Uniform Dispersal of Cheap Flying Robots in the Presence of Obstacles

Attila Hideg[1], László Blázovics[1], Tamás Lukovszki[3], Bertalan Forstner[1]

[1] Department of Automation and Applied Informatics,
Budapest University of Technology and Economics
Magyar tudósok krt. 2, H-1117 Budapest, Hungary
e-mail: attila.hideg@aut.bme.hu, laszlo.blazovics@aut.bme.hu,
bertalan.forstner@aut.bme.hu

[2] Faculty of Informatics, Eötvös Loránd University
Pázmány P. sétány 1/c, H-1117 Budapest, Hungary
e-mail: lukovszki@inf.elte.hu

Abstract: In previous solutions, the authors considered the uniform dispersal problem (or Filling problem) in which inexpensive robots had to disperse in order to cover an a priori not known area, as well as they also examined the possibilities of solving the Filling in two-dimensional regions. The swarm entities had to collectively solve a common task using the simplified cognitive abilities of the robots (i.e., their memory, visibility, and communication capabilities were restricted). In this paper, the authors investigate the possibilities to apply the method for three-dimensional regions. The need for such a solution emerged, as nowadays the number of low-priced flying robots, e.g., quadcopters, drones, has increased heavily. The main research direction is to minimize the hardware requirements of these robots, as doing so is crucial in order to maintain their cost-efficiency. The authors demonstrate that it is still possible to solve the Filling problem in three-dimensional space in the presence of obstacles, while the robots maintain the following hardware requirements: they have a constant amount of memory, minimal visibility, as well as there is no communication between them, and the algorithm terminates in linear runtime. Finally, simulations were carried out to prove the theoretical results.

Keywords: uniform dispersal; distributed robotics; autonomous robots

1 Introduction

Swarm robotics differs from single robot systems in many aspects; in the first one, numerous cheap and simply constructed robots perform sophisticated tasks together, while in the latter, the main focus is on the fact that single robot systems are less scalable, reliable, and fault-tolerant than multi-robot systems. Recently

published studies have focused on the behavioral, so to say cooperative, properties of simple and small robots that have the task of solving incurring problems together, as one team.

Starting from the publication of Reynolds [1], many researchers have started to investigate the issue of tasks requiring the positioning of individuals. In these works, robots did not possess a central control or coordination, as well as they, performed the same distributed algorithm for achieving flocking (collective movement of a swarm). Additionally, the algorithm provided was less complex and only relied on the local perception of the robots.

As for the concept of flocking, there have been other studies published, focusing on the problems of exploration (fully discovering an unknown area), gathering (meeting at given points in the area), pattern formation (the swarm achieving a previously defined shape), coverage (the way robots cover an area), or dispersing (reaching the state of coverage from certain starting points). These problems are included in the publication [2, 3, 4, 5, 6, 7, 8, 9, 10]; see [11, 12] for recent surveys.

In the recent past, largely the public has also been allowed to create big flying swarms by the use of small and inexpensive robots. In the case of a robot being equipped with a measuring sensor, it's possible to monitor specific values of its environment. When cleaning up toxic waste, reducing pollution, or when there is a potential risk of radiation, being present in the area itself might be of high risk for humans. This is where these drones come in handy since they could signal to the human party when a certain threshold of a dangerous substance has been reached.

The authors have already investigated the issue of dispersion when the goal is to evenly distribute robots in the area [13, 14, 15, 16]. A unique case of dispersion like this is the Filling, which was first introduced by Hsiang et al. [9], where the robots enter individually through an entry point (Door), and have to disperse from that point, covering the area subdivided into smaller cells. The Filling becomes completed when all cells contain a robot, and none of the robots collided during the dispersion process. Another requirement is that each cell must contain no more than one robot at the same time.

After they have dispersed, robots are able to measure different environmental factors with the help of built-in sensors. These measurements have to be immediately sent to the network (of robots) by the use of network coding [17], since – given their inexpensive nature – the continuous operation, communication, and measurement-making is not expected from these robots. This is similar to the sensor-bridging communication presented in [18] since the artificial swarm of robots explores the environment from a broader and more complex perceptive than the human cognitive system – for example as opposed to a human being, a robotic system is able to process a wide area at once.

The interdisciplinary area of cognitive infocommunication (CogInfoCom) investigates the connection between the areas of research of infocommunications and cognitive sciences [19]. As such, this area could be used to support any area where the central question is how artificial or natural cognitive systems could be able to work together more effectively [20]. The present paper examines the possible ways of cooperation in the Socio-Cognitive ICT, which is a subfield of CogInfoCom [19].

A simulator has already been built in [16], where a simulation environment was designed in order to examine the interaction between humans and computers. An example of this can be seen in [21] where cooperation was investigated in the VR environment. Many other studies show that these environments have a high impact on the field of CogInfoCom [22, 23], as well.

Another advantage of the simulation is its perspective in the CogInfoCom based education [24, 25, 26]. While observing and interacting with the simulated robots, the human party will have a much more immersive experience, as one of the primary goals of CogInfoCom is to create systems based on human perception, develop or return the cognitive ability of understanding and cognition to the user, all through models based on ICT engineering tools [27].

The robots evolve and learn, as well, which process is highly similar to the human learning process. Therefore, if we want to develop a robotic system, we could not avoid examining the human way of learning since we would mimic this process.

As it was previously mentioned, exploring and dispersing areas that are unsafe for humans, such as extinguishing a building on fire, the use of computer-operated robots could offer a perfect solution, despite being somewhat expensive, to saving human lives by using these robots worth way more than their price. What is more, simulating, analyzing, and testing these scenarios is also very supportive of several fields of study.

In this paper, an algorithm where robots are substituted by flying drones capable of areal movement (i.e., flying and floating) is shown and further analyzed, as opposed to the solutions proposed previously for Filing, where the robots acted in a 2D plane.

The present paper investigates the use of the Virtual Chain Method (VCM) presented in [15]. The new contribution proposed is the use of algorithms for flying drones in 3D areas. When describing and analyzing the algorithms in question, terms, and arguments introduced in [15] are used or repeated.

2 Model

The robots work in a three-dimensional area, which is connected yet unknown for the robots. The whole area is built up from small regions of space, which are virtually subdivided into smaller regions (see Figure 1), allowing no more than one robot to occupy them at any given time. These spaces can be interpreted as vertices of a graph, with having the edges between neighboring spaces. Drones are equipped with built-in compass, allowing them to be able to differentiate between the orientation of North, South, East, and West neighbor spaces horizontally, and Upper or Lower neighbors vertically. The vertices where they enter the graph are called Doors, from where drones can only move to Upper neighbors and continue from those points.

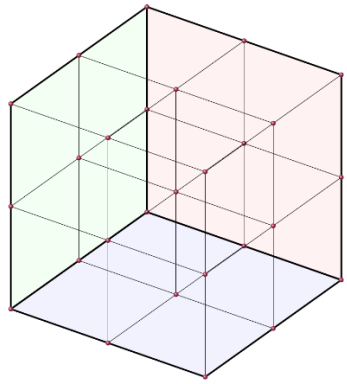

Figure 1
Three-dimensional area decomposed into small regions

Each of the drones is able to get information from its environment by using its sensors. The robots are also equipped with a computational unit, as well as have aerial locomotion capabilities, such as flying and floating. They are *autonomous* (do not have central coordination), *homogeneous* (each has the same capabilities and behaviors), and *anonymous* (they could not identify one another).

Furthermore, they have a sensor for measuring the distance between themselves and nearby obstacles. Obstacle detection can be achieved by attaching IR or ultrasonic distance sensors or any other rangefinders into 6 directions; 4 horizontal in every 90° angles, one to the top, and one to the bottom. The sensors have to tell whether the robots can move in the given direction or not since in case of an obstacle – such as another drone, a tree, or a wall – they will hit when they move in that direction.

They also have limited persistent memory of $O(1)$ bits in the single Door case and $O(\log k)$ bits when k-Doors are present.

The drones act according to the general model of Look-Compute-Move (LCM), where the drones' actions are decomposed into three phases (Look, Compute, and Move phase). In the Look phase, the drones take a snapshot of their environment, in the Compute phase, they perform calculations in order to determine whether they should float in their position or fly to one of their neighboring space, and finally, during the Move phase, they implement the decision they have made in the previous phase. Their movement is atomic, i.e., it is either entirely performed and the robot appears at the destination, or not performed, meaning that the robot does not move at all.

As for the timing of an LCM cycle, the fully-synchronous (FSYNC) model is used, where all drones perform their LCM cycles simultaneously, i.e., each of them takes a snapshot of the environment, computes, as well as performs their movement at the exact same time.

The drones are placed in previously defined positions (Doors), from where they can only move to their Upper neighbor and continue dispersing. At the start of each LCM cycle, a new drone is placed there and performs its first Look-Compute-Move phases during the same cycle in case the Door is empty.

3 Areal Virtual Chain Method

In our previous paper, we described the Virtual Chain Method (VCM). This method was able to solve the problem of Filling by using the general leader-follower method [5, 6, 9, 13].

The Areal Virtual Chain Method (A-VCM) extends this latter method for robots having flying capabilities (e.g., drones or quadcopters).

3.1 Concept

One drone becomes the *Leader* and moves to unexplored areas in the given space, while the others become *Followers*, following the Leader by forming a chain by the way they move. When the Leader reaches a point where it can not move anymore, it switches to *Finished* state, and its immediate Follower becomes the new Leader which will start moving to other parts of the area they are in. These states, as well as the transitions between them, could be seen in Figure 2. In their initial state, drones can either become Leaders or Followers depending on whether, at that time, they have a neighbor or not.

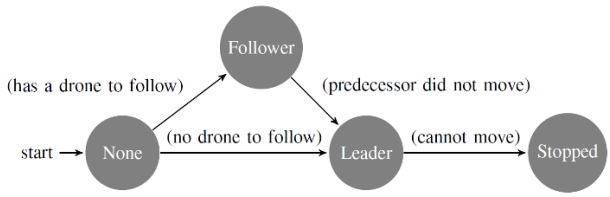

Figure 2

States and state-transitions of the drones. The edges are labeled with the condition for the transition.

3.2 Method

In the algorithm, the following tasks are performed by the drones in their respective states:

- *None*: the starting state, right after the drone has been placed at the Door.
- *Leader*: the first drone placed at the Door switches to Leader state. Only this one moves to previously unoccupied, so-called *unvisited*, vertices. We ensure that there can be no more than one Leader at any given time.
- *Follower*: drones following their *predecessor* (the previously placed drone) are in the Follower state. A Follower can only become a Leader when its predecessor switches to the Finished state.
- *Finished*: this is the final state of the drones, in which state they switch to once they detect they cannot move anymore. A Finished drone can never move anymore; it only floats in its current position. Only the leader is able to switch to Finished state.

3.3 Round Structure

Similar to the VCM, the algorithm also functions according to rounds. A *round* is a sequence of 6 consecutive *steps* where a step means one LCM cycle of the drones. Steps are called by directions of North (N), East (E), South (S), West (W), and Up (U) and Down (D). The rounds and steps are illustrated in Figure 3. During each round, a drone is either an *observer* or an *observed* one.

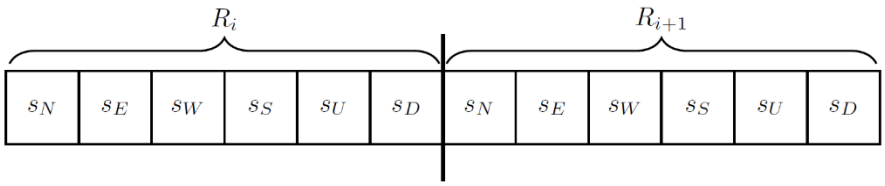

Figure 3

The round structure. Two rounds (denoted by R_i and R_{i+1}) consisting of 6 steps, labeled with the corresponding direction (sN,…).

The observed one has to schedule its movement and is able to move to one of its neighbors in the direction, which is the label of the current step. At the same time, the observer drone counts the steps its predecessor has waited before moving. At the end of each of the rounds, drones switch roles (i.e., the observer becomes the observed and vice versa), however, each drone starts its actions as an observer when placed at the Door. What is more, in order to detect unvisited vertices, they also retain information about the occupancy of neighboring vertices.

Each drone is able to move every second round (in the round their state is observed). After the drones have moved, they do not move again until their next observed round. They are not allowed to backtrack, i.e., move to their previous position. The chain is defined by the current Leader drone's path from the Door, and Followers follow that path, while other drones in the area are in the Finished state.

3.4 Analysis

When analyzing the A-VCM, the area is represented by a connected graph. It is possible to repeat the arguments of the Virtual Chain Method analysis during these Lemmas. It is possible due to the generality of the VCM, meaning that apart from being connected, it did not require any particular attribute of the graph; thus, the Lemmas still hold in three-dimensional areas.

When analyzing the A-VCM, it is essential to prove that two constraints are satisfied during the whole dispersion: *i)* collisions are not possible, *ii)* there is no vertex of the graph which remains unoccupied after the termination of the algorithm.

Lemma 1. Collisions are not possible.

Lemma 2. A drone can determine if a neighboring region is unvisited by observing it for two consecutive rounds.

Lemma 3. The predecessor of the Follower is either in a neighboring vertex *v*, or it was in *v* in the previous round. In the latter case, the Follower moves to the previous position of the predecessor, which is *v*.

Lemma 4. Each robot in the Follower state always knows where its predecessor is.

Lemma 5. Two Followers cannot have the same predecessor.

Lemma 6. The Leader only moves to unvisited vertices.

Lemma 7. There can be at most one Leader at a time.

Lemma 8. Algorithm A-VCM fills the area (represented by the graph).

Theorem 1. By algorithm A-VCM, a three-dimensional area with a single entry point, is filled in *O(n)* rounds without collisions by drones with a visibility range of 1 hop and *O(1)* bits of persistent memory, if they are equipped with a compass.

Proof: The A-VCM fills the area (Lemma 8) without collisions (Lemma 1) if the area is represented by a graph. After placing the drone at the entry point, it is in None state. In the next round, it observes its predecessor moving, and then it moves in the third round. In the same round, the next drone will be placed at the Door. For this reason, the drones are placed at the Door in every third round; as each round consists of 6 steps, it takes $3 \cdot 6 \cdot n = O(n)$ steps to place n drones.

Regarding the memory requirement, the drones require to store: the index of the current step within a round, the unvisited neighbors, the direction of their predecessor (each requiring at most 6 bits of memory), and some additional information, requiring a constant amount of bits: current state, observer/observed role, entry vertex. As a result, *O(1)* bits of persistent memory is required for the Areal Virtual Chain Method.

3.5 Multiple Doors

In case there are several entry points (Multiple Doors) for a given area, the greatest challenge is to make sure that robots entering through different Doors avoid collisions. Two robots could avoid such a collision by mutually agreeing on which one will go (to the same destination) first, typically by setting a priority order. This order can either be visible externally or is communicated between the robots. To arrive to such an agreement, those robots have to 'see' one another, which means they have to have a visibility range of 2 hops; however, the drones used in the present paper are only equipped with range finders, cannot detect others' priority orders, nor can they communicate with one another. In the Multiple Door Areal Virtual Chain Method (MDA-VCM), robots from each distinct Door will form a distinct chain.

Similar to the MD-VCM, a distinct time-slot is allocated for each Door, in which they are able to execute their actions. Contrary to the single Door case, each step is replaced by k steps; therefore, a Round will be a sequence of $6k$ steps. Each drone entering from D_i only performs their actions in $s_{i,*}$ and stays idle during the other steps.

Theorem 2. A three-dimensional area, represented by a connected graph, having several entry points, is filled by the MDA-VCM in $O(k \cdot n)$ rounds, without collisions by robots having a visibility range of 1 hop and $O(\log k)$ bits of persistent memory.

Proof: Similar to Theorem 1, the drones are placed in each Door every third round, given that the chain from that Door is able to move; otherwise, no further drones could be placed at that Door anymore. The worst-case scenario is when a

single chain blocks all the other Doors – in this case, only one Door is used to cover the area.

As for hardware requirements, the drones do not need additional visibility, nor other equipment either. The memory increases to *O(log k)* as the round's length increases, and the current step index has to be stored.

4 Practical Usage

Formals proof has shown that the A-VCM can be utilized in three-dimensional (and also in n-dimensional) areas. This implies that there are numerous practical scenarios where the algorithm can be applied.

In the last years, we have experienced an increasing interest and rapid development of robotics, 'Internet of Things' (IoT) devices, as well as related platforms. The tremendous amount of wirelessly connected devices suggests that it is not suitable to maintain centralized communication, given that in more and more scenarios, low latency and real-time communication might be crucial (e.g., autonomous vehicle fleets, sensor networks). In these particular systems, along with real-time transmission, the intra-group data exchange is also substantial. Consequently, peer-to-peer architecture is the most suitable.

Nonetheless, peer-to-peer data exchange dramatically relies on the links between neighboring entities, especially the quality of those links. This quality can be measured easily in a flat, 2D setup; however, the same can be done more complicatedly in a 3D setup. Furthermore, performing measurements in a moving swarm of drones or a convoy of vehicles is also a high-complexity task.

By using A-VCM, different multi-dimensional drone setups could be created with the communication links between the nodes that can be measured and monitored.

4.1 Sensor Data Propagation

A final goal is that, in the given area, the dispersed drones have to perform constant measurements and maintain continuous surveillance. A problem could arise from the fact that these drones are built by using cheaper components, therefore they might malfunction, hindering communication between them or making them land by leaving their positions. These drones could be seen as nodes in a network, with local information provided by their sensors whose task is to send those data to the network. A method was described in [28] where the nodes could lose network availability or fail permanently, as well as a dynamic repair mechanism allowed them to maintain integrity between data.

Applying the same method to drones, a network of flying sensors could maintain data propagation even in cases when some of the drones malfunction, or could be

used for various problems where the constant monitoring is required even by leaving out human presence, as this one being too dangerous or impossible, e.g., in the case of dangerous environments.

5 Simulation

A simulation framework based on [16], called the RobotCore, is currently under development which is to simulate complex distributed algorithms. With the simulator, the purpose is to validate the formal proofs made in the previous section.

During the simulations, the focus is on the runtime of the algorithm in different areas. First, the runtime was examined by using randomly created three-dimensional grids with obstacles having different sizes in them. The size means the number of cells in the area. Then, the same areas were examined with multiple entry-points. Finally, a large area with 100 drones and cells were filled using more and more Doors.

The graph generation procedure: for a graph with n vertices take a cuboid-shaped three-dimensional area with approximately $2n$ cells. The cuboid has a width of x, a length of y, and a height of z cells. First, x and y are randomly generated between 1 and the square root of $\sqrt{n}$. Then z is chosen, so $x \cdot y \cdot z$ will be closest to $2n$: $z=2n-x \cdot y$ rounded up. From these $2n$ cells, randomly remove n, so the final area has only n cells. Finally, k Door vertices will be added as Doors (and k cells removed from the cuboid). Note that the removed cells can be considered as obstacles in the area.

5.1 Runtime with single Door

In the single Door case, Theorem 1 has shown an $O(n)$ runtime, meaning that, based on the size of the cells, there is a linear growth in the running time. The simulation results can be seen in Figure 4. The horizontal axis shows the number of vertices in the area from 1-1000; the vertical shows the required number of turns to complete the Filling. For each size, graphs were randomly created with the given vertex count, then it was tested how many rounds were required until each vertex was occupied. As it was expected, the runtime has shown a linear growth.

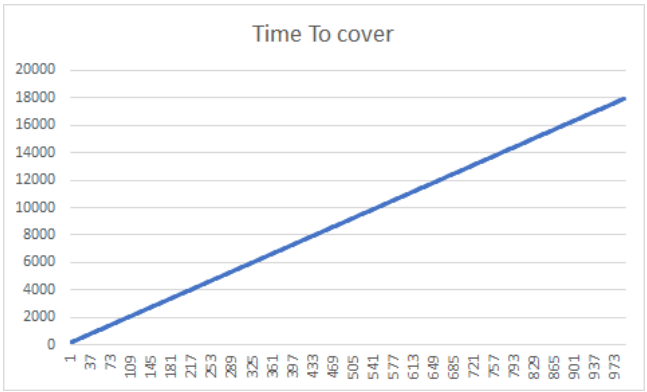

Figure 4

The horizontal axis shows the number of vertices in the area; the vertical shows the required number of turns to complete the Filling.

5.2 Runtime with multiple Doors

In the second test scenario, the claim that multiple Doors can increase the runtime was tested. In some cases, it necessarily improved the runtime, as the robots might have blocked others coming from different Doors. However, in general, the runtime was reduced, which can be seen in Figure 5.

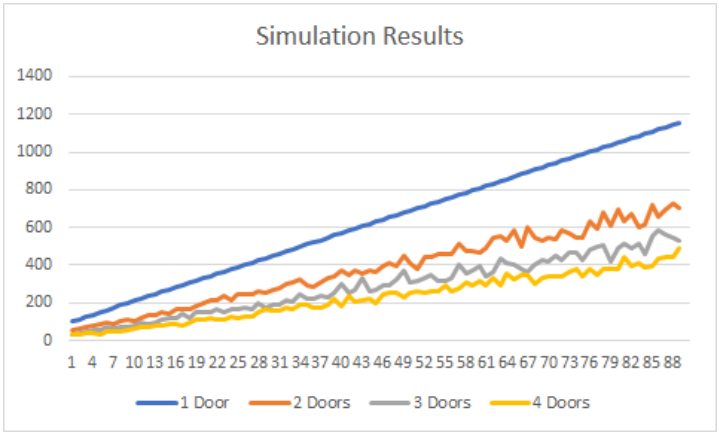

Figure 5

The horizontal axis shows the number of vertices in the area; the vertical shows the required number of turns to complete the Filling

5.3 Runtime reduction test with multiple Doors

When introducing more and more Doors, the runtime can be reduced by a factor of k if the number of Doors in the area is k. During the last test scenario, the authors were curious whether this runtime improvement is achieved or not. The simulation results can be seen in Figure 6, which proves that the runtime is approximately k-times faster with k Doors in the area. In these simulations, an area consisting of 100 vertices was Filled by robots entering through k Doors, where k is from 1 to 100.

Note: this means that more and more vertices become Doors, and in the extreme case of 100, all the vertices are Doors. In such a case, it might not be possible to add enough Doors; thus some of the vertices will be simply treated as Door vertices. This high rate of Door vertices among non-Door vertices become more and more impractical as the rate rises since robots placed on them will just get stuck and become Finished. The simulation is only to validate the $1/k$ characteristic of the runtime improvement.

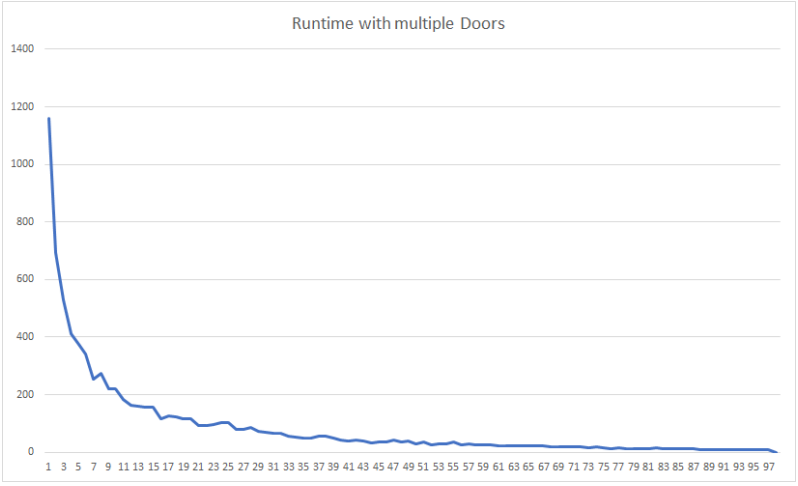

Figure 6

The horizontal axis shows the number of Doors in the area; the vertical shows the required number of turns to complete the Filling

Conclusions

In this paper, an extended version of the Virtual Chain Method was demonstrated for solving the Filling problem in 3D-areas. The algorithm's hardware requirement regarding visibility and communication range is optimal, and it is asymptotically optimal for the memory requirement. As a result, flying drones are able to solve the Filling problem even when they are present in an open space.

The drones do not have to be supplied with more equipment than a range finder in 4 horizontal directions, and 2 for vertical directions, $O(1)$ bits of persistent

memory, a compass, as well as a timing unit for achieving full synchrony. After they have dispersed, robots could provide a subservience or monitoring system in the given area, even in cases when human presence would be dangerous or impossible. Simulations then back up the theoretical results: the correctness, as well as, the runtime of the algorithms is tested. The correctness was validated as no collisions occurred, and the problem was solved completely every time. The performance tests also validate the linear runtime of the algorithm.

In the future, this work could lead to the investigation of possibilities of interaction between humans and robots (HCI) within the same field. Given that with the help of human assistance, it is possible to further improve completely autonomous robot-systems, in which case the human being will not only be a passive participant in the situation, but will also an active one engaging in the robot system itself. This outcome could be hugely beneficial in situations where the framework or the environment is likely to change at a quick pace, and the human being, given its more complex cognitive abilities, is able to adapt faster than the swarm algorithm. Moreover, with the assistance of the simulator, the inter-cognitive communication (communication between parties having a different level of cognitive abilities) between the robots and the human party can be further exploited, for example, by manually controlling a single member of the swarm.

The usage of a game engine in the simulator should also be mentioned since it allows the implementation of gamification elements, i.e., using game elements in non-game contexts [29]. With the help of this, the cooperative development process of the algorithm can be less intimidating and more inclusive for the user; as well as with automation and machine learning algorithms, the development of learning methods will be faster than before.

Acknowledgment

Project no. FIEK_16-1-2016-0007 has been implemented with the support provided by the National Research, Development and Innovation Fund of Hungary, financed under the Centre for Higher Education and Industrial Cooperation - Research infrastructure development (FIEK_16) funding scheme.

References

[1] C. W. Reynolds: Flocks, herds and schools: A distributed behavioral model, Proceedings of 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'87), New York, 1987, pp. 25-34

[2] Albers, Kursawe, and Schuierer: Exploring unknown environments with obstacles, Algorithmica, Vol. 32, No. 1, pp. 123-143, 2002

[3] S. Albers and M. R. Henzinger: Exploring unknown environments, SIAM Journal on Computinh, Vol. 29, No. 4, pp. 1164-1188, 2000

[4] J. Augustine and W. K. Moses, Jr.: Dispersion of mobile robots: A study of memory-time trade-offs, in Proceedings of 19th International Conference on Distributed Computing and Networking (ICDCN '18), 2018, pp. 1:1–1:10

[5] E. M. Barrameda, S. Das, and N. Santoro: Deployment of asynchronous robotic sensors in unknown orthogonal environments, in Algorithmic Aspects of Wireless Sensor Networks: 4th International Workshop (ALGOSENSORS 2008), Revised Selected Papers, 2008, pp. 125-140

[6] E. M. Barrameda, S. Das, and N. Santoro: Uniform dispersal of asynchronous finite-state mobile robots in presence of holes, in Algorithms for Sensor Systems - 9th Int. Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics (ALGOSENSORS 2013), Revised Selected Papers, 2014, pp. 228-243

[7] P. Brass, F. Cabrera-Mora, A. Gasparri, and J. Xiao: Multirobot tree and graph exploration, IEEE Transactions on Robotics, Vol. 27, No. 4, pp. 707-717, 2011

[8] R. Cohen and D. Peleg: Local spreading algorithms for autonomous robot systems, Theoretical Computer Science, Vol. 399, No. 1, pp. 71-82, 2008

[9] T. Hsiang, E. M. Arkin, M. A. Bender, S. P. Fekete, and J. S. B. Mitchell: Algorithms for rapidly dispersing robot swarms in unknown environments, Algorithmic Foundations of Robotics V, Springer Tracts in Advanced Robotics, Vol. 7, pp. 77-93, 2004

[10] N. Megow, K. Mehlhorn, and P. Schweitzer: Online graph exploration: New results on old and new algorithms, Theoretical Computer Science, Vol. 463, pp. 62-72, 2012, special Issue on Theory and Applications of Graph Searching Problems

[11] F. Bullo, J. Cortes, and S. Marttınez: Distributed algorithms for robotic networks, Applied Mathematics Series, Princeton University Press, 2009

[12] P. Flocchini, G. Prencipe, and N. Santoro: Distributed Computing by Oblivious Mobile Robots, Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2012

[13] A. Hideg and L. Blázovics: Area coverage using distributed randomized methods, 2016 Cybernetics and Informatics (K&I'2016), Levoca, 2016, pp. 25-34

[14] A. Hideg and T. Lukovszki: Uniform dispersal of robots with minimum visibility range, in Algorithms for Sensor Systems - 13th Int. Symp. on Algorithms and Experiments for Wireless Networks (ALGOSENSORS 2017), Revised Selected Papers, 2017, pp. 155-167

[15] A. Hideg, T. Lukovszki, and B. Forstner: Filling arbitrary connected areas by silent robots with minimum visibility range, in Algorithms for Sensor Systems - 14th International Symposium on Algorithms and Experiments

for Wireless Sensor Networks, ALGOSENSORS 2018, Revised Selected Papers, 2018, pp. 193-205

[16] A. Hideg, L. Blázovics, and B. Forstner: Multi-robot simulation framework, 2018 IEEE 9th International Conference on Cognitive Infocommunications (CogInfoCom), Budapest, 2018, pp. 159-164

[17] M. Sipos, P. Braun, D. Lucani, F. Fitzek, and H. Charaf: On the effectiveness of recoding-based repair in network coded distributed storage, Periodica Polytechnica Electrical Engineering and ComputerScience, Vol. 61, No. 1, pp. 12-21, 2017

[18] P. Baranyi and Á. Csapó: Cognitive Infocommunications: CogInfoCom. 11th IEEE International Symposium on Computational Intelligence and Informatics, CINTI 2010 - Proceedings. 141-146

[19] P. Baranyi, Á. Csapó, and Gy. Sallai: Cognitive Infocommunications (CogInfoCom) Springer International Publishing, 2015

[20] P. Baranyi and Á. Csapó: Definition and synergies of cognitive infocommunications, Acta Polytechnica Hungarica, Vol. 9, No. 1, pp. 67-83, 2012

[21] V. Kövecses: Cooperative learning in vr environment, Acta Polytechnica Hungarica, Vol. 15, pp. 205-224, 06. 2018

[22] Á. B. Csapó, I. Horváth, P. Galambos and P. Baranyi: VR as a Medium of Communication: from Memory Palaces to Comprehensive Memory Management, 2018 9th IEEE International Conference on Cognitive Infocommunications (CogInfoCom), Budapest, Hungary, 2018, pp. 389-394

[23] Ildikó Horváth: Evolution of teaching roles and tasks in VR/AR-based education, Proceedings of the 9th IEEE International Conference on Cognitive Infocommunications (CogInfoCom), Budapest, 2018, pp. 355-360

[24] A. Kovari: CogInfoCom Supported Education: A review of CogInfoCom based conference papers, Proceedings of the 9th IEEE International Conference on Cognitive Infocommunications (CogInfoCom) 2018, pp. 233-236

[25] I. Horváth: The IT device demand of edu-coaching in the higher education of engineering, 8th International Conference on Cognitive Infoommunications, 2017, pp. 379-384

[26] A. Kovari: CogInfoCom Supported Education: A review of CogInfoCom based conference papers, Proceedings of the 9th IEEE International Conference on Cognitive Infocommunications (CogInfoCom) 2018, pp. 233-236

[27] V. Szűcs, T. Guzsvinecz, and A. Magyar: Movement Pattern Recognition in Physical Rehabilitation - Cognitive Motivation-based IT Method and Algorithms. Acta Polytechnica Hungarica, Vol. 17, No. 2, pp. 211-235, 2020

[28] R. Ahlswede, Ning Cai, S. R. Li, and R. W. Yeung: Network information flow, IEEE Transactions on Information Theory, Vol. 46, No. 4, pp. 1204-1216, July 2000

[29] R. Pinter, S. Čisar, Z. Balogh, and H. Manojlovic: Enhancing Higher Education Student Class Attendance through Gamification. Acta Polytechnica Hungarica, Vol. 17, No. 2, pp. 13-33, 2020