
Proceedings Paper:
Salomons, N., Kapellmann-Zafra, G. and Gross, R. orcid.org/0000-0003-1826-1375 (2016) Human management of a robotic swarm. In: Towards Autonomous Robotic Systems. Towards Autonomous Robotic Systems (TAROS 2016), June 26 - July 1, 2016, Sheffield, UK. Lecture Notes in Computer Science, 9716 . Springer International Publishing , pp. 282-287.

https://doi.org/10.1007/978-3-319-40379-3_29

Human Management of a Robotic Swarm

Nicole Salomons, Gabriel Kapellmann-Zafra, and Roderich Groß

Sheffield Robotics & Department of Automatic Control and Systems Engineering,
The University of Sheffield, Sheffield, UK
nicolejsalomons@gmail.com

Abstract. This paper proposes a management algorithm that allows a human operator to organize a robotic swarm via a robot leader. When the operator requests a robot to become a leader, nearby robots suspend their activities. The operator can then request a count of the robots, and assign them into subgroups, one for each task. Once the operator releases the leader, the robots perform the tasks they were assigned to. We report a series of experiments conducted with up to 30 e-puck mobile robots. On average, the counting and allocation algorithm correctly assigns 95% of the robots in the swarm. The time to count the number of robots increases, on average, linearly with the number of robots, provided they are arranged in random formation.

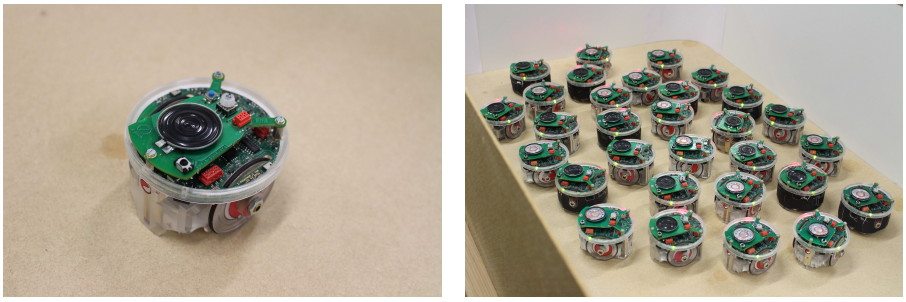
1 Introduction

Swarm robotics is concerned with the study of a large number of robots which are usually simple, and of limited capabilities. When deploying a swarm of robots in a real-world scenario, the swarm is likely to get separated into different groups, for example, due to obstructions in the environment. Moreover, some of the robots might even get lost. A challenge of such real-world environments is that the operator typically has no birds-eye perspective. Yet, the operator would like to know how many robots are currently in an area and within communication reach of one another.

Methods for counting robots have been proposed in [4,1]. Brambilla et al. [1] use an algorithm similar to firefly signalling. The algorithm counts the number of nearby robots that are signalling during a period of time. Ding and Hamann [2] use an algorithm that counts the number of locally connected robots and causes them to self-organize into two different areas based on their assigned colors. The robots have unique IDs, and exchange these IDs via pair-wise communication until all robots within communication range share the same set of IDs.

Different algorithms for task allocation are presented in [3]; our paper uses an algorithm similar to Extreme-Comm. Extreme-Comm is considered reliable and fast compared to the other algorithms, but requires a large number of exchanged messages between the robots. Each robot spreads its ID and keeps a list of the other robots' IDs. According to their position in the list, the robots choose their task.

In both [2] and [1], the counting experiments are performed in simulation, with no human operator involved. This paper proposes an algorithm that allows



(a) (b)

Fig. 1: (a) The e-puck robot used in the experiments; (b) a group of 30 robots, which are performing a count.

an operator to select any robot from a group of real robots, and receive a count of the robots which are in communication range. Subsequently the operator can manage the group of robots by assigning different tasks to subgroups of them, and then releasing them to perform the tasks.

2 Management Algorithm

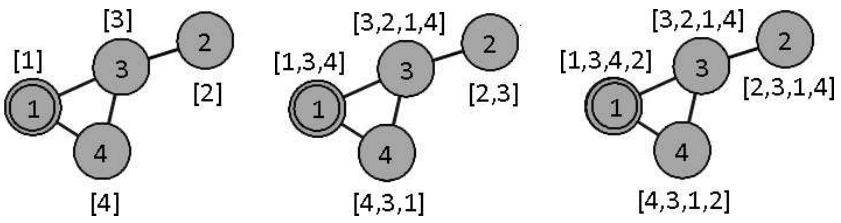
The main purpose of the management algorithm is to provide the operator with advanced capabilities of organizing a robotic swarm. The algorithm was implemented on a swarm of e-puck robots [5]. The e-puck is a differential drive robot, around 75 mm in diameter. It is shown in Figure 1a. It has eight infrared (IR) transmitters, located around its perimeter, which the robots use to communicate with surrounding robots. Their maximum communication range is limited to 15 cm, creating a fairly stable connection between the robots. The e-puck also has Bluetooth, which allows an operator to send and receive information wirelessly via a computer.

2.1 Human Interaction

The management algorithm allows the operator to reassign groups of robots to different tasks during run-time, and to verify the status of the robot swarm without requiring direct visual contact with the swarm. The commands are given to the e-puck robots through a Bluetooth connection with the leader robot. The following commands are available:

– **Request Robot**: The operator can request either a random robot or a specific robot, by providing its ID. The selected robot, hereafter called *leader*, stops its movement, and requests the neighboring robots to stop as well. If a robot receives a stop request, it stops and relays the message.[1] By suspending

[1] The stop message is relayed periodically so if other robots enter communication range, they also stop and participate in the counting of the robots.



(a) (b) (c)

Fig. 2: Example of robots counting (lines denote pairs of robots within communication range). (a) Initially, only their own ID is in the list. (b) The robots include the IDs received by their neighbors. (c) After a further iteration, each robot contains everyones' IDs in their list. Note that perfect communication is assumed. In practice, the e-pucks may be unable to receive multiple messages simultaneously.

the movement within the group of connected robots, their connectivity can be maintained for the duration of the operation.

- **Request Count**: The operator can request a count of robots belonging to the same group as the leader.
- **Assign Tasks**: For each task, the operator sends the task ID and how many robots should be assigned to it. However the operator has no control over which specific robots are selected. The leader may also assign a task to itself if all other robots have already been assigned tasks.
- **Release Robots**: The robots resume their suspended tasks, or perform their newly assigned task, if any.

2.2 Robot swarm counting

The counting algorithm is based on [2], but here implemented on physical rather than simulated robots. Each robot has a list of IDs. Upon initialization, a robot's list contains only its own ID, which is assumed to be unique. The robot iterates through its list and broadcasts each ID. Once reaching the end of the list, the robot starts over again. Whenever receiving an ID, the robot adds it to its list, if not already present.

Figure 2 illustrates the counting process for a group of four robots. At the beginning, each robot has only its own ID. The list is updated when receiving messages from neighboring robots. Notice how robots 1 and 2 are not in communication range of each other, but are both in range of robot 3. Robot 3 will pass the ID of robot 1 to robot 2 and vice versa, and thus robot 1 and robot 2 will, after two iterations, also have each other's IDs in their lists.

2.3 Task assignment

After the operator connected to a leader and performed a count, a number of robots can be assigned to perform a certain task. Similar to counting, each robot

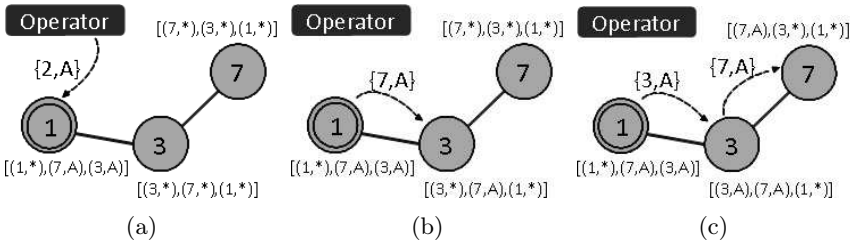


Fig. 3: Example of the leader robot assigning tasks (lines indicate pairs of robots within communication range). The leader robot receives a request by the operator to assign two robots to task A. The leader robot chooses two robots (a), and broadcasts the task assignment information through the swarm (b–c).

keeps a list of IDs which also includes a task ID associated with each robot ID. The list will be broadcast continuously until all the robots know which task all robot IDs are assigned to.

Figure 3 demonstrates how tasks are assigned to subgroups of robots. To start the process, the operator sends the leader task A and requests that 2 robots should be assigned to it. The leader assigns task A to robots 3 and 7 in the list. In our example, the leader and these two robots are not yet assigned any tasks. The leader starts broadcasting the IDs and associated tasks to its neighboring robots. Robot 3 upon receiving a task assignment for robot 7, broadcasts the message and robot 7 receives it. When robots 3 and 7 receive a message with their ID, they update their task but remain static. Once they are released by the leader robot (robot 1), they start task A. When the robots are released, they reset their ID list.

2.4 Message Structure

The robots exchange four types of messages through infrared communication. All messages consist of 5 digits: the first indicates the message type (1–4); the next 3 are the robot ID (001—999) if applicable, otherwise 000; the last digit can be a task ID, checksum or 0.

Once the operator requests a robot, the robot starts broadcasting message type 1 (*stop*) to all surrounding robots. When receiving this message, a robot stops moving and relays the message to surrounding robots periodically.

If the leader receives a request for a count of the swarm, it triggers the counting process described in Section 2.2. The robots use message type 2 (*count*), which contains an ID and checksum. This improves the reliability of the IR communication, and minimizes the risk of wrong IDs being added to the list. If the maximum message size permits, it is possible to send multiple IDs per message instead of just one.

Message type 3 (*task assignment*) contains a robot ID and a task ID. Whenever a robot receives this message, it saves the message to its list. If the ID in

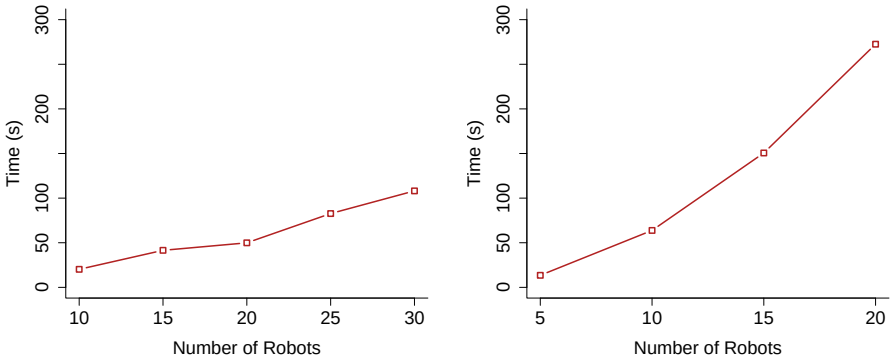


Fig. 4: Counting time for swarms in (a) random and (b) linear formations.

the message is the robot's own ID, it performs this task once released by the leader, but refrains from relaying the message.

The operator can release the group by sending a message of type 4 (*release robots*), which is relayed by the members in the group. The robots then start the newly assigned tasks, or if not assigned a new task, resume their previous tasks.

3 Results

Experiments were conducted to examine the counting and task assignment performance of swarms of physical e-puck robots. Unless otherwise stated, the robots were organized in random formations, that is, they were arbitrarily placed. For an example formation, see Fig. 1b.

3.1 Counting and Task Assignment Accuracy

We analyzed the accuracy of the counting and task assignment process. We assumed that 30% of the swarm are to be assigned to tasks A and B each, and the remaining 40% to no task. We define the overall accuracy as the percentage of robots that ended up with a correct assignment, averaged over multiple trials.

We performed 10 trials with 10 robots and 10 trials with 20 robots. For groups of 10 robots, the accuracy was 95%. For groups of 20 robots, the accuracy was 96%. Errors in the task assignment can happen, for example, when a low battery causes a robot to reset, and hence lose its list information. Errors can also be introduced during message transmission, causing a robot to be assigned the wrong algorithm, or include an ID in its list that does not exist.

3.2 Counting Time

We estimated the time it takes to obtain the count. Figure 4a shows the time (in s) it took to count 10, 15, 20, 25 and 30 robots organized in random formations.

For each group size, 10 runs were conducted and the mean time is reported. The counting time seems to increase linearly with the number of robots.

We also considered linear formations. In other words, the robots were sequentially arranged, starting with the leader robot. Intermediate robots were in communication range of only their two neighbors. This formation can be considered as the worst case. Figure 4b shows the time it took to count 5, 10, 15 and 20 robots organized in linear formation. As can be seen, the counting algorithm scales less favorably when the robots are organized in this worst case formation.

4 Conclusions

We implemented a counting algorithm proposed by [2] on a swarm of physical e-puck robots and expanded it to also include a role assignment mechanism. The algorithm could be easily adapted for different platforms. The only requirements are that the robots have local communication, memory to store the lists and an interface to interact with the operator.

Our results show that the robots are on average 95% accurate when counting and sequentially assigning tasks. The counting time seems to increase linearly with the number of robots, as long as they are sufficiently mixed. For linear formations, however, the algorithms took substantially longer to complete.

Future work will implement these algorithms on other robotic platforms as well as consider refined broadcasting protocols.

References

1. Brambilla, M., Pinciroli, C., Birattari, M., Dorigo, M.: A reliable distributed algorithm for group size estimation with minimal communication requirements. In: Proc. International Conference on Advanced Robotics (ICAR 2009). pp. 1–6. IEEE (2009)
2. Ding, H., Hamann, H.: Sorting in swarm robots using communication-based cluster size estimation. In: Proc. 9th International Conference on Swarm Intelligence (ANTS 2014), LNCS, vol. 8667, pp. 262–269. Springer (2014)
3. McLurkin, J., Yamins, D.: Dynamic task assignment in robot swarms. In: Proc. Robotics: Science and Systems (RSS 2005). Cambridge, USA (2005)
4. Melhuish, C., Holland, O., Hoddell, S.: Convoying: using chorusing to form travelling groups of minimal agents. Robotics and Autonomous Systems 28(2), 207–216 (1999)
5. Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.C., Floreano, D., Martinoli, A.: The e-puck, a robot designed for education in engineering. In: Proc. 9th Conference on Autonomous Robot Systems and Competitions. vol. 1, pp. 59–65. IPCB: Instituto Politécnico de Castelo Branco (2009)