Sensors & Transducers

ISSN 1726-5479
© 2009 by IFSA
http://www.sensorsportal.com

Movement in Collaborative Robotic Environments Based on the Fish Shoal Emergent Patterns

[1]Razvan CIOARGA, [1]Mihai V. MICEA, [1]Vladimir CRETU, [2]Emil M. PETRIU
[1]Department of Computer and Software Engineering (DCSE),
"Politehnica" University of Timisoara, 2, Vasile Parvan Blvd., 300223 – Timisoara, Romania
[2]School of Information Technology and Engineering, University of Ottawa,
800, King Edward, Ottawa, KIN 6N5, Canada
E-mail: razvan.cioarga@cs.upt.ro, mihai.micea@cs.upt.ro, vladimir.cretu@cs.upt.ro,
petriu@site.uottawa.ca

Received: 30 January 2008 /Accepted: 20 February /Published: 23 March 2009

Abstract: Robotic collectives are used for the efficient achievement of complex tasks. There is a significant increase in the interest for emergent, collaborative robotics as a viable alternative to the more centralized classic approach as the dimensions, energy consumption and especially price are becoming required constraints. This paper describes a nature inspired algorithm intended for the movement and communication of such robotic collectives. As a case study, the implementation of the emergent algorithm on a system consisting of LEGO Mindstorm Robots is further discussed along with the most interesting experimental results. *Copyright © 2009 IFSA.*

Keywords: Collective robotics, Emergent behavior, Nature inspired movement

1. Introduction

The evolution of information processing equipment and the need of adaptable digital control systems generate an extraordinary spreading of digital equipment in all fields of life including military applications, service industries, space exploration, agriculture, mining, factory automation, health care, waste management, disaster intervention and domestic applications. Most of such equipment present in diverse human activities is required to be more or less autonomous, meaning that some degree of intelligence must be embedded into it.

The new trend of miniaturization of electronic devices leads to the necessity of small, relatively independent entities which can interact with each other. Therefore, the embedded autonomous equipment must scale well and have a low cost.

Embedded systems [1], robotic systems [2], sensors and digital perception equipments (sensor networks) [3, 4], digital signal and image acquisition and processing (DSP-based and digital image processing systems) [5] are major domains of interest in the present activities of the academic and industrial community. This fact is proven by the huge number of applications and projects that are developed in these domains. The complexity of the applications involving embedded digital intelligence has grown over the last years, reaching the limit where the integration of inter-domain approaches is required. This leads to the necessity of defining and implementing new, specific methodologies in new domains such as: collaborative robotic environments and intelligent sensor networks, distributed artificial perception, reconfigurable and auto-configurable systems, associative behavior patterns.

The employment of robots in numerous industrial and academic applications is common nowadays. Many of the applications of robots imply their movement towards a given goal.

The movement in collective robotics has always been a complex issue. It is usually addressed in two ways: individual navigation of a single entity which does not take into account the movement of the other entities in the system and tries to reach the given goal individually, and distributed navigation where the actions of the robots are known to each other and they collaborate in order to reach their goal.

As long as the robots involved are large enough to carry sufficient computing and sensing capabilities, movement can be designed using a very complex distributed system (the network of robots themselves). The robots involved with these algorithms are very important for the whole system, in that losing one in a dangerous environment may be disastrous for the system. Usually, because of the required constraints (price, dimensions and energy consumption) the robotic collectives are composed of relatively small robots, with little computing power which is not sufficient for most of the movement algorithms. This is a case well suited to apply movement algorithms which were inspired from the study of the natural world. Such emergent behavior patterns like the movement of bees and fish have proven to be extremely efficient when dealing with large number of small entities [6-9].

Applying emergent behavior patterns to robotic collectives [4, 9, 10] meets the constraints of power saving and price which are required in usual applications.

This paper focuses on an improved movement algorithm based on the movement of fish shoals [11-13]. In addition to [11], this paper presents the detailed movement algorithm and the mathematical formalisms involved. The algorithm has been implemented on a robotic collective composed of 6 entities which are able to communicate with each other using a protocol that is also described. A complete set of experiments have been conducted on the robotic collective using the fish shoal inspired movement algorithm and the results have been presented in this article.

1.1. Emergent Behavior Patterns

Emergence is the process by which complex behavior patterns are formed starting from extremely simple rules [4, 10]. The new field of emergent behavior is mainly based on the study of the natural world in order to retrieve simple natural algorithms that can be applied in other fields of interest. The simplicity of these patterns and the interaction between them can solve some of the most important problems in embedded systems such as: energy consumption, communication protocols, efficiency and reliability features, obstacle avoidance algorithms etc.

Emergent phenomena are usually unpredictable, representing non trivial results of simple interactions between simple entities. As a result, there are several combinatorial complexity problems for which finding an optimal solution is difficult or sometimes impractical because of the high dimensions of the search field or because of the computational requirements involved by the existing constraints. In order to solve these problems heuristic techniques are preferred, to generate solutions which are not always optimal [14]. The meta-heuristics used are optimizations based on ant colonies studies (Ant Colony Optimization [15]) and global convergence [16] in which case a great number of emergent behavior systems are exponentially converging to the required result, the convergence rate increasing with the number of entities involved. Another extremely important domain of applicability of emergent behavior patterns is represented by NASA's future spatial missions involving new exploration technologies based on swarm emergent behavior [17, 18].

Although these meta-heuristics are sustained by a significant mathematical basis, there are no formal descriptions of the emergent behavior patterns used, just some partial representations mostly seeking the global effect of the patterns. Also, the local predictable behavior can be formally described, but the interactions between these local behaviors which finally generate the emergence, cannot be quantified because of their great number. As well, the application of behavior patterns to different systems is not standardized. These patterns are usually unpredictable, complex and specific for each application.

1.2 Problems of Collaborative Robotic Environments and Intelligent Sensor Networks

The domains of collaborative environments and intelligent sensor networks are some of the research areas of great interest for international academic and industrial communities. A lot of problems from these domains, highlighted by the literature, remain open-ended. A synthesis of these problems derived by the authors includes the following items:

1) Development of a highly scalable, efficient and precise solution for the coordination of collaborative robotic and intelligent sensor systems;
2) Defining efficient strategies for decision making;
3) Efficient task distribution by exploiting and optimizing the system parallelism;
4) Although the potential and advantages of the emergent behavior patterns has been proven for a wide range of collaborative systems applications like environment exploration and monitoring, there is no real implementation of these patterns for coordination and decision making [17 – 18];
5) The implementation of emergent behavior in different applications is based on a set of empirical recipes without defining a rigorous mathematical basis [15, 16];
6) There is no formalization for describing the emergent behavior patterns [15, 16];
7) The services required by the efficiency of operation of collaborative systems: communication platform (infrastructure + protocol stack), localization solutions with minimal system resource requirements, maintaining a global system clock by using efficient synchronization techniques;
8) Lack of a representation model for collaborative systems, which integrates the infrastructure and coordination level along with the required support services, to design, analyze and evaluate the performance of a system in the context of a certain application.

This article presents an emergent algorithm used for movement in robotic collectives. This emergent algorithm implements a formal, mathematical model that is employed for the coordination of the movement in the robotic collective and requires small amounts of computing power.

2. Nature Inspired Movement in Collective Robotics

According to the model presented in [12, 13] and then later revised in [19], the fish shoal does not have a leader. Each group member moves by the same rules as the others. Also, the movements of the fish are influenced only by the nearest neighbors. Each fish can approximate the distance between itself and its neighbors by vision, and the neighbors' heading and speed by a natural sensor that fishes have, called lateral line that can sense the movement of water currents near them. From the point of view of a particular instance of the emergent algorithm, the fish hosting that algorithm is called "main fish", while all the other fishes are "neighbors". Obviously, each fish in the shoal is "main fish" for the corresponding instance of the algorithm which it hosts.

According to [12, 13], each fish is surrounded by four distinct zones ($z_1 - z_4$), as shown in Fig. 1. Depending on the relative position of the neighboring fish to the main fish, there are four cases as follows:

1) If a neighboring fish is situated in the closest circle around the main fish (Fig. 1a), then their next move will be a repulsion move because they are too close and there is the possibility of collision.
2) If the neighboring fish is situated in the second circle (Fig. 1b), then the distance between it and the main fish is considered to be optimal; the main fish will follow its neighbor's direction describing a parallel direction.
3) In the third case, the neighboring fish is inside the furthermost circle (Fig. 1c), the main fish, which calculates the next move, will try to move according to the heading of its neighbor.
4) When the neighboring fish is located outside the furthermost circle, then it will not be taken into consideration (as in real life it can't be seen).

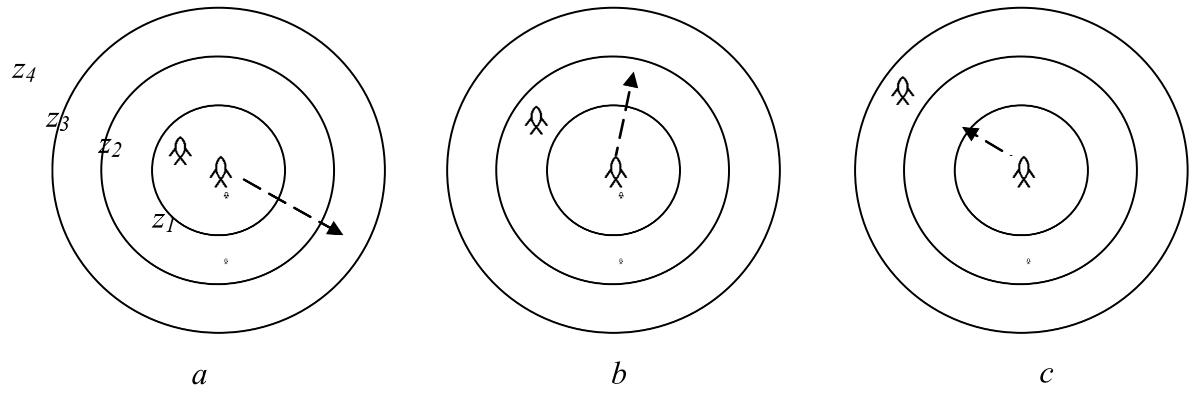

Fig. 1. Basic Fish Shoal Movement.

When there are no fish in its proximity, the main fish will adopt a searching strategy to find the lost group. When one fish has more neighbors it can either take into consideration only one of them (based on the distance and the weight factor given from the beginning to each fish) [12, 13], or take into consideration all of its neighbors [19].

3. Improved Fish Shoal Movement in Collective Robotics

The above cited papers [12, 13, 19] present only three possible influences, which are induced by the neighboring entity (attraction, repulsion and parallel movement). These generate a limited range of motion (parallel to the direction of movement of the neighboring entity in the case of the parallel movement influence or a full turn around in the case of repulsion). In order to give the entities of the robotic collectives a wider range of motions the algorithm presented in this paper introduces a larger range of influences instead of the only three named above. From attraction to repulsion, the robot can move at any orientation angle which can be induced by a neighbor, depending on the distance between them. If the distance is smaller than a given minimum value, which means that a collision is about to happen, then the two entities form a total repulsion system, moving away from each other in opposite direction; if the distance between the entities is equal to a maximum value then the two entities will completely attract each other. If the distance between them is between the constraint values, the angle between the new directions of movement of the fish will differ accordingly. If the distance is bigger than the maximum value then the two entities will not influence each other at all.

An improvement of this situation is presented in [11]: when the main entity is influenced by more than one entity, it calculates a weight distributed mean of all the influences. In order to establish the new position for a fish the algorithm calculates an angle of movement and a distance. The angle gives the new orientation of the fish and the distance gives the length of the movement. To calculate the new position we use a movement vector. This vector is calculated as a sum of the vectors that describe the movement for every neighbor fish. For every fish, the vector is represented by an angle and a distance relative to initial state. When a fish calculates the length of the movement it takes into consideration the distance between it and the neighbors; the further the neighbors, the shorter the movement.

In the opposite case, when the fish is closer to the neighbor, the distance is larger. This consideration has been made so that at any given time the repulsion is more important than the attraction in order to avoid collisions.

For the algorithm to work properly, a known initial state, shown in Fig. 2, is necessary to position the whole system. The algorithm is implemented on a robot collective; the implementation is presented later in this paper. Each fish has been implemented using a robot which has movement and communication capabilities.

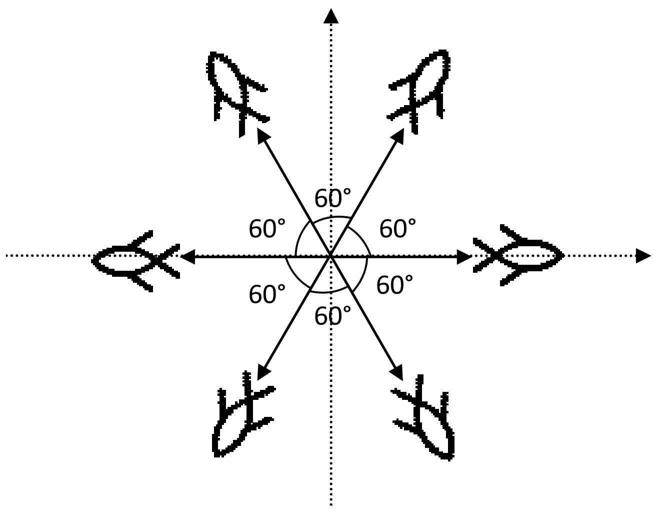

Fig. 2. Initial State.

The entities are placed in a circular shape. They are 19 centimeters apart from the center of the system; they are equidistant and heading in the opposite direction from the center, thus making an equal angle between each two consecutive robots (in Fig. 2 there are 6 robots so the angle is exactly 60°). After the robots are set in position each of them moves forward with a random number of centimeters, after which they stop and compute their coordinates. These will be considered the initial coordinates. There is also a final state that the fish should reach. In real life, this goal can be represented by food or by a safe place to live.

At each step the robot will calculate the distance between it and its neighbors. Depending on the relation between the computed distance and the robot's view range, there are three possible cases as seen in Fig. 3:

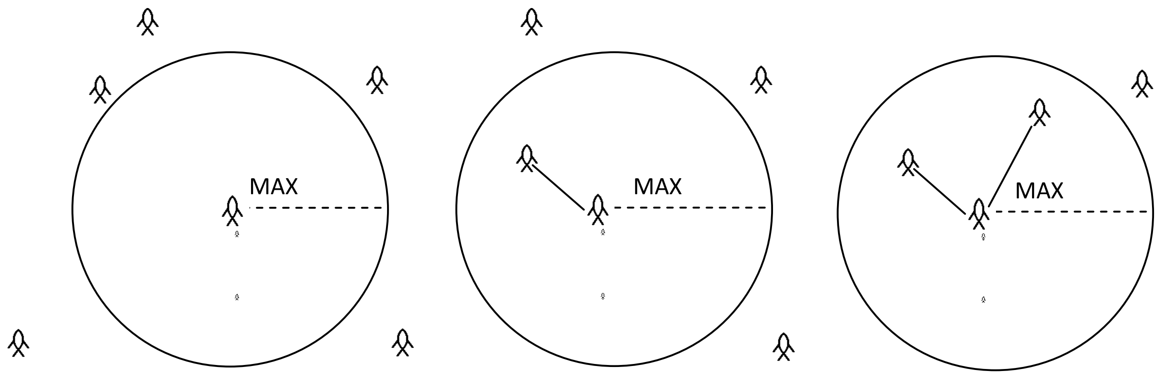

Fig. 3. The three possible cases of positioning.

If the robot has no neighbors in its vicinity, based on a maximum distance at which the robots can "see" each other, it will have to go in a random direction to find some other robots. The robot needs to calculate both the distance it has to cover and the angle of that direction. Because the movement has to be random in this case, the deviation of the angle will be no more than 30°, and the distance covered will be between *mD* (a minimum distance) and *MD* (a maximum distance).

$$Angle = Angle\_to\_target - current\_angle \pm 30° \tag{1}$$

$$Distance = random(0, MD - mD) + mD \tag{2}$$

If the robot has exactly one neighbor in its vicinity, it will move closer or it will move away based on the distance between them.

If the robot has more than one neighbor in its vicinity its movement will be dictated by the resultant of all the forces of attraction and rejection between those robots. Three possible situations can occur.

3.1. The Entity Has no Neighbors in its Proximity

In the case an entity has no neighbors in its proximity, the necessary strategy is to search the group; the robot knows the relative position of its neighbors (all of the robots communicate with each other, and broadcast the data relevant to their positions), it computes the coordinates of the center of the group and, then, it will move towards that center. The explanation retrieved from the natural world resides in the fact that even if a fish cannot distinguish all the other neighboring fish, it is able to sense the whole shoal of fish and can head towards it. Fig. 4 shows how the robot changes its orientation and starts moving towards the group.

In Fig. 4, α is the angle between the current orientation and the desired one, β is the angle between the horizontal axis (*Ox*) and the current direction of the fish (known at each step) and γ is the angle between the horizontal axis and the fish-target direction.

To make the move the movements of the robots using this algorithm seem more natural, we introduced a random angle of ±30°. This angle is added to the angle calculated for fish-target direction.

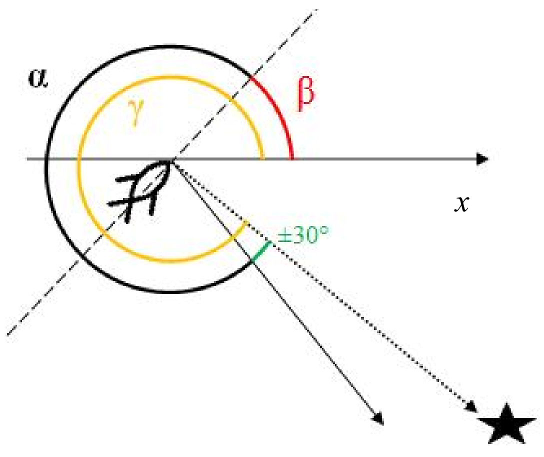

Fig. 4. Movement towards the group of a single entity.

The slope of the fish-target direction without our correction is given by the next formula:

$$m_R = \frac{R_G.y - R_R.y}{R_G.x - R_R.x}, \tag{3}$$

where $R_G=(x, y)$ is the position of the center of the group and $R_R=(x, y)$ is the position of the robot.

The γ angle can be computed with the formula:

$$\gamma = \begin{cases} \tan^{-1}(m_R) & \text{for the first quadrant} \\ \tan^{-1}(m_R) + \pi & \text{for the second or third quadrant} \\ \tan^{-1}(m_R) + 2\pi & \text{for the forth quadrant} \\ \frac{\pi}{2} & \text{if the slope is } -\infty \\ \frac{3\pi}{2} & \text{if the slope is } \infty \end{cases} \tag{4}$$

In the end, the final turn angle (α) is calculated as:

$$\alpha = \gamma - \beta \pm 30° \tag{5}$$

3.2. The Entity Has Exactly One Neighbor in its Proximity

In the case when an entity has exactly one neighbor in its proximity, the influence that appears between the two entities ranges from repulsion to attraction, depending on the distance between them. If this distance is maximum, the fish will attract each other, and if the distance is minimum, theoretically 0, the fish will reject one another. In order to create a general algorithm, we introduced a function that takes as input values the possible distances between the two fish, and in return it gives the final angle that the main fish should follow (θ). This angle is relative to the direction of the fish-neighbor pair and its values can be positive or negative taking into consideration the position of the goal. The final direction of the movement will bring the fish closer to the goal. Fig. 5 shows three possible situations that can arise in this case.

The θ angle corresponding to the chosen influence (from repulsion to attraction) is calculated with the formula:

$$\theta = \frac{180^\circ}{DMAX} \times d\,, \tag{6}$$

where *DMAX* is the constraint defined by the view range and d is the distance between the two entities.

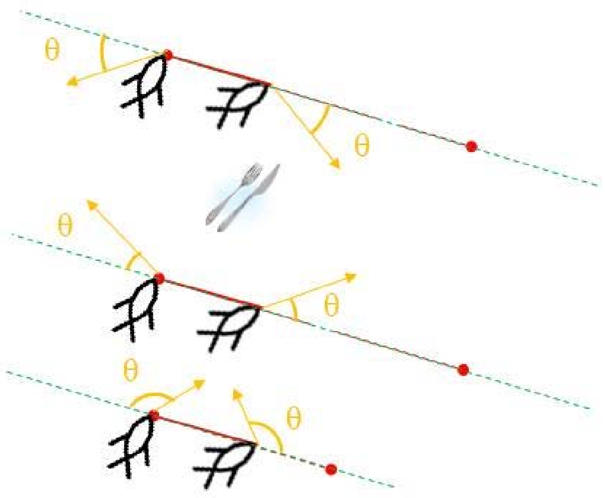

Fig. 5. Relative positioning of the goal and the chosen directions.

Fig. 6 and Fig. 7 present a repulsion case, from the perspective of a pair of fish.

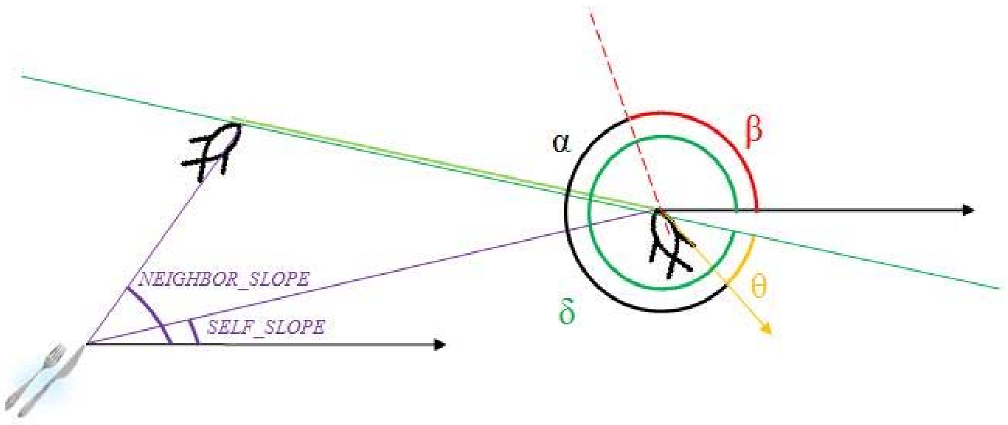

Fig. 6. Repulsion case seen by the right fish.

The α angle and the β angle were explained in the first section of this chapter, and θ was defined in the last paragraph. To calculate the new movement angle, we still need the angle between the horizontal axis and the fish-neighbor direction. This is represented by the δ angle, and its value is determined like the θ in the first section of the chapter, replacing the center of the group with the neighbor. The final angle is given by the next formula:

$$\alpha = \delta - \theta - \beta \tag{7}$$

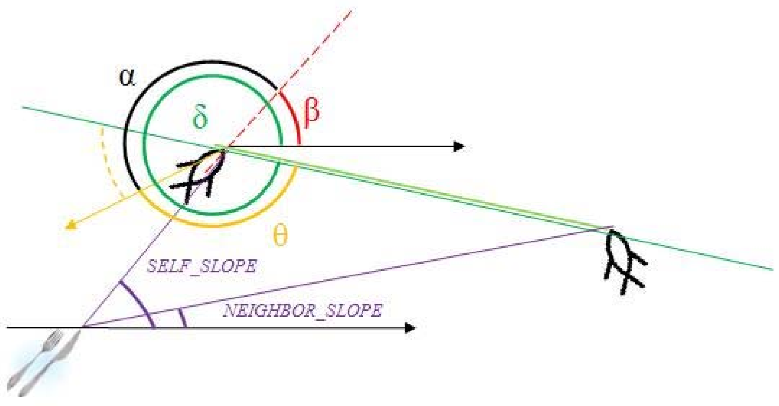

Fig. 7. Repulsion case seen by the left fish.

For the left fish, for the generality of the formula (7), instead of the θ angle, we used its complementary angle, as seen in Fig. 7.

To determine which the left one is, the algorithm calculates the slope formed from the target to each fish. The fish with the biggest slope is considered to be the left one.

3.3. The Entity Has more than One Neighbor in its Proximity

In this general situation, the main entity will compute a resultant vector based on all the vectors computed for each of the neighboring entities.

The robot computes the coordinates of the center of the group and he will make the next move towards it. By doing a random search of the group it would have been very possible that the fish lost the group forever.

4. Case Study: Implementing the Fish Shoal Inspired Movement Using Lego NXT Robots

An application which implements the movement algorithm presented in the previous section has been developed on a robotic collective composed of 6 LEGO Mindstorm NXT robots [20 - 22]. The LEGO Mindstorm NXT robots were programmed using the RobotC programming language [23].

A NXT robot is composed of an “intelligent brick”, as depicted in Fig. 8 (1), that controls a series of actuators and sensors, namely: a touch sensor (2), a microphone (3), a light sensor (4), an ultrasonic sensor (5), and DC motors (6). The intelligent brick also features an advanced communication device based on the Bluetooth protocol.

The NXT intelligent brick features an ARM7-based Atmel main processor [24] which can be programmed using RobotC which is the version of the C language for NXT. The firmware application software was written in RobotC [23] and has been uploaded to the intelligent brick.

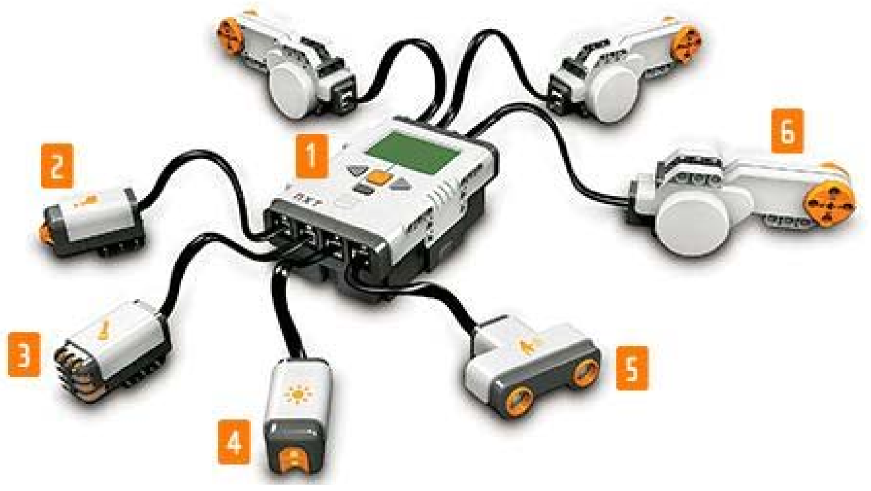

Fig. 8. The Lego Mindstorm NXT intelligent brick, actuators and sensors [20 - 22].

4.1. Implementing Movement on the NXT Robots

Using the LEGO parts and intelligent bricks we have created 6 robots loosely based on Castor Bot [25], as depicted in Fig. 9.

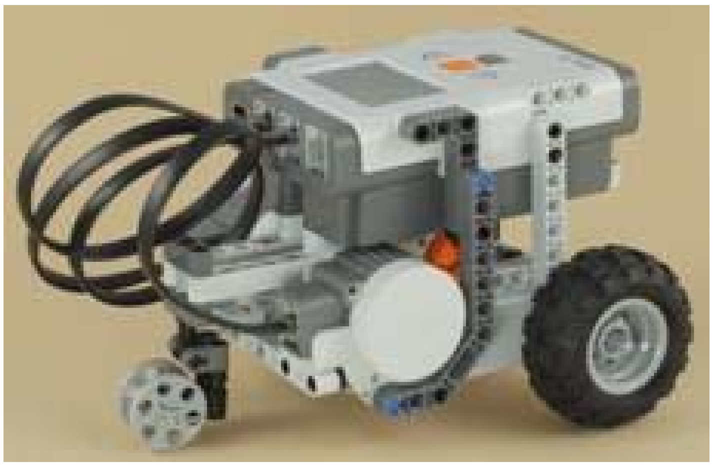

Fig. 9. The Castor Bot [25].

The movement for each robot is composed of a succession of rotations and translations. When implementing these functionalities on the NXT robot, special care was taken in order to make them extremely precise, so that the coordinates computed in the robots memory correspond to the actual coordinates the robot. As seen in Fig. 9, each robot has two active wheels which are used for motion. For each type of movement (either rotation or translation), each wheel has been programmed to turn a very specific and exact number of degrees (for rotation the wheels rotate in opposite ways, and for translation they rotate in the same direction).

The following parameters are used:

- the distance between wheel, *DBW*:

$$DBW = 12 \text{ cm} \tag{8}$$

- the circumference of the circle made by one wheel if the other is stationary, *CC*:

$$CC = DBW * \pi \tag{9}$$

- the distance (in centimeters) which the robot has to cover in order to rotate exactly 1 degree (distance per degree), *DPD*:

$$DPD = CC / 360° \tag{10}$$

- from the wheel diameter, *WD*, the circumference of the wheel, *WC*, can be computed, and also the number of degrees necessary to rotate the wheel so that the robot moves exactly 1 centimeter (degree per centimeter), *DPC*:

$$WD = 5.6 \text{ cm} \tag{11}$$

$$WC = WD * \pi \tag{12}$$

$$DPC = 360° / WC \tag{13}$$

- the number of degrees that the wheel has to move so that the whole robot can rotate exactly one degree (degree per degree), *DegPD*:

$$DegPD = DPD * DPC \tag{14}$$

Based on the physical dimensions of the NXT brick, motors and wheels the parameters presented above are computed. Replacing (11) in (12) and the result in (13), DPC can be calculated as:

$$DPC = 20.4628 \text{ °/cm} \tag{15}$$

Replacing (8) in (9) and the result in (10), and using the value from (15), (14) can be computed as:

$$DegPD = 0.7692 \tag{16}$$

In order to rotate the robot with d degrees, the corresponding wheel should rotate with

$$WR = DegPD * d \tag{17}$$

degrees.

In order to move the robot c centimeters in a straight line, both wheels should rotate with

$$WT = DPC * c \tag{18}$$

degrees.

4.2. Communication in the NXT Robot Network

A computer featuring a Bluetooth connection as shown in Fig.10 is used as a gateway/router for the communication between the robots; the computer is transparent to all the robots, so that they are not aware of its presence (the computer works on a lower level of a layered communication protocol).

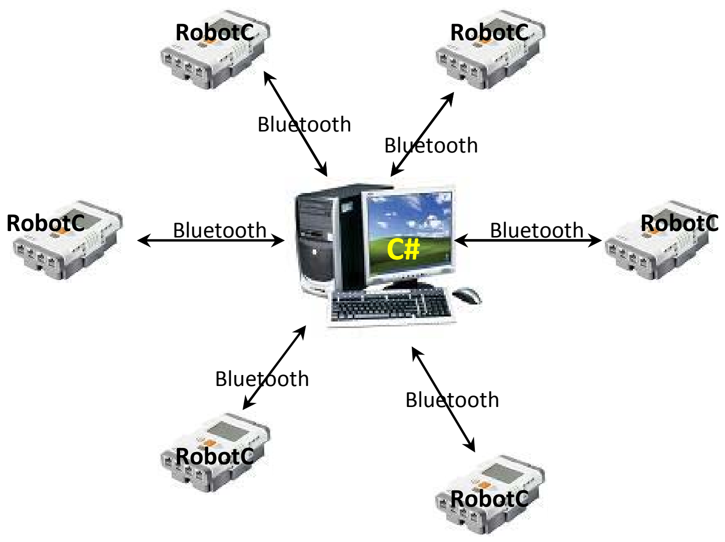

Fig. 10. NXT System Architecture.

The NXT brick and the PC communicate with each other with direct commands. The communication protocol used implements a master-slave communication in which the master is the PC computer and the slaves are represented by the NXT bricks. The commands exchanged between the master and the slave are *MessageRead* and *MessageWrite*. The structure of the *MessageRead* command is presented below:

```
Byte 0: 0x00 or 0x80
Byte 1: 0x13
Byte 2: Remote inbox number (0-9) – the inbox which is receiving the message
Byte 3: Local inbox number (0-9) – the inbox which is sending the message
Byte 4: remove (if true the message is removed from the remote inbox)
```

The *MessageRead* command is issued only by the master. When the *MessageRead* is received by the slave, it responds with a *ResponseRead* message:

```
Byte 0: 0x12
Byte 1: 0x13
Byte 2: Status Byte
Byte 3: Local inbox number (0-9)
Byte 4: Message Length
Byte 5-63: Message (zero filled if necessary)
```

The structure of the *MessageWrite* command is presented below:

```
Byte 0: 0x00 sau 0x80
Byte 1: 0x09
Byte 2: Inbox number (0-9) – the inbox which is receiving the message
Byte 3: Message length
Byte 4-N: Message;
```

The *MessageWrite* command is issued only by the master. When the *MessageWrite* is received by the slave, it responds with a *ResponseWrite* message:

```
Byte 0: 0x12
Byte 1: 0x13
Byte 2: Status Byte
Byte 3: Local inbox number (0-9)
```

The required software resources have been created on both the PC computer master (written in C#) and on the NXT bricks slaves (written in RobotC) Fig. 11.

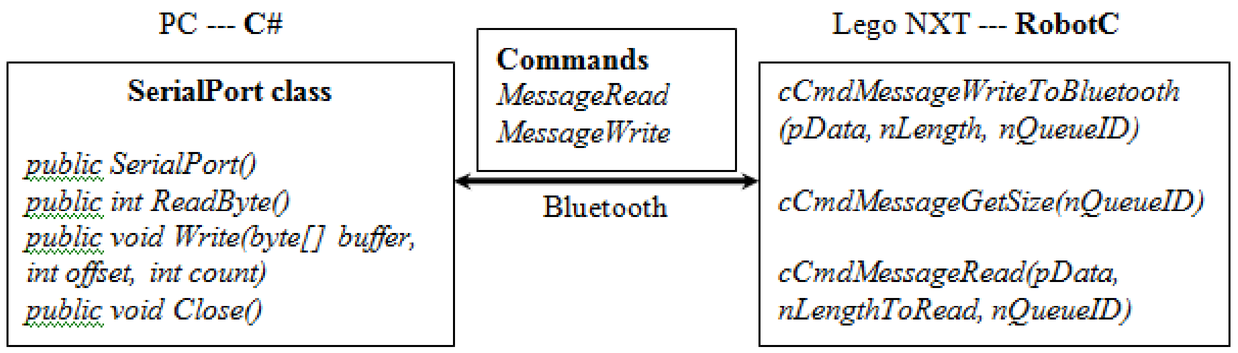
PC --- C#
SerialPort class
public SerialPort()
public int ReadByte()
public void Write(byte[] buffer, int offset, int count)
public void Close()
Commands
MessageRead
MessageWrite
Bluetooth
Lego NXT --- RobotC
cCmdMessageWriteToBluetooth (pData, nLength, nQueueID)
cCmdMessageGetSize(nQueueID)
cCmdMessageRead(pData, nLengthToRead, nQueueID)

Fig. 11. Bluetooth commands for the PC (left) and NXT brick (right).

All the NXT robots know their position relative to the starting position. At each step of the algorithm, the robots compute the new direction of movement, the angle it makes with the original direction and the distance which has to be covered at that step. This information enables the robot to know its exact position relative to the starting position. Also, each movement the robot makes is broadcasted to all the other robots at each step of the algorithm; this is necessary because the algorithm takes into account the position of all the other robots (actually the center of the robot group) when deciding the next movement.

4.3. Experimental Results

The behavior of the 6 robot system has been investigated based on a number of scenarios designed to stress the movement algorithm. Based on these experiments, a success rate has been calculated based on the time to reach the goal. The time to reach the goal is the sum between the number of grouping steps and the number of parallel movement steps.

The experiments have included scenarios to test the following situations:

- Movement towards the target of a number of robots which were not in the larger group's view range;
- Movement towards the center of the group of a robot which was not in the group's view range;
- Using different initial states for the algorithm: using angles of 50° and 60° for the initial placing of the robots;
- Movement of a group of 3, 4, 5 or 6 robots towards the goal.

Scenario No. 1. Movement of a group of 3 robots situated at a distance smaller than the maximum distance from the center of system.

The robots leave from the initial position shown Fig. 12.a, they then regroup as presented in Fig. 12.b and move in parallel until they are close to the target. Finally, the robots change their movement pattern, heading towards the target, as depicted in Fig. 12 d-f.

Because there are only three robots, they are not attracted strongly enough to the target unless they are very close to it.

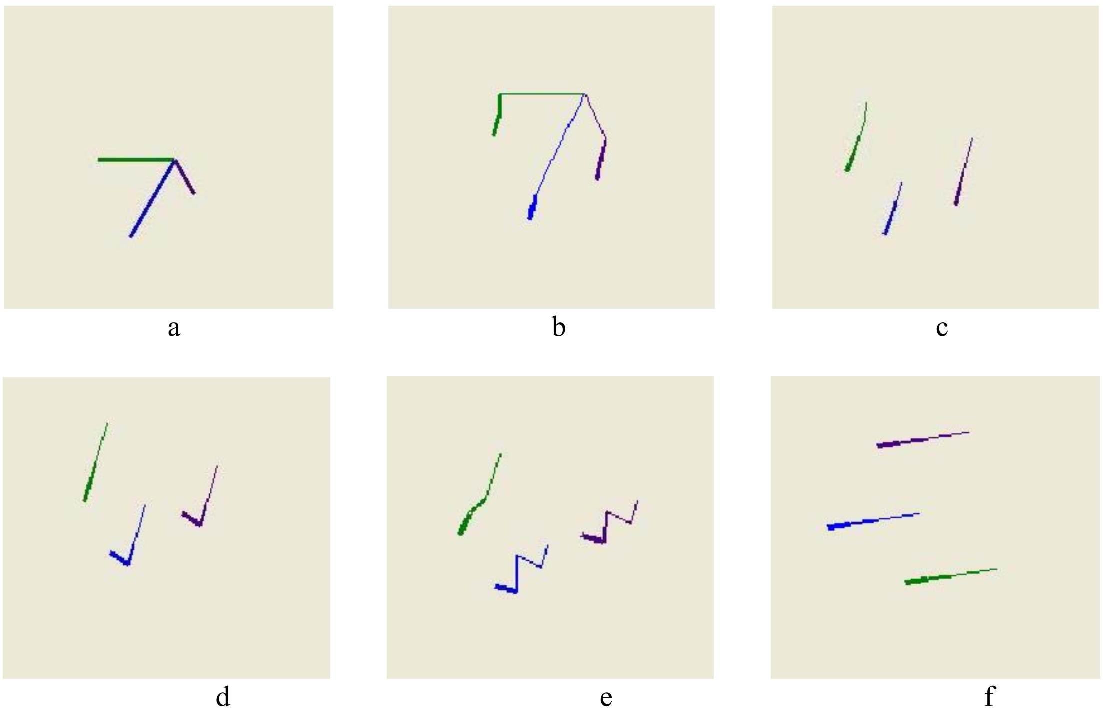

Fig. 12. Movement of a group of 3 robots situated at a distance smaller than the maximum distance from the center of system.

Scenario No. 2. Movement of a group of 4 robots situated at a distance smaller than the maximum distance from the center of system.

The movement of the robots is similar to Scenario No. 1, but here the influence of the fourth robot is making the whole system converge to the target faster, as seen in Fig. 13.

Scenario No. 3. Movement of a group of 5 robots situated at a distance smaller than the maximum distance from the center of system.

The movement of the robots is similar to those in the previous scenarios but now the influence of the supplementary robots is visibly making the whole system converge to the target faster. When some robots change direction, that change is evident only to the robots closest to them; the change in direction if propagated to the closest robots but is also dampened by the distance between them.

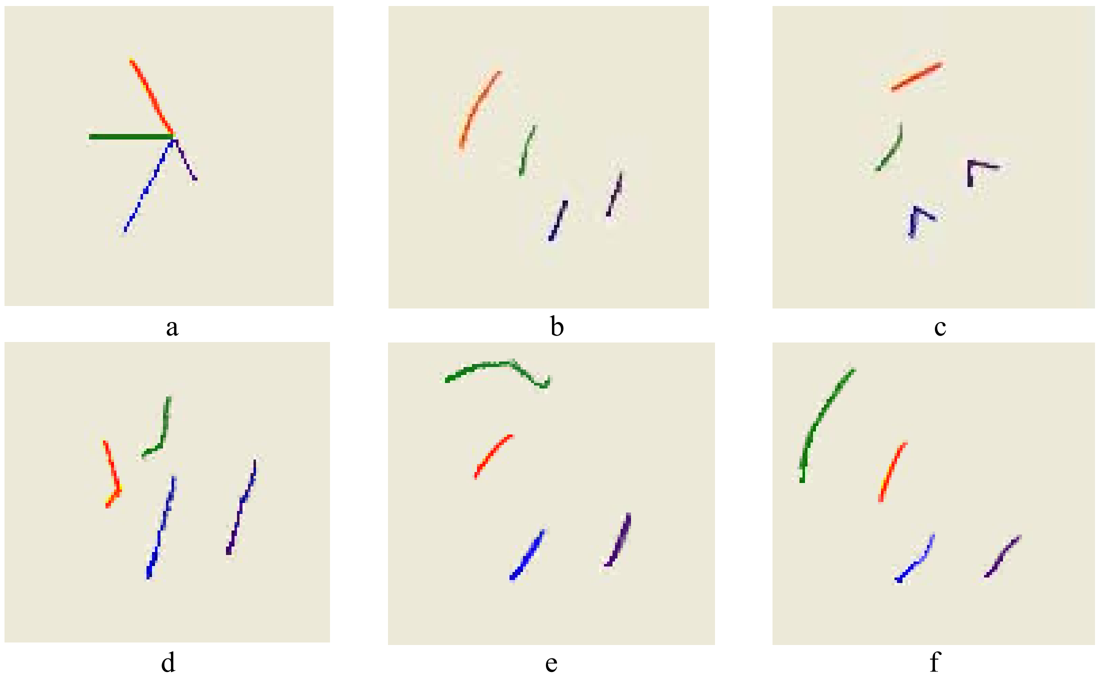

Fig. 13. Movement of a group of 4 robots situated at a distance smaller than the maximum distance from the center of system.

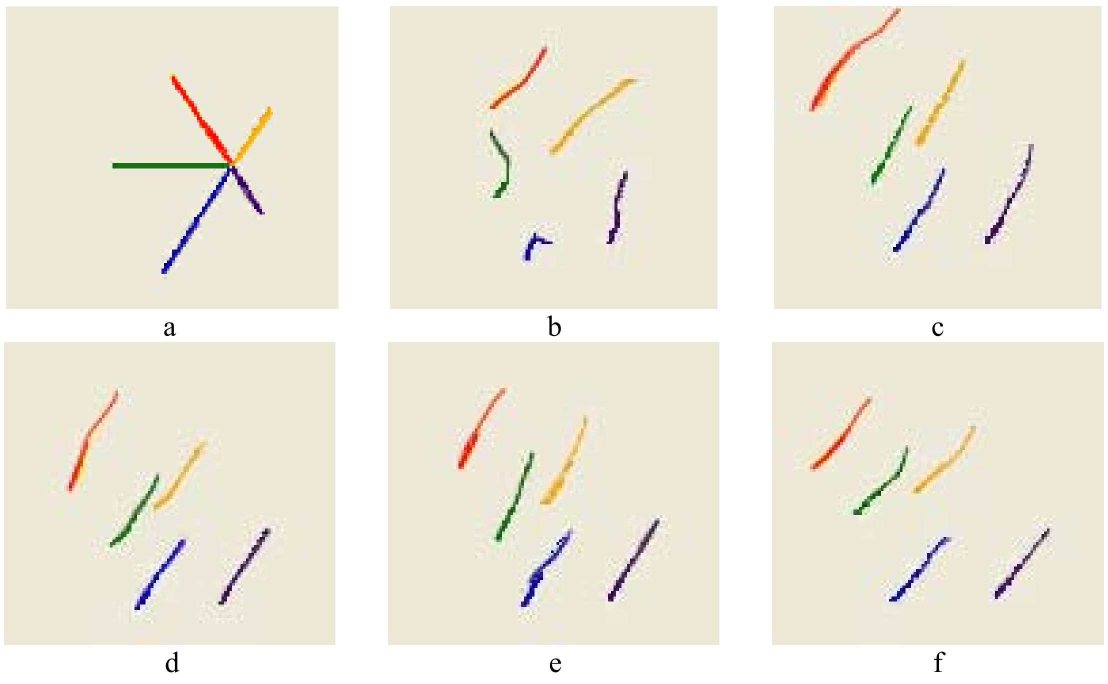

Fig. 14. Movement of a group of 5 robots situated at a distance smaller than the maximum distance from the center of system.

Scenario No. 4. Movement of a group of 6 robots situated at a distance smaller than the maximum distance from the center of system.

Because of the large number of robots, the changes in direction for the robots are more frequent than in the last scenarios. These frequent changes allow the system to converge faster to the target.

The performance of the algorithm for the above studied cases using 3, 4, 5 or 6 robots respectively is analyzed in Table 1 and Fig. 16.

Table 1. Performance of the algorithm when using a varying number of robots.

No. of robots	Grouping [number of steps]	Parallel movement [number of steps]	Reaching the goal [number of steps]	Time to reach the goal [%]
3	3	65	68	100
4	6	60	66	97,1
5	7	58	65	95,6
6	6	51	57	83,8

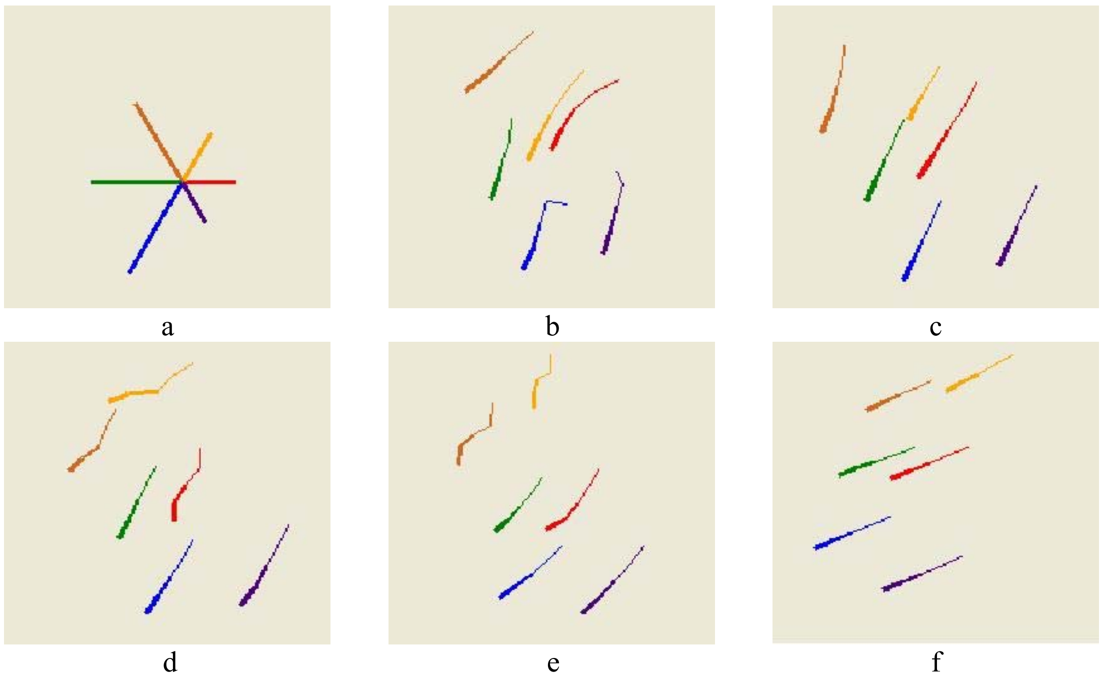

Fig. 15. Movement of a group of 6 robots situated at a distance smaller than the maximum distance from the center of system.

As shown in Fig. 16, the time to reach the goal decreases as the number of robots increases which demonstrates the emergent properties of the system: the more entities there are, the faster the entities converge to the goal.

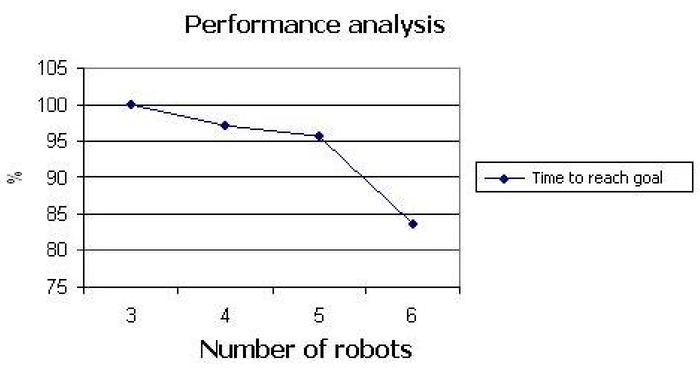

Fig. 16. The time to reach the goal for varying number of robots.

5. Conclusions

This paper presents and discusses the problem of movement in robotic collectives. An algorithm inspired from the study of the natural world has been presented. This algorithm is derived from the study of the movement of fish shoals and has a mathematical basis.

Emergent algorithms inspired by the natural world are part of the distributed navigation algorithms. Such algorithms make use of local data and local interactions in order to build a global view of the system and reach their goal [27-30]. On the opposite side from the normal distributed algorithms, the distributed algorithms inspired from the natural world are very simple, effective and inexpensive. The entities are very homogeneous, and losing one entity has almost no meaning on the functionality of the whole system (the system only integrates local data, so losing a simple entity in another part of the environment has little importance to another entity which is only aware of just a small part of the environment).

The shoal movement algorithms in [12, 13, 19] use only three types of movement: parallel movement, repulsion and attraction, and a larger number of constraints; this implies a limited range of motion for the robots. The improved movement algorithm presented in this paper has fewer constraints and a much wider range of motion which implies that the entities are reaching the goal faster and more efficient.

The algorithm presented in this article has been implemented on a robotic collective which consists of 6 LEGO Mindstorm NXT robots with advanced communication and sensing capabilities. Various experiments have been conducted using this system in order to prove the feasibility and efficiency of the algorithm. All of the experiments were conducted using numerous scenarios in order to test the movement algorithm. The emergent properties of the algorithm were studied based on these experiments.

The experiments presented in this article have been designed in such a manner that they are able to model some of the worst case scenarios of this algorithm. These experiments have validated the algorithm successfully.

The main disadvantage of the algorithm presented in this article is that it does not take into account the presence of obstacles in the environment; this algorithm is based on emergent behavior patterns which are retrieved from the study of fish shoals, and in real world, in the ocean, fish shoals rarely meet any obstacles. The current algorithm can be used in the presence of obstacles only if those obstacles are marked in advance.

Future work will focus on further testing the algorithm in real world scenarios and modifying the algorithm in order to model obstacle avoidance.

References

[1]. M. Popa, A. S. Popa, V. Cretu, M. V. Micea, Monitoring Serial Communications in Microcontroller Based Embedded Systems, in *Proceedings of the International Conference on Computer Engineering and Systems (ICCES '06)*, Cairo, Egypt, Nov. 2006, pp. 56-61.

[2]. Y.-S. Dai, M. Hinchey, M. Madhusoodan, J. L. Rash, X. Zou, A Prototype Model for Self-Healing and Self-Reproduction In Swarm Robotics System, in *Proceedings of the 2nd IEEE Symposium on Dependable, Autonomic and Secure Computing*, Sept. 2006, pp. 3-10.

[3]. Intel Research, Instrumenting the World: An Introduction to Wireless Sensor Networks, Intel Corporation, 2005 (http://www.intel.com/research/exploratory/instrument_world.htm).

[4]. R. D. Cioarga, M. V. Micea, B. Ciubotaru, D. Chiuciudean, D. Stanescu, CORE-TX: Collective Robotic

Environment - the Timisoara Experiment, in *Proceedings of the 3rd Romanian-Hungarian Joint Symposium on Applied Computational Intelligence (SACI 2006),* Timisoara, Romania, May 2006, pp. 495-506.

[5]. P. Gai, L. Abeni, G. Buttazzo, Multiprocessor DSP Scheduling in System-on-a-Chip Architectures, in *Proceedings of the 14th Euromicro Conference on Real-Time Systems (ECRTS '02)*, Vienna, Austria, 2002, pp. 231–240.

[6]. Eric Bonabeau, Marco Dorigo, Guy Theraulaz, Swarm Intelligence: From Natural to Artificial Systems, *The Oxford University Press*, 1990.

[7]. Scott Camazine, Nigel R. Franks, James Sneyd, Eric Bonabeau, Jean-Louis Deneubourg, Guy Theraulaz, *Self-Organization in Biological Systems*, Princeton University Press, Princeton, NJ, 2001.

[8]. Julia K. Parrish, Steven V. Viscido, Daniel Grünbaum, Self-Organized Fish Schools: An Examination of Emergent Properties, *Biological Bulletin*, Vol. 202, No. 3, June 2002, pp. 296-305.

[9]. Y. Inada, K. Kawachi, and H. Liu, Simulation Study of Schooling Motion of Fish based on Two Observed Motions: Approaching Motion and Parallel Orientating Motion, in *Proceedings of Modeling and Simulation (MS 2004)*, California, USA, March 2004, [CD support].

[10]. R. Cioarga, M. V. Micea, B. Ciubotaru, D. Chiciudean, V. Cretu, V. Groza, eBML: A Formal Language for Behavior Modeling and Application Development in Robotic Collectives, in *Proceedings of the International Workshop on Robotic and Sensors Environments (ROSE 2007)*, Ottawa, Canada, Oct. 2007, pp. 80-85.

[11]. R. Cioarga, B. Panus, C. Oancea, M. Micea, V. Cretu, E.M. Petriu, Fish Shoal Inspired Movement in Robotic Collectives, in *Proceedings of the IEEE International Workshop on Robotic and Sensors Environments (ROSE 2008)*, Ottawa, Canada, October 2008, pp. 7-12.

[12]. Aoki, I., A simulation study on the schooling mechanism in fish, *Bulletin of the Japan Society of Scientific Fisheries*, Vol. 48, Aug. 1982, pp. 1081-1088.

[13]. Huth, A., and Wissel, C, The simulation of the movement of fish schools, *Journal of Theoretical Biology*, Vol. 156, 1992, pp. 365-385.

[14]. M. Dorigo, L. M. Gambardella, Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem, *IEEE Transactions on Evolutionary Computation*, Vol. 1, Issue. 1, 1997.

[15]. V. Maniezzo, L. M. Gambardella, F. de Luigi, *Ant Colony Optimization: New Optimization Techniques in Engineering*, by G. C. Onwubolu, B. V. Babu, Springer-Verlag Berlin Heidelberg, 2004, pp. 101-117.

[16]. H. Van Dyke Parunak, S. Brueckner, J. Sauter, R. Matthews, Global Convergence of Local Agent Behaviors, in *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005)*, Jul. 2005, Utrecht, Netherlands, pp. 305–312.

[17]. W. Truszkowski, M. Hinchey, J. Rash, C. Rouff; NASA's swarm missions: The challenge of building autonomous software, *IEEE IT Professional*, Vol. 6, No. 5, Sept-Oct 2004, pp. 47–52.

[18]. M. Hinchey, C. Rouff, J. Rash, Requirements of an Integrated Formal Method for Intelligent Swarms, in *Proceedings of the 10th International Workshop on Formal Methods for Industrial Critical Systems (FMICS 2005)*, Sep 2005, Lisbon, Portugal, pp. 125 – 133.

[19]. N. Bhooshan, The Simulation of the Movement of Fish Schools, *Technical Report for The Institute for Systems Research*, ISR, UG-2001-4, University of Maryland.

[20]. LEGO Group, LEGO MINDSTORMS NXT Hardware Developer Kit, 2006 (http://mindstorms.lego.com/Overview/nxtreme.aspx).

[21]. LEGO Group, LEGO MINDSTORMS NXT Communication protocol, 2006 (http://mindstorms.lego.com/Overview/nxtreme.aspx).

[22]. LEGO Group, "LEGO MINDSTORMS NXT Bluetooth Developer Kit", 2006 (http://mindstorms.lego.com/Overview/nxtreme.aspx).

[23]. T. Friez, D. Swan, "ROBOTC for LEGO® MINDSTORMS™ 1.30 - Users Manual", 2008.

[24]. ARM, "ARM7TDMI, Technical Reference Manual", Rev. 3 (http://infocenter.arm.com/help/topic/com.arm.doc.ddi0210c/DDI0210B.pdf).

[25]. D. J. Perdue, The Unofficial Lego Mindstorms NXT Inventor's Guide, No Starch Press, October, 2007.

[26]. R. Cioarga, B. Ciubotaru, D. Chiciudean, M. V. Micea, V. Cretu, V. Groza, Emergent Behavioral Modeling Language in Obstacle Avoidance, in *Proceedings of the 24th IEEE Instrumentation and Measurement Technology Conference (IMTC 2007)*, Warsaw, Poland, May 2007, [CD support].

[27]. R. Cioarga, I. Nalatan, S. Tura-Bob, M. Micea, V. Cretu, M. Biriescu, V. Groza, Emergent Exploration and Resource Gathering in Collaborative Robotic Environments, *Proceedings of the IEEE International Workshop on Robotic and Sensors Environments (ROSE 2008)*, Ottawa, Canada, October 2008, pp. 13-18.

[28]. Prassler, E., Milios, E., Parallel distributed robot navigation in the presence of obstacles, in *Proceedings of the Second IEEE Symposium on Parallel and Distributed Processing*, Dallas, Texas, USA, 9-13 Dec 1990,

pp. 475 – 478.

[29].Konolige, K. Fox, D. Limketkai, B. Ko, J.; Stewart, B., Map merging for distributed robot navigation, in *Proceedings of International Conference on Intelligent Robots and Systems (IROS 2003)*, Volume 1, 27-31 Oct. 2003, pp. 212 – 217.

[30].D. F. Hougen, J. C. Bonney, J. R. Budenske, M. Dvorak, M. Gini, D. G. Krantz, F. Malver, B. Nelson, N. Papanikolopoulos, P. E. Rybski, S. A. Stoeter, R. Voyles, and K. B. Yesin. Reconfigurable robots for distributed robotics, *Government Microcircuit Applications Conference*, Anaheim, CA, March 2000.

Advertise in
Sensors & Transducers Journal
and Sensors Web Portal
ISSN 1726-5479
SENSORS & TRANSDUCERS 12/08
Sensor Market Trends
SENSORS 11/08
SENSORS & TRANSDUCERS 1/09
vol. 100
Sensor Instrumentation, DAQ and Virtual Instruments
IFSA
http://www.sensorsportal.com/DOWNLOADS/Media_Kit_2009.pdf
sales@sensorsportal.com