Biologically Inspired Multi-Robot Foraging (Demonstration)

Sjriek Alers[1], Daniel Claes[2], Karl Tuyls[2] and Gerhard Weiss[1]

[1] Maastricht University, P.O. Box 616, 6200MD Maastricht, The Netherlands
[2] University of Liverpool, Ashton Building, Liverpool L69 3BX, United Kingdom
{sjriek.alers, gerhard.weiss}@maastrichtuniversity.nl
{daniel.claes, k.tuyls}@liverpool.ac.uk

ABSTRACT

In this demonstration we illustrate the direct usage of the principles of bee-inspired coordination in swarm intelligence on multi-robot systems. We present the first results of this implementation, where a subset of the bee algorithms are implemented on multiple turtlebot robots with the goal to simulate a food foraging application. For this we implemented means for locally detecting the location, speed and direction of the other robots using visual markers, applying collision avoidance algorithms and simulating local communication over wi-fi.

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics

General Terms

Algorithms, Experimentation

Keywords

Robotics, Foraging, Swarm Intelligence

1. INTRODUCTION AND MOTIVATION

Over the recent years we have seen an increasing interest in taking inspiration from natural phenomena for solving computational problems in disciplines such as computer science and robotics. One of these interesting phenomena in nature is the behaviour that can be observed in colonies of social insects such as ants and bees. 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 [7, 4, 5]. These insects have evolved over a long period of time and display remarkable behaviours that are highly suitable for addressing the complex tasks that they are facing. Swarm optimisation algorithms, like ant colony optimisation [3], rely on pheromone trails to mediate (indirect) communication between agents. Pheromone need to be deposited by agents, sensed by agents and decay while time passes by. Though easily to simulate, artificial pheromone is hard to bring into real-life robotic applications. However, recently non-pheromone-based algorithms where developed [6]. Such algorithms are inspired by the foraging and nest-site selection behaviour of (mainly) bees. In general, bees explore the environment in search for good 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. In this demonstration, we aim to implement a subset of the bee algorithms, i.e. the navigation principle, together with local communication throughout the environment on multiple turtlebots to simulate a food foraging application.

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*

2. IMPACT

Swarm robotics can be deployed in a wide range of application domains, such as in the security sector, where mobile guarding robots are considered as an alternative and improvement over fixed security cameras and even humans. In other applications as for instance exploration and identification of hazardous environments (e.g. nuclear plants and fire detection), a swarm of robots could be used as a mobile sensor network, and as such take care of any type of contemporary monitoring or exploration challenge (e.g. in space). By directly transferring the main principles of bee-inspired algorithms in swarm intelligence to swarm robotics we want to achieve a simulation of a food foraging application. This means that multiple robots with limited sensing and computing power randomly explore an unknown environment until a food location is found and start foraging. By using local communication, the robots can ask other robots they encounter for the vector to the closest food location.

3. SYSTEM

In previous work [1], we started implementing these algorithms on e-puck robots. However, due to hardware limitations and memory constraints we were forced to continue and re-implement this work on more capable robots with better sensors. For this we use the turtlebot platform, which is equipped with a laptop with a core-i3 CPU for computation that is running the Robot Operating System framework. In the rest of this section, we briefly describe the key implementations that are needed to detect and avoid other robots and to communicate with each other locally over a global wifi network.

3.1 Sensors

As a main sensing unit the robots are equipped with a Kinect sensor. The full RGBD information is used to detect and locate AR markers. For static obstacle detection, we only use the depth information of the sensor together with three bumpers that are located in front half of the robot. Furthermore, the robot estimates its position by integrating the wheel odometry and gyro information. Hence, no map of the environment is built and the only known reference point is the hive location marker. This can lead to the problem that if the odometry is faulty, the robot does not always find the hive or food location back. As a solution the robots fall back into a search mode, if this is the case. Another solution could be to implement a Northstar like navigation system, by providing a fixed frame of reference which is almost always visible from any location.

3.2 Marker detection

To enable visual robot-robot detection we equipped every Turtlebot with six unique markers, which are oriented in a way that at least one marker is visible from any angle. To track and decode these markers we make use of a toolkit called ALVAR, more specific we use the ROS wrapper[1] of this library. We use a customised bundle detection method to determine the center of the detected robot. Each marker in the bundle encodes its location with respect to the center of the robot and the robot number. This information is used to predict the detected robot's position. Kalman filtering is applied to get better and more stable readings and so a more accurate estimate of the detected robots position, heading and speed. These parameters are used again for collision avoidance.

3.3 (Local) Communication

Communication is realised over wi-fi with a UDP connection to each turtlebot using the LCM library[2]. Even though global communication would be possible, we limit the communication, such that every robot listens only to its own channel. To simulate local communication, the robots can only communicate with another robot when it is in view and in close proximity, i.e. less than one meter away.

3.4 Collision Avoidance

In order to avoid robot to robot collisions, we rely on the marker detection to predict positions and speeds of the other robots. This informations can be used to efficiently compute a non-colliding speed vector as we have developed previously [2]. In contrast to this previous approach, in which the robot-robot detection was avoided by using a global reference frame and broadcasting the positions to all robots via wi-fi, solely the marker detection and the predictions using a Kalman filter are used. This means that a few collisions still might occur due to failure to detect the markers of the other robots and additionally, there are certain configurations in which the robots cannot see each other due to the field of view of the Kinect sensor, e.g. when two robots drive in a V-shape towards each other, the field of view of the Kinect is too narrow to detect the other robot.

4. DEMONSTRATION

In this demonstration, we show multiple Turtlebots performing a foraging task, i.e. starting at the hive (H) location and randomly exploring the unknown environment for a specific food (F) location, as shown in Figure 1. An other way of locating a food location is by asking bypassing robots for a known food location, which is done by simulating local communicating over wifi. When the source is found the robot starts to exploit this source, i.e. driving from the food to the hive location until the food is depleted or a better source is found. A video showing this demonstration can be found in the online material[3].

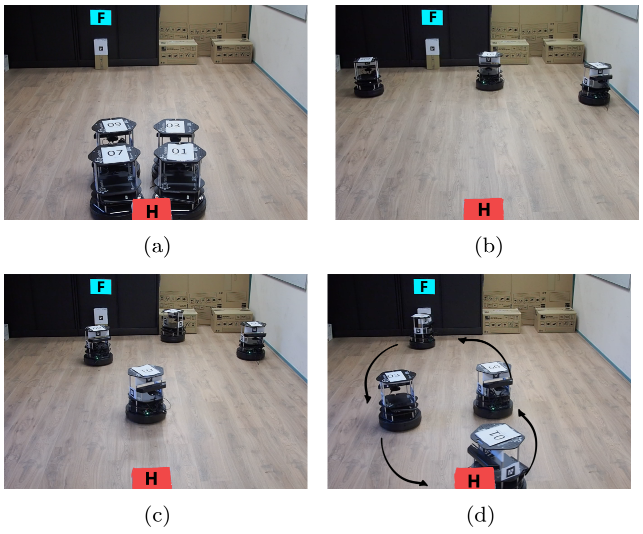

Figure 1: (a) All robots start at the hive (H) location. (b) Robots are exploring the unknown environment randomly. (c) The left two robots have found the food (F) location and are foraging between the hive and the food location. (d) All robots have converged to foraging.

5. REFERENCES

[1] S. Alers, D. Bloembergen, D. Hennes, S. de Jong, M. Kaisers, N. Lemmens, K. Tuyls, and G. Weiss. Bee-inspired foraging in an embodied swarm. In *Proceedings of the Tenth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1311–1312, 2011.

[2] D. Claes, D. Hennes, K. Tuyls, and W. Meeussen. Collision avoidance under bounded localization uncertainty. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012)*, Vilamoura, Portugal, October 2012.

[3] M. Dorigo, M. Birattari, and T. Stutzle. Ant colony optimization: Artificial ants as a computational intelligence technique. *Computational Intelligence Magazine, IEEE*, 1(4):28–39, 2006.

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

[5] D. Floreano and C. Mattiussi. *Bio-inspired artificial intelligence: theories, methods, and technologies*. The MIT Press, 2008.

[6] N. Lemmens. *Bee-inspired Distributed Optimization.* Maastricht University, 2011.

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

[1] `http://wiki.ros.org/ar_track_alvar`
[2] `https://code.google.com/p/lcm/`

[3] `http://swarmlab.unimaas.nl/papers/aamas-2014-foraging`