Simulation of Semiautonomy Mode for IBIS Mobile Robot with Analysis of Sensor Failure Tolerance

Submitted 12th April 2011; accepted 6th June 2011

Jakub Bartoszek, Maciej Trojnacki, Piotr Bigaj

Abstract:

This work is concerned on sensitivity analysis of semiautonomy algorithm of mobile combat robot to environmental sensors' damage. The construction of the robot, semiautonomy algorithm and used sensors have been described. This algorithm takes into account environmental sensors' damage. Simulation research results of semiautonomy algorithm using Matlab/Simulink package was presented. This research was performed for normal environmental sensors' operation and for selected sensors' damage. On that basis, sensitivity of semiautonomy algorithm to selected environmental sensors damage was tested.

Keywords: *mobile robot, semiautonomy, sensor failure.*

1. Introduction

Extraordinarily large funds and and the work of many research centres around the globe are focused on developing increasingly efficient control algorithms for mobile robots. The main objective behind most this work is to achieve a level of development mobile robotics described as full autonomy. The moment in which this goal is achieved will have a profound effect on the direction and speed of the development of our civilisation. Achieving full autonomy will allow for a great increase in functionality and the number of possible uses for autonomous vehicles. The most important asset of these vehicles is going to be their lack of necessity for them to be controlled by a human operator and their need for only minimal supervision.

One of the most discussed issues in the case of research on mobile robot autonomy is the problem of them moving in a dynamic, highly diverse and unknown environment [1]. To put it simply, the issue can brought down to plotting a movement trajectory for the robot, from its current location to a designated target in a way that allows the robot to reach its destination in the shortest time possible while at the same time avoiding collisions [2]. Algorithms enabling the selection of an optimal path based on various quality indicators are being developed in research centres all over the world. Each algorithm makes use of at least 2 types of sensors – state and environment.

The state sensors include all of the sensors that allow for the determination of the robot's current status. In most cases they are only able to determine the robot's position. The methods used to determine position based on available data can be divided into several separate groups. The first of these includes the use odometry and often involves an error rapidly growing as the driver program loops. The second method relies on knowledge of a statistical map of the terrain and determining position based on markers set up on the robot's route or based on landmarks identified with the use of visual sensors, for example. The third group is based on GPS and similar systems. Each of these methods is not fully reliable and must involve safeguards to protect the program from incorrect data. The data may be a result of a sensor malfunction or, in the case of systems similar to GPS, lack of satellite contact.

Environmental sensors are a group that enables the modelling of a virtual environment, based on the actual environment in which the robot is currently operating. Depending on the number and type of the sensors used, the model may be 3 or 2 dimensional. Based on the power of the robot's CPU this model make take various forms. An approach in which a representation of the model of the environment in which the robot was operating has been used in publications [3], [4], it has been omitted, and control was outlined based only on sensor signals in, for example, publication [5]. The main characteristic of this sensor group is their limited range. In the case of mobile robots designed to work in outdoor environments, these sensors share one other common characteristic – they are vulnerable to various malfunctions. These may be caused by external factors such as collisions with potential obstacles or vibrations caused by the robot's movement they can also be caused by internal factors such as overheating electronic components.

In most cases, sensor malfunctions are extremely difficult to predict. Their nature is often possible to determine only after they have occurred and need to be properly diagnosed. This is why the methods of diagnosing and reacting to sensor malfunctions needs to be included in the first stage of designing the entire system. The literature mentions 2 approaches to this problem [6], [8]. One of them is based on a model enabling the detection of sensor malfunctions [9]-[12]. Some of them require interference into the structure of the system because of the detected malfunction [13].

Another, completely different approach to the problem of detecting and reacting to a sensor malfunctions has been presented by Martin Soika in [6]. The goal of his work was the development of a method that will enable the detection of a previously unnoticed sensor malfunction and an adequate reaction to it. His method is based on determining the level of credibility of the data sent by the sensors. Healey [14] created a system based on a model of sensor malfunctions, however he utilised an artificial neuron net to detect the malfunctions

themselves. This solution enables the implementation of the method into the robot's driver in a way not limiting its autonomy, unfortunately this also increases the complexity of the algorithm.

An important factor enabling automated detection and diagnosis of sensor damage is their redundancy. In this case, the environmental model is being created based on data from a larger number of sensors, which allows for relatively easy detection of a potential malfunction in one of them. A larger number of sensors, often using different measurement methods, requires the use of more elaborate and better optimized algorithms designed for constructing a model of the robot's surrounding environment. This in turn necessitates the use of more efficient CPUs in the robot's driver. Because of this, it is necessary to give up on sensor redundancy in some cases.

This article presents the methods used in reacting to sensor malfunctions used in the control algorithm for the IBIS robot. This algorithm has been described in greater detail in [15].

2. Mobile combat robot

The commercial version of the IBIS (Fig. 1a) is a pyrotechnical and combat robot. It has been adapted to operate on difficult and diverse terrain, such as snow, sand or rocky terrain. The robot's high speed enables it to perform its tasks dynamically. The manipulator gives it a long range of operation, while the design of its drive gives it a great deal of movement flexibility as far as speed is concerned.

a)

b)

Fig. 1. IBIS robot: a – commercial version with manipulator, b –autonomy research variant.

The basic technical specifications of the robot are as follows: mass: 295 kg, dimensions (length x width x height): 1,3 m x 0,85 m x 0,95 m, maximum speed: 8,5 km/h, manipulator lifting capacity: 30 kg, manipulator maximum range: 3,15 m [16], [17].

A version of the IBIS without the manipulator arm has been developed for research purposes (Fig. 1b). This version of the robot has been also equipped with an additional installable frame. This contains the module of the robot responsible for performing semiautonomy. The module contains microprocessors as well as a set of four cameras and sensors designed to detect and locate obstacles. The robot can operate in two modes: teleportation – when controlled by an operator, semiautonomy – when it performs its tasks by itself under the supervision of an operator.

3. Robot sensors

The mobile base of the robot has been extended with a specially designed frame enabling precise positioning of sensors as well as navigational and positioning controllers. The physical architecture of the system is comprised of 4 blocks separated physically and functionally: sensors, positional controller (determining the robot's current position), navigational controller (processing outgoing data and facilitating control) and the engine drivers. Movement controls in teleportation mode is transmitted to the robot *via* an ISM modem, while movement in semiautonomy mode is calculated entirely by the navigational controller.

The localization sensors are mounted on the positional controller and are used to determine the robot's position and orientation. Position is determined in 3 dimensions: longitude, latitude and altitude according to NMEA specifications, thanks to this it can be presented in any form of GIS programming. A monophase GPS receiver supplemented by INS is being used to pinpoint the current position of the robot. Pinpointing of the robot's coordinates in the WGS-84 system is achieved with the use of Kalaman filtration, GPS positioning and inertial navigation. The robot's azimuth is determined by use of a digital compass with inclination compensation. Inclination sensors (inclinometers and accelerometers) are being used to determine the robot's inclination and declination.

Four types of obstacle detection sensors have been mounted on the frame: a 2 dimensional laser scanner, ladars, true-presence radar sensors and tactile sensors (bumpers). These sensors are placed in a way that (Fig. 2), they are able to cover the entire area around the robot and so that most of the gathered information concerns the front.

The purpose of the 2D laser scanner is the detection of obstacles in front of the robot. Its angle scope has been set to 100º. Its maximum range of 80 m is the result of technical limitations of the sensor itself, however the present LOS depends on the angle, at which the beam is directed. The scanner is tilted downward at a small angle, which enables it to detect obstacles smaller than the height on which the scanner is mounted.

Ladars are used to detect obstacles in the immediate vicinity of the robot. They are mounted at various angles facing downwards, which allows them to detect both concave and convex obstacles. One of them is tilted

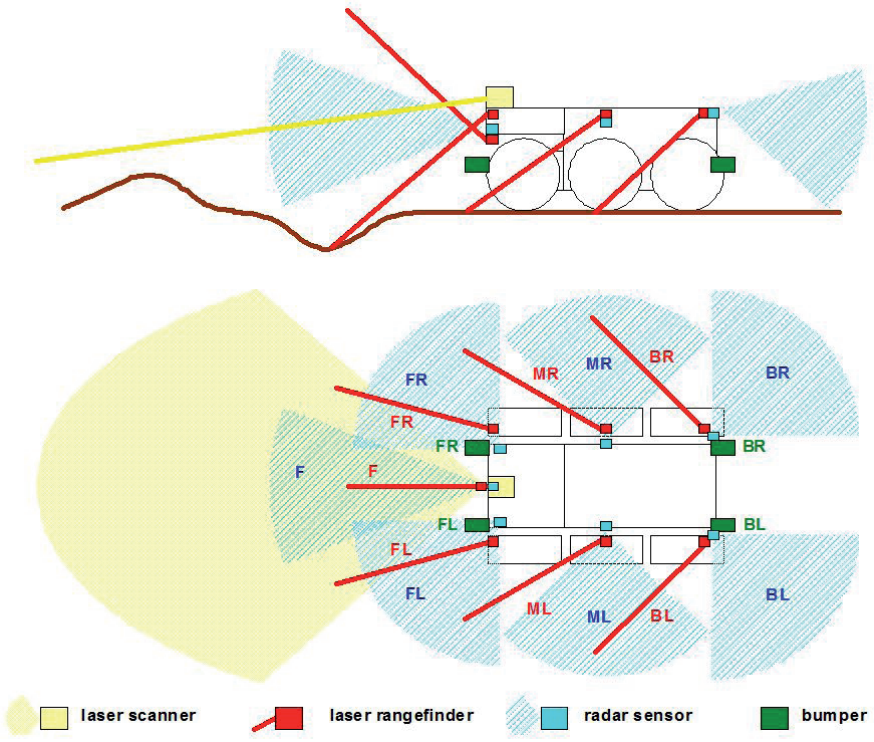

Fig. 2. Sensor placement: a – left side view, b – top side view.

upwards in order to detect obstacles that are too low for the robot. The ladars used enable measuring the distance to the nearest obstacle within a range of 0,5–10 m.

Radar sensors are designed to detect obstacles at long range. The only information they provide is, whether or not there is an object at the specified distance. They work at ranges of 2–15 m. They allow for early warnings about obstacles within the robot's vicinity.

The tactile sensors are tasked with detecting obstacles that have not been detected by the other sensors i and triggering an emergency stop in case the robot collides with something.

4. Semiautonomy algorithm

None of the algorithms designed so far have allowed mobile robots to perform their tasks without error. This is due to the fact that the creators of algorithms are unable to predict all possible situations, as well as sensors being unable to detect and classify all obstacles. Because of this, the following article uses the term, semiautomy. This means that the robot will perform its task by itself, but under the supervision of an operator, who can halt the robot at any time.

The goal of the research described in this paper was to analyse the effect of sensor malfunctions on the algorithm enabling the IBIS to perform its objective, while avoiding detected obstacles.

The control algorithm is comprised of 4 subroutines. The first one of these is responsible for turning the robot towards the target. The second of these is responsible for setting the value of velocity depending on the distance between the robot and the nearest obstacle. If the robot is far away from the target, and the nearest obstacle is located beyond sensor range, the robot will move at maximum speed. The third behaviour is connected to the laser scanner, which is treated as 101 single beams. Modified VHF algorithms [18], [19] are used in order to process the environmental data received from the scanner. The fourth subroutine is patterned on the Braitenberg algorithm, which is based on directly connecting the sensors with actuators, with every connection having its own weighting factor. Depending on the sensor readings and weighting factor the robots able to perform various tasks. The semiauthonomy algorithm as well as its division into separate subroutines has been described in greater detail in [15].

The weighting factor for each of the subroutines is determined based on the active routine. If the current routine is ineffective the virtual tank (WaterTank) overflows and a different routine is activated. The new routine is decided based on an assessment of the locations of the target and obstacles in relation to the robot. Eg., if there are obstacles in front of the robot and the robot must go forward and turn right to reach it, at the moment when the „move to target and avoid obstacles" routine is interrupted, the robot may switch to the „observe obstacles on the left side" routine. The WaterTank is first emptied before switching to a new routine.

One of the most important characteristics of modern control algorithms is their resistance to sensor malfunctions. Malfunctioning sensors may cause significant changes to the robot's behaviour which may cause damage to it or pose a threat to people in the vicinity. In order to avoid such situations the presented algorithm has been modified in order to detect improper data transmitted by damaged sensors.

The location sensor in the form of a GPS receiver will trigger a switch from semiautonomy toteleoperation mode if the signal from the transmitters is lost or there are high discrepancies in the robot's perceived location (above 10 m). The main factors determining the proper functioning of the localisation sensors are discrepancies in the robot's location and its distance to the target in each iteration of the driver.

The environmental sensors on the robot are seven ladars and a 2D scanner. A malfunction in these sensors may manifest itself in different ways, depending on its type.

The first type of malfunction appears when it is impossible to establish contact with the sensor or if the sensor itself is sending an error message after it has been initialised. In such a case, the microprocessor responsible for gathering information about the sensors and communication with them sends a malfunction message to the main microprocessor. This information is sent in the same way as the proper value sent back by the sensors but the value is 0.01 m. This is equivalent to a situation in which an obstacle is at that distance from the sensor, which is impossible under normal working conditions.

The second type of malfunction is harder to detect. It appears when the sensor is returning false data. A malfunction of this type may be caused by multiple factors such as for example large vibrations of the mobile platform which may occur during fast movement on rough terrain or the deflection of a laser beam of a flat surface tilted at a low angle. Detecting and filtering false data in such cases is extremely difficoult. In steering algorithm only values going beyond measurement range which arising from the construction of a mobile robot and installing manner of sensors are filter off.

Detecting and diagnosing a malfunction causes a change in the weighting factor for the proper subroutine. In the case of detecting a sensor malfunction the weighting for subroutine 3 is set to zero, which means it no longer has any effect on the functioning of the algorithm and the robot itself. In the case of one or more ladars malfunctioning the weighting for subroutine four is being changed. The weighting is changed based on reliability, which is determined using the method shown below.

The controls for the robot's wheels are determined depending on:

$$u_i = \sum_{j=1}^{4} c_j u_{ij} \ , \tag{1}$$

where: i = {L, R}, L and R – respectfully the left and right wheels of the robot, j – the number of the subroutine, c_j – the weighting factor of subroutine j defining the influence this algorithm has over general control, u_{ij} – controls for wheel i and subroutine j.

The weighting for particular subroutines is as follows: $c_1 = 1$ (malfunctions of the sensor related to the determination of the robot's orientation are not being analysed), $c_2 = 1$ (malfunctions of the sensor related to the robot's speed are not analysed), $c_3 = 0$ in case of a laser scanner malfunction, $c_3 = c_3^* = 0{,}6$ if the scanner is functional, while the weighting for the last subroutine is calculated with:

$$c_4 = c_4^* \Delta \tag{2}$$

where: $c_4^* = 0{,}4$ means the weighting of the subroutine related to the ladars if all of them are functional, Δ – the factor of reliability of the information coming from the ladars depending on the damage sustained by them.

It is worth noting that in case of a malfunction of any of the sensor there is a relation:

$$c_3 + c_4 < 1 \, . \tag{3}$$

An approach has been adopted in which the robot's maximum speed is limited in the case of an environmental sensor malfunction, in order to ensure safety for the robot. In a different approach the weighting could be increased to the subroutine, for which the sensors are functional, so that $c_3 + c_4 = 1$.

The reliability factor is determined in relation:

$$\Delta = \frac{1}{n + 0{,}88} - 0{,}13 \tag{4}$$

where: n means the number of malfunctioning ladars.

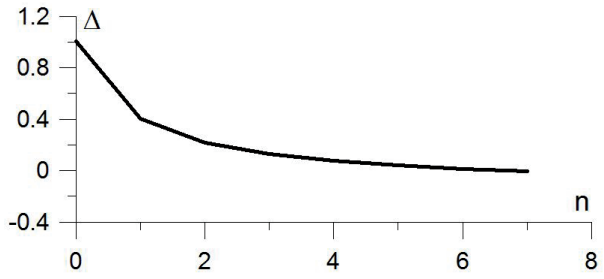

Fig. 3. Change in the reliability factor in relation to the number of malfunctioning ladars.

The changing of value for this factor in relation to the number of malfunctioning ladars has been illustrated in Fig. 3.

5. Simulation reasearch

In order to perform simulation research on autonomy methods, a simulated environment has been prepared based on the Matlab/Simulink packet. The programming responsible for environmental simulations and determining the data displayed by virtual sensors has been separated for the programming performing the robot's control algorithm in teloperation and semiautonomy modes. This second programming has been prepared so that, to implement it into the robot's microcontroller quickly and easily. The robot's simulated environment has been prepared with the V-Realm Builder program objects and scenes have been approximated with the use of cuboids and cylinders.

This work presents the research data in the case where 4 obstacles were placed on the robot's route to the target. The robot preformed the complex „move towards the target and avoid obstacles" routine. As part of the simulation research the mobile robot's semiautonomy algorithm has been evaluated in terms of vulnerability to the malfunction of select environmental sensors.

In order to receive a conclusive assessment of the results, the following quality indicators have been input:

a) The square sum of the robot's distance to the target

$$E = \sum_{i=1}^{n} e_i^2 \Delta t \ , \tag{5}$$

where e_i – the robot's distance to the target in the i iteration, n – the number of iterations before the robot reaches its target or the simulation is ended before the target is reached,

b) Standard variance in the robot's speed

$$S = \sqrt{\frac{\sum_{i=1}^{n} (v_i - \bar{v})^2}{n-1}} \ , \tag{6}$$

where: v_i – robot's speed in the i iteration, $\bar{v}$ – average movement speed,

c) Length of the route from the starting position to the target

$$s = \sum_{i=1}^{n} v_i \Delta t \ , \tag{7}$$

d) The time it takes for the robot to reach the target T, assuming the target is achieved for $e \leq 0{,}5$ [m],

e) Robot's maximum speed v_{max} (within a timeframe from t_0 to T, where t_0 is the moment when the sensor malfunction occurred),
f) Robot's average speed $\overline{v}$ (within a timeframe from 0 to T).

It should be noted that for designators a – d one should move towards minimallisation, while for designators e and f their maximalisation.

In the case in which the robot cannot reach the target within the assumed time T_{max}, it is assumed that quality designators s and T reach the value of $+\infty$, while remaining designators reach the values calculated for T_{max}. This work assumes that T_{max} = 100 [s].

Simulation 1 – all environmental sensors are functional

The first presented simulation has been preformed with the assumption that all o the robot's environmental sensors are functional. The results of this simulation presented in Fig. 4 and in Tab. 1, are used as reference for the remaining simulations, in which the malfunctions of select sensors are assumed.

The ladar indicators shown in Fig. 4e are consistent with Fig. 2. While marking the laser scanner beams on pic. 4f they have been designated Lα, where α is the scanner's angle (in degrees). A positive angle means a tilt towards the left while a negative one means a tilt towards the right. The readings from every tenth scanner beam have been shown in Fig. 4f. The control values for the left and right

Quality indicators	E [m²s]	S [m/s]	s [m]	T [s]	v_{max} [m/s]	$\overline{v}$ [m/s]
Value	8 533	0,284	28,2	38,8	1,34	0,726

Tab. 1. Received values for quality indicators in simulation 1.

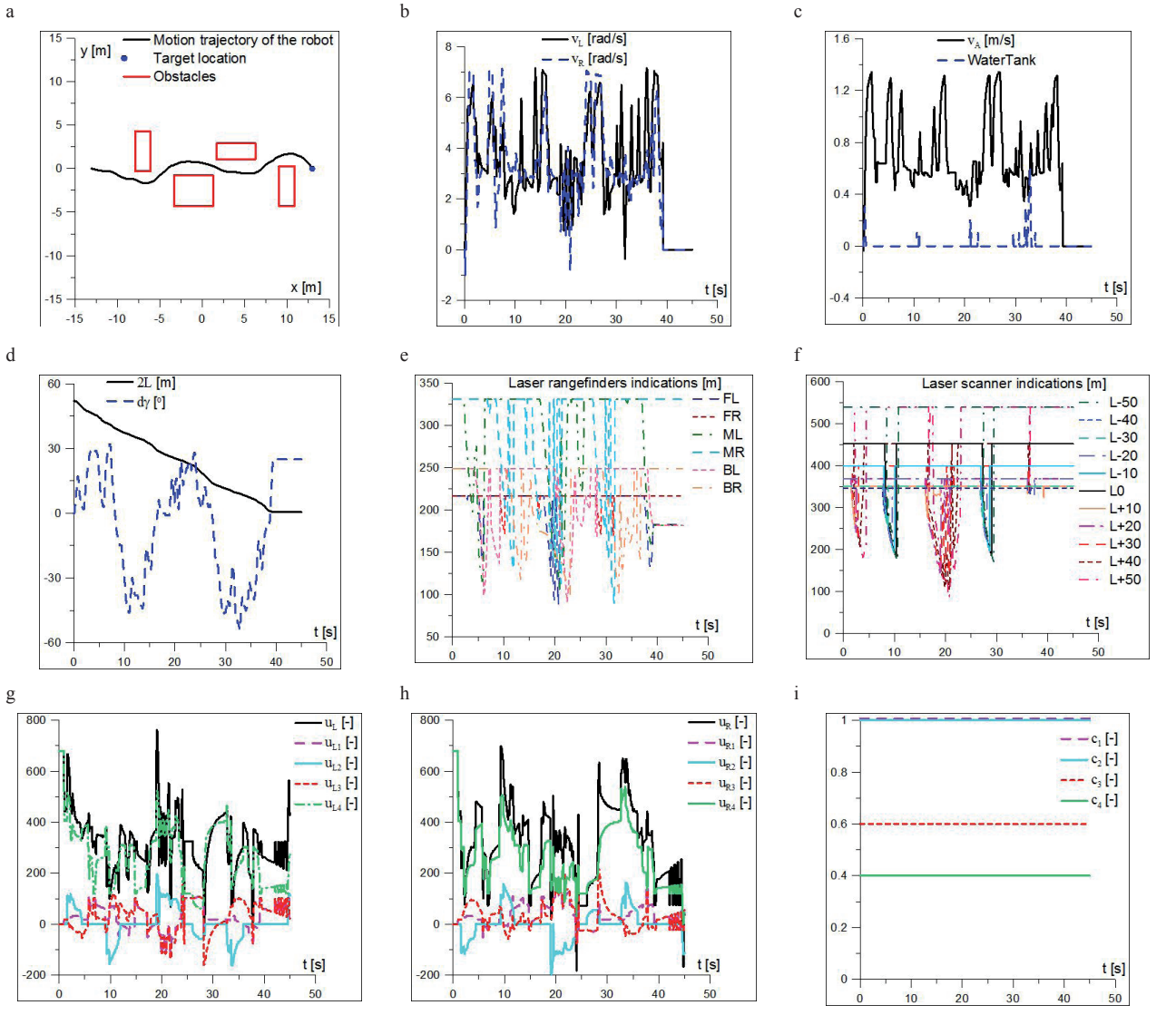

Fig. 4. *Simulation 1 results: a – robot's movement trajectory, with the location of the target and obstacles being marked, b – linear speeds for the robot's left and right wheels, c – robot's movement speed and changes in the WaterTank parameter, d – robot's distance and angle to the target, e – ladar readings, f – scanner readings for select 11 beams, g-i – control values and weighting for specific routines.*

wheels of the robot (u_{Li} and u_{Ri}, i = 1 ... 4) for the previously described subroutines as well as general control (u_L and u_R) have been presented in Figs. 4g and 4h. The weighting for each of the robot's subroutines remains the same (Fig. 4i) during the robot's movement. The weighting is changed only if an environmental sensor malfunction is detected.

The simulations indicate that the developed method enables the robot to avoid obstacles and reach its designated target. The robot moves at a reduced speed, which is related to the fact that it moves slower the closer it is to an obstacle. The distance between the target and the robot decreases constantly during its movement. While evading obstacles, the work of the ladars and laser scanners becomes visible; their readings are subject to noticeable change. The effectiveness of the routine is determined with the WaterTank parameter, which has the largest values when the robot is between obstacles, which is visible in Fig. 4c.

Simulation 2 – the ladar marked as ML malfunctions after the time of 1 s

The second simulation has been performed in the case of a malfunctioning the ladar marked in Fig. 2 as ML. The malfunction occurred after 1 s of the simulation. The results of this simulation have been illustrated in Fig. 5 and in Tab. 2. After the ladar malfunction has been detected by the semiautonomy algorithm the weighting has been modified for the first subroutine (Fig. 5i) related to the modified Braitenberg algorithm based on readings from the ladars.

Quality indicator	E [m²s]	S [m/s]	s [m]	T [s]	v_{max} [m/s]	$\bar{v}$ [m/s]
Value	10 459	0,282	29,1	57,0	1,36 [1)]	0,511

[1)] Maximum values at the begining have been omited, as they are the effect of the algorithm's functions before the malfunction.

Tab. 2. Received values for quality indicators in simulation 2.

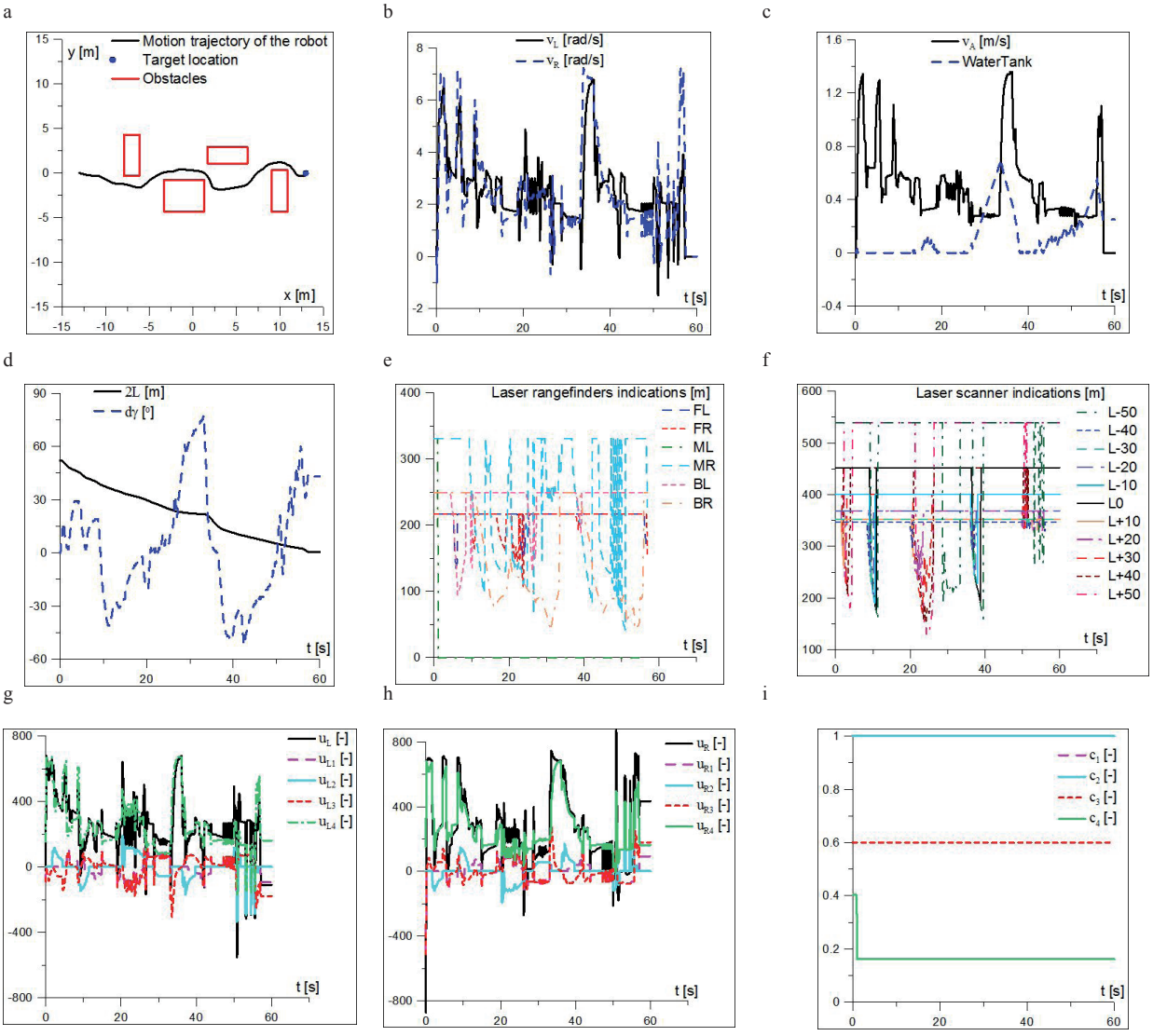

Fig. 5. *Simulation 2 results: a – robot's movement trajectory, with the location of the target and obstacles being marked, b – linear speeds for the robot's left and right wheels, c – robot's movement speed and changes in the WaterTank parameter, d – robot's distance and angle to the target, e – ladar readings, f – scanner readings for select 11 beams, g-i – control values and weighting for specific routines.*

Be comparing the results of simulations 1 and 2 it can be concluded that in the case of a malfunction in the ML ladar, the robot achieved its target after a longer time. Lower values for quality indicators E, s, T and $\overline{v}$ have been noted. While indicators S and v_{max} achieved marginally higher values.

Simulation 3 – laser scanner malfunctions – after a time of 1 s

The last simulation has been performed under the assumption that the laser scanner has malfunctioned after 1 s. The results of the simulation are displayed in Fig. 6 and in Tab. 3.

In this case lower values for all indicators except indicator S have been noted when compared with Simulation 1 which is related to the fact that the robot was moving at a lower and more stable speed. When compared with Simulation 2 the robot achieved its objective faster with a higher average speed which is related to the fact that the robot had less information about obstacles so it moved more freely.

As part of this work simulation research has been conducted assuming the malfunction o remaining sensors in the case of a single sensor malfunctioning.

Based on all simulations preformed it can be concluded that, in most cases, the developed semiautonomy algorithm is capable of dealing well with the malfunction of a single sensor.

The robot animations for the described simulations have been done with the Simulink 3D Animation tool and can be found under address [17].

Quality indicator	E [m²s]	S [m/s]	s [m]	T [s]	v_{max} [m/s]	$\overline{v}$ [m/s]
Value	10 564	0,177	28,9	47,2	0,85[2]	0,6133

[2] Maximum values at the begining have been omited, as they are the effect of the algorithm's functions before the malfunction.

Tab. 3. Received values for quality indicators in simulation 3.

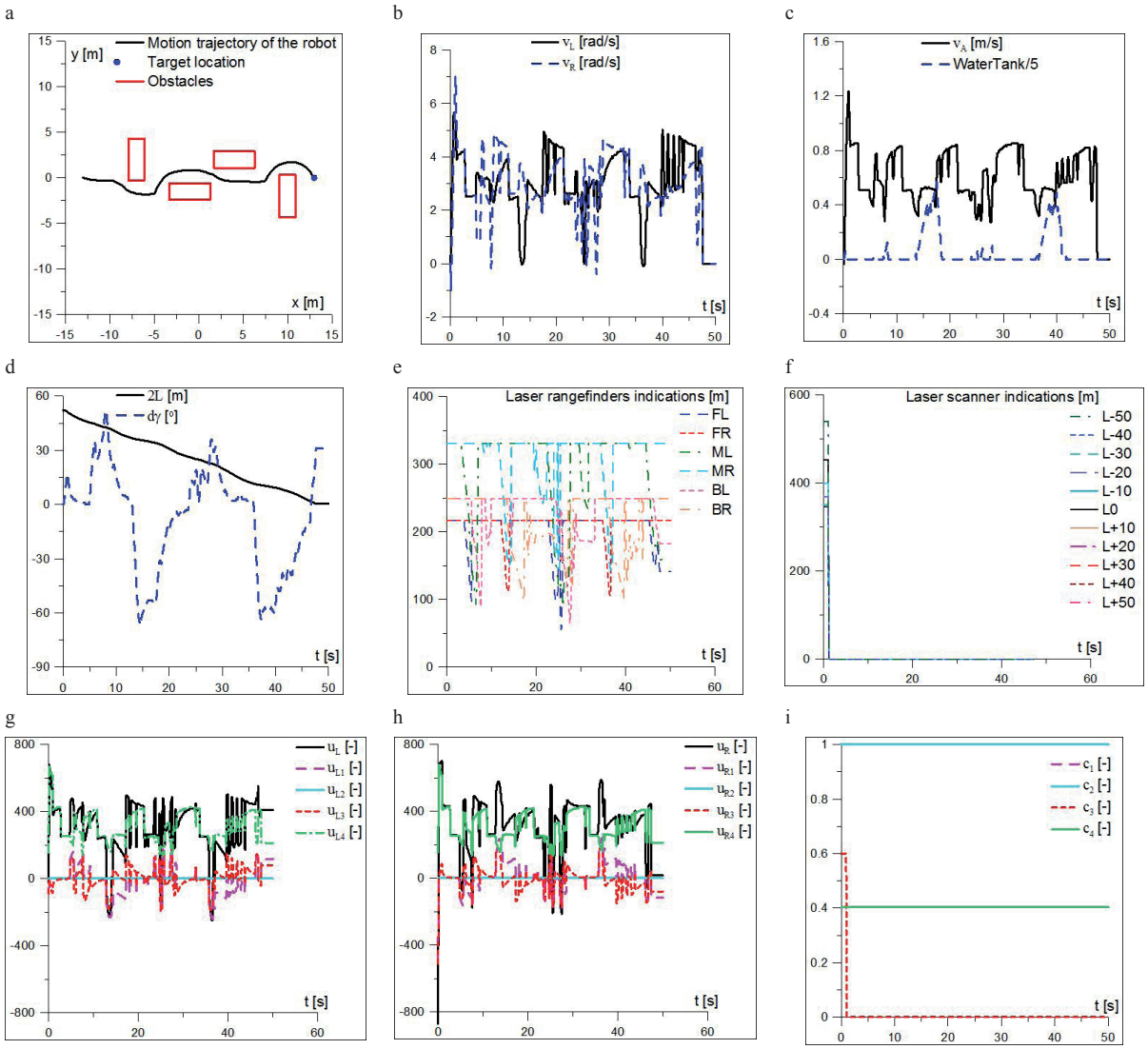

Fig. 6. *Simulation 3 results: a – robot's movement trajectory, with the location of the target and obstacles being marked, b – linear speeds for the robot's left and right wheels, c – robot's movement speed and changes in the WaterTank parameter, d – robot's distance and angle to the target, e – ladar readings, f – scanner readings for select 11 beams, g-i – control values and weighting for specific routines.*

6. Summary and conclusions for further research

The results of research into the vulnerability of mobile robot semiautonomy algorithms to environmental sensor malfunctions have been described in this paper. The developed semiautonomy algorithm included the possibility of a sensor malfunction, appropriately modifying the robot's subroutines. The results of simulation research carried out with the Matlab/Simulink packet for cases of full functionality and single sensor malfunctions, has been presented. A comparison of the results using 6 quality indicators has been carried out. The data gathered indicates that the developed semiautonomy algorithm exhibits high resistance to sensor malfunctions. What's more, in the case of malfunctions in some sensors better values in the quality indicators are achieved. This behaviour can be explained by the robot passing closer to some obstacles because of the malfunction and as a result shortening its route. However, this does not mean that the algorithm functions better under these conditions or that the malfunctioning sensor should be.

Further research will focus on simulating the malfunctions of a larger number of sensors as well as the implementation of a modified semiautonomy algorithm in the mobile robot and conducting experimental research related to analysing the vulnerability of this algorithm to environmental sensor malfunctions.

ACKNOWLEDGMENTS

The paper is a result of PROTEUS Project (POIG.01.01.02-00-014/08) and is co-financed by The European Union from The European Regional Development Fund under the Operation Programme Innovative Economy, 2007-2013.

AUTHORS

Jakub Bartoszek*, **Maciej Trojnacki**, **Piotr Bigaj** – Industrial Research Institute for Automation & Measurements PIAP, Al. Jerozolimskie 202, PL-02-486 Warsaw, Poland

*Corresponding author. E-mail: jbartoszek@piap.pl

References

[1] A. Wołoszczuk, M. Andrzejczak, P. Szynkarczyk, "Architecture of mobile robotics platform planned for intelligent robotic porter system – IRPS project", *Journal of Automation, Mobile Robotics & Intelligent Systems,* vol. 1, 2007, pp. 59-63.

[2] M. Trojnacki, P. Szynkarczyk, "Tendencje rozwoju mobilnych robotów lądowych (3). Autonomia robotów mobilnych – stan obecny i perspektywy rozwoju", *Pomiary Automatyka Robotyka,* no. 9, 2008, pp. 5-9. (in Polish)

[3] A. Ghorbani, S. Shiry and A. Nodehi, "Using Genetic Algorithm for a Mobile Robot Path Planning". In: *Proc. of International Conference on Future Computer and Communication*, 2009, Kuala Lumpur, Malaysia, pp. 164-166.

[4] M. Wang, J. N. K. Liu, "Fuzzy logic-based real-time robot navigation in unknown environment with dead ends", *Robotics and Autonomous Systems,* vol. 56, 2008, pp. 625-643.

[5] X.-J. Jing, "Behaviour dynamics based motion planning of mobile robots in uncertain dynamic environments", *Robotics and Autonomous Systems,* vol. 53, 2005, pp. 99-123.

[6] M. Soika, "A sensor failure detection framework for autonomous mobile robots". In: *Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS '97,* 1997, vol. 3, pp. 1735-1740.

[7] N. Ranganathan, M. I. Patel, R. Sathyamurthy, "An intelligent system for failure detection and control in an autonomous underwater vehicle", *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans,* vol. 31, 2001, pp. 762-767.

[8] K. Bouibed, A. Aitouche, M. Bayart, "Sensor fault detection by sliding mode observer applied to an autonomous vehicle". In: *Proc. of International Conference on Advances in Computational Tools for Engineering Applications, ACTEA '09,* 2009, pp. 621-626.

[9] I. J. Rudas, I. Ori, A. Toth, "Design methodology and environment for robot diagnosis". In: *Proceedings of IEEE International Symposium on Industrial Electronics, ISIE'93,* Budapest, 1993, pp. 367-372.

[10] K. Kroschel, A. Wernz, "Sensor Fault Detection and Localisation Using Decorrelation Methods", *A Sesnsors and Actuators,* vol. 25-27, 1991, pp. 43-50.

[11] G. J. S. Ra, S. E. Dunn, "On-Line detection for AUV". In: *IEEE Symp. Autonomous Underwater Vehicle Technology*, 1994, pp. 383-392.

[12] T. J. Farrel, B. Appleby, "Using leraning techniques to accommodate unanticipated faults". In: *IEEE Trans. Control Syst. Technol.*, 1993, pp. 40-49.

[13] M. L. Visinsky, J. R. Cavallaro, J. D. Walker, "Expert System Framework for Fault Detection and Fault Tolerance in Robotics", *Computers & Electrical Engineering,* vol. 20, no. 5, 1994, pp. 421-435.

[14] A. Healey, "Toward an automatic health monitor for autonomous underwater vehicles using parameter identification". In: *Amer. Control Conf.*, 1993, pp. 585-589.

[15] P. Bigaj, M. Trojnacki, J. Bartoszek, "Robot IBIS – realizacja ruchu w trybie teleoperacji i semiautonomii", *Prace naukowe Politechniki Warszawskiej,* series*: Elektronika,* Zeszyt 175, vol. 1, 2010, pp. 135-148. (in Polish)

[16] *IBIS Robot designed by PIAP – an offer* (24/11/2010). Available at: http://www.antyterroryzm.com/robot_bojowy.php (in Polish)

[17] http://www.youtube.com/user/osmpiap

[18] J. Borenstein, Y. Koren, "High-speed obstacle avoidance for mobile robots". In: *Proceedings., IEEE International Symposium on Intelligent Control*, 1998, pp. 382-384.

[19] I. Ulrich, J. Borenstein, „VFH+: reliable obstacle avoidance for fast mobile robots," in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, 1572-1577, 1998.