Pomiary Automatyka Robotyka, ISSN 1427-9126, R. 23, Nr 4/2019, 25–30, DOI: 10.14313/PAR_234/25

Mobile Robot Simulation Framework

Maciej Ciurej
AGH Akademia Górniczo-Hutnicza, Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej, Al. A. Mickiewicza 30, 30-059 Kraków

Abstract: This paper presents a programming framework for simulation of the mobile robot. Model of Khepera III robot with IR proximity sensors was considered as a base model for tests of proposed system. Architecture of designed software was presented with the use of UML class diagram.

Keywords: mobile robotics, trajectory, simulation, agent based, programming framework

1. Introduction

Simulation of the mobile robot trajectory can be performed with the use of several specialized and dedicated robot simulation platforms. Among a vast of possibilities, one of the most known is Visual Robot Experimentation Platform (V-REP) [13]. It includes significant number of useful modules, as well as its own simulation framework. Process of modelling and further simulation with the use of V-REP environment was presented in [8]. Same software bundle, but used to propose conditioned anxiety mechanism in mobile robotics, can be found in [7]. Overview of path planning using V-REP was presented in [10].

Another example is ARGoS environment, which allows user to perform robot simulation, however it lacks several features which can be found in V-REP package. It was designed to cover the area of large-scale heterogeneous robot swarms simulation which was not provided by already existed simulators [2]. ARGoS simulation of 10 marXbot robots performing a dispersion behavior can be found in [12].

Gazebo is yet another simulation framework used e.g. in mobile robotics. It is still actively developed since 2002. Among large set of features, Gazebo introduces components such as Dynamics Simulation, Advanced 3D Graphics and Cloud Simulation – which allows user to run simulation on Amazon AWS and GzWeb. Example of simulation system based on ROS and Gazebo for RoboCup Middle Size League is described in [3]. More insight on differences between V-REP and Gazebo, can be found in [6].

To contrast an approach of great and powerful simulations packages (and its variety of possible usage), this paper is intended to propose minimalistic version of simulation framework. Author took an effort to create a proposal of the software framework architecture, which can be easily implemented.

Main motivation of this paper is to present flexible structure of the simulation framework, which can be implemented independently with the use of open source components, by anyone interested in such an area of study.

Architecture of proposed solution was introduced with the use of UML notation, which has numerous examples of usage across several fields of study. Proposal of robot behavior modelling with the use of UML notation was presented in [1]. Use of UML in modelling of multi-agent system (MAS) was presented in [9]. Additionally to the UML usage, proposed solution is based on agent, state-space model of mobile robot. Agent approach presented in [11], which incorporated elements from systematic way of designing control systems for fields robot [4] was used. Simulations presented in this paper employed Braitenberg algorithm to control the robot during movement to the target. Additionally, discrete model of kinematics presented in [11] was used.

2. Preliminaries

2.1. State-space agent model

Adjusted agent model for Khepera III robot proposed in [11] is presented below:

$$A = \{p_n,\ e_m,\ c,\ f\}, \tag{1}$$

where p and e are sets of virtual receptors and effectors, c is a control system and f is a nonlinear transition function.

In a simulation of robot movement, virtual receptors and virtual effectors can be treated as equal to its real equivalents. In real, physical systems it is needed to make clear distinction between real and virtual pairs of effectors and receptors. It is essential to organize data transfer and responsibilities between abstraction layers: real and virtual one. More insight on this approach can be found in universal model of robotic system [4]. During software simulation this case is not valid. Lack of additional system physical layer of the system concludes no distinction between virtual and real receptors and effectors (issue of e.g. data transfer and its pre and post-processing is not applicable).

Accordingly, three basic transition functions needs to be defined. For the considered model of the robot, goal of the simulation is to reach final destination (defined as two-dimensional restricted area). In each step on its way to the target, one of the three function is executed: transition function (f), arrival to target function (f^{τ}) or error function (f^{err}). Possible transitions between those functions are presented in figure 1.

Autor korespondujący:
Maciej Ciurej, mciurej@agh.edu.pl

Artykuł recenzowany
nadesłany 19.06.2019 r., przyjęty do druku 26.11.2019 r.

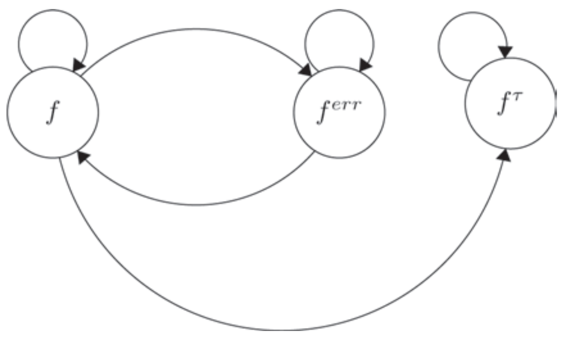

Fig. 1. Transitions between simulation functions
Rys. 1. Diagram tranzycji pomiędzy funkcjami symulacji

3. Framework structure

Internal structure of an embodied agent presented in [4] can be taken as an aggregation of control subsystem, virtual/real receptors and effectors. Selection of current behavior of the agent, implies data exchange between each of the internal modules.

Simulation of a mobile robot based on such an approach required architecture based on object oriented programming (OOP) [5]. Beside relations inside of an agent, simulations object needs to be defined in order to constraint all elements and actions required during single trial.

3.1. Components of the simulation

UML class diagram is presented in figure 2. Class *Model* provides a concept of the robot representation. Class *Robot* extends base class with robot dependent variables declarations.

Definitions of *Receptor* and *Effector* were separated from *Model* in order to provide atomization of each component. This approach allows the user to enhance code with the new solutions, e.g. definition of a class extends *Receptor* or *Effector*. *Model* is composed with the use of *Effector* and *Receptor* instances. Key functions aforementioned in 2.1 were defined as methods of *Model* class and can be extended with the use of *Robot* methods definition.

Object of class *Simulation* itself is setting up preliminary values for desired simulation, such as: start position of the mobile robot, position of the target, maximum number of iterations (steps) to be performed. It is composed with the use of *Robot* class instance and *Obstacle* class instance. According to figure 2, *Obstacle* object instance is not mandatory to perform the simulation.

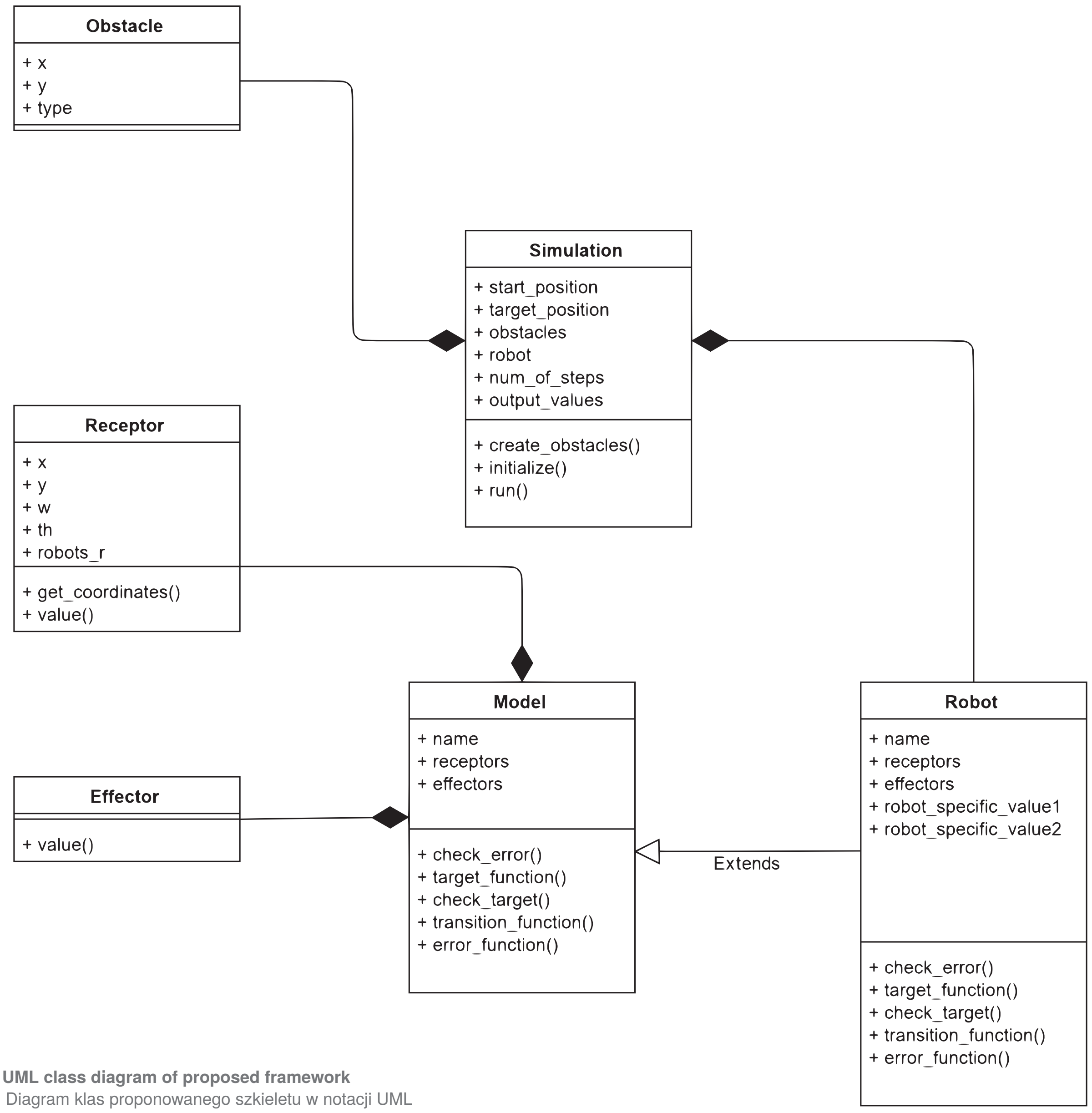

Fig. 2. UML class diagram of proposed framework
Rys. 2. Diagram klas proponowanego szkieletu w notacji UML

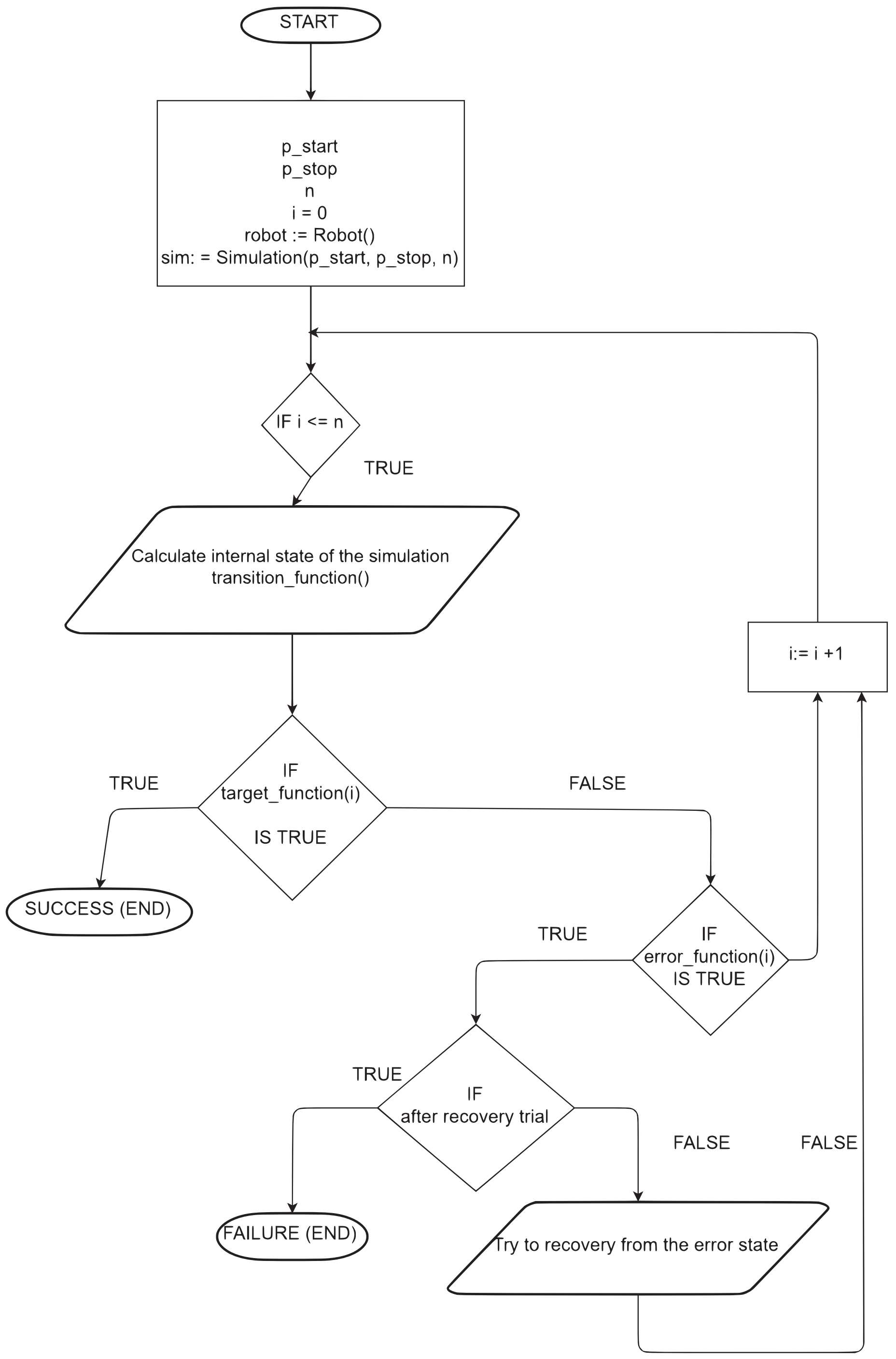

Fig. 3. Proposed algorithm of a simulation
Rys. 3. Proponowany algorytm symulacji

3.2. Simulation algorithm

In order to perform a single simulation, there is a need to create an instance of class *Simulation* with desired parameters. Starting and target points coordinates as well as maximum number of iterations (allowed number of steps) must be defined. Schema of the simulation algorithm is presented in figure 3.

4. Simulations

Implementation of the simulation framework, beside the source code which incorporate hierarchy from figure 2, requires additional components. What can be observe in figure 3, application executes framework logic and stores all of the results values such as current state or placement of the obstacles presented across the simulation environment.

Graphical User Interface (GUI) should be in place as it takes part in evaluation of the results correctness. Example of components types and relations is shown in figure 4.

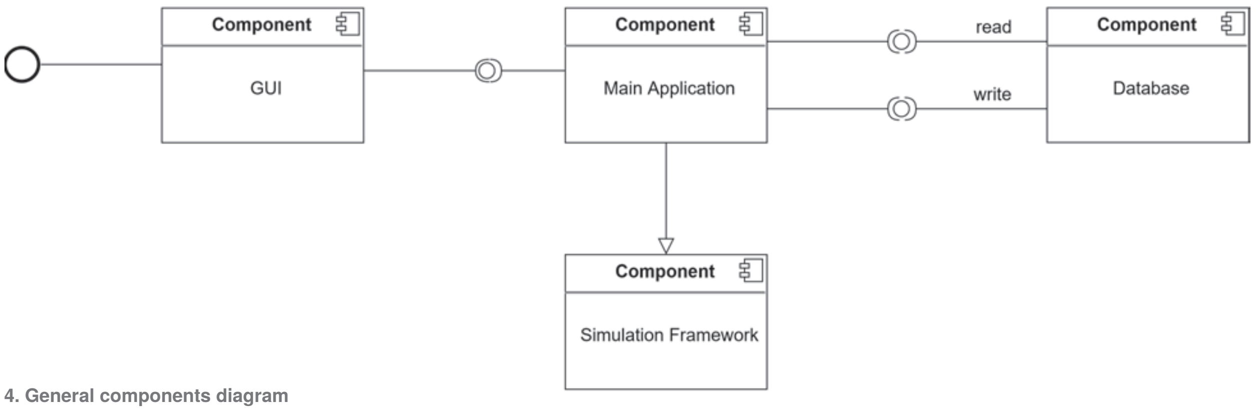

Fig. 4. General components diagram
Rys. 4. Diagram komponentów wchodzących w skład rozwiązania

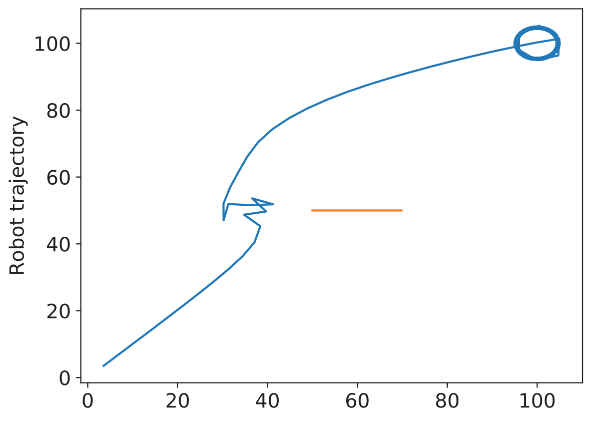

Fig. 5. Trajectory simulation with obstacle as oblong shape
Rys. 5. Symulacja trajektorii z przeszkodą o podłużnym kształcie

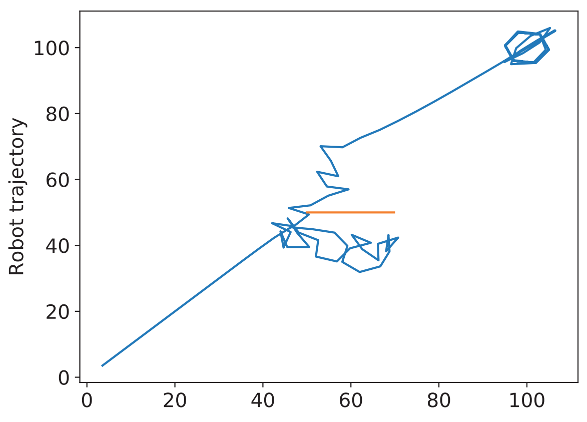

Fig. 8. Trajectory simulation with obstacle as oblong shape with defect of frontal sensors
Rys. 8. Symulacja trajektorii z przeszkodą o podłużnym kształcie oraz defektem przednich sensorów

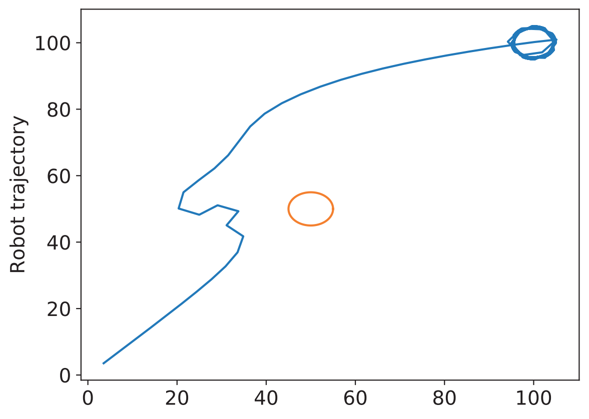

Fig. 6. Trajectory simulation with obstacle as circle
Rys. 6. Symulacja trajektorii z przeszkodą o okrągłym kształcie

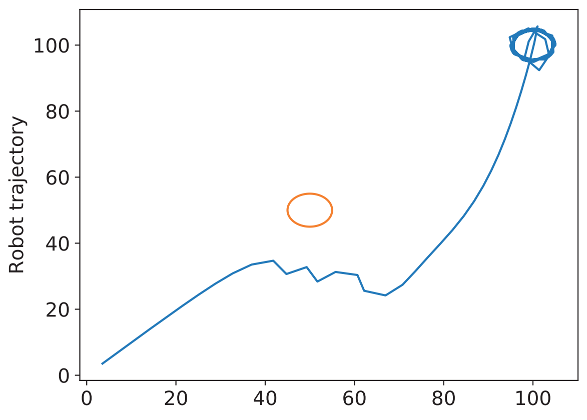

Fig. 9. Trajectory simulation with obstacle as circle with defect of right-side sensors
Rys. 9. Symulacja trajektorii z przeszkodą o okrągłym kształcie oraz defektem prawostronnych sensorów

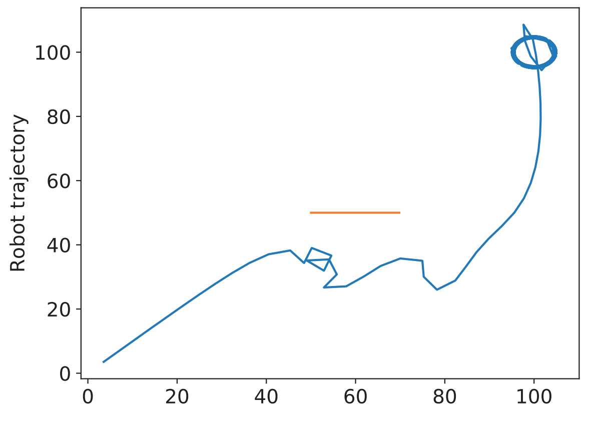

Fig. 7. Trajectory simulation with obstacle as oblong shape with defect of right-side sensors
Rys. 7. Symulacja trajektorii z przeszkodą o podłużnym kształcie oraz defektem prawostronnych sensorów

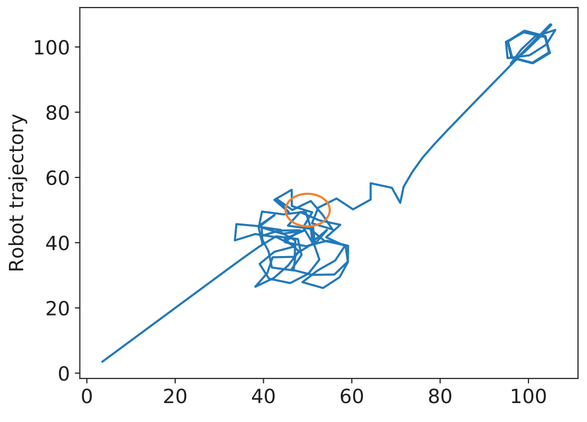

Fig. 10. Trajectory simulation with obstacle as circle with defect of frontal sensors
Rys. 10. Symulacja trajektorii z przeszkodą o okrągłym kształcie oraz defektem przednich sensorów

To verify correctness of the proposed solution, several tests (related to those presented in [11]) has been performed. Khepera mobile robot model was used. In figure 5 simulation with single obstacle defined as oblong shape has been presented. In figure 6 simulation with single obstacle defined as circle has been presented. For the test purposes, number of receptors and its weights were defined same way as in [11]. Additionally, two more test scenarios for each of those two types of obstacle shapes (oblong and circle) were defined: first one with the defect of left-side sensors, second one with the defect of frontal set of sensors.

What can be observed in each of the proposed test schemas, robot behaved as it was expected, especially in scenarios, when particular group of sensors were not used by the simulations. While right-side sensors were deactivated, left side sensors provided data, which was enough to conduct robot trajectory in the right side of the obstacle. Figure 10 presents the vulnerability of simulation algorithm. Error function (f^{err}) implementation and enhancement of the mechanism responsible for the recovery from the error state needs an improvement in order to avoid such cases in the future simulations.

The goal of described simulations was to verify correctness of simple behaviors of framework as a whole. Feasibility of applying changes inside of its definition was crucial to achieve. Framework can be easily modified across all of the modules defined in entire solution. Separation of receptors and effectors modules from control module, allows user to introduce specific changes only in the scope of particular single responsibility of each module.

5. Final conclusions

In this paper, simulation framework for mobile robot based on agent approach was presented. Simulation tests with Khepera III model applied were successfully conducted with the use of Braitenberg algorithm. Architecture of proposed solution can be generalized onto several classes of mobile robots – availability to specify core modules of described framework is required. It can be enhanced with custom, specialized elements (objects) of simulation in order to provide more detailed scenarios of use.

Modules separation is the very first step on the way to customize simulation in order to fulfil all of the users' requirements. Built solution will help the author in research of several mobile robots movements in bounded environment. It can also help in investigation of e.g. conditioned anxiety of mobile robots.

Further improvements of proposed system can be performed by implementing all of the steps of systematic method of designing control systems presented in [4]. For example extraction of behavior selection automaton into separate class can be performed in order to provide more flexibility inside framework architecture.

References

1. Gogolla M., Vallecillo A., (*An Example for*) *Formally Modeling Robot Behavior with UML and OCL*. Software Technologies: Applications and Foundations 2018, 232–246. Springer International Publishing.
2. Pinciroli C., Trianni V., O'Grady R., Pini G., Brutschy A., Brambilla M., Dorigo M., *ARGoS: A modular, multi-engine simulator for heterogeneous swarm robotics*. IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2011, DOI: 10.1109/IROS.2011.6094829.
3. Yao W., Dai W., Xiao J., Lu H., Zheng Z., *A simulation system based on ROS and Gazebo for RoboCup Middle Size League*. 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO), 2015, DOI: 10.1109/ROBIO.2015.7414623.
4. Zieliński C., Kornuta T., Winiarski T., *A systematic method of designing control systems for service and field robots*. 19th International Conference on Methods and Models in Automation and Robotics (MMAR), 2014, DOI: 10.1109/MMAR.2014.6957317.
5. Mitchell J., *Concepts in programming languages*. Cambridge, UK New York: Cambridge University Press, 2003.
6. Nogueira L., Comparative Analysis between Gazebo and V-REP Robotic Simulators. Unpublished, 2014.
7. Bielecki A., Bielecka M., Bielecki P., *Conditioned Anxiety Mechanism as a Basis for a Procedure of Control Module of an Autonomous Robot*. Artificial Intelligence and Soft Computing, Springer International Publishing. 2017, 390–398, DOI: 10.1007/978-3-319-59060-8_35.
8. Ciszewski M., Mitka Ł., Buratowski T., Giergiel M., *Modeling and Simulation of a Tracked Mobile Inspection Robot in Matlab and V-Rep Software*, "Journal of Automation, Mobile Robotics and Intelligent Systems", Vol. 11, No. 2, 2017, 5–11, DOI: 10.14313/JAMRIS_2-2017/11.
9. Xiong L., Wei C., Zhenkun Z., Zekai H., Jing G., *Modeling for Robotic Soccer Simulation Team Based on UML*. 2008 International Conference on Computer Science and Software Engineering. IEEE, DOI: 10.1109/CSSE.2008.1462.
10. Reddy K.K., Praveen K., *Path Planning Using VREP*, "International Journal of Research in Engineering and Technology", Vol. 2, No. 9, 2013, 94–97.
11. Oprzędkiewicz K., Ciurej M., Garbacz M. *The agent, state--space model of the mobile robot*. „Pomiary Automatyka Robotyka", Vol. 22, No. 3, 2018, 41–50, DOI: 10.14313/PAR_229/41.
12. O'Keeffe J., Tarapore D., Millard A.G., Timmis J., *Towards Fault Diagnosis in Robot Swarms: An Online Behaviour Characterisation Approach*. Towards Autonomous Robotic Systems, 2017, 393–407, Springer International, DOI: 10.1007/978-3-319-64107-2_31.
13. Rohmer E., Singh S.P.N., Freese M., *V-REP: A versatile and scalable robot simulation framework*. 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, DOI: 10.1109/IROS.2013.6696520.

Szkielet oprogramowania do symulacji robota mobilnego

Streszczenie: W artykule zaprezentowano szkielet oprogramowania do symulacji kołowego robota mobilnego. Jako przykład rzeczywistego robota przedstawiono robot Khepera III z czujnikami IR do wykrywania i omijania przeszkód. Zaprezentowany szkielet jest niezależny od fizycznej reprezentacji robota mobilnego.

Słowa kluczowe: robotyka mobilna, symulacja, szkielet oprogramowania

Maciej Ciurej, MSc Eng.
mciurej@agh.edu.pl
ORCID: 0000-0001-8714-3255

Graduated (2015) from the AGH University of Science and Technology. He is a PhD student at Faculty of Electrical Engineering, Automatics, Computer Science and Biomedical Engineering of AGH UST. His current research interests include: mobile robotics, artificial intelligence and machine learning.