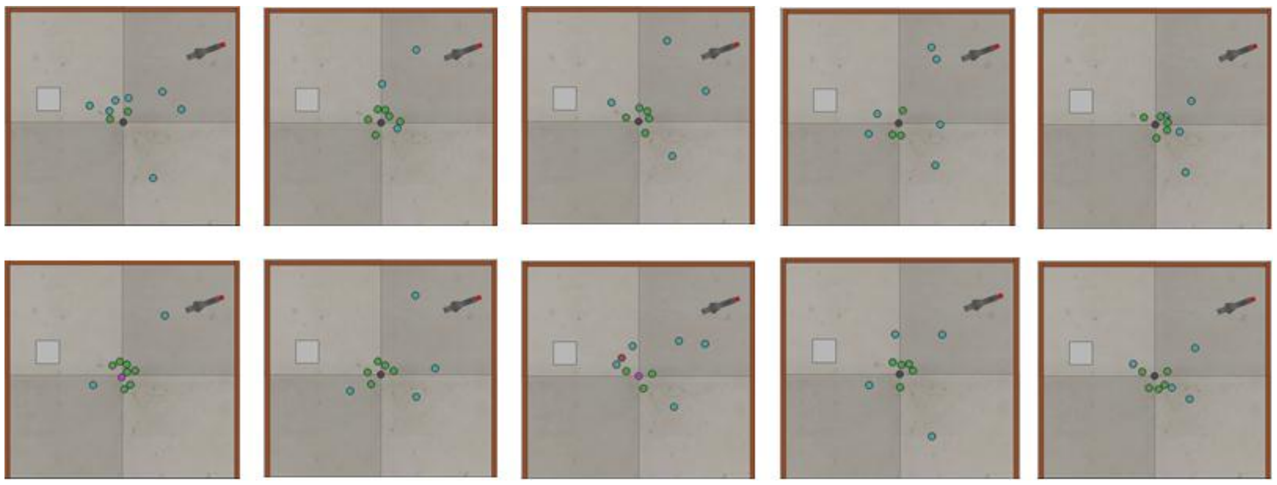

Figure 4. Results of the Simulation II

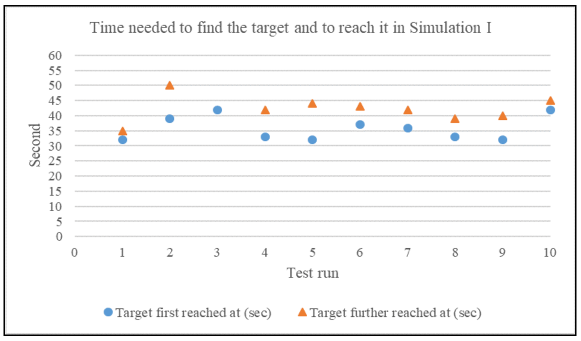

Figure 5. Time required to find the target and to reach it in Simulation I

Like in Simulation I, the searchers in Simulation II first walked towards the target, showed spreading behavior and acted according to the interaction with other searchers after roughly 10 seconds. The change of state was also well observed in Simulation II.

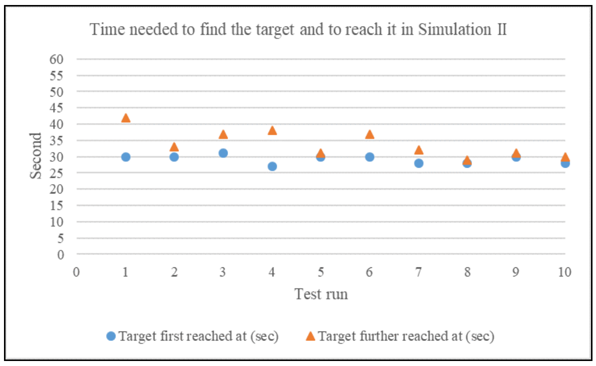

Figure 6. Time required to find the target and to reach it in Simulation II

VII. Discussion and Analysis

In this section, we compare and discuss the result of the actual implementation of the Kilobots and their simulation on the computer. Additionally, we report observations we made during the implementation and simulation.

A. *Results of the Actual Implementation of the Robots and of the Simulations*

The dispersal strategy and the communication with other searchers were crucial to find the target in terms of the number of searchers that reached the target and of the time required to reach it. While searching the target, it seems that the searchers in the actual implementation were more willing to explore the surrounding area. In most test runs, the searchers spread right at the beginning. This dispersal behavior favors the finding of the target at the beginning. This might explain why the time required to find and reach the target for the first time in the actual implementation was less than in both simulations.

However, the searchers in the simulations caught up quickly by orbiting fellows that had already found the target. This might explain why the time needed to find and reach the target by a further searcher was shorter in Simulation II than in the actual implementation given the same desired distance.

Noticeably, the orbiting fellow strategy was not observed in the actual implementation. Indeed, the searchers approached the target on their own (i.e. the LED was yellow). The absence of the orbiting strategy clearly made the finding of the target less efficient.

The inefficient search lead to a higher probability that the searchers would spread further than actually needed and finally lose connection to the other searchers. This phenomenon was more prevalent in the actual implementation where more searchers were located far away from the target when the test run ended.

Explanations for the absence of the orbiting strategy in the actual implementation might be due to the dispersal of the searchers and the fact that there were less searchers that found the target and that many searchers still held the gradient value of 0 during the entire test run, which did not trigger the orbiting behavior.

An important observation made in the actual robot implementation and simulation, is that a Kilobot can receive a message from only a single source at the same time. A preference of the message source could not be pre-defined. This might explain that the searcher approached the target on its own instead of orbiting.

The search behavior had implications on the success rate. Overall, there was a higher probability to find the target in the simulations than in the actual implementation of the robots. Moreover, the increase of the desired distance from 4 cm to 6 cm in the simulations clearly had an effect on the success rate of the finding, but not on the searchers that were close to the target at the end of a test run. Interestingly, the number of targets found and reached in Simulation I (desired distance: 4 cm) outperformed the actual implementation (desired distance: 6 cm). This outperformance again stresses the discrepancy of the performance in the simulation and in the real world. Interestingly, the results of Simulation I with a desired distance of 4 cm were closer to the results of the actual implementation where the desired distance was 6 cm.

Comparing the time required for a further searcher to reach the target, the searchers in Simulation I needed more time than the searchers in Simulation II. This difference could be explained by the fact that the desired distance was smaller in Simulation I. A further searcher in the actual implementation needed more time than in Simulation II, but less than in Simulation I.

In addition, the time required to find and reach the target for the first time was less arbitrary in the simulation than in the actual implementation. Compared to the actual implementation, the variances across all test runs were smaller in the simulation (see Fig. 2, Fig. 5 and Fig. 6). While in the actual implementation a target that was found and reached by two searchers almost simultaneously was only observed once in test run No. 4, whereas the same phenomenon could be observed four times in Simulation II.

Certainly, there were searchers in the simulations that lost connection to the cohort. However, in comparison with the actual implementation the number of searchers that were lost was smaller. This was also reflected by the number of searchers that were close to the target at the end of the test run. To tackle the problem of lost searchers, the communication architecture proposed by Jha et al. might be used [20].

B. *Locomotion of the Kilobots*

While preparing the actual implementation, we noticed that the calibration value of the Kilobots used for the actual implementation revealed large manufacturing differences. According to the developers of the Kilobot [19], values between 60 and 75 achieve the best rotation. In our case, four Kilobots exceeded the lower limit of this recommendation.

The goal of the calibration is to allow the Kilobot smooth moves on a given surface. The calibration, however, does not guarantee a locomotion at the same speed. In a preliminary test, we let the Kilobots turn right, left and walk straight. Our observation indicated that the Kilobots performed at a different speed with similar calibration values. For instance, Kilobot No. 7 finished as many left-turning rounds as Kilobot No. 5 even though the power level for turning left of Kilobot No. 5 is higher than that of Kilobot No. 7 (61 vs. 55). On the other hand, given the same duration of time, Kilobots No. 8 and No. 4 finished the same numbers of left-turning rounds, but both were slower than Kilobot No. 5 and No. 7 in turning left.

The simulation software V-Rep does not consider manufacturing differences. Neither could we calibrate the Kilobots with the simulation software, nor did they perform individually.

The difference in speed has implications on the actual implementation. For example, while two Kilobots perform the same action such as to turn left and then go straight, one of the Kilobots might make a larger rotation than the other one and crash into it, thus adding some degree of uncertainty to the operation. Fine-tuning the calibration of the Kilobots to the same speed might add stability, however.

Furthermore, the time interval for receiving messages is roughly twice every second. This is good for determining the course of action but might be less optimal for the motion. We observed a strange behavior (e.g. side slipping) when the Kilobot performed acute rotation.

C. *Algorithm*

The collision avoidance worked better in the actual implementation than in the simulations. In the implementation, we observed clearly that a Kilobot succeeded to avoiding collision with another Kilobot in the majority of test runs. This was less visible in the simulation. Instead, the Kilobots were still standing next to each other shortly after they had avoided a collision.

In addition, it happened often in the simulation that three Kilobots clashed. In this case, the current algorithm for collision avoidance worked less effectively. This might be explained by the fact that, as mentioned before, a Kilobot receives the message arbitrarily and that in the current algorithms a Kilobot had no memory. It reacted according to the incoming message. In the case of multiple Kilobots, this configuration proved unfavorable for the algorithm responsible collision avoidance. For instance, Kilobot A receives a message from Kilobot B and decides to walk away from it. Half a second later, Kilobot A receives a message from Kilobot C and decides to walk away from it as well. By doing so, Kilobot A might reject the previous collision avoidance of Kilobot B and clash with it.

Another illustrative example is the algorithm for approaching the target. Currently, while approaching the target, a Kilobot does not perform any collision avoidance and therefore might run into another searcher as shown in the simulation in which the searcher while approaching the target sometimes pushed other searchers away. Due to this communication characteristic, incorporating avoidance collision in approaching a target would be very ineffective because the searcher has no information about other Kilobots standing between it and the target. In fact, it prevents the searcher to come close to the target. To solve this problem, we propose a memory-based approach combined with synchronization and localization of nearby Kilobots.

VIII. Conclusion and Outlook

The presented algorithm based on dispersal and orbiting enabled a successful search- and target-surrounding strategy. While the dispersal strategy facilitated the findings in terms of time, the orbiting strategy increased the number of searchers that found and reached the target. The interplay between the dispersal and the orbiting affects the entire operation.

The comparison of the results shows differences in the number of searchers that found the target and in the time required to reach the target. Overall, there were more searchers reaching the target in the simulation than in the actual implementation, while the time required to reach the target was faster than in both simulations in the actual implementation. The increase of the desired distance affected the number of searchers that reached the target in the simulations. Apart from that, no significant difference could be observed within both simulations.

Moreover, the proper calibration of the motor of the Kilobots and the time interval for changing the motion are important for a smooth locomotion. Further research can direct its effort on a memory-based approach combined with synchronization and localization of nearby Kilobots and on the fine-tuning of the interplay between the dispersal and the orbiting. These would make the search and the collision avoidance more effective.

REFERENCES

[1] V. J. Zhong, R. R. Umamaheshwarappa, D. Rolf, and H. Thomas, “Comparison of a real Kilobot robot implementation with its computer simulation focussing on target-searching algorithms,” presented at the 2018 International Conference on Intelligent Autonomous Systems, Singapore, 2018.

[2] M. Rubenstein, C. Ahler, and R. Nagpal, “Kilobot: A low cost scalable robot system for collective behaviors,” in *Proc.-IEEE Int. Conf. Robot. Autom*, pp. 3293–3298, 2012.

[3] M. Rubenstein, A. Cornejo, and R. Nagpal, “Programmable self-assembly in a thousand-robot swarm,” *Science*, vol. 345, no. 6198, pp. 795–799, Aug. 2014.

[4] E. R. Magsino, F. A. V Beltran, H. A. P. Cruzat, and G. N. M. De Sagun, “Simulation of search-and-rescue and target surrounding algorithm techniques using kilobots,” pp. 70–74, 2016.

[5] E. Șahin and A. Winfieldq, “Special issue on swarm robotics,” *Swarm Intell.*, vol. 2, no. 2, pp. 69–72, 2008.

[6] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, *Swarm Robotics: A Review from the Swarm Engineering Perspective*. 2013.

[7] R. Vaughan, “Massively multi-robot simulation in stage,” *Swarm Intell.*, vol. 2, no. 2–4, pp. 189–208, Dec. 2008.

[8] J. Craighead, R. Murphy, J. Burke, and B. Goldiez, “A Survey of commercial open source unmanned vehicle simulators,” in *Proc. 2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 852–857.

[9] P. Castillo-Pizarro, T. V. Arredondo, and M. Torres-Torriti, “Introductory survey to open-source mobile robot simulation software,” in *Proc. 2010 Latin American Robotics Symposium and Intelligent Robotics Meeting*, 2010, pp. 150–155.

[10] S. Ivaldi, J. Peters, V. Padois, and F. Nori, “Tools for simulating humanoid robot dynamics: A survey based on user feedback,” in *Proc. 2014 IEEE-RAS International Conference on Humanoid Robots*, 2014, pp. 842–849.

[11] D. Cook, A. Vardy, and R. Lewis, “A survey of AUV and robot simulators for multi-vehicle operations,” in *2014 IEEE/OES Autonomous Underwater Vehicles (AUV)*, 2014, pp. 1–8.

[12] G. Biggs and B. Macdonald, “A survey of robot programming systems,” in *in Proc. the Australasian Conference on Robotics and Automation, CSIRO*, 2003, p. 27.

[13] B. Balaguer, S. Carpin, and S. Balakirsky, “Towards quantitative comparisons of robot algorithms: Experiences with SLAM in simulation and real world systems,” in *ResearchGate*, San Diego, 2007.

[14] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper, “USARSim: A robot simulator for research and education,” in *Proc. 2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 1400–1405.

[15] M. Kudelski, L. M. Gambardella, and G. A. Di Caro, “RoboNetSim: An integrated framework for multi-robot and network simulation,” *Robot. Auton. Syst.*, vol. 61, no. 5, pp. 483–496, May 2013.

[16] Self-Organizing Systems Research Group, *kilobotics-labs/disperse.c*. 2016.

[17] Self-Organizing Systems Research Group, *kilobotics-labs/orbit_star.c*. 2016.

[18] Self-Organizing Systems Research Group, *kilobotics-labs/orbit_planet.c*. 2016.

[19] SSR Lab, Harvard University, “Calibration,” *Documentation*. [Online]. Available: https://www.kilobotics.com/documentation. [Accessed: 05-Feb-2018].

[20] A. Jha, K. Gupta, and M. Sen, “M2M communication system for networked robots with low memory footprint,” in *2014 International Conference on Information Technology Systems and Innovation (ICITSI)*, 2014, pp. 310–316.

Vivienne Jia Zhong completed her master’s degree in business information systems at the University of Applied Sciences and Arts Northwestern Switzerland (FHNW) in 2018. She has been working as a research assistant at the Institute for Information Systems of the FHNW for three years. Her research fields include robotics, computational intelligence, human-machine interaction and business information systems.

Prof. Dr. Rolf Dornberger holds a PhD and a Diploma in Air- and Aerospace Engineering. He is full professor and lecturer at the University of Applied Sciences and Arts Northwestern Switzerland FHNW (since 2002). He is the head of the Institute for Information Systems at the School of Business FHNW (since 2007). Before becoming a professor, Prof. Dr. Dornberger worked in industry in different management positions as a consultant, IT officer and senior researcher in different engineering, technology, and IT companies in the field of energy, software, IT, and airline business. His current research interests include artificial intelligence, particularly computational intelligence and optimization, robotics, human-machine interaction, the Internet of things as well as innovation management and learning didactics.

Prof. Dr. Thomas Hanne received his master’s degrees in Economics and Computer Science, and his PhD in Economics. From 1999 to 2007, he worked at the Fraunhofer Institute for Industrial Mathematics (ITWM) as senior scientist. Since then he has been a professor for Information Systems at the University of Applied Sciences and Arts Northwestern Switzerland. Since 2012, he has been the head of the Competence Center Systems Engineering. Prof. Dr. Hanne is the author of more than 100 journal articles, conference papers, and other publications and editor of several journals and special issues. His current research interests include multicriteria decision analysis, evolutionary algorithms, metaheuristics, optimization, simulation, systems engineering, software development, logistics, and supply chain management.