Dispersion Algorithms for Swarm Robots Navigation

***Abstract*—A robot swarm is a multi-robot system which consists of a large number of small and relatively simple individual units. This approach is based on the idea that relatively simple and primitive individual behaviours will produce a set of complex emergent swarm behaviours. These systems are scalable by using local inter-robot communications. Most of the applications for robot swarms must perform a previous important task: disperse efficiently the robots in an unknown environment. This document summarizes several swarm robotic platforms developed in the last years and several algorithms from different authors to achieve the dispersion goal.**

I. Introduction

The most desirable applications for robots are jobs that are dangerous, dirty, or dull. Many of these jobs lend themselves to being performed by groups of robots working together rather than by single robots working alone. Some tasks can achieve efficiency gains as a direct function of the number of robots applied. The use of a large number of robots increase the fault tolerance and leads to the synergy of an emergent behaviour. Other applications can benefit even more, as radically different techniques can be employed to solve a problem with ten thousand robots than with ten. As robots become more common in everyday's life, the shift to multirobot systems will become the rule, rather than the exception. Swarm robotics is a relatively new field that focuses on controlling these large-scale homogeneous multirobot systems in such applications like vacuum cleaning, earthquake rescues or exploration and mapping of new areas.

The algorithms designed for these systems are based on the idea that complex behaviours can emerge from simple local interactions between agents. Robotic swarms have several advantages over other more complex individual robots and are the results of using many robots instead of just one [1]:

- Robot swarms are able to cover a larger area than an individual robot.
- Swarm robots are fault tolerant because the algorithms designed for them don't need robots to depend on one another.
- Robot swarms usually increase their effectiveness with the number of members.

A great variety of algorithms have been implemented to be run on robotic swarms. All of them produce different emergent behaviour but its main features include [1]:

- Simple and elegant. The robot controller that dictates the behaviour of the individual robots is very simple and is usually represented as a state machine with few states and edges.
- Scalable. Algorithms are designed for any number of robots and scale well as new robots are added to the swarm.
- Decentralized. The robots in a swarm are autonomous and do not follow any exterior commands. Although a member of a swarm can be influenced by the behaviour of another, the choice is under its own accord. Being scalable and decentralized are often related features.
- Usage of local interactions. Local interactions are used over broadcasting messages in the majority of these algorithms. The scalability of the system depends greatly on this concept.

The main goal in almost every application for robotic swarms typically involves two different phases [2]:

1) The swarm fills the environment as quickly as possible.
2) The robots perform some computational tasks.

This document presents some algorithms designed to perform the dispersion task included in the first phase previously presented. These algorithms have different approaches and strategies to solve the dispersion problem and describe the path to follow in this context.

This paper is structured as follows: in the second section a brief overview of several robotic swarm platforms is presented, in the third section some algorithms with different disperssion approaches are described, and finally, obtained conclusions from this work are presented in the fourth section.

II. Robot swarm systems

Multiple swarm robotics platforms have been designed in the last years to develop different types of algorithms and behaviours. In this section a description of three robotic swarms is presented which are different in terms of structure, motion, sensing and comunication design. This is a brief sample of the great variety of architectures for robot swarms developed all around the world.

A. i-Robot

This platform is the world's largest swarm, with over one hundred individual robots at the Massachusetts Institute of Technology, and has been used for a multitude of projects and experiments. The goal of the project is to develop distributed algorithms for robotic swarms composed of hundreds of individual robots [3].

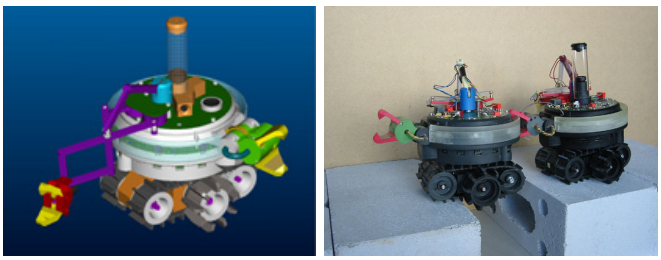
An individual i-Robot unit packs a comprehensive sensor suite, a high-performance 32-bit microprocessor, in the form of a 5 inch cube (see figure 1). Each robot uses the ISIS infrared

Fig. 1. Individual i-Robot unit

communication system that provides obstacle detection, localization and communications at 125 kbps [4]. Robots that are close to each other are able to communicate and to determine the locations of each other. Messages are dispersed through the entire network topology dynamically formed by all the robots in the swarm using a multihop protocol. Most communication protocols and basic obstacle avoidance are handled by the underlying system so researchers can use this platform to focus on emergent behaviour.

B. Swarm-bots

Swarm-bots was a project completed on 2005 and was sponsored by the Future and Emerging Technologies program of the European Community aimed to study new approaches to the design and implementation of self-organizing and self-assembling artifacts [5][6]. A swarm-bot consists in the physical construction of at least one s-bot. Each s-bot is a fully autonomous mobile robot capable of performing basic tasks such as autonomous navigation, perception of its surrounding environment and grasping of objects. A s-bot is also able to communicate with other individual units and physically join either rigidly or flexibly to them, thus forming a swarm-bot. The s-bot design is shown in figure 2a and a configuration to pass a large gap is shown in figure 2b.



(a) (b)

Fig. 2. (a) S-bot design. (b) Swarm-bot configuration to pass a large gap.

Its main features are its compact size (116 mm in diameter of the main body and 100 mm in height), all-terrain mobility, rigid and flexible arms, several sensors as accelerometers, force, temperature, torque and light sensors and an omnidirectional camera.

S-bots don't use the typical radio communication system, they communicate with each other in a very simple way in terms of the colours of their light sensors. The s-bot behaviour is related with the measures of colours and distance to other s-bots obtained from the images captured by its camera.

C. Scout

In [7] a set of small robotic systems called scouts is presented. As shown in figure 3, the scout is a cylindrical robot 11.5 cm in length and 4 cm in diameter that can use their wheels to travel on two ways. The scouts can transmit video from a small camera to a remote source for processing and can also transmit and receive digital commands over a separate communications link that uses an ad hoc packetized communications protocol. By interleaving packets destined for different robots with different and unique ID, multiple scouts can be controlled simultaneously. Actuator, sensor and communication controls are handled by two onboard microprocessors.

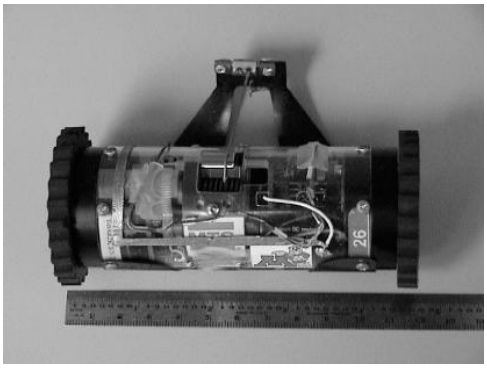


Fig. 3. Scout robot

III. Dispersion algorithms for robot swarms

In this section different strategies to perform the dispersion task for a robot swarm are presented.

A. Rapid dispersion

Hsiang et al. present in [2] several algorithms for dispersing robot swarms in an unknown environment, where the primitive objective is to minimize the time to fill the entire region. An arbitrary region is formed of connected pixels, where a subset of these pixels is formed by doors, which serve as sources of incoming robots (see figure 4). Robots move discretely on the grid of pixels, so if a time t a robot occupies a pixel (i,j) of region R and a neighbouring, for example, $(i+1,j)$ of R is unoccupied at time t, then the robot may take a step and occupy that pixel at time $t+1$. Obviously, two robots can't move to the same pixel, so a priority among the four possible directions is established.

Two algorithms are given for a single door case based on leader-follow strategies adapted from depth-first and bread-first search to apply to robot swarms.

1) Depth-First Leader-Follower (DFLF): This strategy is inspired by depth-first search in a graph. At any given time *t* there is exactly one leader *r* which is on a pixel and is looking for a frontier pixel, one that has never been occupied by a robot, and selects it as its next destination. If the leader has no frontier pixel next to it, it stops and tells its successor to assume the leadership. If the leader has no successors, then the algorithm halts. Any robot that is a follower (not the leader) simply follows its predecessor. It's important to note that at any point in time there's only one leader and that once a robot stops, it never moves again.

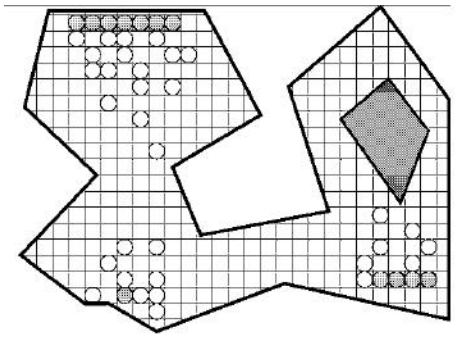


Fig. 4. Region where special door pixels are shown in grey

2) Breadth-First Leader-Follower (BFLF): This algorithm is based on the breadth-first leader-follower alternative strategy and, while still being optimal in terms of makespan just like the previous one, often has advantages in terms of other metrics of performance as total travel of robots, maximum travel of any robot, total relative distance, etc.

In this strategy there can be multiple leaders and it has been introduced a new robot state. As before, a robot can be in a moving state or a stopped state, but now the robot can pause temporarily and wait to be able to move. This is called a waiting state. Once again, if the robot stops, it never moves again.

Initially there's only one leader *r*, the robot at the door. When another robot gets at the door, it chooses to follow the previous robot that left the door. A leader always tries to go to a neighbouring frontier pixel, but makes sure it does not stray far from its successor. If there are no neighbouring frontier pixels, the leader waits for its successor to arrive. When the immediate successor catches up, the leader stops and its immediate successor becomes the new leader. If the leader *r* at pixel *u* has several neighbour frontier pixels, it chooses any one of them and heads to it. If there are frontier pixels adjacent to *u* remaining, the follower *r'* of *r* will choose one of them as it heading when it arrives at *u*. If there remains a frontier pixel adjacent to *u*, then *r'*'s follower *r"* chooses this pixel as its heading when it arrives at *u*. This way, *r'* and *r"* becomes relabeled leader and pixel *u* becomes a branching point.

This strategy tries to create as many paths as possible at all times, so the visited pixels form a tree that guides the directions of the robots. At branching points, robots alternate the direction they travel so the flow gets balanced.

Both strategies achieve the goal of filling the entire region in a quick manner, but only if there are enough robots available.

B. Direct Dispersion

In [8] algorithms for dispersing large swarm of robots into an enclosed space are presented where only inter-robot communication and processing is used. This algorithms have been successfully tested on the i-Robot Swarm platform.

The Directed Dispersion algorithm tries to spread the robots throughout the enclosed region in a quick and uniform manner, while keeping each robot connected to the communication network. The behaviour is accomplished by using two algorithms that alternate running on the swarm: Disperse Uniformly and Frontier Guided Dispersion. The Disperse Uniformly algorithm spreads robots evenly, using boundary conditions to limit the dispersion. The Frontier Guided Dispersion algorithm spreads robots towards unexplored areas and is designed to perform well both in open and constricted environments.

1) Disperse Uniformly: Physical walls and a maximum dispersion distance between any two robots of r_{safe} are used as boundary conditions to prevent the swarm from spreading too thin and fracturing into multiple disconnected components. r_{safe} is the maximum distance that provides reliable positioning when using the ISIS inter-robot infrared communication system installed on every i-Robot.

Each robot moves away from the vector sum of the positions of their *c* closest neighbors. The magnitude of the velocity vector that is given to the motor controller of each robot is such that it will tend to expand the swarm to fill the space available in the region. Once this space is occupied, robots will position themselves so the energy is minimum.

2) Frontier Guided Dispersion: This algorithm guides the swarm towards new areas using the robots that are on the frontier of explored space. Robots must identify themselves as occupying one of three positions in the network using several thresholds in their signal measurements: wall, frontier or interior. Once they know their positions, the frontier robots source a gradient message so a tree is formed such that it guides the swarm towards the frontier robots. To avoid that newly discovered frontiers pull robots away from previously explored areas, robots move away from children in the frontier tree. Thus, a reliable network is built because robots only move if they are in contact with at least two children in the frontier tree. Progress of distant frontiers is slowed as interior robots disperse towards the frontiers because the gradients of the messages are based on hop count.

Directed Dispersion combines the two previous algorithms, so if there are any frontier gradients in the network, the swarm runs the Frontier Guided Dispersion algorithm. Otherwise, it runs Disperse Uniformly to equalize inter-robot spacing. Disperse Uniformly tends to push robots into open spaces and tight constrictions, which can cause new frontiers to form. This activates the Frontier Guided Dispersion behavior on the rest of the swarm, which causes a directed dispersion towards the frontiers.

These algorithms also achieve the dispersion goal of filling an area in a quick manner as the rapid dispersion approach, but the number of robots used can be reduced thanks to the communication network involved.

C. Dispersion algorithms for Scouts

In [9] four reactive dispersion algorithms for the scout platform described in the second section are presented. These methods are close to those in [2], but now it is assumed that there are not enough robots to cover the entire region and to guarantee that every robot can remain within sensor range of other robots. Furthermore, these algorithms are constrained not to require explicit communications, they only use their sensors to communicate implicitly by observing the environment. This type of communication is very used in swarm robotics and is called *stigmergy* [10].

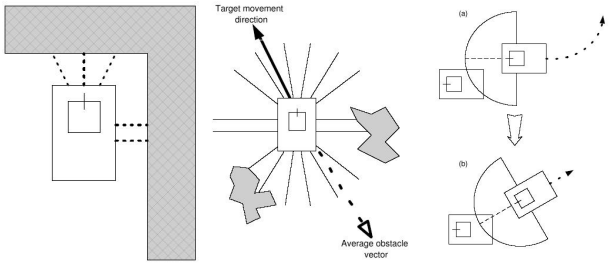


Fig. 5. Left: Follow Wall algorithm. Center: Seek Open algorithm. Right: Fiducial Movement algorithm. (a) A robot detects another one behind in and turns so the detected robot will be immediatly behind. (b) The robot has positioned the detected robot behind it so continues straight forward.

1) Random Walk: A robot using the Random Walk algorithm can be in random forward movement or obstacles avoidance states. In random forward movement, the robot moves forward with a small random turn factor changed at random intervals, ensuring the robot doesn't end up running in circles. Once the robot detects an obstacle, it enters the obstacle avoidance state, where the robot turns around a random amount and goes back to the random forward movement state.

2) Follow Wall: A robot running this algorithm will search for an obstacle (presumably a wall) and follow it indefinitely. The robot can be in four different states: find wall, align to wall, follow wall and navigate corner (see figure 5 left). If the robot loses the obstacle in any of the three non-find-wall states, it will go back to the initial find wall state and search for a new one to follow. The major problem is that it is assumed that every obstacle is a wall. Thus, when many robots are together, they will tend to perceive each other as walls and try to align themselves to each other, which is not an effective strategy to spread the robots throughout the environment.

3) Seek Open: The goal of the Seek Open algorithm is to motivate the robots to disperse as quickly as possible. The average obstacle vector is calculated for all the obstacles in sensor range such that its magnitude is large for objects close to the robot and small for objects far away. After this vector is computed, the goal of the robot is to move in the opposite direction of the average obstacle vector. This allows the robot not to run near to walls and disperse from other robots (see figure 5 center).

4) Fiducial Movement: This algorithm needs a fidual device in every robot to find beacons that are attached to all of them. This allows a given robot to know the polar coordinates of other robots respect to its own position. Whenever a robot detects another robot within sensor range, it adjusts its movement so that it is moving away from the detected robot. When no robots are in sensor range, a robot simply moves according to the random walk algorithm. If at any time a robot encounters a physical obstacle such as a wall, the obstacle avoidance technique takes precedence over whatever movement algorithm the robot is currently executing (see figure 5 right).

The reactive characteristic of these algorithms is an advantage in terms of speed, but it could lead the swarm to fill the region in a not uniform manner.

IV. Conclusion

Swarm robotics is a novel approach to the coordination of large numbers of robots designed such that a desired collective behaviour emerges from individual ones. This paper describes briefly three swarm robotics platform developed in the last years and three different approaches to achieve the typical dispersion task of this kind of systems. Rapid Dispersion seems a quick manner to fill a entire region but only if there are enough individual robots available. Direct Dispersion achieves the same goal for a fewer number of robots and guarantees they're all connected to the communications network. The dispersion algorithms for Scouts are reactive, robots don't need information from each other, so they're faster, but could disperse themselves in a not uniform way.

References

[1] D. Miner: Swarm Robotics Algorithms: A Survey. Technical Report, University of Maryland, 2007.

[2] T. R. Hsiang, E. Arkin, M. A. Bender, S. Fekete, and J. Mitchell: Algorithms for rapidly dispersing robot swarms in unknown environments. In *Proc. 5th Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2002.

[3] iRobot Corporation official website. http://www.irobot.com

[4] M. Schwager, J. McLurkin and D. Rus, Distributed Coverage Control with Sensory Feedback for Networked Robots. In *Proceedings of Robotics: Science and Systems*, 2006.

[5] F. Mondada, L. M. Gambardella, D. Floreano, S. Nolfi, J.-L. Deneubourg, and M. Dorigo: SWARM-BOTS: Physical interactions in collective robotics. *IEEE Robot. Autom. Mag.*, vol. 12, no. 2, pp. 21−28, 2005.

[6] G. Pettinaro, I. Kwee, L. Gambardella, F. Mondada, D. Floreano, S. Nolfi, J. Deneubourg, and M. Dorigo: Swarm Robotics: A Different Approach to Service Robotics. In *Proceedings of the 33rd ISR (International Symposium on Robotics)*, October 7-11, 2002.

[7] P. E. Rybski, S. A. Stoeter, M. Gini, D. F. Hougen, and N. Papanikolopoulos: Performance of a distributed robotic system using shared communications channels. *IEEE Trans. on Robotics and Automation*, 22(5):713−727, October 2002.

[8] J. McLurkin and J. Smith: Distributed Algorithms for Dispersion in Indoor Environments using a Swarm of Autonomous Mobile Robots. *7th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, 2004.

[9] R. Morlok and M. Gini: Dispersing robots in an unknown environment. In *Proceedings of the 7th International Symposium on Distributed Autonomous Robotic Systems*, Toulouse, France, June 2004, Springer-Verlag.

[10] E. Bonabeau, M. Dorigo, and G. Theraulaz: *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press, 1999.