A Telepresence-Robot Approach for Efficient Coordination of Swarms

Karl Tuyls[2]⋆, Sjriek Alers[1], Elisa Cucco[2⋆], Daniel Claes[2] and Daan Bloembergen[2]

[1]Maastricht University / Fontys University of Applied Sciences, Eindhoven the Netherlands
[2]Department of Computer Science, University of Liverpool, UK
⋆k.tuyls@liverpool.ac.uk, Elisa.Cucco@liverpool.ac.uk

Abstract

In this paper we explore a novel perspective on surveillance robotics, which is based on a coordination principle of honeybees, and on the integration of an autonomous telepresence robot in such system. Coordination principles, based on biological systems such as ant, bee and termite colonies, show several properties which are essential to multi-robot surveillance, including low computation load, robustness, scalability and adaptability. In this paper we aim to improve on the efficiency of such a robotic swarm by taking a human in the loop by means of a telepresence robot. The human operator controlling the telepresence robot will aim to speed up the convergence of the swarm. The experiments, which evaluate the proposed multi-robot coordination system both in simulation and on real robots, show how the telepresence robot substantially increases the efficiency of the process.

Introduction

In recent years there has been a rapidly growing interest in using teams of mobile robots for automatically surveilling environments of different types and complexity. This interest is mainly motivated by the broad spectrum of potential civilian, industrial, and military applications of multi-robot surveillance systems (Kuorilehto et al., 2005; Folgado et al., 2007). Examples of such applications are the protection of safety-critical technical infrastructures, the safeguarding of country-borders, and the monitoring of high risk regions and danger zones which cannot be entered by humans in the case of a nuclear incident, a bio-hazard, or a military conflict.

Triggered by this interest, today automated surveillance is a well-established topic in multi-robot research, which is considered to be of particular practical relevance. Despite the remarkable progress made on this research topic so far, there is still a huge gap between theory and practice of multi-robot surveillance systems, and as a consequence there are still only very few on-field deployments. The reason for this is that many basic questions about coordination among mobile robots are not yet answered in a satisfactory way.

In this paper a new approach on multi-robot surveillance systems is proposed, which is based on a bio-inspired coordination principle from swarm intelligence and on the integration of an autonomous telepresence robot in such system.

Natural entities, such as ant and termite colonies improve their collective performance by influencing one another through local messages they deposit in their shared environment. In computer science, robotics and economics a number of computational variants have been developed, and it has been shown that they allow for very efficient distributed control and optimization in a variety of problem domains. For instance, recent work shows a strong potential in creating artificial systems that mimic insect behaviour that can solve complex coordination tasks such as e.g., routing on the internet, mobile ad hoc network routing, robotic tasks, etc. (Lemmens and Tuyls, 2012; Dressler and Akan, 2010; Floreano and Mattiussi, 2008).

Swarm optimisation algorithms, like ant colony optimisation (Dorigo et al., 2006), rely on pheromone trails to mediate (indirect) communication between agents. These pheromones need to be deposited and sensed by agents while they decay over time. Though easy to simulate, artificial pheromones are hard to bring into real-life robotic applications. However, recently non-pheromone-based algorithms were developed as well (Lemmens, 2011). Such algorithms are inspired by the foraging and nest-site selection behaviour of (mainly) bees. In general, bees explore the environment in search for high quality food sources and once returned to the hive they start to dance in order to communicate the location of the source. Using this dance, bees recruit other colony members for a specific food source. The algorithm we used draw inspiration from these insect behaviours with the goal to create intelligent systems for distributed coordination that can be deployed in real world settings.

The key idea put forward in this paper is that a telepresence robot can improve upon the efficiency of such a swarm. Telepresence robotics is a form of teleoperation, namely the extension of a person's sensing and manipulation capability to a remote location, in which a human operator act as a supervisor intermittently communicating information about goals and

actions relative to a specific task. The human operator will receive information about accomplishments, difficulties and, as requested, raw sensory data, while the subordinate telepresence robot executes task based on information received from the human operator plus its own sensing and artificial intelligence (Sheridan, 1989). In the approach we propose in this paper the human operator controlling the telepresence robot can observe the environment and will aim to steer the behaviour of the swarm by means of direct communication.

In the following sections we introduce telepresence robotics and the biological background of our foraging approach. Then we show our experiments and discuss the efficiency of the algorithm and the improvement obtained by integrating a telepresence robot in the system.

Telepresence robotics

Already more than 30 years ago, artificial intelligence pioneer Marvin Minsky (Minsky, 1980) laid out an ambitious plan calling for the development of advanced teleoperated robotics systems that would result in a remote-controlled economy. He coined the term "telepresence" to describe these systems, which in his futuristic vision would transform work, manufacturing, energy production, medicine and many other facets of modern life. Although the idea of a teleoperated robot for remote presence is not new, only recently telepresence robots become available to the broader public (Lazewatsky and Smart, 2011; Takayama et al., 2011; Tsui et al., 2011). Basically, telepresence robotics systems can be described as embodied video conferencing on wheels, providing a physical presence and independent mobility in addition to communication, unlike other video conferencing technologies, allowing the user to interact more naturally in the remote office environment.

However, telepresence robots can be deployed in a wide range of application domains: the informal meeting scenario in offices, in hospitals to allow doctors to provide consultations from a distance (Tsui et al., 2011) or to pay a virtual visit when it is not possible to be present in person, or to give people with restricted mobility a new way to interact beyond their possibility. Furthermore, many work-sites are hazardous to human health or even survival. With telepresence robotics it will be potentially possible to operate in dangerous environments without such risks.

Adding a level of autonomy to a telepresence robot can greatly improve the experience of the user, as it reduces their cognitive load. This allows to focus more attention on the interaction and to the task and less on controlling the robot (Tsui et al., 2011). However, it remains important for the operator to have control over the behaviour of the system. Indeed, as a telepresence robot is controlled from a remote location, precise control and feedback of the robot is required. One possible solution, assisted navigation, is investigated by Takayama et al. (2011). Adding more autonomy and integrating the findings of recent AI research into the platform can greatly increase the usability of these robots.

Biological coordination

A great deal of research in swarm intelligence is situated in the area of bio-inspired computation; more precisely in the area that investigates algorithms that find inspiration from nature in order to develop novel computational models, often to solve coordination problems. Foraging is one of the coordination problem in this domain. Essentially it consists of two sub-problems: path construction/planning and path exploitation/repair. The task of foraging consists of gathering objects out of the environment and returning them to a central point, most often the starting location. A commonly used method for solving foraging problems focuses mainly on the behaviour of social insects such as ants and bees.

Ants deposit pheromone on the path they take during travel. Using this trail, they are able to navigate toward the food location and communicate with other members of the colony, not directly but by accumulating pheromone trails in the environment. Pheromone strength indicates the "fitness" of a trail but is not able to indicate direction, therefore an ant is not able to know a priori to which destination it is travelling. When a trail is strong enough, other ants are attracted and will follow it towards a destination which results in a reinforcement of the trail. This is known as an autocatalytic process: the more ants follow a trail, the more that trail becomes attractive for being followed. Short paths are reinforced more often over time and will eventually be preferred. This principle is used to address several problems, such as Routing Problem (Di Caro et al., 2005) and area coverage with robots (Wagner et al., 1999; Ranjbar-Sahraei et al., 2012).

On the other hand, bees and desert ants do not use pheromones to navigate in unfamiliar environments. Their navigation mainly consists of Path Integration (PI). The PI vector represents the continuously updated knowledge of direction and distance and, as a consequence, bees are able to return to their starting point by choosing the direct route rather than their outbound trajectory. More precisely, when the path is unobstructed, the insect exploit previous search experience. However, when the path is obstructed, the insect has to fall back on other navigation strategies such as exploration (Collett and Collett, 2009). For recruitment bees communicate with other colony members by means of a waggling dance performed in the hive. The direction of the food source is read from the angle between the sun and the axis of a bee's waggle segment on the vertical hive comb, while the duration of the waggle phase is a measure of the distance to the food and the "fitness" of a solution

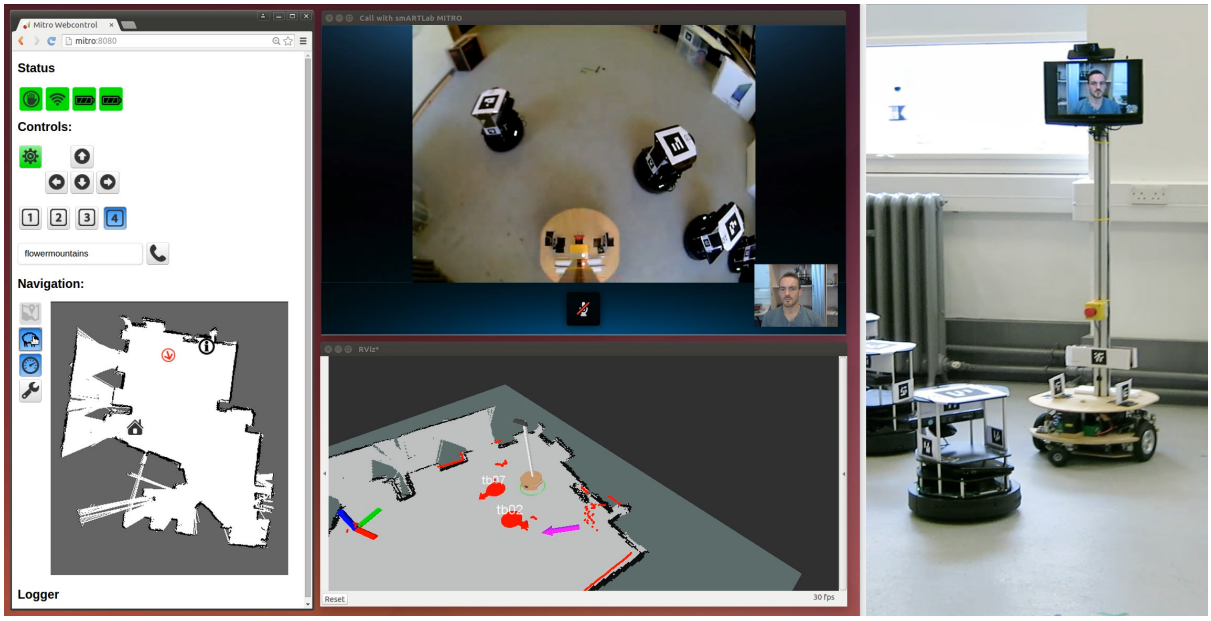


Figure 1: MITRO interface and Turtlebots foraging supervised by a telerobot.

(von Frisch, 1967). More precisely, depending upon the strength of the dance, more bees are attracted and follow the PI vector toward a destination. Furthermore, the more bees follow a PI vector, the more that destination will be communicated and the more it will attracts other bees. Eventually, the best solution prevails.

Transferring these principles to algorithms is the domain of computational swarm intelligence. Comparisons of these algorithms (Lemmens et al., 2008) show that the bee-inspired mechanism is able to collect all the items in the environment faster than the ant-inspired mechanism in a relatively unobstructed environment. However, in an environment with more obstacles and/or dynamic environment, the bee-inspired mechanism is less adaptive.

System & Approach

The main idea of the proposed approach is to integrate swarm algorithms with telepresence robotics. We build on previously developed algorithms in swarm robotics, aiming to achieve a food foraging application in the real world guided by a telepresence robot that will be shepherding the swarm.

Similar to the Path Integration principle, the robots in our swarm estimate their positions by integrating information coming from the gyroscope and the wheel odometry. Using this, the robots can always compute the home vector (HV), and if the food location is seen, the path integration (PI) vector can be used to communicate the location to other robots.

Therefore, no map of the environment needs to be built by the robots and the only common reference point that is needed for the correct communication of the food locations is the hive location, i.e. the HV.

As a consequence, if the odometry is faulty, the robot might not find the hive or food location, and if this problem occurs the robots fall back in search mode. As soon as the hive or the food are seen again, the robots update their internal reference system.

In contrast to the honeybees's behaviour, we also allow communication outside of the hive, since it is very likely that the robots see each other under way. Additionally, there's also a probability that the robots return to the hive after being in the search state for a long time, in order to increase the chance to meet another robot that might already be in foraging mode.

This approach has been demonstrated to work reasonably well for small environments (Alers et al., 2014a,b). However, there was no human supervision involved and also no simulation runs were performed to gain empirical insights about the performance of the swarm, i.e. how long it takes the swarm to converge on the food locations, what is the throughput of the system, etc.

In this paper, we propose a novel approach to add a human shepherd to the system, which can supervise the swarming robots and help to enable faster convergence. The idea is that a human can interact with the swarm using a telepresence robot as a shepherd. The human operator can have more knowledge of the environment, i.e. a map and a camera. After a food location has been found, the shepherd can steer the swarm towards that location or catch "lost" swarm robots.

We implemented the approach using the Turtlebot[1] platform as swarm robots and a custom-built telepresence robot MITRO (Alers et al., 2013) as shepherd. These platforms will be explained in more detail in the next subsections.

[1] `http://www.turtlebot.com`

Figure 1 shows an overview of the system. On the left, the interface for the human controlling MITRO is shown. It gives an overview of the system's status, allows the user to control the robot and shows the live video feed of the environment. Additionally, the internal view of the robot is shown, below the video feed. The reference frame is depicted as axis, and the two circles with arrows are the detected robots. On the right, a picture of the real-life experiment is shown, where MITRO is shepherding in the middle of several Turtlebots.

Swarm robots

As explained before, for the real world experiments, we use the Turtlebot platform. It has a laptop on board with a core-i3 CPU for computation, running the Robot Operating System (ROS) (Quigley et al., 2009) framework. The robots are also equipped with a Kinect sensor and the RGBD information is used to detect and locate AR markers, see black and white markers in Figure 1. This sensor is also used for the obstacle detection, together with three bumpers located in front half of the robot.

To enable visual robot-robot detection every Turtlebot has six unique markers, oriented in a way that at least one marker is always visible. To track and decode these markers we use the ROS wrapper of the ALVAR toolkit [2]. We use a customised bundle detection method to determine the center of the detected robot. Each marker in the bundle encodes the robot number and its location with respect to the center of the robot. This information is used to predict the position of the detected robot. Kalman filtering is also applied to get more stable and accurate estimation of the detected robots position, heading and speed. These parameters are also used for the collision avoidance.

Communication between Turtlebots is realised over wi-fi using a UDP connection to each Turtlebot. Even though global communication would be possible, we limit the communication of each robot to its own channel and allow only communication after visual detection of its peer. Therefore, the robots can communicate only with another robot if it is in close proximity.

In order to avoid collisions between robots we rely on the marker detection to predict positions and speeds of the other robots. The obtained information could be used to efficiently compute a non-colliding speed vector (Claes et al., 2012). In contrast to the previous approach, in which the robots avoided each other by using a global reference frame and broadcasting the positions to all robots via Wi-Fi, we adapted this method to only rely on the marker detection and the predictions using a Kalman filter. However, a few collisions still might occur due to the failed detection of other robots and additionally in such configurations in which the robots cannot see each other because of the field of view of the Kinect sensor.

[2] http://www.wiki.ros.org/ar_track_alvar

Telepresence robot

In addiction to the Turtlebot platform we also use a custom-built telepresence robot, shown in the right panel of Figure 1 (Alers et al., 2013). The advantage of using a custom-built system over a commercial platform is the flexibility, extendibility and knowledge of the complete system, that for our purpose is crucial.

The robot has a height of approximately 160 cm and is based on the Parallax Mobile Robot Base kit, which includes the base plate, powerful motors and 6 inch wheels with pneumatic tires. The sensors include a low-cost LIDAR, an Asus XTION PRO 3D sensor, sonar sensors, and two cameras (one pointing forward for conversations, one fish-eye camera pointing downwards for driving). The robot is also running ROS.

Since the robot is controlled from a remote location, we implemented low level autonomy on the robot in the form of assisted teleoperation. With assisted teleoperation the robot follows the steering commands of the operator except for a situation when there is a high chance of collision. This can easily occur when the user is not experienced in navigating the robot, the network connection is delayed or an obstacle suddenly appears in front of the robot. Additionally, the video feed can be switched from front- to down-facing, and is augmented with a projection of the expected navigational path. Furthermore, the robot is able to perform SLAM (simultaneous localization and mapping) to build a map of its environment (Thrun et al., 2005); this map is used subsequently for localization and autonomous navigation to a chosen destination, or back to his charging location, all to ease the remote operation.

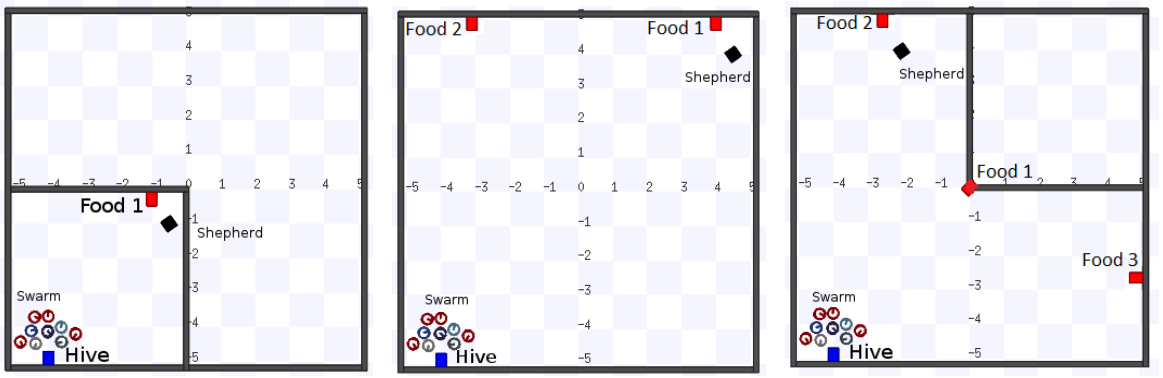
Experiments

In our experiments the Turtlebots are performing a foraging task, starting at the hive (H) location and randomly exploring the unknown environment for a specific food (F) location. The robots can also locate the food location by asking bypassing robots for a known food location, see Figure 2. When the source is found the Turtlebots start to exploit this source, driving from the food to the hive, where they drop the food, until the food is depleted or another source is found. The telepresence robot works as a "shepherd" sending relative location information to the Turtlebots.

We implemented our approach on the real robots as in simulation for getting additional statistics. In this experiment section we will describe the simulation results, in the demonstration section the real-world setting is shown. Simulations are run in real time using

(a) **(b)** **(c)**

Figure 2: Multi-Robot foraging. **(a)** All robots start at the hive (H) location. **(b)** Robots are exploring the unknown environment randomly. The left two robots have found the food (F) location and are foraging between the hive and the food location. **(c)** All robots have converged to foraging behaviour.



(a) 5x5 simulation environment with 1 food source.

(b) 10x10 simulation environment with 2 food sources.

(c) 10x10 L-shaped simulation environment with 3 food sources.

Figure 3: The different simulation environments with the shepherding robot (black square), food sources (red square), and 9 robots located at the hive-location (blue square)

Stage (Gerkey et al., 2003; Vaughan, 2008). We use simulated Turtlebots and a simple differential drive robot as telepresence robot. For the detections, mock-ups are written, so that the same state-machine is run on the real robots and in simulation. Having the simulation setup allows us to investigate the system performance for different scenarios and using more repetitions than would be feasible in the real world.

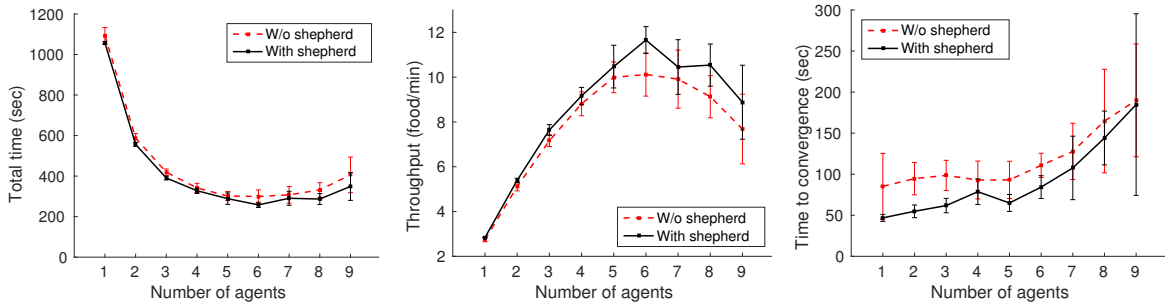
The main goal of our experiments is to compare the performance of the original bee-inspired algorithm with the newly proposed approach that has the telepresence robot in the system. We evaluate the proposed approach in simulation for 3 different environments: 5x5 meters square shaped, 10x10 meters square shaped, and 10x10 meters L-shaped, shown in Figure 3.

In the first case, we compare the performance of the swarm for different numbers of Turtlebots involved in the foraging task. We evaluate the throughput, the speed of convergence and the efficiency of the foraging process with and without the shepherding telepresence robot. We also collect statistics showing the user effort, expressed as the number of times the user interfered (i.e. corrected a Turtlebot's navigation), and the total distance driven by the telepresence robot. We repeat the same experiments in the 10x10 world and in the 10x10 L-shaped environment with 9 Turtlebots, and for these cases we evaluate the convergence of the algorithm after moving the food to a different location, e.g. due to depletion of the first food source. Each experiment lasts until 50 food units have been transported from the source to the hive. Similarly, in the 10x10 environments, a food source becomes "depleted" after 50 food units, upon which a new source becomes active. Every experiment is repeated 10 times, and the results are averaged.

Discussion

Figure 4 shows the results of simulations in the 5x5 world. In this relatively small environment the swarm will often converge without the interference of the telepresence robot, except for a few cases when the number of robots get too large for the environment, leading to collisions, and robots getting stuck. However, through minimal user effort the shepherd still improves the efficiency of the process.

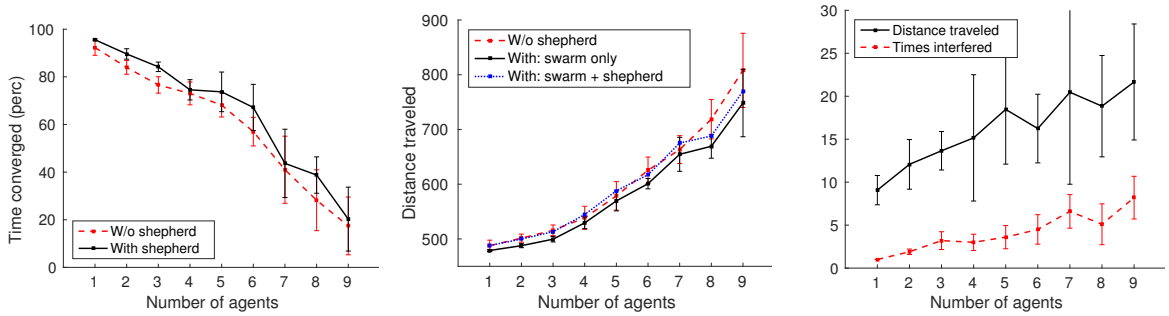
Figure 4(a) shows the total time, in seconds, needed to complete the task (i.e., transport 50 food units), with the error bars representing the standard deviation intervals. We observe that the optimal swarm size is reached at 6 robots, both with and without shepherd. When the swarm size increases beyond this point, the small environment becomes too clut-



(a) Total time needed for 50 food drops.

(b) Average throughput in food drops per minute.

(c) Time to first convergence.



(d) Percentage of time during which the swarm was converged.

(e) Total distance traveled by the swarm.

(f) Distance traveled and number of times interfered by shepherd.

Figure 4: Results for the 5x5 world with different swarm size.

tered as robots start colliding, hindering each other's performance. The same trend can be observed when looking at the total throughput in Figure 4(b), measured in units of food delivered per minute. Here again we see that the optimum is reached for a swarm size of 6 robots. Shepherding significantly improves the performance of the swarm in both cases.

We also investigate the convergence performance of the system. In Figure 4(c) we show the time needed in seconds until the whole swarm is converged, meaning that all robots are aware of the food location and are continuously going back and forth between the hive and food location to transport food units. The figure shows that the time needed to converge stays more or less stable up to 5 robots, after which the environment becomes more cluttered, preventing the robots from converging quickly. Additionally, the converged state can be lost again, e.g. due to collisions, or robots driving in each others line of sight preventing them from relocating the food. In Figure 4(d) we plot the percentage of experiment time during which the whole swarm is converged, and note that this value decreases approximately linearly with an increasing swarm size. The fact that robots may get in each other's way can also be observed by looking at the total distance travelled (in meters) by the swarm during the course of one experiment, which increases exponentially with the swarm size (Figure 4(e)). This shows that even though a swarm of size 6 is optimal in both time and throughput, it is not necessarily the most efficient in terms of per robot performance.

Finally, in Figure 4(f) we look at the effort required by the user to guide the swarm. The figure shows the distance travelled by the telepresence robot, as well as the number of interferences, i.e. the number of times that the user has corrected a swarm robot's navigation target. We can see that the required effort doesn't necessarily grow with the number of swarm robots, indicating that the robots are able to relay the new information among the swarm.

We now move on to the larger environment. Table 1 shows the results for the 10x10 world (with terms between parenthesis representing the standard deviation) with and without moving the food source. In both experiments the shepherd can significantly improve the performance of the system. In particular, after moving the food source the swarm without shepherd takes more than twice as long to re-converge (third column in the table) as the swarm with shepherd. Also note that when moving food, without shepherd the swarm only fully re-converged in 3 out of 10 runs, while with shepherd this happened in 9 out of 10 runs.

Results for the 10x10 L-shaped environment are shown in Table 2. Again, shepherding significantly improves the performance of the swarm with relatively limited effort. However, this task is clearly harder, as the food source is moved twice. A breakdown of time to re-convergence, as well as the num-

Table 1: Results for the 10x10 world with and without moving the food source.

	Total time	Time to conv. (1)	Time to conv. (2)	% of time converged	Throughput	Shepherd distance	Times interfered
Static							
W/o shepherd	510.5 (56.4)	304.7 (68.5)	n/a	38.5 (4.9)	5.9 (0.6)	n/a	n/a
With shepherd	389.7 (12.9)	191.9 (42.4)	n/a	47.7 (10.1)	7.7 (0.3)	44.8 (12.6)	8.0 (2.8)
Moving food							
W/o shepherd	922.6 (68.2)	328.1 (75.1)	316.9 (81.7)	20.0 (5.6)	6.5 (0.5)	n/a	n/a
With shepherd	677.7 (42.9)	181.2 (51.3)	146.7 (37.7)	46.0 (11.3)	8.8 (0.5)	88.9 (19.4)	20.4 (3.2)

Table 2: Results for the L-shaped world with and without shepherding.

	Total time	Throughput	Distance traveled	Shepherd distance	Times interfered
W/o shepherd	1068.8 (167.4)	8.3 (1.0)	3347.9 (466.6)	n/a	n/a
With shepherd	895.2 (73.4)	10.1 (0.8)	2984.5 (96.9)	121.2 (14.0)	26.8 (6.6)

Table 3: Break-down of convergence times in the L-shaped world with and without shepherding. Food is moved twice. Convergence times are listed for the three food locations, as well as the number of runs that did converge.

	Time to conv. (1)	Time to conv. (2)	Time to conv. (3)	Nr. of conv. (1)	Nr. of conv. (2)	Nr. of conv. (3)
W/o shepherd	223.3 (57.0)	235.6 (91.3)	278.2 (60.7)	8/10	6/10	4/10
With shepherd	188.9 (73.2)	137.6 (39.0)	201.4 (67.6)	10/10	10/10	9/10

ber of runs that fully re-converged, is given in Table 3. Both metrics are significantly improved by the shepherd. We note that the first time the food is moved, the shepherd is able to make a big difference, as the distance between both food locations is easy to overcome. In contrast, the last food location lies in the opposite side of the L-shape, around the corner. This makes it harder for the swarm to re-locate the food, even with the help of the shepherd.

Demonstration

We have also undertaken a real-world experiment in which 5 Turtlebots are foraging in an unknown environment. All the robots are initially located around the hive and they start to explore the environment randomly for the food location. An operator supervises the group using the MITRO telepresence robot. The user is able to send the food location information to individual Turtlebots, e.g. when they get stuck. A video showing this demonstration can be found online.[3] In this physical implementation the sheperd robot increases the efficiency of the foraging process and speed up the convergence of turtlebots, especially when the food is moved.

Conclusion and further work

We have proposed a new approach for swarm robotics systems, which is based on both the coordination principle of honeybees and on human-robot interaction through telepresence robotics. In order to validate the approach we performed swarm experiments, i.e., a foraging task in a unknown environment, both in simulation and in a situated environment. Our results show that the telepresence robot, acting as a shepherd, can substantially increase the efficiency of the foraging process, especially in dynamic and complex scenarios, in which food sources change over time. Only a limited effort by the telepresence robot can already make a great difference in performance. In future work we aim to integrate an augmented telepresence robot in a swarm, allowing interaction between a human operator and the multi-robot system in a complex, potentially dangerous surveillance task. The human operator would be able to steer the behaviour of the swarm from a remote location by means of direct communication.

[3] `http://smartlab.csc.liv.ac.uk/shepherding/`

References

Alers, S., Bloembergen, D., Claes, D., Fossel, J., Hennes, D., and Tuyls, K. (2013). Telepresence robots as a research platform for AI. In *Proc. of the AAAI Spring Symp. on Designing Intelligent Robots: Reintegrating AI II*, pages 2–3.

Alers, S., Claes, D., Tuyls, K., and Weiss, G. (2014a). Biologically inspired multi-robot foraging. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems (AAMAS)*, pages 1682–1684.

Alers, S., Tuyls, K., Ranjbar-Sahraei, B., Claes, D., and Weiss, G. (2014b). Insect-inspired robot co-

ordination: Foraging and coverage. In *Artificial Life 14*, pages 761–768.

Claes, D., Hennes, D., Tuyls, K., and Meeussen, W. (2012). Collision avoidance under bounded localization uncertainty. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 1192–1198.

Collett, M. and Collett, T. S. (2009). Local and global navigational coordinate systems in desert ants. *Journal of Experimental Biology*, 212(7):901–905.

Di Caro, G., Ducatelle, F., and Gambardella, L. (2005). Swarm intelligence for routing in mobile ad hoc networks. In *Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE*, pages 76–83.

Dorigo, M., Birattari, M., and Sttzle, T. (2006). Ant colony optimization – artificial ants as a computational intelligence technique. *IEEE COMPUT. INTELL. MAG*, 1:28–39.

Dressler, F. and Akan, O. B. (2010). A survey on bio-inspired networking. *Computer Networks*, 54(6):881 – 900. New Network Paradigms.

Floreano, D. and Mattiussi, C. (2008). *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies*. The MIT Press.

Folgado, E., Rincón, M., Álvarez, J. R., and Mira, J. (2007). *Nature Inspired Problem-Solving Methods in Knowledge Engineering: Second International Work-Conference on the Interplay Between Natural and Artificial Computation, IWINAC 2007, La Manga del Mar Menor, Spain, June 18-21, 2007, Proceedings, Part II*, chapter A Multi-robot Surveillance System Simulated in Gazebo, pages 202–211. Springer Berlin Heidelberg, Berlin, Heidelberg.

Gerkey, B. P., Vaughan, R. T., and Howard, A. (2003). The player/stage project: Tools for multi-robot and distributed sensor systems. In *In Proceedings of the 11th International Conference on Advanced Robotics*, pages 317–323.

Kuorilehto, M., Hännikäinen, M., and Hämäläinen, T. D. (2005). A survey of application distribution in wireless sensor networks. *EURASIP J. Wirel. Commun. Netw.*, 2005(5):774–788.

Lazewatsky, D. A. and Smart, W. D. (2011). An inexpensive robot platform for teleoperation and experimentation. In *Proceedings of ICRA 2011*.

Lemmens, N. (2011). *Bee-inspired Distributed Optimization*. SIKS dissertation series. Maastricht University.

Lemmens, N., De Jong, S., Tuyls, K., and Nowé, A. (2008). *Adaptive Agents and Multi-Agent Systems III. Adaptation and Multi-Agent Learning: 5th, 6th, and 7th European Symposium, ALAMAS 2005-2007 on Adaptive and Learning Agents and Multi-Agent Systems, Revised Selected Papers*, chapter Bee Behaviour in Multi-agent Systems, pages 145–156. Springer Berlin Heidelberg, Berlin, Heidelberg.

Lemmens, N. and Tuyls, K. (2012). Stigmergic landmark optimization. *Advances in Complex Systems*, 15(8).

Minsky, M. (1980). Telepresence. *Omni*, pages 45–51.

Quigley, M., Conley, K., Gerkey, B. P., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*.

Ranjbar-Sahraei, B., Weiss, G., and Nakisaee, A. (2012). Stigmergic coverage algorithm for multi-robot systems (demonstration). In van der Hoek, W., Padgham, L., Conitzer, V., and Winikoff, M., editors, *AAMAS*, pages 1497–1498. IFAAMAS.

Sheridan, T. B. (1989). Telerobotics. *Automatica*, 25(4):487–507.

Takayama, L., Marder-Eppstein, E., Harris, H., and Beer, J. M. (2011). Assisted driving of a mobile remote presence system: System design and controlled user evaluation. In *ICRA*, pages 1883–1889. IEEE.

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic robotics*. MIT press Cambridge.

Tsui, K. M., Desai, M., Yanco, H. A., and Uhlik, C. (2011). Exploring use cases for telepresence robots. In *Proceedings of the 6th International Conference on Human-robot Interaction*, HRI '11, pages 11–18, New York, NY, USA. ACM.

Vaughan, R. (2008). Massively multi-robot simulation in stage. *Swarm Intelligence*, 2(2):189–208.

von Frisch, K. (1967). *The dance language and orientation of bees*. Belknap Press of Harvard University Press.

Wagner, I. A., Lindenbaum, M., and Bruckstein, A. M. (1999). Distributed covering by ant-robots using evaporating traces. *IEEE T. Robotics and Automation*, 15(5):918–933.