Ministry of Education of Republic of Belarus
Brest State Technical University

Robotics and Artificial Intelligence. Problems and perspective

PROCEEDINGS of International Conference
4-6 November 2013

Brest 2014

УДК 004.896
ББК 32.816

Робототехника и искусственный интеллект. Проблемы и перспективы: материалы международной научно-практической конференции, Брест, 4-6 ноября 2013 г. / Брестский государственный технический университет; под ред. В.С. Рубанова [и др.] – Брест: изд-во БрГТУ, 2013. – 60 с.

ISBN 978-985-493-313-9

Представлены статьи по современным проблемам робототехники в научных и прикладных исследованиях, распознавания образов и анализа изображений, искусственного интеллекта и нейронных сетей. Представлены проекты Студенческой научно-исследовательской лаборатории «Робототехника».

УДК 004.896
ББК 32.816

ISBN 978-985-493-313-9

COMMITTEES

Program Committee of RAIPAP 13

Chair: Vladimir Rubanau, Brest State Technical University, Belarus

Vice-chair: Vladimir Golovko, Brest State Technical University, Belarus
Valery Raketsky, Brest State Technical University, Belarus
Akira Imada, Brest State Technical University, Belarus
Rauf Sadykhov, Belarus State University of Informatics and Radioelectronics, Belarus
Vladimir Golenkov, Belarus State University of Informatics and Radioelectronics, Belarus
Victor Krasnoproshin, Belarus State University, Belarus
Alexander Doudkin, National Academy of Science, Belarus
Alexander Krot, National Academy of Science, Belarus
Arunas Lipniskas, Kaunas Technical University, Lithuania
Kurosh Madani, University of Paris XII, France
Ralf Stetter, Ravensburg-Weingarten University, Germany
Andreas Pachinsky, Ravensburg-Weingarten University, Germany
Hubert Roth, University of Siegen, Germany
Grzegorz Granosik, Lodz Technical University, Poland
Richard Balogh, Bratislava State Technical University, Slovak
David Obrdzalek, Charles University, Czech Republic
Dmitriy Bagaev, Kovrov State Technical Academy, Russia
Vladimir Ivahiv, Lviv Politechnic University, Ukraine
Anatoly Sachenko, Ternopil Technical Academy, Ukraine

Organizing Committee

Andrey Dunets
Valery Kasyanik
Uladzimir Dziomin
Ivan Dunets
Anton Kabysh

GRIP RECOGNITION AND CONTROL OF A THREE FINGER GRIPPER WITH A SENSOR GLOVE

Igor Zubrycki, Grzegorz Granosik
Lodz University of Technology
tel +48 42 6312554
Email: igor.zubrycki@dokt.p.lodz.pl, granosik@p.lodz.pl

Abstract—We propose the structure of the control system to intuitively operate the multi-finger gripper SDH from Schunk. The sensor glove can be used as an input device. However, the transformation from 5 fingers of human hand down to 3 fingers of the gripper is not trivial. Our approach includes a grip recognition phase followed by a grip execution phase. This paper shows some preliminary tests of the performance of this controller.

I. Introduction

This paper details the design of 3-finger gripper controller using a sensor glove as an input device. Proposed system has the ability to recognise different grip types and, as a result enables to control all 7 degrees of freedom of the 3-finger gripper with our simple in house made sensor glove. Paper describes sensor glove functionality and provides a comparison between two learning methods used to enable grip recognition.

As a continuation of our previous work [1], where we designed a system to control a virtual model of the 3-finger gripper, this paper describes some control aspects related to using real gripper and presents results of several tests performed on the 3-finger gripper SDH from Schunk.

II. Motivation

Dexterous grippers provide a method to manipulate objects in constrained environments in ways that would be otherwise impossible with only 6-dof manipulator and parallel-jaw grippers [2]. However, coordinated motion of such gripper in not trivial and, in fact, the whole system can be considered as a robot with three manipulators working in the common space. Therefore, controller not only has to produce grip that is optimal for a certain task, but also has to deal with transitions stages and avoiding self-collisions. In the proposed system, we have decided to use a sensor glove as a main input device. Intentions of the operator moving his or her hand (with a sensor glove) are translated to movements of 3-finger gripper having 7 DOF. This transformation has to be accurate and intuitive for the user.

Resulting solution provides a way to teleoperate a dexterous gripper without significant expertise [3] as well as gives us a platform for learning grasp methods from human demonstrations [4].

III. Proposed control structure

The proposed system to control 3-finger gripper consists of 7 parts (shown in Fig. 1):

1. Sensor glove which is a two-layer textile glove with flexion and force sensors attached to it. We have used resistive sensors and added appropriate voltage dividers to safely connect output voltages directly to A/D ports of Arduino board.

2. Data filtering program. This program reduces effects of the sensor noise and its temperature sensitivity.

3. Gesture recognition. Filtered sensor data are used to recognize the grip type. Program is based on the machine-learning algorithm and is explained in detail in further sections.

4. Planner of pose change. This program is used in case of the grip change. As different types of the grasp can require rapid changes of the poses of three fingers, this process may sometimes be danger (risk of the self-collision of SDH hand). To reduce the risk we propose the planner that would generate safe trajectories to change the pose in case of different grip recognition.

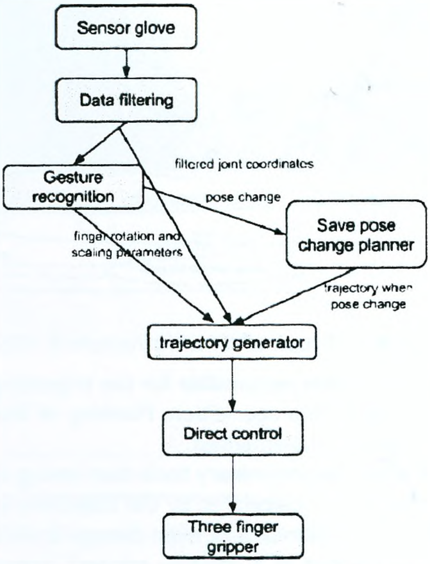


Figure 1 - Proposed control structure

5. Trajectory generator. This program consists of a state machine that manages the gripper behavior in different scenarios. For each recognized grip, the relation between readings from flex and force sensors and the pose of the gripper is different. Therefore, movements of the operator's fingers are transformed to the movements of 7 DOF in the specific way. In case of grip change the trajectory generator gives priority to the trajectory generated by the planner of pose change.

6. Direct control node. This node realizes the trajectory by sending velocity commands to the gripper, with position feedback from SDH hand. It also transfers to the system information about the current state of the gripper.

IV. Physical control layout

To realise the proposed control structure, we have implemented a ROS based network of nodes, each with some part of the aforementioned functionality, as shown in Fig. 2. To simplify development, nodes are implemented on two PC computers, interconnected by the TCP protocol, with master-slave configuration. Master computer has also the ability to control the slave's functionality (e.g, switching-on devices connected to it) using secure ssh connection.

Master computer has two key functionalities - direct control of the SDH gripper and managing communication between different ROS nodes using built-in DNS-like functionality.

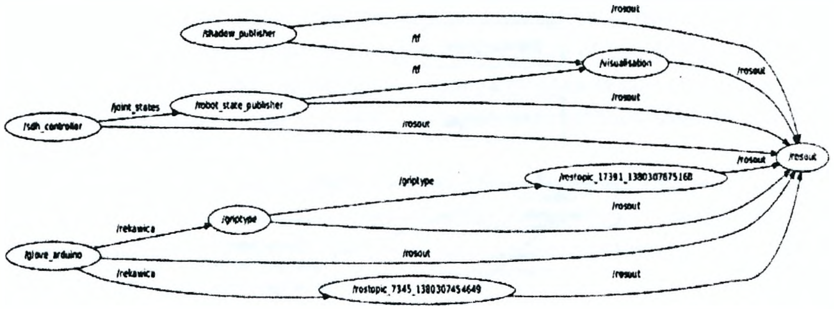


Figure 2 - Graph of control structure in ROS

Slave computer provides nodes responsible for the trajectory generation, gesture recognition, data filtering and data acquisition. Planning of the safe movement between different grips is not yet implemented.

We have observed during the preliminary tests that having visual presentation of the target position of the gripper calculated by the trajectory generator is very useful for the operator, who can calibrate it or even change it while gripper is still moving. Such visualization is provided as a "shadow gripper", shown in Fig. 3. Solid texture is used to present actual pose of the SDH hand while transparent picture represents output from the trajectory generator. When the system is in operation the solid picture follows the shadow. As the SDH hand is not very fast, that can be read from bode plots presented in Fig. 4 the operator has some time to make corrections in the grip before the fingers reach the object.

Visualisation in done in the tool available in the Robot Operating System named Rviz. It allows for easy presentation of 3D models, described by Unified Robot Description Format (URDF). Special node (robot_state_publisher, Fig. 2) computes forward kinematics for each coordinate frame attached to robot's joints, that further allows moving any object on the 3D scene (in our case 3D meshes described by COLLADA files).

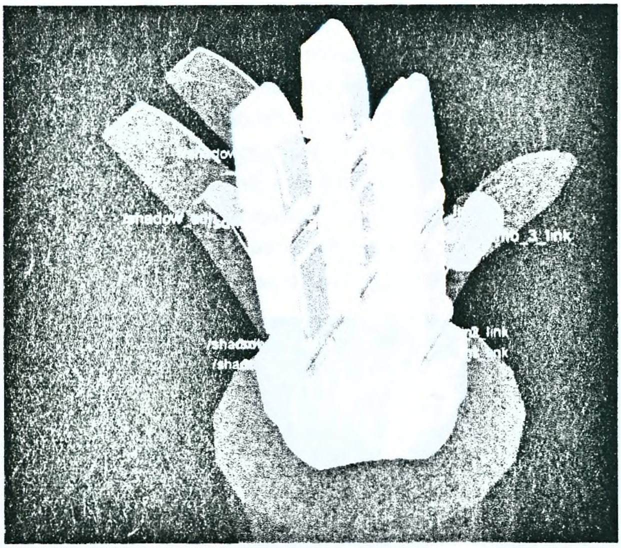


Figure 3 - Visualisation of gripper pose and shadow hand in Rviz

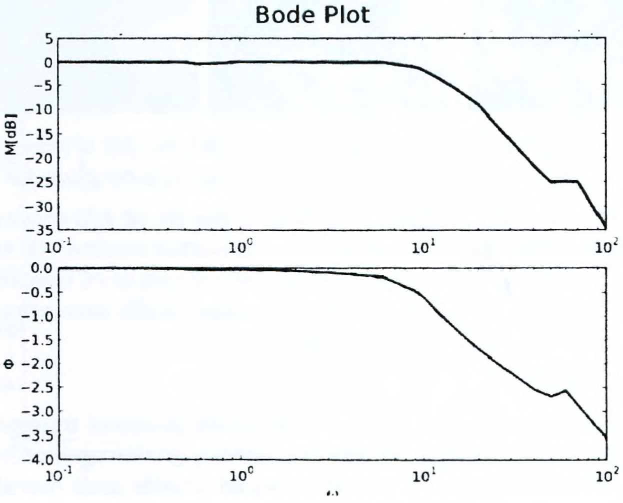


Figure 4 - Bode plot of gripper's joint

V. Sensor glove

Sensor glove designed for hand movement and grip estimation consists of 10 flex sensors: five positioned on the upper side of the hand, as shown in Fig. 5a, another set of five is positioned on the bottom side of hand as shown in Fig. 5b, the force sensors are mounted on fingertips and in the middle of palm, as shown in Fig. 5c. The two-directional FS Flex sensors have 25 kOhm flat resistance and maximally 125 kOhms of bend resistance. They are combined with 10 kOhm resistors to compose voltage dividers. Finger joints have limited range of movement of about 90 degrees. At this range all sensors had a nearly linear characteristic, as shown in Fig. 6. One can see that these characteristics vary in both inclination coefficient and offset from sensor to sensor between 18.7 kOhm and 28.8 kOhm. This called for the need of individual calibration for each sensor. Force sensors have much less linear characteristic in the range of 0-15 N, as shown in Fig. 7, but they are more homogenous in the set of 6 sensors we have tested.

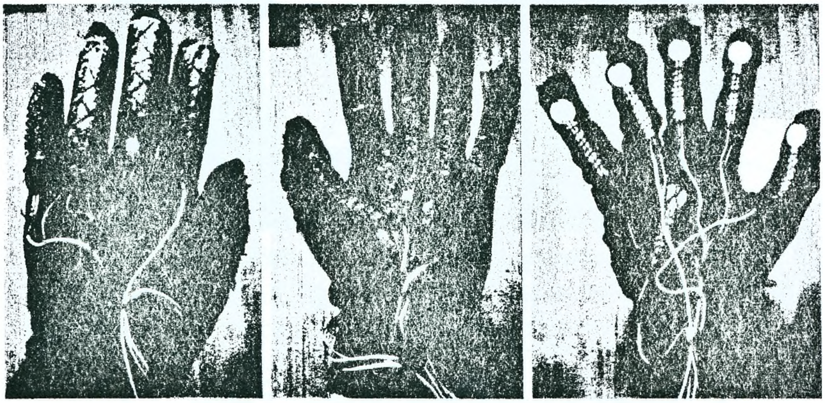


Figure 5 - Sensor placement on our sensor glove (for left hand) [5]: a) inner glove, upper side flex sensors, b) inner glove, bottom side flex sensors c) outer glove, force sensors

Sensor glove is connected to the 16 channels of the 10- bit A/D converter built in Arduino Mega 2560 board. The acquired data is formatted into the ROS Message directly on the Arduino Board. Then data is serialized and sent to PC computer by USB port. Special *rosserial*_node controls communication, checks data correctness and publishes topics created by Arduino on the ROS System.

VI. Learning grips

Robust and fast grip recognition is essential for the proposed trajectory generation scheme. There are several procedures for learning grip recognition from sensor data. We assumed that successful algorithm would provide good overall precision and recall, and would not require re-learning for each new user.

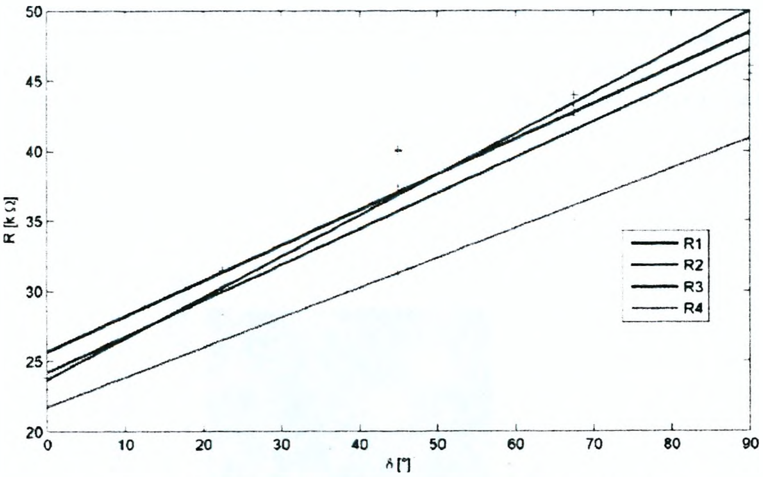


Figure 6 - Static characteristics of the flex sensors [?]
(delta - bending angle, R - sensor's resistance)

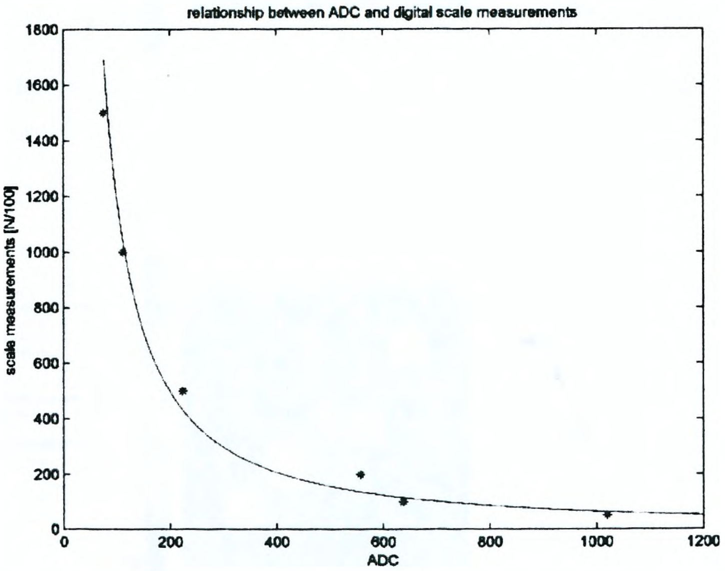


Figure 7 - Relationship between ADC and measured force

In our research we have tested two algorithms - Forest of Randomized Trees (FoRT) and Support Vector Classification (SVC) as they provide good performance and generalisability.

To teach our algorithms we have used 230 training vectors, that were acquired by asking six users to perform grips on different objects and with different intentions. Performance was tested on additional 70 vectors.

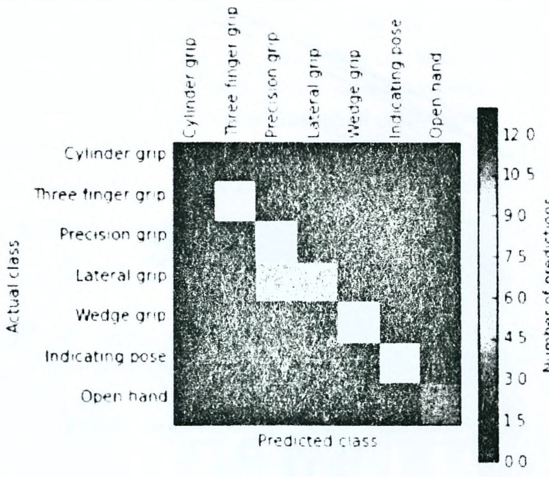


Figure 8 - Confusion matrix for Forest of Randomized Treees method

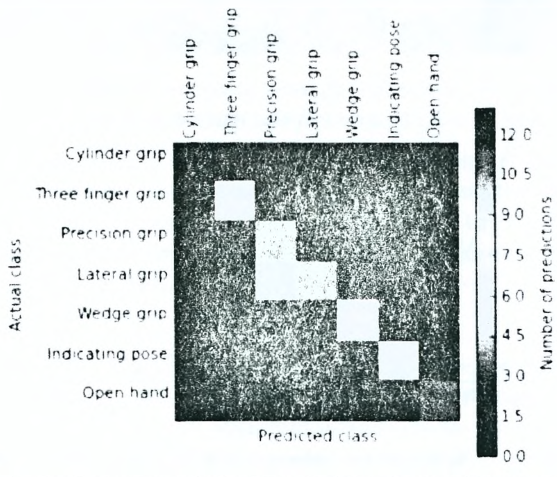


Figure 9 - Confusion matrix for Forest of Randomized Treees method

To find the best parameters for learning algorithms, we have used 10 fold cross validation. In case of Forests of Randomized Trees, we have used grid search for number of estimators (trees) ranging from 1 to 40 and minimum sample split ranging from 1 to 30, and maximum number of features ranging from 1 to 16. Best results were found for number of estimators 19, maximum number of features 9, and minimum sample split of 6. In that case, for test data algorithm had f-1 score of 0.77, precision 0.8 and recall 0.78. Confusion matrix is shown in Fig. 8.

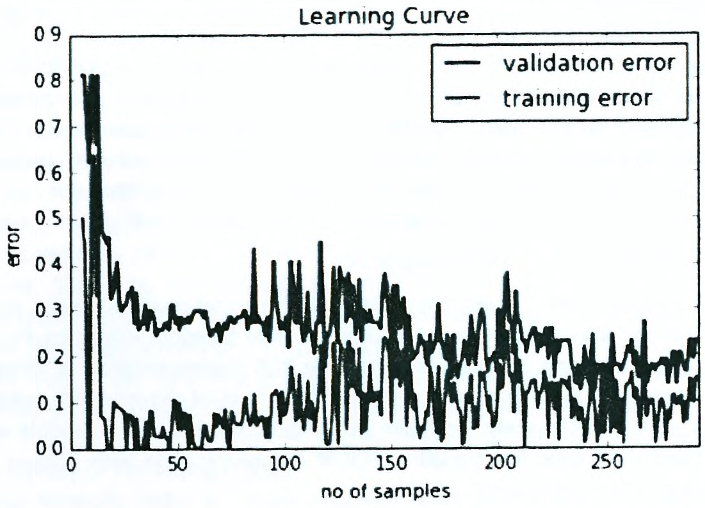


Figure 10 - Learning curves for SVM algorithm used for grip recognition

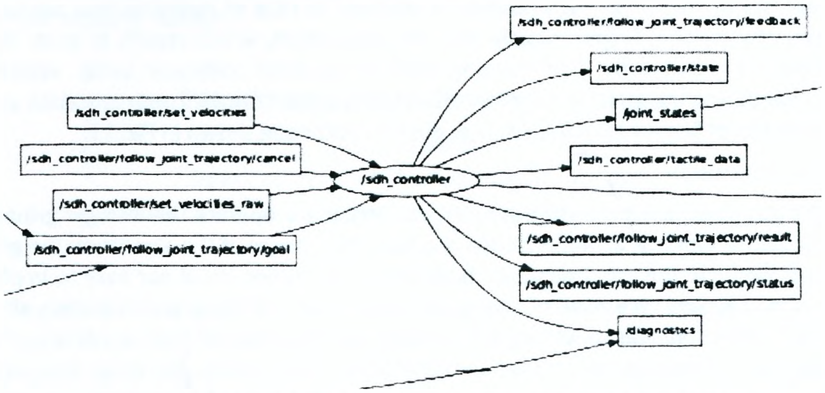

/sdh_controller/follow_joint_trajectory/feedback
/sdh_controller/state
/joint_states
/sdh_controller/set_velocities
/sdh_controller/follow_joint_trajectory/cancel
/sdh_controller/tactile_data
/sdh_controller
/sdh_controller/set_velocities_raw
/sdh_controller/follow_joint_trajectory/goal
/sdh_controller/follow_joint_trajectory/result
/sdh_controller/follow_joint_trajectory/status
/diagnostics

Figure 11 - Schunk SDH controller input/output flow diagram

Similar prediction results were obtained for the second of the algorithms we have tested with mean precision score of 0.78 and mean recall score of 0.77 (obtained f1 value 0.76). Confusion matrix (Fig. 9) shows that precision and lateral grips were the most difficult to tell apart.

In order to decide when we should finish the teaching process, we have conducted tests on in-sample (training) and out-of sample errors to assert the state of learning as a function of number of training vectors [6]. For both algorithms learning curves, shown in Fig. 10 indicate that errors are diverging, so we can assume that additional training would only marginally improve performance for the learning models chosen.

The biggest difference between tested algorithms was in the prediction performance. We have measured the prediction time for both algorithms on the randomly chosen data and for the 10000 repetitions. The results show that for the SVC algorithm single prediction takes on average 79 µs while for the FoRT algorithm the mean value is 2.65 ms, being over 30 times slower. Considering that our algorithm is going to be used for online grip recognition, we have chosen SVC.

VII. Schunk SDH 3 finger gripper properties

Schunk's SDH gripper is a very sophisticated device that enables precise control of a grip angle and a grip shape as it has 7 degrees of freedom driven by DC motors with low gear ratio and precise encoders. The ROS implementation of Gripper API (initially developed by [7]) allowed us to control most of the gripper's parameters directly from the ROS system. To make the original solution more flexible we have implemented additional functionality: TCP/IP communication with gripper (RS232 and CAN was already available).

To use the sensor glove to control the gripper we also had to change the behaviour of the gripper when receiving new position - which is normal in situation when operator's hand (wearing sensor glove) moves. The default hardware position controller has to stop after each position is reached. In case of receiving new position during the motion, fingers rapidly stop the movement, which results in jerks. We solved this problem by implementing software position controller using velocity commands and the position feedback from the gripper. Control, communication and diagnostics of Schunk SDH Gripper was put to a ROS node shown in Fig. 11.

VIII. Conclusion

We have proposed the control system to intuitively operate the 3-finger gripper using sensor glove as an input device and two main stages of the algorithm: the grip recognition and the grip execution. Both tasks are neither trivial nor easy to implement in real time. We have shown quite good results of the grip recognition after the SVC algorithm was taught by 230 training vectors obtained from 6 subjects. We have also tested some performance of the SDH hand, namely: dynamics and position control in the continuous teleoperation mode. We have shown effective control of the SDH hand using sensor glove and the shadow hand philosophy. The visualisation of the results of the transformation from the sensors readings to the expected gripper's movement helps operator to correctly show his/her intentions and on-line control the 3-finger gripper.

IX. Acknowledgment

Research partially supported by the National Centre for Research and Development under grant No. PBS1/A3/8/2012.

REFERENCES

[1] Zubrycki, Igor; Granosik, Grzegorz, "Test setup for multi-finger gripper control based on robot operating system (ROS)," Robot Motion and Control (RoMoCo), 2013 9th Workshop on , vol., no., pp.135,140, 3-5 July 2013 doi: 10.1109/RoMoCo.2013.6614598

[2] Onda, H., and T. Suehiro. "Motion planning and design of a dexterous gripper-graspability, manipulability, and virtual gripper." Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on. IEEE, 1998. - Vol. 1.

[3] Kang, Sing Bing, and Katsushi Ikeuchi. "Toward automatic robot instruction from perception-mapping human grasps to manipulator grasps." Robotics and Automation, IEEE Transactions on 13.1 (1997): 81-95.

[4] Lin, Yun, et al. "Learning grasping force from demonstration." Robotics and Automation (ICRA), 2012 IEEE International Conference on. IEEE, 2012.

[5] M. Fraszczyski, Interface design to control the multi-finger gripper, BSc Thesis, Lodz University of Technology, 2013

[6] Abu-Mostafa, Yaser S., Malik Magdon-Ismail, and Hsuan-Tien Lin. Learning from data. AMLBook, 2012. p.66-68

[7] Schunk Modular Robotics ROS Repository. Fraunhofer-Institut Produktionstechnik und Automatisierung, 2013. Web. 7 Oct 2013. https://github.com/ipa320/-schunk modular robotics

EVALUATION OF THE OPTONCDT ILR SENSOR

Richard Balogh
Slovak University of Technology in Bratislava
Faculty of Electrical Engineering and Information Technology
Bratislava, Slovakia
richard.balogh@stuba.sk

Abstract—In this paper we evaluate metrological properties of the optoNCDT ILR sensor and compare it with the producer's datasheet. Then we describe its use for the purposes of map creation in mobile robotics. Results are promising so we can use this sensor for cheap map-making mobile robot applications.

Keywords—laser scanner, distance sensor, mapping, mobile robotics

I. INTRODUCTION

In mobile robotics, we often use laser scanner sensors (e.g. Sick [1], [2] or Hokuyo rangers [3],[4]) as a primary source of information for map creation. Unfortunately, although their performance is very good, high price limits their use in laboratory experiments. Many other attempts to obtain reliable map of the environment were described, e.g. [5] or [6]. In this paper we describe our experiences with standard industrial laser distance sensor optoNCDT produced by the Micro-Epsilon company [7].

II. SENSOR DESCRIPTION

Sensor considered for tests is a compact and reliable laser distance sensor with an analogue output signal produced by the German company Micro-Epsilon Messtechnik. According to the producer, sensor is appropriate for monitoring and positioning of cranes, measuring the liquid level in a tank, measuring motion of the conveyors and positioning [7], see also Fig. 1.

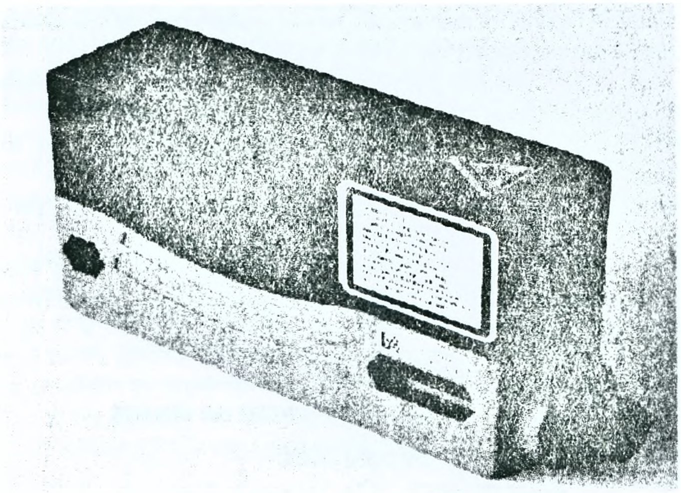


Figure 1 -Sensor optoNCDT ILR 1030-8 by the Micro-epsilon company.

The sensor operation principle is based on the measuring of the pulse propagation time (time-of-flight technology). The sensor sends a short (5 ns) pulse which is reflected from the object being measured and is then detected by the sensitive element of the sensor. Distance of the object is derived from the pulse flight time to the object and back. As a source of pulse is used a laser diode (Class 2) with a wavelength of 660 nm and beam divergence 1 mrad. Pulses of length 5 ns are repeated with a frequency of 250 kHz. This technology should minimize the impact of the scanned surface, properties of the material as well as the impact of lighting object and should thus always provide reliable and consistent data.

The sensor has an analog output 4-20 mA, the measurement range can be set in so called learning mode. In addition to the analog output, also single switching output is available. Its switching level again can be set in learning mode of the sensor. Values can not be entered exactly, only on the basis of the actual measurement. Effective measurement range is limited by the color of the target surface. For dark surfaces the effective range is more than half of the light ones (see Fig. 2).

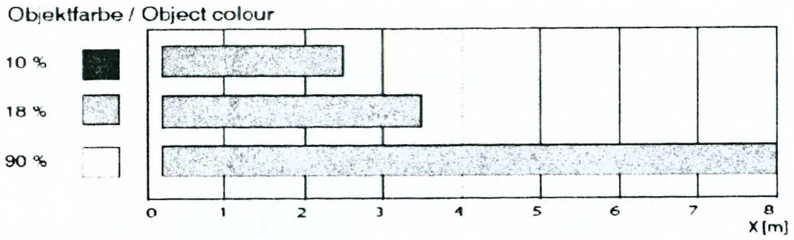


Figure 2 - Measuring ranges of the sensor for different colors of objects.

Table I -Technical Parameter

Measuring Range	0,2 ... 2,5m [a]
Linearity	±20 mm
Resolution	1 mm
Repeatability	<5 mm
Response time	10 ms
Operation temperature	-30° ... +50°C Protection class IP 65
Power Supply	10 - 30VDC, class 2
Weight	90g
Dimensions	25,8 x 54,3 x 88 mm (with connector 102)

a) for black surface, up to 8 m for white

III. MEASUREMENTS WITH THE SENSOR

Technical parameters of the sensor are completed in Tab. 1. We measured the metrological properties of the sensor according the EN 60 770 standard. We performed 3 sets of measurements for increasing and 3 for decreasing values, then following properties were calculated:

A. Inacuracy.

Maximum positive error was at 10 % input and its value is 0,5 %. Maximum negative error is at 90 % input and its value is -0,9 %.

B. Measured error

Maximum positive difference from an average is at 10% input and its value is 0,28 %. Maximum negative difference from an average is at 90 % input and its value is -0,58 %. Measured error is 0.58 %.

C. Non-linearity (terminal based)

Line connecting end points has a slope -0,003796 and offset 0,279630. Maximal deviation of average values from this line is at 90 % input and the value of non-linearity is -0,37 %.

D. Hysteresis

Maximum difference between the increasing and decreasing measurements is at 90% input and its value is 0,7 %.

E. Non-repeatibility

Non-repeatibility value is 0,75 at 85 % input.
We also measured additional sensor-specific properties of the device.

F. Size of the laser spot

We measured diameter of the measuring spot in various distances. Real diameter of the spot was difficult to measure due its high intensity which enlights also the surrounding, especially for small distances. It explains also paradox that spot decreases its diameter with increasing distance. For larger distances the intenzity of the spot is lower and measurement is more precise so we are able to measure just the spot itself and not the secondary lighted area. Results of the measurement are in Tab. 2

G. Influence of the surface color

Measurement on the 100 cm range show almost no dependance of the measured range on the color of the measured object. Results of the measurement are summarized in Tab.3.

H. Power consumption

When powering the sensor from the 12 V DC source (exact value 12,45 V) we measured current consumption 98,9 mA in steady state.

I. Dynamic properties

We measured dynamic properties of the sensor at the step change of the input distance. Distance was changed by the shutter in the laser beam line from the 50 to 100 cm using a position servo. Change of the output value was measured by the oscilloscope as a voltage across the measuring resistance. Measured response time was in the range 10-12 ms, once up to the 14 ms. Response time is not depending on the change direction.

TABLE I. SPOT DIMENSIONS

Distance [cm]	spot dimensions [mm]
50	4,8 x 4,8
100	4,8 x 4,8
150	2,4 x 4,8
200	2,5 x 4,4
250	2,5 x 4,4
300	2,5 x 2,5

TABLE II. COLOR SURFACES

color	current
D=100cm	[mA]
white	6,62
yellow	6,62
[illegible]	6,62
green	6,61
[illegible]	6,61
black	6,61
black foam	6,61
mirror	6,62

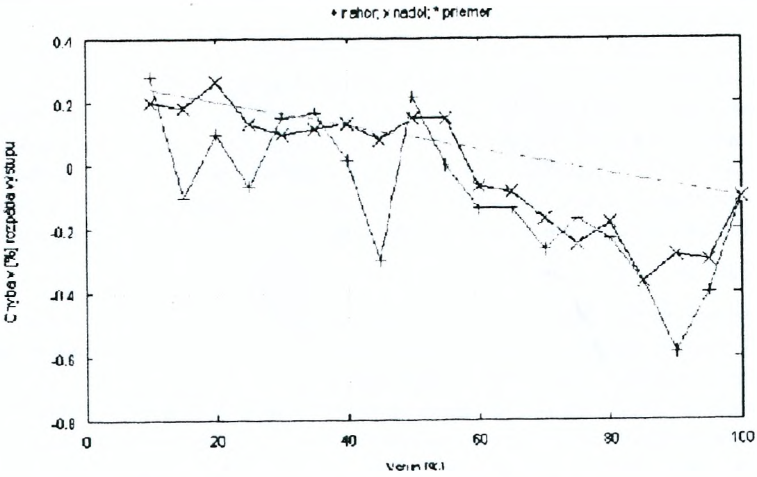


Figure 3 - Measured error of the sensor.

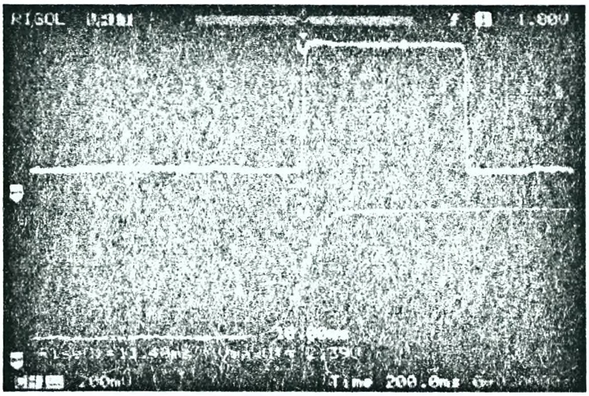


Figure 4 - Dynamic response of the sensor on the step change.

VI. MAP CREATION

For the purposes of the mobile robotics we tried to use this sensor for as a source of data for map creation. For this experiment, we mounted relatively light-weighted sensro on the positioning RC servo head. We were rotate the sensor in the range of 0 to 180 degrees with increments of 1 degree. The servo gear characteristics are unfortunately not quite linear. Then we converted measured value of the current by the 10-bit A/D converter (0-1023) with an input range of 0 to 5 V. Current of 4-20 mA was converted to a voltage using a 233 ohms resistor. Sensing range was for practical reasons limited to 0-200 cm.

Then we created a model room in which the robot has to move (see Fig. 5). The measurement was carried out statically from the point with coordinates [0,0]. During the measurement we exploited three different surface materials . The segments AB and EF white plastered wall , in the section DE ceramic tiles in the remaining sections of BD , FH and JL chipboard . Created model allows as to test several problematic parts - the intern corners ABC, CDE and EFG as well as outer corners BCD. Also we tested the behaviour of the sensor for holes (e.g. window, door, ...) GJ – see Fig. 6.

V. RESULTS

During the measurement the mobile robot with rotating sensor was placed at the point O. Readings with 1 degree increments were plotted in polar coordinates (angle in degrees, distance in cm) on the following diagram.

During the measurement we observed that too fast moving head results in delayed measurements. When faster movements occur the higher the variance of the measured values due to vibration throughout the head Experimental results proved the measured dynamic properties of the sensor and shows that it is impossible to perform a single measurement faster than each 10-12 ms.

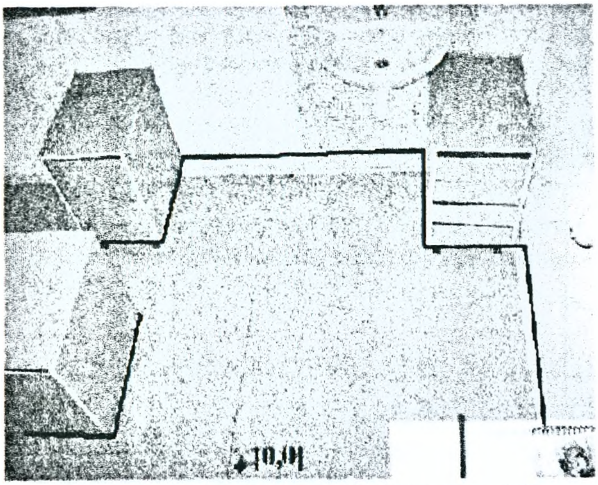


Figure 5 - Photo of the measured room model.

Measurements shows that the approximate time required for full scan in the range of 0 to 180 degrees with a resolution of 1 degree takes about 2,1 seconds. It doesn't seem to be appropriate for mobile robot, but relatively long time is balanced by the reliable and accurate measurement, which don't need to be repeated. Sensor can be also exploited for faster measurements with a possible compensation of dynamic errors or for measurements with more than 1 degree of resolution.

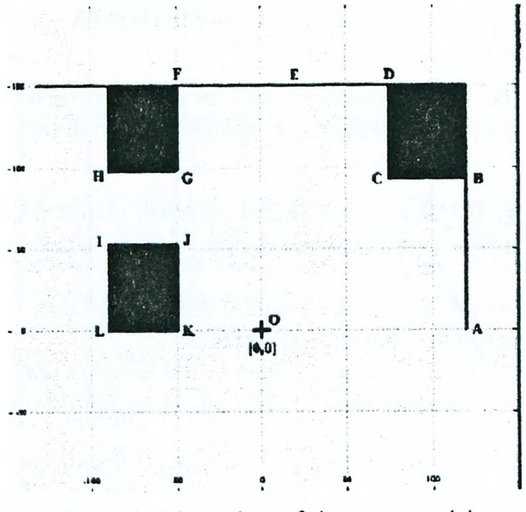


Figure 6 - Dimensions of the room model

VI. CONCLUSIONS

The present sensor ILR 1030-8 is a very promising component for robotics. After correcting the positioning head allows relatively precise distance measurements, it is robust and reliable. Its use is enforces its low weight (which can be further reduced by removing the sensor case). Its approximate price is around 280,-€.

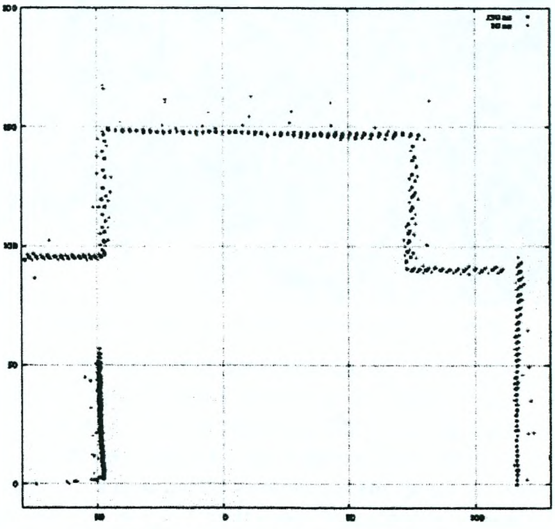


Figure 7 - Results of the measurements.

ACKNOWLEDGMENT

Above experiments were performed thanks to the company Micro-Epsilon Messtechnik, Czech division. We appreciate their helpfuilnes when lending the sensor device.

REFERENCES

[1] Teresa Escrig: The SICK laser sensor is currently mandatory for autonomous robots. Available on-line: <http://teresaescrig.com/the-sick-laser-sensor-is-currently-mandatory/>

[2] SICK Inc.: Solutions for Robotic industry. Available on-line: <http://www.sick-automation.ru/images/File/pdf/Industry/robots.pdf>

[3] Kneip, Laurent, et al. "Characterization of the compact Hokuyo URG-04LX 2D laser range scanner." Robotics and Automation, 2009. ICRA'09. IEEE International Conference on. IEEE, 2009.

[4] Okubo, Yoichi, Cang Ye, and Johann Borenstein. "Characterization of the Hokuyo URG-04LX laser rangefinder for mobile robot obstacle negotiation." SPIE Defense, Security, and Sensing. International Society for Optics and Photonics, 2009.

[5] Ye, Cang, and Johann Borenstein. "Characterization of a 2D laser scanner for mobile robot obstacle negotiation." Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on. IEEE, 2002. - Vol. 3.

[6] Duchon, F., Dekan M., Jurisica, L. and Vitko A. "Some applications of laser rangefinder in mobile robotics." Journal of Control Engineering and Applied Informatics 14.2 (2012): 50-57.

[7] Laser distance sensors optoNCDT ILR 1030/1031. Micro-Epsilon Messtechnik. Available on-line: <http://www.micro-epsilon.com/ displacement-position-sensors/-laser-distance-sensor/optoNCDT_ILR_ 1030_1031 /index.html>

LANGUAGE OF DESIGN OF MODEL OF BEHAVIOR OF COMPLEX SYSTEM

Harchenko S.L., Bagaev D.V.
Kharkov National University of Radioelectronics
Kharkov, Ukraine
Kovrov State Technological Academy named after V.A. Degtyarev
Kovrov, Russia
khsln@yandex.ru, dmitrybag@gmail.com

Abstract – The paper deals with creating a complex pattern of behavior controls technical system based on the use of the formal language of design. Which provided visualization used in the model of concurrency control system behavior and the possibility of action sequences (the track) with the performance. The resulting pattern of behavior may be useful in the verification and validation of the control program of the technical system.

Keywords – pattern of behavior, a process, asynchronous parallel processes, multiprocessor control system, the graph model of behavior.

I. INTRODUCTION

One of the urgent tasks of software development - is the creation of programs for complex technical systems, such as robot control systems. Of the total list of works carried out in the solution of this problem in the early stages of design, it is possible to allocate the task of creating a model of the future behavior of the robot control system. Perform extensive quality indicators such work can be substantially increased if the design of the management model to use formal language that allows to visualize the execution trace, organization and interaction of parallel processes, using a multiprocessor system.

Besides, complexity of modern control systems demands to provide application of the multiuser operating mode over the project. That provides possibility of simultaneous work on the project to several groups of the independent developers united by one ultimate goal.

Arising needs for resources when performing such works, indicate the need of application of a various automation equipment without which application time implementation of the project much more exceeds admissible intervals of expectation of the end result.

Searches of the complex solution of such complex problems of design led to that one of possible versions of the solution of a task is realization of methodology of design which is based on creation of model of behavior of the control system which description can be automatically transformed to sequence equivalent to it (or to some equivalent sequences) operators of one of programming languages.

As the consensus about methodology of creation of the software product in which decisions with use of series-parallel algorithm or functioning are applied as asynchronous parallel processes isn't present, it is possible to choose representation of model of behavior in the form of the count.

Such approach will allow to display through columns all options of branching in algorithm of model of behavior, and action which are correlated to edges of the count, can be written down in any formal language. The record form a file script can serve one of such examples of record.

It is obvious that realization of such model of behavior imposes restrictions on a choice of an operating system. If to consider definition "action" as any sequence of executed operators, such approach is characteristic for UNIX of similar operating systems in which such sequence of operators is treated as process. For such OS, performance of process can be considered as execution of operators by virtual car. In this case the virtual cars created by an operating system have opportunity to communicate among themselves through the general resources of the environment of performance of processes.

II. PROBLEM DEFINITION

For today software creation for difficult control systems is one of actual tasks. In the general list of works which are performed at the solution of such task at early design stages, there is a creation of model of behavior of future system. Performance of this work can baht is significantly accelerated if to use formal language at design of model of behavior of a difficult control system. Especially it should be noted need for creation of the mechanism of visualization of designs of language of design which display overlapping at realization of interactions a component of a difficult control system.

One of the main requirements to language of design of model of the behavior, difficult control system, will be that the created designs of language of design of model of behavior have to be suitable for machining. It will allow to transform further by formal rules the description of the designed model of behavior of a difficult control system to sequence of operators of one of programming languages.

Formal language of design of model of behavior the count for visualization of interactions uses a component and branchings of sequences of executed actions in a control system. In such count this or that action of model of behavior is associated with each edge, and tops of the count display current state of process of modeling. On the other hand, the model of behavior needs to be considered as component structure which can декомпозировать any element to an indivisible condition (component). And everyone to a component, at all levels of decomposition, model of behavior it is possible to consider as process in a control system.

I ruled records of action which is associated with an edge or top of the count, are regulated by the interpreter applied in the project for which the set of rules of analysis and interpretation of an entrance stream is in advance defined.

III. Process in behavior model

If we call the count of model of behavior as *P*, and in this count of an edge of the count will designate actions, and condition tops. In this case functioning of model of behavior can be considered as system transitions on actions from one condition (s_i) in another. Generally the model of behavior can be presented as a set of actions of *Act(P)* which *P* process, thus, for any process can execute

$$Act(P) \subseteq Act,$$

where *Act* set of all possible actions.

The choice of a set of actions of model of behavior depends on a situation for which some set of actions is characteristic. If to divide a set of actions on entrance $\alpha?$ days off $\alpha!$ and internal τ i.e. which aren't connected with environment, it is possible to designate an infinite set of names of objects of process (*Names*) which can be entered or removed. Then the set of actions can be defined as

$$Act = \{\alpha? | \alpha \in Names\} \cup \{\alpha! | \alpha \in Names\} \cup \{\tau\}.$$

In this case process in model of behavior can be defined as

$$P = (S, s^0, R),$$

where S – a set of conditions of *P*;

s^0 – an initial condition of *P*, s^0, *S*;

R – look transitions (s_1, α, s_2), where subset of a type of $R \subseteq S \times Act \times S$.

And process of *P*, in behavior model, it is possible to consider as passing of a set of transitions of a type of $s^{0a0} \to s_1^{a1} \to s_2$ and performance of actions of *a0*, *a1*, ... Process will continue work until then while there is a transition from *R* and will stop work in the absence of that.

Let's define a set of all actions of process of *P* as *Act*(*P*) from *Act*\{τ}, i.e.

$$Act(P) \overset{def}{=} \{\alpha \in Act \setminus \{\tau\} | \exists (s_1^a \to s_2) \in R\}.$$

Process will be final, if sets of *S* and *R* have the termination.

The final sequence of *a0, a1, ...* set of *Act* for which there is a sequence of conditions of *s0, s1, s2, ... P* process will be the route of process of *P*. In this case the set of all routes of process can be designated *P* models of behavior as *Tr*(*P*).

If to remove all unattainable conditions and all transitions at which there are unattainable conditions, from process of model of behavior the turned-out process of *P'* (the achievable part of process of *P*) will have the same behavior, as well as *P* process therefore they can be considered as identical.

During the work with model of behavior replacements of conditions in the course of *P* are possible. If we replace $s \in S$ with any $s' \in S$ and we will designate process as *P'* which turns out from *P* by replacement of *s* by *s'* in sets of *S* and *R*, transition of a type of $s^{a} \to s_1$ can be replaced with transition of a type of $s'^{a} \to s_1$. In this case *P'* process will possess the same behavior as *P*. If to make multiple replacement of conditions in the course of *P*, and to consider it as replacement of a subset of conditions of process, display of this process can be presented as $f: S \to S'$ and result will be

$$P' = (S'', (s')^0, R'),$$

where $(s')^0 \overset{def}{=} f(s^0)$ and for everyone vapors $s_1, s_2 \in S$ and $\alpha \in Act$

$$(s_1^{a} \to s_2) \in R \Leftrightarrow (f(s_1)^{a} \to f(s_2)) \in R'$$

In this case behavior of processes identical, therefore, processes are identical.

IV. OPERATIONS WHICH CAN BE CARRIED OUT OVER PROCESS IN BEHAVIOR MODEL

By consideration of process of modeling it is possible to come to conclusion that over it the limited number of actions can be executed. Now we will consider those actions which can be realized in process.

In case for process in model of behavior there are no transitions, and there is one entrance condition - it call empty and designate as *0*.

Let's consider a case of addition to process of *P* prefix actions. In this case to a set of conditions of process the condition of s which is an initial condition of new process is added. It leads to that to a set of transitions $s^{a} \to s^{0}$ transition is added, and it is possible to call the process which has turned out in this case as *α.P*.

If there is an alternative composition, that couple of processes of P_1 and P_2 needs to construct process of *P* which will function also as P_1 or P_2 is supposed. The choice, what branch will be used by *P* process, depends on a choice of the process or on an environment choice.

So, if $P_1 = \alpha?.P_1'$ and $P_2 = \beta?.P_2'$ and the environment can enter the *α*, but can not enter the *β*, then *P* must select only possible behavior - P_1. Thereafter, the process *P* can not change its decision. Then the alternative formulation is as follows

$$Pi = (S_i, s_i^0, R_i)\ (i = 1, 2)$$

and a set of conditions of S_1 and S_2 in this case have no general elements.

In this case alternative composition will be

$$P_1 + P_2 = (S, s^0, R),$$

where: - *S* is $s_1 \in s_2$ to which the new condition of s^0 which will be initial for $P_1 + P_2$ is added;

- R contains all transitions from R_1 and R_2;
- for each transition from R_i *(i = 1, 2)* a type of $s^0_i{}^a \to s$, *R* contains $s^0 \, {}^a\!\!\to s$ transition.

If sets of S_1 and S_2 have the general elements, for $P_1 + P_2$ need to be replaced those elements which enter into S_1 in S_2 and as appropriate to execute S_2 and s^0_2 modification.

As the model of behavior is difficult system, in it can be used and parallel compositions from several interacting a component. If to consider two Sys_1 and Sys_2 systems (subsystems) which are components of one Sys system, i.e.

$$Sys \overset{def}{=} \{Sys_1, Sys_2\}\ .$$

That behavior of systems can be presented by P_1 and P_2 processes respectively, and behavior of Sys_i *(i = 1, 2)* as a part of Sys system will be presented to the corresponding P_i. Let's designate $\{P_1, P_2\}$ as process which describes behavior of system. It can be treated as round column *P*, we will consider thus all transitions from a condition in a condition in the column as instant. The fact of performance of action, in this case, we will fix at the time of transition.

Each entrance or output action of P_i (*i* = 1, 2) represents result of interaction of P_i with process not entering into set $\{P_1, P_2\}$, or as result of interaction with P_j, where $j \in \{1,2\} \setminus \{i\}$, or it internally process action. If internally process action that P_i (*i* = 1, 2) transfers it a certain object, and P_j it accepts $P_j (j \in \{1,2\} \setminus \{i\})$.

Each possible option of behavior of process of P_i (i = 1, 2) can compare a thread (in treatment of UNIX OS is a process, i.e. sequence of operators), which we will designate as ⍰*i*. It will allow to define option of behavior of process of ⍰*i* (*i* = 1, 2), for P_i as a part of processes $\{P_1, P\}$.

If to designate set of all options of behavior of processes as *Beh*$\{P_1, P_2\}$, each of which corresponds to one of options of functioning $\{P_1, P_2\}$. That can be assumed that process $\{P_1, P_2\}$ functions consistently, i.e. that at any option $\{P_1, P_2\}$ form the linear ordered sequence of *tr* = (*act*$_1$, *act*$_2$, ...) actions which are ordered on performance time. If to designate With as one of sequences, the set of indexes of elements of sequence can be designated as *Ind*(*tr*), and *a* set of points as *Points*(*C*).

The sequence of *tr* is linearization With (it is supposed that in parallel processes carrying-out operators aren't synchronized on time, therefore, it is possible to construct some ordered sequence) if there is a display

$$Lin: Points \to (Ind\ (tr).$$

In this case the answering condition can define the description of process $\{P_1, P_2\}$ as construction P

$$Tr(P) = \bigcup_{C \in Beh\{P_1, P_2\}} Lin(C)$$

with $Beh\{P_1, P_2\}$

In this case in the course of P all linearizations of processes of P_1 and P_2 answering to any their joint behavior are presented.

If processes look like $P_i = (S_i, s^0_i, R_i)$ $(i = 1, 2)$, and tr is the route (S, s^0, R) which components can be defined as

$$-S \overset{def}{=} S_1 \times S_2 \overset{def}{=} \{(s_1, s_2) | s_1 \in S_1, s_2 \in S_2\}$$

$$-s^0 = (s^0{}_1, s^0{}_2)$$

- for each transition $s_1{}^{\alpha} \rightarrow s'_1$ of R_1 and each state of $s \in S_2$. R comprises a passage $(s_1, s)\,{}^{\alpha} \rightarrow (s'_1, s)$. And for each transition $s_2{}^{\alpha} \rightarrow s'_2$ of R_2 and each state $s \in S_1$, R comprises a transition $(s, s_2)\,{}^{\alpha} \rightarrow (s, s_2)$.

For each pair of transitions $s_1{}^{\alpha} \rightarrow s'_1 \in R_1$ and $s_2{}^{\alpha} \rightarrow s'_2 \in R_2$, R contains the transition $(s_1, s_2)\, \tau \rightarrow (s'_1, s'_2)$.

It is possible to claim that this process of P is parallel composition of processes of P_1 and P_2 which can be designated as $P_1 | P_2$.

During the work with parallel processes restrictions which follow from logic of functioning of processes of P_1 and P_2 are applied. If the L any subset of a set of *Names*, in this case restriction on L looks like $P\backslash L = (S, s^0, R')$ which turns out by removal of all transitions having tags from a subset of L, i.e.

$$R' \overset{def}{=} \{(s^{\alpha} \rightarrow s') \in R | \alpha = \tau \text{ or } name(\alpha) \in L\}.$$

It is necessary to remember and that during the work the set of conditions of process of P decides on parallel compositions as work of conditions of processes of P_1 and P_2 entering into this process.

As the model of behavior is under construction on the component principle, it can lead to repeated use of the same component in model. It leads to that tags of transitions and names of actions repeat. For permission of the such conflicts renaming of names f is carried out: *Names* $\rightarrow$ *Names* and change of tags $\alpha?, \alpha!$ on $f(\alpha)?$ and $f(\alpha)!$. The turned-out process can be designated as $P[f]$.

At nonidentical renaming when names from the list $\alpha_1, \ldots, \alpha_n$ also occurs displays in names $\beta_1, \ldots, \beta_n$. In this case process will be designated as $P[\beta_1/\alpha_1, \ldots, \beta_n/\alpha_n]$ [1].

V. TREATMENT OF DESIGNS OF THE COUNT OF MODEL OF BEHAVIOR

Recognizing that we consider behavior model, and it means work with such categories as a condition of system, action, an event, transition from one condition in another, transition conditions, sequence of events, from column P it is possible to allocate information (J) and the managing director (G) of the column. If to compare transitions in columns J, G and transitions in the column of the description of model of behavior, it is possible to receive the statement about their route equivalence of Tr (P) = Tr(G) = Tr(J) that will allow to operate with each of counts at the solution of specific problems of creation of the software.

So to column J it is possible to refer all actions which change internal state of model of behavior. Now we will define that action - any operation which makes signing up in port of input-output is meant the term or carries out an splitting of new process in system. It is necessary to remember that the declaration of assignment need to carry to operations of change of internal state of process.

For all these listed elements it is possible to execute operation of replacement of action on the corresponding time equivalent on each layer of decomposition in the operating count of model of behavior. Such action, further, will simplify a task of the analysis and verification of the operating count.

As the form of record of the information count is suitable for machining, further it is possible to use the interpreter which in an automatic mode will replace the action elements of the information count with the corresponding sequences of operators of the chosen programming language.

Having carried out the analysis of the managing director column G it is possible to allocate some moments, so transition from a condition in a condition can be unconditional or it can be limited to performance of some condition. Such condition influences a choice of the route and depends or on a condition of objects connected with internal or entrance actions. Most conveniently for routes defining a choice to write down conditions over an edge of the count. It will significantly improve visualization of a condition of transition. The type of such record assumes the following options:

- lack of record over an edge, unconditional transition;

- record existence over an edge, transition will be executed if expression over an edge is true. Especially it is necessary to emphasize that in the column surely there has to be an alternative route which will be realized in case of condition non-performance;

- existence over an edge of «*» symbol means that unconditional performance of transition (it is used only in an alternative design) is realized.

The following situation in the operating count needs to be considered when the multiprocessor platforms on which it is realized parallel asynchronous processes are used. In this case it is necessary to consider parallel processes as two or more components. Symbols "&&>" will be a sign of the organization of parallel processes, and as a condition of end of overlapping symbols ">&&" or ">//" will serve. These designs have to be connected with a condition, i.e. top of the count.

It should be noted that the organization of parallel processes is possible only according to one scenario, and end of a parallel segment assumes two scenarios. In the first case the scheme "and" when further work will be possible at completion of parallel processes is realized. In the second case, the competitive scheme is realized "or" - process which came to the end earlier has to destroy competing parallel process with release of its resources.

As the branch condition and performance of cyclic operators to implement with the selection of the most optimal design, it is at the moment the developer is recommended to indicate the most appropriate statement of branching, for example, "*>if/case*", which is associated with a vertex of realizing input action. Failing this, the interpreter will implement the scheme "*if*". Similar rules apply to the choice of the operator of the cycle. If there is no clear indication to choose the type of cycle, such as "*>for/while*", the interpreter selects the implementation of the design "*>while*".

VI. RULES OF DESIGNING OF THE COUNT AND PERFORMANCE OF INSCRIPTIONS

For convenience of reading columns it is designed at the left on the right and from top to down. All elements of the count (tops and edges) have unique marking. As one top (condition) can leave some edges (actions), the generating top of the count is duplicated with the indication of a sign of circulation – « ' » in the top identifier. The similar design arises and in a situation when some edges enter into top of the count. At transfer of elements for "*a new line*" identifiers of tops repeat with the indication of operation of transfer « '' ».

At design of parallel structures, cycles and branchings on a condition it is offered to use the component principle of creation of structure with further carrying out decomposition, i.e. to project model by the principle "*from above – down*". It will increase "*readability*" of model of behavior. If these elements have "indivisible" character – simply them to write down that in the corresponding places of the count. As the general recommendation, at creation of model, it is necessary to consider performance of the principle of layer-by-layer design and "*not to overload*" each layer of decomposition of model of behavior.

Concerning inscriptions on a process graph. It is necessary to allocate some types, these are recommendatory, performance and action conditions. Their general characteristic will be that they are characterized as a stream of symbols written down by some rules. Now we will consider them one after another.

The group of recommendatory inscriptions treats only "*entrance*" tops of the count (conditions). Their characteristic feature is existence of a pair symbol «"», as restrictions of a stream of symbols. These are "entrance" recommendatory designs: "*>if*", "*>case*", "*>for*", "*>while*". If they are absent as recommendatory, the interpreter is obliged to realize situationally standard designs of *if* and *while*.

There is one more design which belongs to "entrance" - start of parallel processes "&&>". This design is the managing director and assumes existence of several parallel processes. In other situations use of this design is forbidden. In model of behavior designs ">&&" and ">||" which define the mechanism of completion of parallel processes. By record rules these designs are associated with top of the count for which the scenario transition is realized.

Now consider an item such as the execution condition. This stream of characters written on the edge of the graph, and involves the ability to record any kind of logical expression, including the composite. The result of the logical expression is obtaining certification - the "**true**" or "**false**". This design is most applicable in the implementation cycle of the while statement and the statement of branching if.

However, in a situation where design case used to compare the value of the variable pattern and the comparison result decision of the possibility of performing an action associated with the edge of the graph . Since in this case must be provided in "**template**", then the value of the variable with which comparison is made, must be determined by the operator in case, therefore, a complete record of the design suggested should be represented as ">case" $ **<variable name>** . Between constructs ">*case*" and $ **<variable name>** use spaces that are ignored when parsing the string. When writing templates themselves must use the construct " $ **<variable name>**, with the last template in the operator always has to be a template $*. His performance is the lack of coincidence of all the previous template with the value stored in the variable associated with the job conditions of the design case.

Considering the design conditions for the situation "> *for*", it should be noted that in this case given a range of values of the variable changes and which tend to cyclically change its value . Therefore, the condition that is written on the edge of the graph should be shaped sequence of groups of characters separated by spaces or other special characters. An example of the formation of such a stream of characters:

$ <variable name> = <the minimum value of the variable (constant)> $ <variable name> = $ <variable name> <type of operation> {$ <variable name> or <constant>} <maximum value of the variable (constant)>

All elements of the text structures must be separated by spaces or special characters. Attention should be paid to such a structure as a text element <**type of operation**>. In this case, should be considered only arithmetic operations. For the $ **<variable name>** = $ **<variable name> <type of operation>** {$ **<variable name>** or **<constant>**} can be used and unary operations with the construction of a suitable design.

Now consider the character stream that carries the information about the action. It is assumed that the number of elements to the edges of which are characterized as "**action**" is not limited. Therefore, the separator of elements should act or a sign of the completion of the flow (push it «button» Enter) or a special character, " ; " that separates the operation effect. For the structure itself "action" there is only one rule - the use of space characters, spaces or special characters for the separation of the structure.

It should be noted, and feature the use of variable names, as in the elements of "condition" and in the elements of the "**action**". A common feature of the variable name is the symbol "$" is in the first position of the sequence of characters.

The general rules should include the fact that the elements of the text constructs do not use special characters if their application not stated in special agreements for the interpreter.

VII. CONCLUSIONS

Design of model of behavior of a control system is a first step to achievement of a goal. And if when performing this work it is possible to reach reduction of quantity of mistakes of "a human factor" and it can happen only at introduction in process of design of formal language to high extent of visualization, as shows the solution proposed in article.

As the created model has all signs of a scripting programming language that obviously that with use of such representation of model of behavior it is possible to check correctness of implementation of key decisions in a control system.

When obtaining such conclusion, it is possible to start the following phase of implementation of the project – generation of a code of the operating program according to the formal description of model of behavior of a control system.

It is supposed that the solution proposed in article will allow to introduce new approaches to design of the software of control systems, and it will allow to reduce significantly as terms of performance of work, and will increase reliability of a created product.

References

[1] A.M.Mironov A.A. Theory processes / Internet http://www.twirpx.com/file/617525/ resource /

REVIEW OF GROUP CONTROL ALGORITHMS

Norseev S.A., Bagaev D.V.
KSTA named Degtyarev V.A.
Kovrov, Russia
Norseev@gmail.com, dmitrybag@gmail.com

Problem of distributing task between some systems and coordination interaction this systems is actual problem. In this article will review main methods of task distribution between different system elements and adjustment interaction these elements.

algorithm; behavior; swarm; multiagent system

I. MAIN TERMS

Agent – it is object that solves narrow range of specific tasks. In this role can be: processor, microcontroller, computer, robot and other. Typically, single agent performs a specific simple operation. For example, defining readings of sensor and sending these data in machining center (usually, other agent); monitoring perimeter, limited by means of technical vision; transportation object on a given route; other tasks.

Multiagent system – it is system, that consists from several interact agents. This system can solve difficult tasks, by distributing these tasks between agents. Examples these tasks are: machining information, which receive from several remote sensors, and decision-making based on this information; monitoring area beyond the capabilities of means of technical vision of single agent, and coordination agents, which perform monitoring different parts of this area; other tasks. Main difficulty in designing of the multiagent system is development and realization effective algorithms of interaction of agents, which lead to effective solution of the whole task. In this article will review several approaches to designing of such algorithms.

Static multiagent system – it is multiagent system, whose architecture and configuration does not change during system work. Such system is designed once in case of implementation. Subsequently, changes of systems are minimal and, typically, not effect the system architecture.

Behavior of static multiagent system is deterministic and defines by precise rules. In case, when system make management decision, use heuristic algorithms and/or neural networks. However, one must understand, that these algorithms uses for controlling of small number (in most cases, one) of agents from whole system. Behavior of other agents is strictly deterministically.

Such systems well suited to manage system, working in static environment. For example, monitoring system the technically complex object (manufacturing, nuclear power plant, other).

Static multiagent systems poorly suited for cases, when system working in little-known and unpredictable environment. Of course, such systems uses for research unknown area (for example, American system MSSMP or Israeli system Avantguard). But, such system capabilities in these cases are severely limited, versus capabilities of dynamic multiagent systems.

Dynamic multiagent system – it is multiagent system, whose architecture and configuration continuously changes during system work. In such systems, connections between particular agents has temporary nature. Behavior of such system has random nature and defines with help behavior algorithms.

These systems well suited for work in unknown, unpredictable and ever-changing environment. However, It are ineffective during work in static conditions, when changes of environment are minimal.

"Collective" – it is multiagent system, in which every agent "know" about other agents in same system.

Main advantage of "collective" is deterministic behavior. Typically, all connections between agents in "collective" defined and debugged on design and implementation stage. Agents in "collective" guided by simple and strict rules. Their behavior is strict documented and predictable.

Main disadvantage of "collective" is difficulty of maintaining base of "friends" in actual condition in memory of every agent. In ever-changing environment it is difficult. Similarly, there is problem of adding new agent in "collective". "Newcomer" must become acquainted with all other "members of collective".

Therefore, "collective" model used in designing of static multiagent systems.

"Swarm" – it is multiagent system, in which every agent "familiar" only with small count of other agents of same "swarm". For example, if "swarm" consists from N agents, then every agent in this "swarm" does not "familiar" with N-1 agents (like in "collective"), but it "familiar" with M agents; at that M < N-1.

Agents, with which concrete agent A is "familiar", will call "friends" of agent A. Necessary understand, that during work of "swarm", amount and composition of "friends" for every agent in "swarm" ever-changing.

Idea of organization agents in "swarm" taken from wildlife. Therefore, many behavior algorithms developed by monitoring the swarms, flocks, colonies, shoals. Examples of such algorithms are SWARM (also known as "birds"), formic algorithm, bees algorithm and other.

Main advantage of "swarm" is it's dynamism. Behavior of agents in "swarm" has random character. Therefore, behavior of whole "swarm" also is random and unpredictable. It allow use "swarm" for solve tasks, initial data in which is contradictory and insufficient for solve task with help deterministic algorithms (for example, with help "collective"). Example of such task is task of research unknown area.

Other advantages of "swarm" are its resistance to failure of one or some agents (other agents does not "notice" the disappearance of "friends", because does not keep a list of agents in "swarm", so it occurs in "collective") and scalability (ability to easily add new agents in "swarm").

Inapplicability of "swarm" for solve a number of problems, for which already developed deterministic algorithms, is consequence of random behavior of "swarm". Examples of tasks solved with help "swarm" are: patrolling the area, research unknown area, search in little-known area, other tasks. Examples of tasks, in solving which "swarm" is inapplicability, are: accumulating and machining information, which receive from several remote sensors; manage complex system consisting from several interact subsystems (for example, manage the robot, machine); monitoring and control of production processes; other tasks. System, developed on base of "collective" model, more effective solves such tasks.

Therefore "swarm" model applicability only for organization dynamic multiagent systems.

II. "COLLECTIVE" MODELS

Logic of agent behavior in "collective" determined by list of challenges facing the "collective" and architecture of "collective". The most widely used architectures are centralized and hierarchical architectures. It show on picture below.

Simple static multiagent systems developed on base of centralized architecture. For complex systems use hierarchical architecture.

Main feature of these architectures is availability of main agent, which responsible for management all other agents.

III. "SWARM" MODELS

In this part enumerate main "swarm" algorithms. They are rarely used in its pure form. In most cases use different combinations of these algorithms.

A. SWARM algorithm

This algorithm was formulated by Craig Reynolds for definition of birds behavior in flock. Behavior of every bird in this algorithm must comply with three rules.

1. Rule of separation: every bird must try to avoid a collision with other birds.
2. Rule of alignment: every bird must move in the same direction as nearby birds.
3. Rule of solidarity: birds must try move the same distance from each other, moving to mass center of flock.

Computer modeling of flock behavior, managed by these rules, performed by Reynolds, showed that it is similar to behavior of bird flock.

Advantages of algorithm.

1. Simple logic of separate agents.
2. Equivalence and interchangeability of agents.

Disadvantages of algorithm.

1. Lack of leader in swarm leads to difficulty of managing move direction of swarm.

Need to monitor position of mass center of swarm. It is difficult if swarm consist from many agents.

B. Formic algorithm

Initially, ants move in random direction and, finding food, return to their colony, paving the pheromone trail. If other ants find such trails, they will go on these trails. Instead of storing trail, ants strengthen trail, if they find food. Pheromone trail eventually evaporates and its attractive force weakens. The more time needed for traversing the path to the target and back, the stronger evaporate pheromone trail. For short trail, traversing the path will quicker and, whereupon, pheromone density remains high. Thus, when ant find short way from colony to food, other ants will go to this way and pheromone trails leads all ants to shortest way.

This algorithm used for searching some resource (food, in case of ants) in unknown environment. For its adaptation to task of bypass area propose following changes.

- Ants do not return to colony, but keeps a given distance from colony.
- Ants move in random direction.

These changes help achieve "swarming" ants in place.

Advantages of algorithm.

1. Equivalence and interchangeability of agents.
2. Simple logic of separate agents.
3. Simple scalable.
4. For estimate the distance to colony separate agent must evaluate the distance to their "friends". It's easier, than tracking position of colony mass center, as in case of SWARM algorithm.

Disadvantages of algorithm.

- Lack of leader in colony leads to difficulty of managing move direction of swarm.

C. Bees algorithm

Initially, from hive in random direction takes few bees, which search areas having nectar. After returning in hive, these bees tell other bees about position and amount of nectar. Thereafter, other bees fly to these areas. More than a predetermined area expected to find nectar, the more bees flies to the area.

This algorithm, as well as formic algorithm, used for searching some resource (nectar, in case of bees) in unknown environment.

Advantages of algorithm.

1. Equivalence and interchangeability of agents.
2. There is no need to track position of colony mass center, as in case of SWARM algorithm.
3. Simple scalable.

Disadvantages of algorithm.

- Lack of leader in colony leads to difficulty of managing move direction of swarm.

D. Movement algorithm shoal of fish

This algorithm was proposed by B. Filho and L. Neto in 2008.

Movement fish shoal determined by activity of more purposeful zooids. If they move somewhere, then their "friends" can move with them. This movement can cover the entire shoal.

This mechanism is fairly balanced. As rule, zooid just having escaped from shoal, immediately return to it.

Advantages of algorithm.

1. There is no need to track position of colony mass center, as in case of SWARM algorithm.
2. Simple scalable.
3. Ability to easily control the direction of movement of entire shoal.

Disadvantages of algorithm.

• Lack of uniform distribution of agents in the study area.

E. Fireflies algorithm

This algorithm was proposed by X. Sh.Yang in 2007.

All fireflies attract each other. Attractiveness of firefly is proportional to its brightness. Less attractive fireflies move to more attractive fireflies. Brightness of firefly for other glowworm decreases with increasing distance between them. If firefly do not see more bright firefly than it, then it move in random direction.

Advantages of algorithm.

1. Equivalence and interchangeability of agents.
2. There is no need to track position of colony mass center, as in case of SWARM algorithm.
3. Simple scalable.

Disadvantages of algorithm.

1. Lack of leader in colony leads to difficulty of managing move direction of swarm.

PATHFINDING ALGORITHMS EFFICIENCY ESTIMATING IN DISCRETE LABYRINTH BASED ON SOFTWARE SIMULATION

Krasnov E.S., Bagaev D.V.
Instrument engineering dept.
Kovrov State Technological Academy named after V.A. Degtyarev
Kovrov, Russia
krasnoves42@gmail.com, dmitrybag@gmail.com

Abstract – This article continues the research, started in the first article – «Strategy of analyzing most common algorithms for path finding in discrete labyrinth using software statistic data collector» [1]. It is dedicated to experiment's overview and summarizing its results. The common structure of the experiment, its stages, collecting data and methods of its processing are described. The main conclusions are made at the end of this article.

Keywords – algorithms, maze solving, data analyzing, software simulation

I. INTRODUCTION

Statistics was collected via special simulation software, previously described in the first article. It was improved in ways of usability, results' displaying, but not in way of changing calculation methods, described in the first article [1]. The detailed description of the software will be given below.

II. SOFTWARE DESCRIPTION

The software, used in the experiment, is meant to be run on Microsoft Windows ® platform. It is a sort of «sandbox» for creating two-dimensional discrete labyrinths (also maps or mazes; for more detailed description see article [1]) and manipulating with them. Also, it can perform different pathfinding algorithms on created map (see [1] for the list of used algorithms), collecting the required stat data. Here the main features of the software:

• creation of two-dimensional discrete labyrinths (via built-in simple graphics editor, similar to Microsoft Paintbrush, allowing to draw obstacles) and saving them to/loading them from file;

• running selected algorithm on the drawn map with displaying the result route, obtained via this algorithm;

• sequential running of all built-in algorithms with collecting stat data (see [1] for the list of collecting values) and its saving for further analysis;

• viewing and analysis of stat data and exporting for further processing in the third party software;

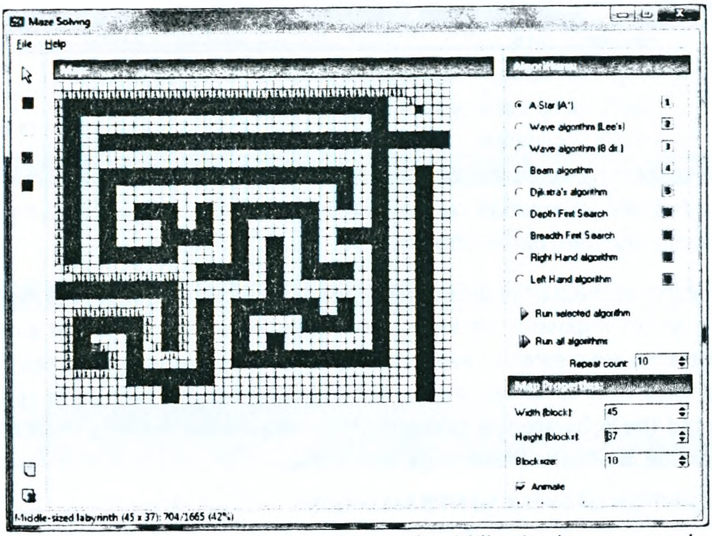
The figure below represents the modified and improved interface of the simulation software, that is displaying the middle-sized map (filled for 42% with obstacles) and a result route, built with A-Star (A*) algorithm (Fig. 1).

Let's introduce a notion: test (session) – it is single run of all of the built-in pathfinding algorithms on the current map (labyrinth), with measuring all the required values and saving collected data. The test (session) could be run by pressing the «Run all algorithms» button.

To obtain a reliable time value of algorithm run, software needs to repeat the run of every algorithm multiple times (from 1 to 500 times; it is a user-defined value). It enables to obtain the average value of the time value.

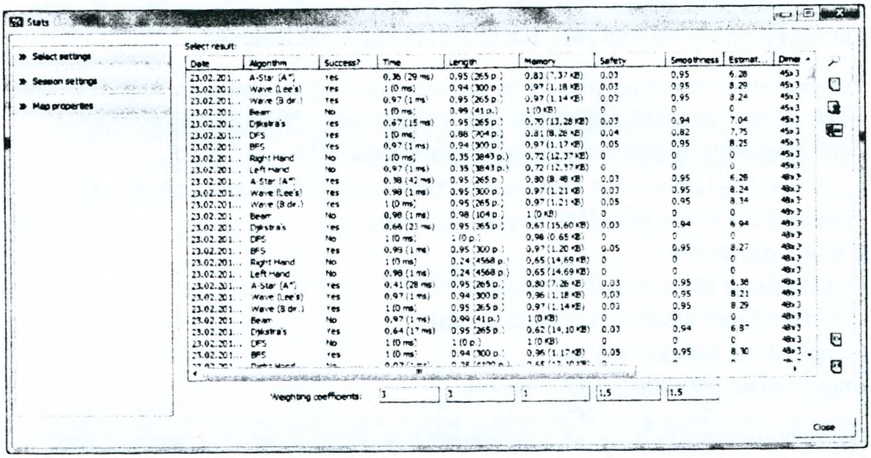
Running of all algorithms was initially programmed to take the special order, for avoiding influence of processor's cache memory mechanism (it could give a erroneous values). Such a thing could happen if the algorithms would be run in the next order: 1, 1, 1, 1, ... 2, 2, 2, 2, ... 3, 3, 3, 3, 3... and so on. To avoid this, the next order was used: 1-2-3-4..., 1-2-3-4... and so on, where 1-9 are the numbers of the algorithms. But the experiment showed, that it doesn't actually happen, and moreover, the selected order makes impossible to measure the time of some algorithms (due to their extremely high speed). For example, the measuring of single run of the classic wave algorithm always resulted in zero milliseconds. So, the order of algorithms' running was restored, and the full time of full repeat cycle for every algorithm was measured. Later it was divided at by the number of repeats.

After finishing the session, the collected data is being saved for further analysis (single algorithms runs are ignore, see explanation below). The viewing of collected data and its partial analysis can be done via stat form window, displayed on the figure below (Fig. 2).



Repeat count: 10
Width (block): 45
Height (block): 37
Block size: 10
Middle-sized labyrinth (45 x 37): 704/1665 (42%)

Figure 1 - Software graphic interface with middle-sized map example.



Select result:

Date	Algorithm	Success?	Time	Length	Memory	Safety	Smoothness	Estimat...	Dimer
23.02.201...	A-Star (A*)	Yes	0,36 (29 ms)	0,95 (265 p.)	0,83 (7,37 KB)	0,03	0,95	6,28	45x3
23.02.201...	Wave (Lee's)	Yes	1 (0 ms)	0,94 (300 p.)	0,97 (1,18 KB)	0,03	0,95	8,29	45x3
23.02.201...	Wave (8 dir.)	Yes	0,97 (1 ms)	0,95 (265 p.)	0,97 (1,14 KB)	0,03	0,95	8,24	45x3
23.02.201...	Beam	No	1 (0 ms)	0,99 (41 p.)	1 (0 KB)	0	0	0	45x3
23.02.201...	Dijkstra's	Yes	0,67 (15 ms)	0,95 (265 p.)	0,70 (13,28 KB)	0,03	0,94	7,04	45x3
23.02.201...	DFS	Yes	1 (0 ms)	0,88 (704 p.)	0,81 (8,28 KB)	0,04	0,82	7,75	45x3
23.02.201...	BFS	Yes	0,97 (1 ms)	0,94 (300 p.)	0,97 (1,17 KB)	0,05	0,95	8,25	45x3
23.02.201...	Right Hand	No	1 (0 ms)	0,35 (3843 p.)	0,72 (12,37 KB)	0	0	0	45x3
23.02.201...	Left Hand	No	0,97 (1 ms)	0,35 (3843 p.)	0,72 (12,37 KB)	0	0	0	45x3
23.02.201...	A-Star (A*)	Yes	0,38 (42 ms)	0,95 (265 p.)	0,80 (8,48 KB)	0,03	0,95	6,28	48x3
23.02.201...	Wave (Lee's)	Yes	0,98 (1 ms)	0,95 (300 p.)	0,97 (1,21 KB)	0,03	0,95	8,24	48x3
23.02.201...	Wave (8 dir.)	Yes	1 (0 ms)	0,95 (265 p.)	0,97 (1,21 KB)	0,05	0,95	8,34	48x3
23.02.201...	Beam	No	0,98 (1 ms)	0,98 (104 p.)	1 (0 KB)	0	0	0	48x3
23.02.201...	Dijkstra's	Yes	0,66 (23 ms)	0,95 (265 p.)	0,63 (15,60 KB)	0,03	0,94	6,94	48x3
23.02.201...	DFS	No	1 (0 ms)	1 (0 p.)	0,98 (0,65 KB)	0	0	0	48x3
23.02.201...	BFS	Yes	0,99 (1 ms)	0,95 (300 p.)	0,97 (1,20 KB)	0,05	0,95	8,27	48x3
23.02.201...	Right Hand	No	1 (0 ms)	0,24 (4568 p.)	0,65 (14,69 KB)	0	0	0	48x3
23.02.201...	Left Hand	No	0,98 (1 ms)	0,24 (4568 p.)	0,65 (14,69 KB)	0	0	0	48x3
23.02.201...	A-Star (A*)	Yes	0,41 (28 ms)	0,95 (265 p.)	0,80 (7,26 KB)	0,03	0,95	6,38	48x3
23.02.201...	Wave (Lee's)	Yes	0,97 (1 ms)	0,94 (300 p.)	0,96 (1,18 KB)	0,03	0,95	8,21	48x3
23.02.201...	Wave (8 dir.)	Yes	1 (0 ms)	0,95 (265 p.)	0,97 (1,14 KB)	0,03	0,95	8,29	48x3
23.02.201...	Beam	No	0,97 (1 ms)	0,99 (41 p.)	1 (0 KB)	0	0	0	48x3
23.02.201...	Dijkstra's	Yes	0,64 (17 ms)	0,95 (265 p.)	0,62 (14,10 KB)	0,03	0,94	6,87	48x3
23.02.201...	DFS	No	1 (0 ms)	1 (0 p.)	1 (0 KB)	0	0	0	48x3
23.02.201...	BFS	Yes	1 (0 ms)	0,94 (300 p.)	0,96 (1,17 KB)	0,05	0,95	8,30	48x3
23.02.201...	Right Hand	No	0,97 (1 ms)	0,25 ([illegible] p.)	0,65 ([illegible] KB)	0	0	0	[illegible]

Weighting coefficients: 3 | 3 | 1 | 1,5 | 1,5

Figure 2 - Stats viewing window.

The main features of the stats module:

- collecting and keeping of the stat data;
- calculation of the normalized and non-normalized characteristic values (see [1]) and the F value (estimation of the algorithm, also described in [1]), calculated with user-specified weighting coefficients k_T, k_L, k_M, k_H, k_A;

- selection of the partial data by the following criterions:
  - session's date;
  - map's filename;
  - width, height and type of the map (by size);
  - map's occupancy;
- data export (selected test or the whole amount) to the HTML-file (for more comfortable information representation, its further editing/formatting and transfer into third party software).

The selection of the partial data by described criterions is designed to filter the whole test, so it's impossible to view results only for A* algorithm, for example. These constraints are created to avoid the improper characteristic values calculation (they have sense only as relative values, not as absolute ones). With some test data being deleted the software will calculate the wrong values, what is unacceptable and tests can be deleted or filtered as a whole only.

III. DESCRIPTION OF EXPERIMENT'S METHODS

The only improvement of the calculations' methods is as follows: the points of the map, close to map's edge are considered as hazardous during calculating an *H* value.
For collecting all required data, the special method was invented.

The main map's characteristics are its size and occupancy by obstacles. As you can see in article [1], there's a following classification of maps made (three occupancy-types):
- poorly-filled (less than 10% of coverage by obstacles);
- medium-filled (more than 10% and less than 40% of coverage by obstacles);
- strong-filled (more than 50% coverage by obstacles).

and four types by size:
- small (about 10x10 blocks);
- medium (from 40x40 up to 60x60 blocks);
- large (about 100x100 blocks);
- huge (about 200x200 blocks or more).

So, its necessary to run at least 12 groups of tests (3 x 4), which enables to make a conclusion about algorithms' efficiency in every group. The number of tests (sessions) in each group is about one hundred. Every test is run on the map with a little modified size, occupancy and a structure.

After all the tests run, the average value of *F* (see [1]) is calculated (for every group), which allows to make a number-supported conclusion about each algorithm's efficiency.

TABLE I - RESULT OF THE PRIMARY SERIOES OF TESTS

	Poorly-filled		Medium-filled		Strong-filled	
Small	A*	6,82	A*	6,41	A*	6,05
	Wave	8,16	Wave	8,01	Wave	7,53
	Wave8	**8,27**	**Wave8**	**8,10**	**Wave8**	**7,72**
	Beam	1,53	Beam	0,97	Beam	0,53
	D	6,54	D	6,46	D	6,44
	DFS	4,32	DFS	4,16	DFS	3,05
	BFS	**8,17**	**BFS**	**8,02**	**BFS**	**7,57**
	R	5,62	R	3,71	R	7,2
	L	5,78	L	3,72	L	7,24
Medium	A*	6,88	A*	6,71	A*	6,5
	Wave	**8,63**	**Wave**	**7,71**	Wave	7,67
	Wave8	**8,59**	**Wave8**	**7,80**	**Wave8**	**7,84**
	Beam	2,34	Beam	0,32	Beam	0
	D	7,35	D	5,99	D	6,05
	DFS	8,12	DFS	0,92	DFS	0,56
	BFS	**8,41**	**BFS**	**7,7**	**BFS**	**7,68**
	R	6,2	R	5,69	R	5,55
	L	6,44	L	5,81	L	5,49
Large	A*	6,22	A*	6,18	A*	6,06
	Wave	**8,14**	**Wave**	**8,05**	**Wave**	**7,98**
	Wave8	**8,13**	**Wave8**	**8,06**	**Wave8**	**8,05**
	Beam	3,65	Beam	0,11	Beam	0
	D	6,69	**D**	**6,58**	D	6,33
	DFS	1,23	DFS	0,42	DFS	0
	BFS	**8,04**	**BFS**	**8,02**	**BFS**	**7,97**
	R	5,65	R	5,8	R	5,78
	L	5,74	L	5,76	L	5,81
Huge	A*	6,51	A*	6,31	A*	6,2
	Wave	8,04	Wave	8,02	Wave	7,98
	Wave8	**8,27**	**Wave8**	**8,15**	**Wave8**	**8,14**
	Beam	1,03	Beam	0,05	Beam	0
	D	7,04	D	6,58	D	6,48
	DFS	0	DFS	0	DFS	0
	BFS	**8,31**	**BFS**	**8,11**	**BFS**	**8,09**
	R	5,41	R	5,43	R	5,45
	L	5,50	L	5,49	L	5,7

Note: the following designations were made:

- A* – A-Star algorithm;

- Wave – Wave algorithm (Lee's);
- Wave8 – Wave algorithm (8 directional);
- Beam – Beam algorithm;
- D – Djikstra's algorithm;
- DFS – Depth First Search;
- BFS – Breadth First Search;
- R – right-hand algorithm;
- L – left-hand algorithm.

TABLE II - RESULTS OF THE SERIES OF TESTS

	Poorly-filled		Medium-filled		Strong-filled	
Small	R	7,26	R	7,81	R	7,31
	L	7,84	L	7,83	L	7,26
Medium	R	7,35	R	7,69	R	7,64
	L	7,22	L	7,81	L	7,42
Large	R	7,46	R	7,62	R	7,88
	L	7,71	L	7,56	L	7,73
Huge	R	7,33	R	7,4	R	7,68
	L	7,44	L	7,32	L	7,59

The F value can be calculated by the following equation:

$$F_{a\lg} = k_T \cdot T_N + k_L \cdot L_N + k_M \cdot M_N + k_H \cdot H_N + k_A \cdot A_N, \quad (1)$$

which is described in details in [1].

The following weighting coefficients were used during experiment:

$$k_T = 3;\ k_L = 3;\ k_M = 1;\ k_H = 1.5;\ k_A = 1.5;\ ,$$

which are limiting the *F* value to range: from 0 to 10 (see [1] for more). Time and route-length characteristics values T_N and L_N are the most important, while the M_N value have the lowest weighting coefficient, since it is the least significant.

Also, the additional batch of tests was run to answer a question, asked in [1] – which algorithm is better – right-hand or the left-hand one? Since both algorithms can find a route only if the end-point is close to the wall, the test were run with such a condition, taken into the account (unlike to the main series of tests, in which it wasn't mentioned for avoiding the improper high *F* values for these algorithms).

Experiment's results. In this section the main results of the stat data analyzing are given (without listing all the processed data due to its huge size and unobviousness). All calculations were made in Micrisoft Excel ®.

The results are given as a table, which has 12 sections (one for every group of tests), divided into a smaller blocks, that represent average F values for each algorithm. The «best» algorithms are highlighted. Here it goes:

The results are given as a table, which has 12 sections (one for every group of tests), divided into a smaller blocks, that represent average F values for each algorithm. The «best» algorithms are highlighted. Here it goes (Table I).

IV. CONCLUSIONS

According to experiments results, the following alignment of forces has taken its place:

1. A* and Djikstra's algorithms have the lowest performance and require more memory than other algorithms. But route, they provided is often the shortest.

2. Classic Wave algorithm (Lee's) is faster than the 8-directional one (about 30-60% faster), but considering their great speed it doesn't make any real difference. At the other hand, the route, provided by the 8-directional Wave algorithm is much smoother with a low number of turns. Also it is as short as A* and Djikstra's ones most of the times.

3. Beam algorithm has extremely high fail rate, so it can be used only for very liitle situations and studying examples, with the simple maps only. Still it has the the great performance rate. Using it with real problems is mostly inappropriate.

4. Depth first search can't reach the end-point in large and huge labyrinths, due to overflowing of the stack in recursion. It faster even than the both Wave algorithms, but can be used for large maps.

5. Breadth first search is comparable to the 8-directional Wave algorithm, but provide a little longer routes. In common it's a little better, than the classic Wave algorithm.

6. The right/left hand algorithms are the fastest (in case of succesing), but they have about 50% of failing rate. The secondary series of tests cleared that these algorithms are equal in common.

So the 8-directional Wave algorithm seems to be the best one. It is fast enough, consumes little memory and provides a short and smooth route, which is medium-safe.

Thus the main questions, asked in [1], have been answered. The results are statistically reliable and there's only one «winner».

REFERENCES

[1] Krasnov E.S., Strategy of analyzing most common algorithms for path finding in discrete labyrinth using software statistic data collector// « East-West Design & Test Symposium (EWDTS 2012)», p. 34-41.

PREDICTION AND COMPARISON FUNCTIONS IN SEMI-AUTOMATIC CONTROL

Veronika Kozhukh
United Institute of Informatics Problems
of the National Academy of Sciences of Belarus
nika.rfe@gmail.com

Mobile robotics is one of the major directions in the field of intelligent robotics systems. It gives an opportunity to presence in areas human cannot or does not want to access. Mobility provides a great range of various applications for robots. Mobile robotics is on the way to be autonomous but still it can't operate in complex environment without outer controller. Nevertheless, there exists a lot situation where controller command can be wrong and leads to robot's failure [1].

The most demanded control for mobile robotic systems is remote control. System does exactly what says controller; controller chooses a command based on the given data from system. Common remote control is shown on the figure 1.

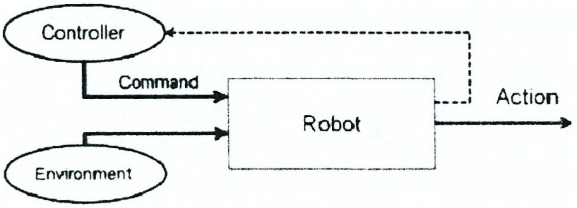


Figure 1 – Common remote control

Let's consider the situation when controller command doesn't reach mobile robot or controller can't receive the information about environment. Such cases can arise by several reasons; e.x. data link is out of reach. So, when robot doesn't have controller command it just take action by command from itself (we consider the one which has all needed computing skills). It is semi-automatic control model and represented by having several control commands: controller command and command generated by robot (figure 2).

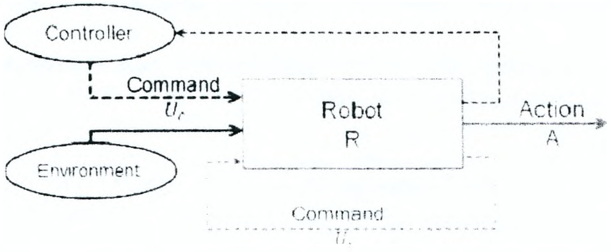


Figure 2 – Semi-automatic remote control

It is obvious when MR takes the action due his own control command only if he doesn't have any connection with the controller. But in other case we have two commands mobile robot to do. So somehow, robot has to choose one. The easiest solution would be when robot command is active only at the time when connection is lost. But loosing connection is not single condition to consider cases of incorrect controller commands. We have to deal with other conditions, such as mistaken interpretation of controller commands or controller wrong commands (the ones that lead robot damage).

On the figure 3 is shown case when controller makes wrong decision (controller command is in red color and robot command is in orange color). In the narrow corridor with obstacle straight and exit on the left. Controller doesn't see the obstacle and his command "straight" in the best way stop robot and in the worse way it can lead robot failure. At that time robot calculates the exit from the corridor. To stay intact it has to ignore controller command and to move left.

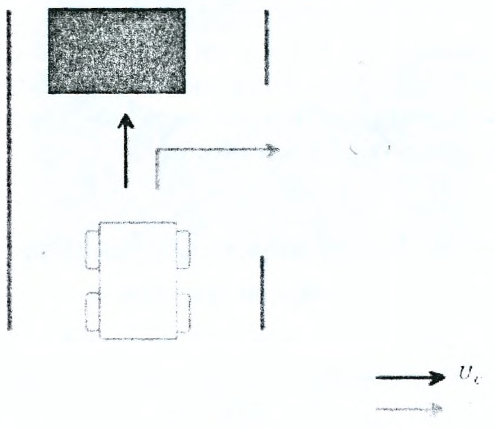


Figure 3 – Two possible commands

So, to find out what command is better robot has to calculate results of each command. For this case can be used prediction function (   ) [2].

Prediction function determines robot state in the next time step.

In that way are counted prediction functions for all types of controllers.

- prediction function of controller
- prediction function of robot

But our goal is some action to take. To figure it out is needed to compare the results of each command.

$$A = C(F_p(R, U_c)F_p(R, U_r), CL)$$

This formula shows what action takes robot. C is comparison function to choose which of predicted results is better. But this function also has to consider who makes less errors, whom robot can trust. For that reason we've included a variable ***CL***, which means a confidence level. It's a dynamic parameter that collects all result of previous evaluations and aggregates wrong decisions for each type of control. More actions are taken - more information about controllers is collected.

Moreover, in prediction function can be calculated not only appropriate state at the next moment of time, but some general goal. For instance, case when robot is to reach some point and complex of controllers commands leads other direction is set as wrong and robot has to search way by itself. Another of the ways to use such approach is controlling a group of robots by one controller. Controller doesn't need to communicate with every agent in group but sends some general commands.

References

[1.] Dudek, G. Computational Principles Of Mobile Robotics. Second Edition / G.Dudek, M. Jenkin - UK: Cambridge University Press,2010. - 11 p.

[2.] Norwig, P. Artificial Intelligence, A Modern Approach. Second Edition / S. Russel, P. Norwig – USA. Prantice Hall, 2003. - 557 p.

ИНТЕЛЛЕКТУАЛЬНАЯ СИСТЕМА УПРАВЛЕНИЯ АВТОНОМНЫМ МОБИЛЬНЫМ РОБОТОМ

Дёмин В.В., Кабыш А.С., Головко В.А.
Брестский государственный технический университет,
vvdemin@bstu.by

Рассматривается система эффективного управления мобильным роботом, в основе которой лежат алгоритмы обучения с подкреплением для сети колесных модулей. В рамках предлагаемого подхода эта сеть рассматривается как многоагентная система, в которой координация поведений агентов осуществляется виртуальным лидером. Предложена модифицированная модель обучения с подкреплением для адаптивной координации индивидуальных стратегий. Модифицированный Q learning алгоритм проводит обучение агентов эффективному управлению каждым колесом, в контексте группы, что позволяет агентам подстраиваться друг под друга.

Мобильные роботы, обучение с подкреплением, многоагентные системы, алгоритмы интеллектуального управления.

I. ВВЕДЕНИЕ

Эффективное управление мобильным роботом на производстве позволяет повысить общую производительность робота. Энергоэффективное управление роботом повышает время автономной работы, уменьшается общее энергопотребление, разумно расходуется общая энергия робота. Эффективная система управления обеспечивает прокладывание оптимальной траектории, что повышает маневренность при перевозке габаритных грузов в ограниченном пространстве, что в комбинации с энергоэффективным управлением дает возможность перевозки более тяжелых грузов на более длинные расстояния.

Задача энергосбережения в общем случае должна обеспечиваться подсистемами управления. Например, проблема энергопотребления моторов решается при их проектировании [1]. Подсистема управления не сможет влиять на КПД моторов, но должна обладать стратегией эффективного управления (оптимальная скорость мотора, оптимальный разгон, плавная функция торможения).

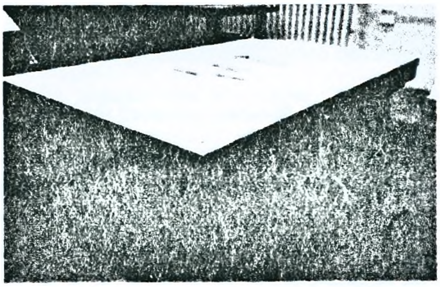


Рисунок 1 – Производственная грузовая мобильная платформа

Оптимальное планирование траектории, как правило, реализуется на уровне подсистемы планирования [2]. Такая подсистема строит траекторию до цели и разбивает ее на части, которые могут быть представлены в виде кривых определенного радиуса и прямолинейных промежутков. Система управления роботом позволяет передвигаться (по возможности без остановок) по этой траектории, затрачивая как можно меньше энергии батарей.

Задача эффективного управления при перевозке тяжелых грузов на современных производствах является актуальной для автономных мобильных грузовых платформ. Одна из таких платформ (рис. 1) – производственный грузовой робот, разработанный в лаборатории университета Равенсбург-Вайнгартена [4]. Основные характеристики платформы: размер 1200 см на 800 см, максимальная грузоподъемность 500 кг при комплектации 4-мя модулями, ёмкость аккумуляторов 52Ah, минимальная скорость 1 м/с, независимое управление каждым модулем. Представленная на рис. 1 платформа использует четыре модуля, но так же возможно собрать платформу и с большим количеством модулей.

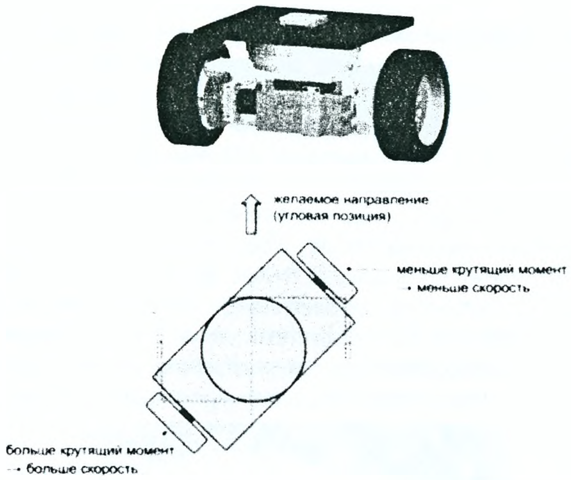


Рисунок 2 – Инновационный модуль

Платформа построена на базе инновационных модулей [4]. Такой модуль (рис. 2) состоит из двух колес, приводимых в движение двумя независимыми моторами, и имеет дифференциальную схему управления. К платформе такие модули подсоединены подшипником, что позволяет им поворачиваться относительно платформы на любой угол.

В статье рассматривается проблема эффективного управления многоколесным роботом на примере производственной грузовой мобильной платформы. Ключевым вкладом этой статьи является предложенная модель координации колесных модулей на основе виртуального лидера и обучения с подкреплением. Представленная модель решает проблему кругового движения платформы относительно центра разворота, даже если он динамически меняет свое положение. Подход требует лишь информации о положении агентов относительно центра платформы. По сравнению с аналогичными подходами кругового движения [3], предложенный алгоритм позволяет повысить эффективность потребления энергии роботом.

II. УПРАВЛЕНИЕ МОДУЛЕМ

Традиционный подход для управления платформой – расчет кинематики и моделирование инверсной кинематики [4]. В таком случае, если будут добавлены или демонтированы модули платформы, это потребует повторных расчетов и повторной конфигурации подсистемы управления. Существующий расчет кинематики может быть применен только для симметричного движения по кругу, при движении по которому центр разворота находится на центральной оси SG (пример предыдущего расчета на рис. 3). К примеру, невозможно использовать автомобильную схему управления или любую другую.

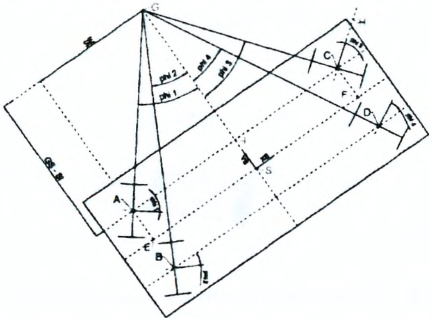


Рисунок 3 – Кинематическая модель платформы с четырьмя модулями для симметричного кругового движения

Кинематическая схема управления модулем описывается дифференциальным уравнением движения [5]:

$$\dot{x}(t) = \frac{R}{2}(v_r + v_l)\cos\theta(t) \tag{1}$$

$$\dot{y}(t) = \frac{R}{2}(v_r + v_l)\sin\theta(t) \tag{2}$$

$$\dot{\theta}(t) = \frac{R}{L}(v_r - v_l) \tag{3}$$

Здесь $[x, y, \vartheta] \in R^2 \times [-\pi, \pi)$ определяют положение робота, v_r и v_l – скорости правого и левого колес соответственно, R – радиус колеса, L – расстояние между левым и правым колесом [5]. Параметры v – линейная скорость, и w – угловая скорость используются для управления модулем:

$$v = \frac{R}{2}(v_r + v_l) \tag{4}$$

$$w = \frac{R}{2}(v_r - v_l) \tag{5}$$

III. ОБУЧЕНИЕ КОЛЕСНОГО МОДУЛЯ

Проведем декомпозицию роботизированной платформы на независимые колесные модули-агенты. Каждый модуль является самостоятельной автономной единицей, с индивидуальным поведением. Агенты располагаются в двумерной среде с привязкой к маяку, как показано на рисунке 4. Местоположение маяка определяется координатами (x_b, y_b). Маяк – точка в пространстве, обозначающая центр разворота робота. Радиус разворота ρ – это расстояние от центра модуля до маяка. Ошибка угла поворота вычисляется по формуле 6 (φ_{center} и φ_{robot} известны из среды).

$$\varphi_{err} = \varphi_{center} - \varphi_{robot} \tag{6}$$

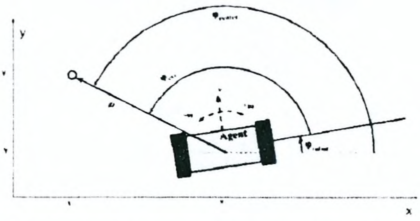


Рисунок 4 – Состояние агента по отношению к маяку

При объединении модулей в целостную группу, агентам необходимо координировать свои действия для поддержания целостности формации. Одним из способов управления формацией агентов является создание виртуальной структуры [6]. Основная идея данного подхода состоит в определении виртуального лидера, задающего виртуальные координаты. Таким образом, состояние каждого агента будет определяться относительно виртуального лидера или виртуального центра координат.

На рисунке 5 (x_i, y_i) и (x_i^{opt}, y_i^{opt}) представляют координаты реального и целевого положения i-го модуля соответственно, $\bar{d}_i^{err}$ представляет вектор отклонения для *i*-го модуля от правильного положения в платформе (7).

$$\bar{d}_i^{err} = \bar{d}_i' - \bar{d}_i^{opt} \tag{7}$$

где $\bar{d}_i'$ – вектор расстояния до виртуального центра от текущего положения модуля, и $\bar{d}_i^{opt}$ – вектор эталонного расстояния между виртуальным центром и *i*-м агентом, которое получено из топологии платформы.

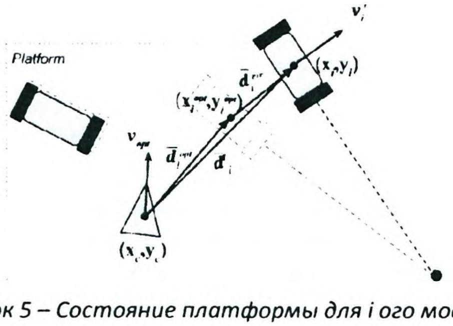


Рисунок 5 – Состояние платформы для i ого модуля

IV. ИНТЕЛЛЕКТУАЛЬНАЯ СИСТЕМА УПРАВЛЕНИЯ МОДУЛЕЙ

Интеллектуальная система управления [7] построена на основе обучения с подкреплением и решает две задачи: позиционирует модули относительно точки вращения (6) и координирует согласованное движение модулей (7). Обучение с подкреплением является методом обучения автономных агентов для нахождения оптимальных стратегий поведения в неизвестной среде [8]. Метод основан на исследовании агентом пространства состояний и нахождении тех пар состояние-действие, которые обеспечивают суммарный ожидаемый максимум награды.

Обучение агента позиционированию означает положительное подкрепление тех действий, которые минимизируют угол ошибки. В результате обучения и обобщения полученной стратегии, агент способен поддерживать значение угла поворота при больших отклонениях, позиционироваться относительно любых углов поворота, даже если они изменяются во время движения. Обученные агенты способны держать строй формации, минимально мешая друг другу при движении реального робота и тем самым увеличивая энергоэффективность управления.

Для обучения скоординированному поведению агентов используется расширение стандартной модели многоагентного обучения с подкреплением, основанное на использовании виртуального лидера [9]–[11]. Идея подхода в том, что виртуальный лидер оценивает влияние поведения каждого агента на общую целостность платформы. Конструктивные влияние рассматриваются как положительные подкрепления, а деструктивные – как негативные.

Модель многоагентного обучения с подкреплением с использованием виртуального лидера, решающая задачу кооперативного движения, изображена на рисунке 6.

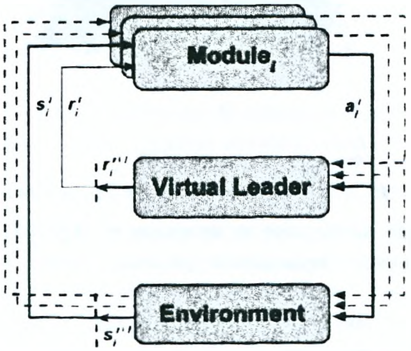


Рисунок 6 – Архитектура подкрепляющего обучения для мультиагентной системы

Модуль i, находясь в состоянии s_i^t, выбирает действие α_i^t, используя текущую стратегию выбора действий, и переходит в следующее состояние s_i^{t+1}. Платформа получает данные об изменениях после выполнения действия, вычисляет и присваивает награду r_i^{t+1} модулю как обратную связь успешности данного действия.

Схожий с Q-learning алгоритм (8) используется для обновления стратегии модуля. Главное отличие заключается в том, что в (8) награда назначается виртуальным лидером, а не окружающей средой:

$$\Delta Q_i(s_i^t, a_i^t) = \alpha[r_i^{t+1} + \gamma \max_{a \in A(s_i^{t+1})} Q_i(s_i^{t+1}, a) - Q_i(s_i^t, a_i^t)] \qquad (8)$$

V. РЕЗУЛЬТАТЫ МОДЕЛИРОВАНИЯ

Первый этап моделирования заключается в позиционировании модулей относительно маяка. В результате агенты занимают правильное положение для движения по кругу. Обучение происходит один раз для одного агента перед кооперативным этапом моделирования [7]. Полученные правила сохраняются и копируются для других агентов. Топология Q-функции, которая обучалась в течение 720 эпох, показана на рис. 7.

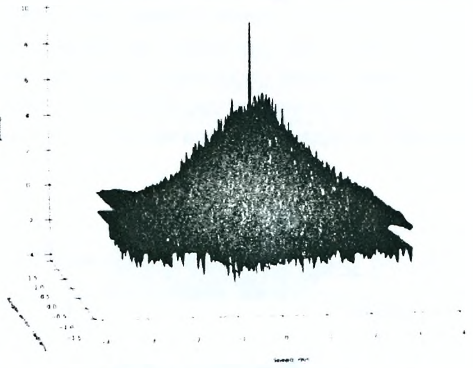
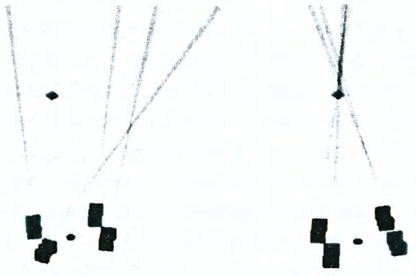


Рисунок 7 – Топология Q-функции после обучения одного модуля

На рис. 8а показано начальное положение платформы, на рис. 8б показан результат автоматического позиционирования агентов, используя обученную стратегию.

На рис. 9 показан результат эксперимента совместного движения платформы после обучения. Такое обучение в среднем занимает 11000 эпох. Внешние параметры моделирования: шаг обучения $\alpha = 0,4$, коэффициент обесценивания $\gamma = 0,7$, оптимальная скорость $\omega_{opt} = 0,8$ рад/с, угол торможения $\varphi_{stop} = 0,16$ рад.

Для моделирования ω_{opt} используется как константное значение скорости, чтобы показать применимость такого подхода. Для реального робота должны производится расчёты такой функции используя документацию на моторы [12] и другие параметры (передаточное число, размер, загруженность). φ_{stop} в данном случае подбирается вручную. В дальнейших исследованиях планируется разработать алгоритм автоматического подбора угла торможения.



а) б)

Рисунок 8 – Результаты моделирования обученных агентов платформы задаче поворота

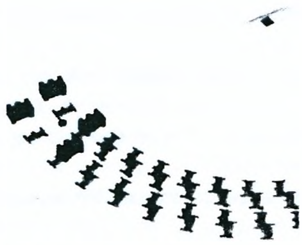


Рисунок 9 – Результаты моделирования обученных агентов платформы задаче совместного движения

VI. ВЫВОДЫ

Экспериментальная часть демонстрирует успешное применение многоагентного подхода на основе виртуального лидера с использованием обучения с подкреплением для задачи эффективного управления многоколесной роботизированной платформой. Предлагаемый подход включает множество Q-learning агентов, которые определяют оптимальное управление модулями относительно виртуального лидера. Достоинства разработанного подхода:

• Декомпозиция обозначает, что вместо построения глобальной Q-функции мы строим множество локальных;

• Адаптивность – платформа адаптирует свое поведение для динамически изменяемого маяка и перенастраивает свою траекторию;

• Масштабируемость и обобщающая способность – один метод обучения используется для множества агентов, для любой позиции маяка и для любой позиции агента на платформе.

REFERENCES

[1] Walters, D.G. The Whole Life Efficiency of Electric Motors // Energy Efficiency Improvements in Electric Motors and Drives. – Springer, 1997. – P. 81–94.

[2] Mei Y., Lu Y.-H., Hu Y. C., Lee G. Energy-Efficient Motion Planning for Mobile Robots // Robotics and Automation. Proceedings of IEEE International conference on ICRA'04. IEEE. – 2004. – Vol. 5. – P. 4344–4349.

[3] Benedettelli, D., Garulli, A. and Giannitrapani, A. Experimental validation of collective circular motion for nonholonomic multi-vehicle systems // Robotics and Autonomous Systems. 2010. No 58. P. 1028–1036.

[4] Stetter, R., Ziemniak, P., Pachinski, A. Realization and Control of a Mobile Robot // Research and Education in Robotics-EUROBOT 2010, Communication in Computer and Information Science. Springer. – 2011. – Vol. 156. – P. 130–140.

[5] G. Dudek and M. Jenkin, Computational Principles of Mobile Robotics. Cambridge University Press. – 2010.

[6] Ren, W. Distributed coordination architecture for multi-robot formation control / W. Ren, N. Sorensen // Robotics and Autonomous Systems. – 2008. – Vol. 56, № 4. – P. 324–333.

[7] Dziomin, U.; Kabysh, A. A multi-agent reinforcement learning approach for the efficient control of mobile robot / U. Dziomin; A. Kabysh, V. Golovko; R. Stetter // Intelligent Data Acquisition and Advanced Computing Systems (IDAACS). – 2013. – Vol.02. – P. 867–873.

[8] Sutton, R.S. Reinforcement Learning: An Introduction / R.S. Sutton, A.G. Barto // MIT Press. – 1998. – 322 p.

[9] Kabysh, A. General model for organizing interactions in multi-agent systems / A. Kabysh, V. Golovko // International Journal of Computing. – 2012. – Vol. 11. Issue 3. – P. 224–233.

[10] Kabysh, A. Influence Learning for Multi-Agent Systems Based on Reinforcement Learning / A. Kabysh, V. Golovko // International Journal of Computing. – 2012. – Vol. 11, Issue 1. – P. 39–44.

[11] Kabysh, A. Influence model and reinforcement learning for multi agent coordination / A. Kabysh, V. Golovko, K. Madani // Journal of Qafqaz University, Mathematics and Computer Science. – 2012. – № 33. – P. 58-64.

[12] Maxon motor uk ltd. Brushless motor RE35 Graphite Brushes, 90 Watt, 12V DC motor datasheet. Available: http://www.maxonmotor.com/medias/sys_master/8806653427742/13_104_EN.pdf.

ВОЗМОЖНОСТИ И ОГРАНИЧЕНИЯ ПОЛЬЗОВАТЕЛЬСКОГО ЛАЗЕРНОГО 3D - СКАНЕРА

Франкевич И., Ситка В.

Научный руководитель - к.т.н., доц. каф. ПТМ Вельган Р.

Национальный университет «Львівська політэхника»,

ИКТА, кафедра ПТМ

Современное промышленное производство характеризуется активным использованием компьютерных технологий. В частности, на стадии проектирования используются программы для твердотельного моделирования, а на стадии производства - 3D-сканеры. Компьютерная обработка данных, полученных от сканера, делает возможным определение геометрических параметров и формы детали, что в свою очередь дает возможность реализовать ее контроль.

Большинство известных средств контроля базируются на контактных методах. Но они не всегда соответствуют современным требованиям, точности и быстродействия измерений [1]. Для быстрого и бесконтактного получения трехмерных координат поверхности используются лазерные 3D - сканеры. Обычно такая система состоит из источника лазерного излучения, цифровой камеры и компьютера с программным обеспечением для управления компонентами системы, сбора данных от камеры и вычисления 3D - координат. Лазерный сканер "просматривает" поверхность объекта точка за точкой и формирует соответствующий набор ее координат. Кроме определения геометрических размеров, областями применения таких сканеров являются компьютерная графика, робототехника, промышленный дизайн, медицинские исследования, археология, реверс- инжиниринг, мультимедиа и веб-дизайн. Однако для таких устройств характерно сложное и дорогостоящее оборудование и программное обеспечение. Для учебных и исследовательских целей в университетах существует потребность в недорогих вариантах таких устройств с обеспечением возможности изменения оборудования. Почти безальтернативным в этом классе является DAVID - laserscanner [2]. Плюсы этого сканера: низкая цена, простота и функциональность. В сканере DAVID предусмотрена возможность изменения направления освещения, что позволяет избежать проблемы лазерной тени. Программное обеспечение имеет гибкий и простой интерфейс, управляет 3D - сканированием и преобразовывает полученные наборы данных в модели. Его положительным аспектом является возможность получить данные в форматах STL, OBJ и PLY, которые можно импортировать в большинство 3D - редакторов. Это дает широкие возможности для последующей обработки данных. Кроме того, этот программный продукт является аппаратно - независимым, что позволяет гибко менять аппаратное обеспечение в зависимости от задачи сканирования.

Целью этой работы было ознакомиться с особенностями использования этой системы, исследовать ее возможности и ограничения, разработать способы улучшения результатов сканирования и решения для автоматизации сканера.

В результате выполненных экспериментов выделены следующие проблемы сканирования:

1) зависимость от условий окружающего освещения;

2) проблемность сканирования объектов с экстремальными рефлективнымы свойствами;

3) зависимость от точности фокусировки лазерного луча;

4) воздействие механических вибраций;

5) трудоемкость операции сканирования.

Для устранения указанных проблем предусмотрены такие меры:

1) разработаны рекомендации по выбору параметров экспозиции, хотя наилучшие результаты (большая плотность точек и небольшое количество выбросов) получены при использовании ширмы окружающего освещения;

2) на данном этапе лучшим решением выглядит покрытие детали слоем меловой пыли;

3) разработан небольшой программный модуль, который контролирует точность фокусировки;

4) предусмотрены меры изоляции от механических вибраций;

5) разработана конструкция поворотного механизма для лазерной указки. Конструкция предусматривает использование шагового двигателя. Это позволит плавно сканировать объект.

Результатом проделанной работы, кроме полученных 3D-моделей, являются разработанные рекомендации для эффективного использования сканера.

Литература

[1.] Ильченко, В.М. Методы и средства контроля деталей компьютеризированными лазерными информационно-измерительными системами: автореф. дис. канд. тех. наук / В.Н. Ильченко, Национальный авиационный университет. - Киев, 2009. - 19 с.

[2.] DAVID 3D Scanner [electronic resource] / DAVID 3D Solutions GbR.- http://www.david-3d.com/

ИСПОЛЬЗОВАНИЕ ПЧЕЛИНОГО И МУРАВЬИНОГО АЛГОРИТМОВ ДЛЯ ОРГАНИЗАЦИИ РАБОТЫ АВТОМАТИЧЕСКОГО СКЛАДА

Жулкэвський В.
НУ «Львивська политэхника», Львив, Украина, ИКТА, Кафедра ПТМ
Науч. рук. к.т.н., асист. каф. Рэпэтыло Т.

Сегодня все большее распространение получают автоматические склады продовольствия, деталей машин и т.д. Они имеют ряд преимуществ перед традиционными, а именно: большая скорость доступа к продукции, дешевле в обслуживании (из-за малого количества персонала), возможность при меньшем объеме складского помещения разместить больше продукции. Поэтому встает вопрос организации таких складов.

Проблема в том, что из-за их больших объемов неэффективно просто заполнять свободные ячейки, нужно учитывать удобство доступа к ним (меньший путь перемещения роботов к выходу) и то, что продукция может быть востребована в определенной последовательности (например, сначала каркас устройства, затем электроника, затем двигатели и т.д.). Поэтому система должна решать не менее двух задач: находить группы свободных ячеек и находить кратчайший путь доставки. С такими задачами можно справиться с использованием идеологии коллективного интеллекта [1]. Системы коллективного интеллекта, как правило, состоят из множества агентов (многоагентная система), которые локально взаимодействуют между собой и с окружающей средой. Сами агенты обычно довольно простые, но все вместе создают так называемый коллективный интеллект [1]. Такими агентами у нас будут транспортные роботы. Две поставленные задачи лучше решать различными алгоритмами.

Сначала рассмотрим задачу нахождения наибольших групп свободных ячеек на складе. Для этой цели наиболее подходит пчелиный алгоритм (Artificial Bee Colony). Это алгоритм для нахождения глобальных экстремумов (максимумов или минимумов) сложных многомерных функций. В информатике и исследовании операций пчелиный алгоритм на основе алгоритма поиска впервые разработан в 2005 году [2] . Он имитирует поведение питания стаи пчел. В базовой версии алгоритм выполняет своего рода соседний поиск в сочетании со случайным поиском и может использоваться для комбинаторной оптимизации и функциональной оптимизации. Кроме этого, метод роя пчел можно эффективно разделить на несколько параллельных процессов, за счет чего значительно увеличится его скорость [3].

Каждая пчела в рое рассматривается как частица или агент. Все частицы роя действуют индивидуально согласно одному управляющему принципу: ускоряться в направлении лучшей персональной и лучшей общей позиции, постоянно проверяя значение текущей позиции. Позиция - аналогично расположению пчелы на поле представлены координатами на плоскости. Однако в общем случае можно расширить эту идею в любое N - мерное пространство в соответствии с поставленной задачей. Это N - мерное пространство является областью решений для задачи, где каждый набор координат представляет решение. Пригодность - по аналогии с примером пчелиного роя, функция пригодности будет иметь плотность цветов: чем больше плотность, тем лучше позиция. Функция пригодности служит средством связи между физической проблемой и алгоритмом оптимизации. Персональная лучшая позиция - по аналогии с пчелиным роем, каждая пчела помнит позицию, где она сама обнаружила наибольшее количество цветов. Эта позиция с наибольшим значением годности, обнаруженная пчелой, известна как персональная лучшая позиция (ПНП). Каждая пчела имеет собственное ПНП . В каждой точке вдоль пути движения пчела сравнивает значение годности текущей позиции со значением ПНП . Если текущая позиция имеет значение пригодности выше, значение ПНП заменяется на значение текущей позиции. Глобальная лучшая позиция - каждая пчела также каким-то образом узнает область наибольшей концентрации цветов, определенную всем роем. Эта позиция наибольшей пригодности известна как глобальная лучшая позиция (ГНП). Для всего роя это одна ГНП, к которой стремится каждая пчела. В каждой точке в течение всего пути каждая пчела сравнивает пригодность ее текущей позиции в ГНП. В случае, если любая пчела обнаружит позицию с более высокой пригодностью, ГНП заменяется текущей позицией этой пчелы. ГНП будет нашей самой большой группой свободных ячеек.

Теперь нужно выбрать алгоритм для транспортировки продукции кратчайшим путем, то есть решить задачу коммивояжера. Один из эффективных алгоритмов для нахождения приближенных решений задачи коммивояжера, а также аналогичных задач поиска маршрутов на графах является муравьиный алгоритм. Суть алгоритма заключается в анализе и использовании модели поведения муравьев, ищущих пути от колонии до еды. Алгоритмы муравья основаны на применении нескольких агентов и обладают специфическими свойствами, присущими муравьям, и используют их для ориентации в физическом пространстве. В основе алгоритма лежит поведение муравьиной колонии - маркировка удачных дорог большим количеством феромона [4]. Важным свойством муравьиных алгоритмов является неконвергентность: даже после большого числа итераций одновременно исследуется множество вариантов решения, в результате чего не происходит длительных временных задержек в локальных экстремумах. Все это позволяет рекомендовать применение муравьиных алгоритмов для организации транспортных роботов на автоматических складах.

Таким образом, система автоматического склада должна использовать комбинацию из двух алгоритмов: пчелиного для нахождения группы свободных ячеек и муравьиного для нахождения кратчайшего пути доставки продукции из склада / на склад.

Литература

[1] Beni, G., Wang, J. Swarm Intelligence in Cellular Robotic Systems, Proceed. NATO Advanced Workshop on Robots and Biological Systems, Tuscany, Italy, June 26-30 (1989)

[2] Естественные алгоритмы. Алгоритм поведения роя пчёл [Электронный ресурс]. – Режим доступа: http://habrahabr.ru/post/104055/

[3] Pham DT, Ghanbarzadeh A, Koc E, Otri S, Rahim S and Zaidi M. The Bees Algorithm. Technical Note, Manufacturing Engineering Centre, Cardiff University, UK, 2005.

[4] M. Dorigo & L. M. Gambardella, 1997. «Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem». IEEE Transactions on Evolutionary Computation, 1 (1): 53-66

ВОЗМОЖНОСТЬ ИСПОЛЬЗОВАНИЯ ОРЛИНОГО АЛГОРИТМА ПРИ УПРАВЛЕНИИ РАСПРЕДЕЛЕННОЙ МЕХАТРОННОЙ СИСТЕМОЙ

Черняк Р.
НУ «Львивська политэхника», ИКТА, ПТМ, Украина
Научный руководитель – к.т.н., асист. каф. ПТМ Репетыло Т.

Часто при управлении распределенными мехатронным системами возникает необходимость глобальной оптимизации. Большинство задач глобальной оптимизации являются нелинейными и, следовательно, трудно решимыми. Такие задачи можно решать с помощью идеологии коллективного интеллекта.

Целью этой работы было ознакомиться с особенностями орлиного алгоритма и возможностью его приспособления к управлению распределенной мехатронной системой.

В статье [1] авторы Xin-She Yang и Suash Deb предоставили новый двухступенчатый гибридный метод поиска под названием «Орел» (с англ. Eagle). Здесь рассмотрена идеализированная двухступенчатая стратегия на примере поведение орла на охоте: орел выполняет прогулку во всей области поиска, как только он находит добычу, он переходит к стратегии погони. Стратегию погони можно рассматривать также как интенсивный локальный поиск с использованием любых эффективных алгоритмов, таких как Particle Swarm Optimization (PSO) или Firefly (FA). Глобальный оптимум в принципе может быть найден из любого начального. Прежде чем провести оптимизацию, нужно либо начать с более широкой области и уменьшить его, либо использовать меньший размер области, а затем постепенно расширять его.

В [1] проведено сравнение (с использованием Matlab) стратегии «Орел» (ES) с PSO для различных стандартных тестовых функции. Было доказано, что успешность нахождения глобального оптимума для алгоритма Орел (ES) ровна 100 %. В то время как для PSO она составила 90-100%.

Мы видим, что алгоритм Орел (ES) может быть эффективнее алгоритма PSO, но нуждается в дальнейшем исследовании и сравнении с другими алгоритмами коллективного интеллекта в контексте управления распределенными мехатронным системами. На что может быть направлена дальнейшая наша работа.

Литература

[1] Xin-She Yang, Suash Deb. Eagle strategy using levy walk and firefly algorithms for stochastic optimization, in: Nature Inspired Cooperative Strategy for Optimization (NISCO 2010), (2010).

СОДЕРЖАНИЕ

Научное издание

Robotics and Artificial Intelligence. Problems and perspective

PROCEEDINGS of International Conference
4-6 November 2013

Ответственный за выпуск: Дунец А.П.
Редактор: Боровикова Е.А.
Корректор: Никитчик Е.В.
Компьютерная верстка: Соколюк А.П.

Издательство БрГТУ.
Свидетельство о государственной регистрации издателя, изготовителя, распространителя печатных изданий № 1/235 от 24.03.2014 г.
Подписано к печати 12.11.2014 г. Формат 60x84 $^{1}/_{16}$.
Бумага «Снегурочка». Гарнитура Calibri.
Усл. печ. л. 3,5. Уч.-изд. л. 3,75. Заказ № 936.
Тираж 30 экз. Отпечатано на ризографе учреждения образования «Брестский государственный технический университет».
224017, г. Брест, ул. Московская, 267.

ISBN 978-985-493-313-9

9 789854 933139