Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 119 (2017) 147–156

Procedia Computer Science

www.elsevier.com/locate/procedia

6th International Young Scientists Conference in HPC and Simulation, YSC 2017, 1-3 November 2017, Kotka, Finland

Simulating robot groups with elements of a social structure using KVORUM

Rovbo M.A.[a*], Ovsyannikova E.E.[a]

[a]*National Research Centre "Kurchatov Insitute", Akademika Kurchatova pl, 1 , Moscow, 123182, Russia*

Abstract

This paper describes the architecture of KVORUM, the agent-based modeling environment developed by the authors to explore certain design decisions and facilitate simulation usage in the field of group robotics, specifically for research of robotic systems with elements of a social structure. KVORUM is a prototype of a simulation system that should adequately abstract away complexities of a physical system while providing convenient interfaces and library modules for modeling groups of mobile land robots and individual agent's intrinsic structures. It was built from the ground up as a modular, highly extensible system focused on speed of calculations that is to be achieved by using proper simplification of physical and other effects, and by the ability to perform simulations using parallel computing. In KVORUM the developed abstractions and limitations of a parallel architecture are tested without fully adhering to parallel computing requirements. It is shown that the proposed method of simulating such systems is applicable to a wide variety of problems from the field of group robotics with elements of a social structure (and swarm robotics in general). Some features of the architecture and models used in KVORUM allow for it to be extended in next iterations to fully support simulations on parallel computing systems.

Peer-review under responsibility of the scientific committee of the 6th International Young Scientist conference in HPC and Simulation

Keywords:artificial life; simulationmodeling; swarm robotics; agent-based simulation;robot;social behavior modeling; development tools; bioinspired systems; eusociality.

* Corresponding author. Tel.: +7-985-305-2967.
E-mail address:rovboma@gmail.com

1877-0509
Peer-review under responsibility of the scientific committee of the 6th International Young Scientist conference in HPC and Simulation
10.1016/j.procs.2017.11.171

1. Introduction

The main task of group robotics is the construction of such a system that would give new qualities and improved characteristics due to the interaction of many individual agents possessing relatively simple rules. The main method of studying these systems is simulation and agent modeling, for which researchers use different environments and libraries.

The term “agent” means anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors [17] and Autonomous agents are computational systems that inhabit some complex, dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are designed[12].

One of the areas of research in group robotics is the study of biological systems in order to apply the principles of their organization to build a team of robots, as they demonstrate many desirable qualities, such as resistance to external perturbations, ability to work in unknown environments and efficiency. A new and very promising direction in this area is the approach based on social behavior and the various mechanisms associated with it [6]. We will further refer to artificial groups that have these mechanisms as “groups with elements of a social structure”.

The review of various existing simulation systems as well as analysis of the common problems in this field allowed to obtain a set of requirements and showed that no existing system fully satisfies them. First of all, let us briefly not some characteristic modeling systems, as well as their features in connection with the problem under consideration

Gazebo[10]is a program and a set of libraries that allows one to simulate the behavior of robots with regard to physical effects. Capabilities of the simulation system can be expanded using readily available plugins or by writing new plugins, for example, by adding infrared rangefinders, a video camera or physical effects of movement through a liquid. However, Gazebo is poorly suited for simulating systems with a large number of agents because of the rather high computational cost (it is difficult to model more than 1000 agents).

AnyLogic [3] and Repast[1]provide very versatile tools for agent-based (and other types in the case of AnyLogic) modeling. Both systems are undoubtedly powerful and advanced modeling tools, but they are aimed at a greater range of tasks than simulating robotic systems with social structure elements, and as a result, as far as we know, they do not provide any special libraries for modeling systems in this field.

Unlike systems that support detailed modeling of physical processes, or the universal agent-based and simulation modeling systems, there are simulations and modeling libraries made explicitly for social systems. The examples are NetLogo[19], Myrmedrome [4] and AntMe! [18]. NetLogo, despite the power of the language itself, has very limited functionality for modeling robotic systems with elements of social structure, and the other two programs are not modeling libraries, but they rather illustrate and simulate some elements of behavior of ants.

Myrmedrome [4] is an agent-based ant colony simulator, built on the principle of ants' reaction exclusively to local events. Ants interact with each other using chemical signals (Fig. 1a, white lines – pheromones’ paths). Agent control algorithms are nondeterministic: at every step, when all the parameters of the ants are updated, there is a possibility that the selected action will not be performed. This makes the system flexible to unforeseen situations. Myrmedrome imitates the life of an ant’s colony in a restricted area of the environment. The social organization is based on a caste system, in which there are workers and soldiers. The first perform basic functions to find food, which they store in joint stomachs (and which are shared with other ants). Workers will try to kill prey upon encountering it and will consider ants from other colonies a threat and announce their presence using pheromones. Soldiers defending the nest will find a source of danger and seek to destroy it. The program provides the user with the ability to manipulate the environment and ants: add food, move ants, change the parameters of pheromone, the number of ants and some other (Fig. 1a). It is an anthill simulation but not a library for modeling, and the source code of this program is not open for modification.

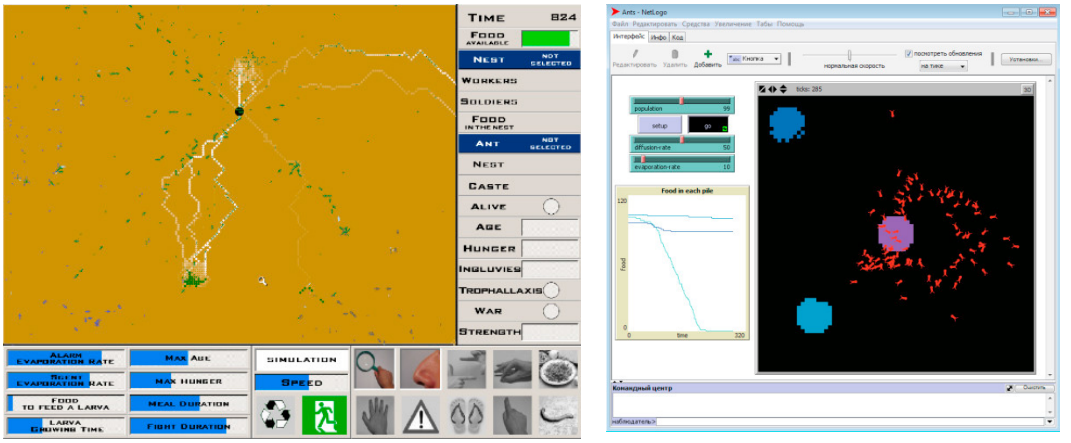


Fig. 1. (a) the simulation window; (b) NetLogo: Ants.

AntMe!is a simulation software for Windows, written in C#. Supported languages are C# and Visual Basic. AntMe! was created with the purpose of teaching the basics of programming. In AntMe! one have to program the vital activity of ants – collecting food (sugar and apples) and defending themselves from attacks of bugs. There is an option of specializing in ants: some collect food, and some fight with beetles. The environment's rules are not subject to change, its rules are set by the developer. The simulation supports built-in visualization and has a debugging mode. The main limitations of AntMe! is its inability to perform complex calculations for a large number of ants (agents) due to low performance (it cannot improve performance using computing clusters), the limitation on the number of commands for agents and the closed code of the program that does not allow to make substantial changes.

NetLogo [20] is a flexible and extensible agent modeling tool with a large set of provided libraries. It is also a name of the corresponding language used in it. While very versatile, for our purposes it has the following drawbacks — it does not implement the various mechanisms necessary to build a group with a social structure (for example, emotions, language communication), leaving this problem for the researcher, its language imposes some limitations [11], unlike the programming languages used in many other environments and, while supporting parallel computation, it does so in a way that limits it to multiple runs speed-up, but each instance of the environment has to be run sequentially since its runtime model isn't naturally parallelizable. The developers of NetLogo explain that this happens because most of the information in the model is stored in agents, which have variables written and read many times per tick. As for KVORUM system, information about the world is stored in layers, which can be processed separately from each other and only local areas are processed at a time, so the performance can be increased by applying the parallel computations within each run.

The existing model of the anthill is too limited, although it is extensible: it demonstrates only the standard foraging mechanism with the help of pheromones (Fig. 1b).

In the development of the KVORUM we have considered the requirements for the system indicated in the above work:

- Scalability, leverage of parallel computing systems;
- Support of certain features of agent systems in general and their implementation in the form of libraries: concept of an agent, interaction between agents, environment, laws of interaction with the environment, space, time;
- Basic structures and mechanisms of social behavior:
    - individual mental differences;
    - differentiation of functions;
    - local interaction of individuals and language communication;

- Formation of coalitions, emergence of a hierarchical structure;
- Support for creating models of the internal and external world of the agent;
- Support of models with several interacting (but distinct) groups of agents.

A more detailed review of the state-of-the-art systems and the motivation behind the requirements can be found in[16], which provides an analysis of the available specialized simulation systems suitable for solving the problems of collective robotics oriented to the creation of systems with elements of social structures.

This motivated us to create a prototype of such a system that was called KVORUM. In this article its architecture is described and selected experimental results are showcased with the hope to demonstrate its ability to simulate systems for most of the problems encountered by researchers in this field. Despite the fact that this prototype itself is not designed to model large systems using distributed computing, its architecture and features allow us to test some concepts at a basic level, which makes it possible in the future to build upon it a specialized simulation system that leverages parallel computing. The developed KVORUM system implements a number of the aforementioned properties of the modeling system in addition to scalability for a distributed computing system. Since KVORUM is an experimental prototype, the library of high-level elements, such as the mechanisms of coalition formation, the model developer has to implement independently under the model instead of using the ready-made module, however, the extensibility of the system allows supplementing it with the necessary functionality.

2. KVORUM architecture

The KVORUM simulation system is designed to simulate the behavior of large groups of agents. The user can select several types of agents and any number of agents within each type. Each agent is equipped with "virtual sensors" that simulate real sensors on robots, such as sensors, locators, superlocators, position sensors, etc.

KVORUM is a software tool that simulates the robots and their environment. A robot is a computer-controlled machine and involves technology closely associated with automation. Industrial robots can be defined as a particular field of automation in which the automated machine (that is, the robot) is designed to substitute for human labor[5]. One of the features of the system is the availability of interfaces that allow you to manage both virtual agents and real technical devices. Virtual agent is an agent without a physical body, for example, a computer simulation of a robotic agent. The configuration of the architecture for a particular control system is carried out in the corresponding user program modules. For example, the configuration for the TMU architecture is defined in the module tmurobot.py.

Unlike modeling systems that allow simulating a variety of physical effects, KVORUM uses a simplified physical model to save resources and accelerate development. In particular, based on the characteristics of the subject field, a 2D world model was chosen as a basis, in which the movement of agents and many interactions are geometricized. For example, the operation of ultrasonic and infrared rangers is modeled by calculating the rays and evaluating the "visibility" zone of the sensor. This approach allows to model larger groups of agents, preserving important elements for the region and abstracting their physical details.

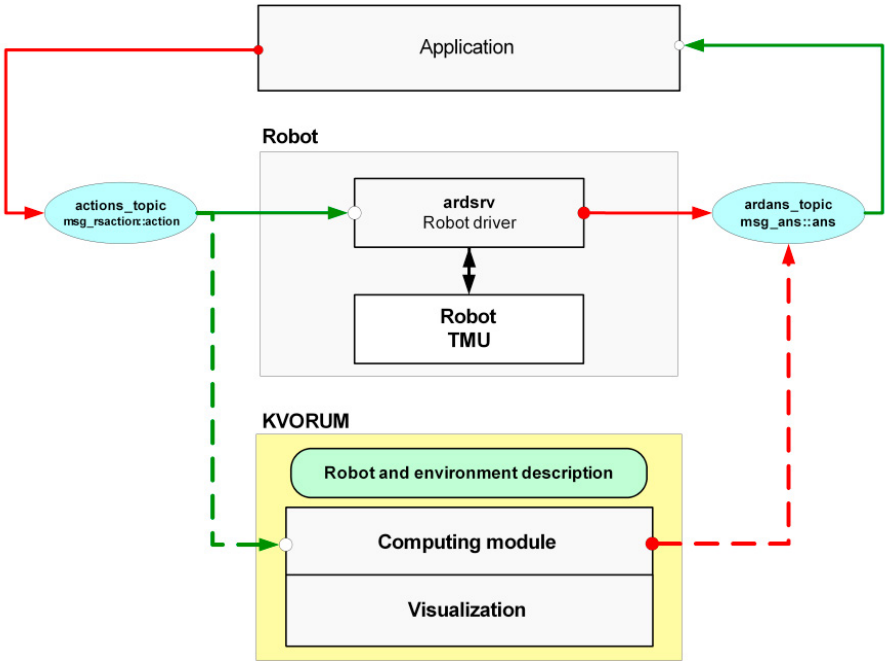


Fig. 2. Conceptual scheme.

The modeling system runs under Linux Kubuntu 14.04 and uses the ROS framework [14], which transfers data between system modules. Modules, scripts, application subsystems can be written in any supported ROS language. The main requirement is the implementation of interfaces through a message system of the so-called topics.

The modeling system is based on two modules — KVORUM_m and KVORUM_v which form the core of the system. The modules are written in Python 2. The main task of the kernel of the system is to model behavior of a group of real technical objects. The interfaces of the kernel modules are defined in such a way as to maximally abstract the control program from the control object.

Fig. 2 presents a conceptual diagram of the KVORUM simulation system, consisting of the core of the system (calculation module + visualizer + robot and environment description), robot models and application programs. The core of the system interacts with the robot simulation model and the application programs through the following topics: input (action_topic) and output (ardans_topic). The model simulating a robot consists of a control node and a logical model of the TMU robot. The application programs are created by the user depending on the problem at hand.

The abstraction of the control program from the control object (a virtual or real robot, or a group of robots) is accomplished by the means of interaction between the kernel components and the user application, namely through the exchange of messages (Fig. 3).

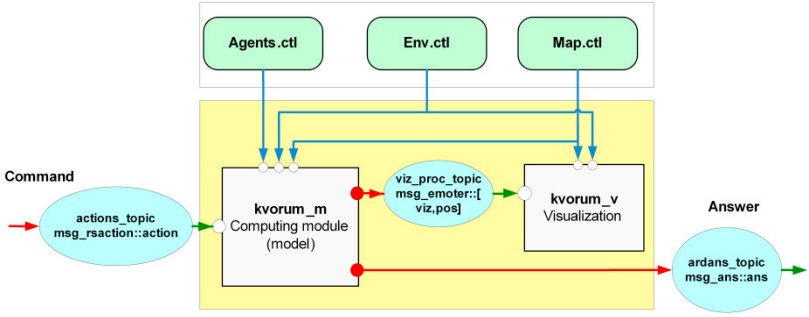
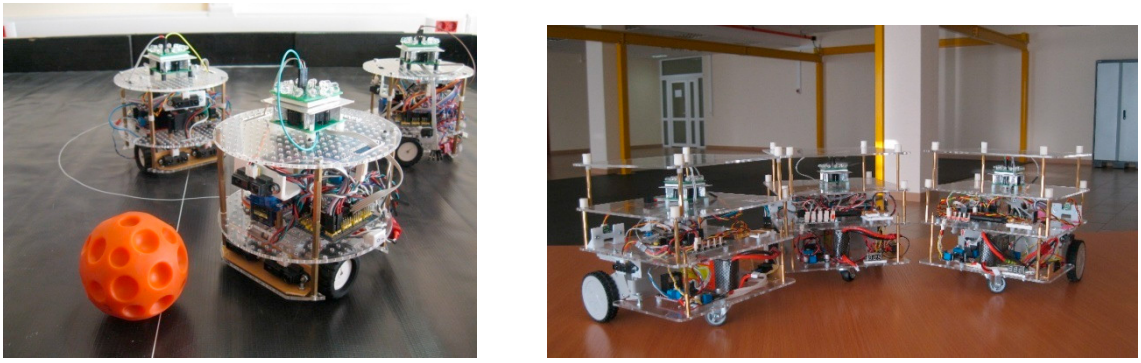


Fig. 3. Core of the system.

This architecture has an important property of portability of the code of model elements between the simulator and real robots. In particular, the robots developed by the authors, named YARP-2 and YARP-3 (Fig. 4), have a TMU architecture and therefore operate under the same control scheme.



a) b)

Fig. 4. (a) YARP-2; (b) YARP-3.

The KVORUM modeling system is based on a two-dimensional representation of space. This is a convenient simplification, since many tasks of group robotics are considered in the context of ground mobile robots. The partition of this space into discrete elements, for example, cells, allows to further reduce the number of necessary computations, requesting and processing only those objects that are in the corresponding cells. The use of only continuous coordinates would lead to the need to search through a large number of objects to compare their coordinates with the given constraints. At the same time, despite the division into cells, continuous coordinates remain as parameters of objects and allow both to not degrade accuracy when simulating the displacement and estimating the distances between objects. A square grid is also easy to implement and allows to enjoy the advantages of the matrix calculation speed up techniques used in modern systems. However, it should be noted that this prototype is not suitable for testing the latter assertion, and only its modification that can be physically run on parallel computing systems will allow to explore it.

When working with a discrete grid the operation of sensors, such as range finders, is modeled in a special way. Consider an ultrasonic range finder. To obtain a measurement a beam is drawn from the robot in the direction in which the range finder is looking. We restrict the length of the ray to some maximum value. The cells that it crossed are marked using the Bresenham's algorithm. We select all objects located in these cells from the memory of the system. If they satisfy the conditions checked by the sensor (in this case — they must reflect the ultrasonic signal), then the smallest distance from the computed ones is returned. It is worth noting that instead of a ray, you can build a cone and select all the cells that are crossed by it. Thus, this approach allows some flexibility in describing operation of the sensors.

The environment in which agents live, is a multitude of layers, each of which is responsible for representing a particular physical or logical sign. This means that each point of space (x, y) is described by a certain vector. The system represents the habitat in the form of a multidimensional array with dimensions [x, y, level]. For example, the interpretation of layers, depending on the task, can look like this (level values):

- 0 (LEVEL_LIGHT) —luminosity;
- 1 (LEVEL_COLOR) — the color of the surface;
- 2 (LEVEL_IR) — "ether" (IR area)
- 3 (LEVEL_GROUND) — the surface;
- 4 (LEVEL_ONBOARD) — on-board tuning sensor (virtual level);
- etc.

The names of layers are defined in the dictionaries of the system.

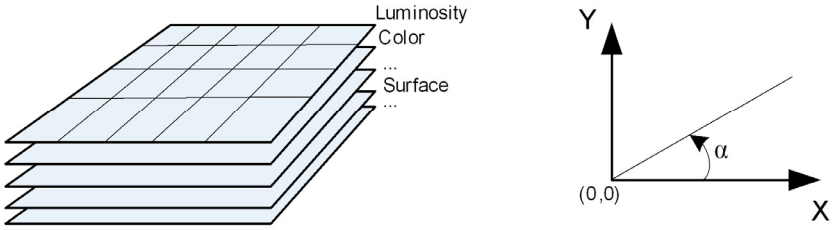


Fig. 5. Representation of the field as layers.

Such a representation allows us to separate various characteristics into different areas of memory and explicitly designate the independence of the pseudophysical characteristics of the medium. In addition to a more structured view of the world, this also has certain technical advantages: since different sensors tend to work with different characteristics of the world, and therefore in different layers, simulation of measuring and changing actions can be parallelized, simultaneously performing operations in different layers in each thread. This also largely unifies these operations, allowing similar features to handle different characteristics.

Fig.5 shows the representation of the field in the form of layers. Each sensor has its own layer containing information about existing agents and objects on it. When creating sensors, you can change the values of the layer. The coordinate axes are represented by the Cartesian system. The coordinates of the point are always positive, the angles lie in the first quarter and are counted counterclockwise from the horizontal (x-axis).

The main types of sensors are point detector, range finder, locator and superlocator.

Rangefinder. Returns the distance to the object found on the LEVEL_GROUND layer.

Locator is a sensor that is able to register surrounding objects in a certain sector, returning the results as a vector of values. And depending on the type of the locator, the vector of the returned values can contain either the distance to the detected objects on the given layer, or the values of the detected objects.

A superlocator is a locator whose return values are a vector of pairs of the form (signal_value, distance). This allows not only to determine the distances to objects, but also to identify them. As for the locator, it is necessary to determine the correspondence between the angle and the index in the array of readings.

The calculation module is responsible for step-by-step simulation of the system state and provides an interface between agents and user requests: obtaining data from the agent's sensor, setting the speed of the agent's forward and rotational motion, removing, initializing agents, rendering and initializing the modeling system.

The visualization module provides a map and agents on the screen.

To transfer the control system to real robots, it is necessary to perform additional tuning and use a robot-specific node that provides the required interface. Such a node is called ardsrv. It translates unified commands passed through actions_topic to signals perceived by the robot. In the case of YARP-2 and YARP-3, these commands are translated into the appropriate format and sent over wireless communication to on-board computers executing them. The answer coming from robots to the ardsrv node is translated back into a unified format and passed to the control system via ardans_topic. Thus, the modules of the simulation system are substituted for the robot driver module, and the communication topics remain unchanged together with the rest of the system.

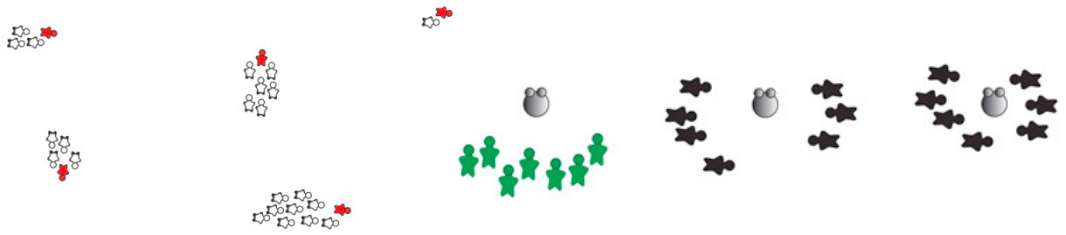
3. Results

Below we give examples of modeling some representative problems.

3.1. Sign-based communication

One of the important elements of the construction of group robotic systems is the language interaction between robots [8]. It’s important element is symbolic structures, on the basis of which it is also possible to build control systems that implement some basic mechanisms of systems with social structure (for example, contagious behavior), which is illustrated by the work [5]. A special feature of the task is the modeling of communication between agents, in particular using elements of language communication.

In KVORUM, communications between agents are modeled explicitly: one agent is the transmitter of the signal, and the other is the receiver. Communications are carried out by interpreting these signals. Communication capabilities were applied in the work [7], where a group of agents used them to coordinate efforts in a hunting task (Fig. 6).



a) b)

Fig. 6. (a) Groups of agents are hunting red agents; (b) agroup is coordinating an attack.

A leader of the group is selected, then the group is divided by roles that determine which direction the subgroup will be approaching the victim from.

3.2. Presentation of a Route for a Mobile Robot Based on Visual Guides

In [9] a method for constructing a path along visual landmarks by a mobile robot is proposed. This task belongs to the category of modeling the internal representation of the external world in an agent.

Description of the route is based on spatial relationships. Simulation experiments were carried out for the foraging problem using the KVORUM simulation system.

The reconnaissance robot performed a random walk to find an object of a given color on a polygon of n × m cells, along the way forming a description of the route. After reaching the goal, the description was transferred to the second robot, which repeated the route (Fig. 7).

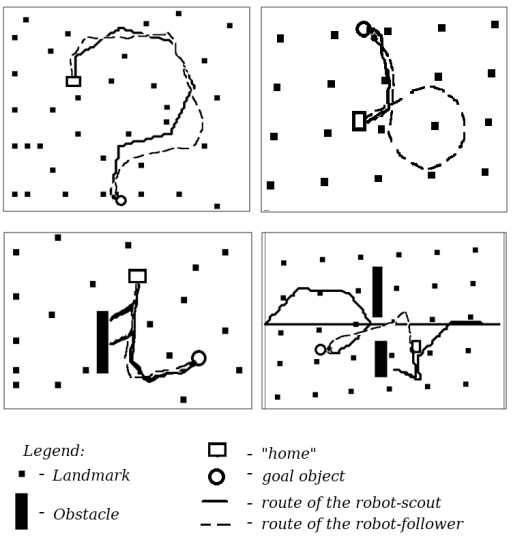


Fig. 7. Movement of the robot along the route discovered by a scout robot.

3.3. Energetically self-sufficient robot group study kit

One of the problems of collective robotics is the energy independence of the team. At the same time, both the task of

fulfilling the goal function simultaneously with maintaining a required energy level of agents, and the task of collecting energy from various sources can be considered. An example of the statement of the problem is given in [15], in which a multi-criteria evaluation of the robot's goals is formulated. The peculiarity of the problem from the point of view of the modeling system is the dynamical nature of the interactions of agents and the world: collection of various resources, changing parameters of the robot (energy level), the generation of energy in the central "nest". In this model agents communicate via stigmergia in the form of directed pheromones [2]. Parametric regulation of individuals' behavior variables is used to control the group's performance. It is implemented by designating two roles and switching between them. Each role corresponds to a different set of characteristics, in particular the carrying capacity and the time after which the agent returns to the nest if no food has been collected. The mechanism used in this work is essentially a stochastic automaton with two states φ1 and φ2 with the corresponding strategies of role switching that are defined by special matrices for returning with food and for returning without food where and are the probabilities of a scout switching to a forager role and vice versa.
In [13], the problem of maintaining a sufficient level of energy for functioning of the group is considered with the constraints of a necessity to use several resources and a centralized generator of energy. Several objects are placed on the test map: resource sources and a base station. The base station determines the resource requirements and informs of them the agents that are nearby. Agents are activated and begin to search for resources, extract them and bring them to the base. An agent can only collect one resource at a single run. Each agent has a limited supply of energy which he uses during the search. If it dries up before the agent reaches the base it "dies" (deactivates). The base plays the role of a charging station of the collective, producing energy from resources extracted by agents (Fig. 8). The model for making the decision is based on Pareto optimization. The agents evaluate the discovered sources of food according to the following factors: how much time it had been searching c_t; how severe is the need for this type of resource was in the nest c_n; what was its designated goal resource c_n.

The influence of the factors is determined by the coefficients in the formula. In general, the objective function f_e is constructed in the following way:

$$f_e = c_t f_t(t) + c_n f_n(n,s) + c_g f_g(s,g) \quad (1)$$

where $c_t + c_n + c_g = 1$ are the coefficients of the corresponding factors' influence.

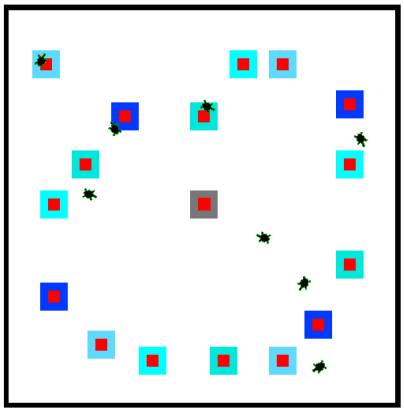


Fig. 8. Homogeneous agents collecting resources of different type for the base.

On this picture there are 4 different types of resources denoted by similarly colored squares. The square in the center that is colored uniquely is the base, where agents store collected resources. The small turtles denote the robots.

The system model was implemented to be simulated by KVORUM. Computational experiment allowed to evaluate the dynamics of the number of robots in the team for different parameters of the control system (the "characters" of agents).

4. Conclusion

The KVORUM simulation system was developed and tested on some representative group robotics problems, which validated viability of the proposed concepts and models that will form the basis of a specialized modeling system for collective robotics for parallel computing simulation. Another important property of the proposed and implemented architecture is the portability of the code of model elements between the simulator and real robots. It allows the same control code that was tested in simulation to be deployed in real world systems, which allows for a clearer interpretation of results obtained using such a simulation system. This capability was verified on real robots and the prototype version of the simulator.

The next step in developing the system is to fully implement support for parallel computing systems and to test in on a computer cluster. The prospective gain in computing speed is expected to allow for simulation of tens of thousands of locally communicating agents in real time while maintaining an adequate representation of the simulated world.

5. Funding

This work was supported, in part, by the Russian Science Foundation, grant 16-11-00018 (which funded the section that contains the review (Introduction), 3.2 (Presentation of a Route for a Mobile Robot Based on Visual Guides) and partially 3.3 (Energetically self-sufficient robot group study kit) of this paper) and by the Russian Foundation for Basic Research, grant ofi-m 16-29-04412 (which funded the second (KVORUM architecture) section).

References

[1] Argonne National Laboratory Repast Simphony [Electronic resource]. URL: http://repast.sourceforge.net/repast_simphony.php# ((date of access: 01.01.2017).
[2] Boissard E., Degond P., Motsch S. Trail formation based on directed pheromone deposition // Journal of Mathematical Biology. 2013. No 6 (66). pp. 1267–1301.
[3] Borshchev A. The Big Book of Simulation Modeling: Multimethod Modeling with AnyLogic 6 / Amazon Digital Services LLC, 2015. 614 p.
[4] Cacace S., Cristiani E., D'Eustacchio D. Myrmedrome: Simulating the Life of an Ant Colony edited by. M. Emmer, Milano: Springer Milan, 2013. pp. 201–210 .
[5] Dorf R.C., Bishop R.H. Modern Control Systems / New Jersey: Pearson Education, Inc, 2008. 1018 p.
[6] Karpov V. Models of social behaviour in the group robotics // UBS. 2016. No 59. pp. 165–232. (in Russian)
[7] Karpov V., Karpova I. Leader election algorithms for static swarms // Biologically Inspired Cognitive Architectures. 2015. No 0 (12). pp. 54–64.(in Russian)
[8] Karpova I.P. Concerning Presentation of a Route for a Mobile Robot Based on Visual Guides // Mekhatronika, Avtomatizatsiya, Upravlenie. 2017. No 2 (18). pp. 81–89. (in Russian)
[9] Karpov V.E. About Some Mechanisms of Parasite Manipulation of Robot's Behaviour Kaufering: b-Quadrat Verlag, 2016.pp. 241–245. (in Russian)
[10] Koenig N., Howard A. Design and use paradigms for gazebo, an open-source multi-robot simulator // 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566). 2004. (3). pp. 2149–2154.
[11] Lytinen S.L., Railsback S.F. The evolution of agent-based simulation platforms: a review of NetLogo 5.0 and ReLogo // European Meetings on Cybernetics and Systems Research. 2010. pp. 1–11.
[12] Maes P. Artificial life meets entertainment: lifelike autonomous agents // Communications of the ACM. 1995. No 11 (38). pp. 108–114.
[13] Malyshev A.A. Research and Development Algorithms and Models of Searching Energy Sources by Robot. 2017. (in Russian)
[14] Quigley M. [and oth.]. ROS: an open-source Robot Operating System 2009. 5 p.
[15] Rovbo M.A., Malyshev A.A. Energetically self-sufficient robot group study kit // Open Education. 2017. No 2 (18). pp. 68–77. (in Russian)
[16] Rovbo M.A., Ovsyannikova E.E., Chumachenko A.A. Review of simulation modeling tools for robot groups with social organization elements // Software & Systems. 2017. No 3 (30). pp. 425–434. (in Russian)
[17] Russell S.J. [and oth..]. A Modern Approach, 1995. 106-10 p.
[18]. Wendel T. AntMe [Electronic resource]. URL: https://service.antme.net/ (date of access: 01.01.2017).
[19] Wilensky U. NetLogo Ants model // Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL [Electronic resource]. URL: http://ccl.northwestern.edu/netlogo/models/Ants (date of access: 05.16.2017).
[20] Wilensky U. NetLogo // Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL [Electronic resource]. URL: http://ccl.northwestern.edu/netlogo/ (date of access: 16.05.2017).