Implementation of Orienteering Methods for Advanced Autonomous Robot

Jaromir Konecny, and Michal Prauzek

***Abstract*—The autonomous robots are very popular research field. The scientist make effort to do autonomous robot, which will able to translate from start position to goal and there carry out mission. The missions are diverse. It can be simple mission such make a beep, or more complicated mission such detect toxic gas or bomb disposal. This article proposes robot prototype, which is intended for roboorienteering competition, where enthusiasts try to reach several goals in as short time as possible. This article presents robot control system, implementation and hardware resources of robot prototype. The navigation strategy is also mentioned.**

***Index Terms*—Autonomous robot, navigation, map design, AutoCAD extension, object oriented programing.**

I. Introduction

THIS research is follow-up to previous research in the roboorienteering field. Autonomous robot for indoor use shows the publication [1]. Another research [2] deals with exploratory vehicle for outdoor use. This research deals with outdoor autonomous device such it was published e.g. in [10] [11]. This article proposes the concept of autonomous robot, which is navigated by maps. The robot primary uses GPS localization system to establish the position in the map. It uses the maps, which are drawn in AutoCAD software. The map includes roads, walls, forests and another useful informations. The navigation counts with complex terrain, where it is much complicated recognize all obstacles and find the optimal way to reach goal. The route planning uses Dijkstra algorithm, which was published in [8] and [9]. The navigation concept especially proposes solution for orientation in areas, where direct azimuth does not lead for the goal. The complex control application is written in C# language and it runs Windows XP Embedded operating system and it takes advantages of object orienting programing.

II. Robot hardware resources

This section describes hardware construction of the robot. Robot chassis is constructed from aluminum profiles. These profiles are fitted with wheels, motors and electronic subsystems. Robot chassis is designed as a six-wheeled construction. The left front wheel is complementarily cross associated with the left rear wheel and the front right wheel is complementary cross connected to the right rear. Left center wheel and the right center wheel are controlled separately. The change of the robot direction is possible by different speeds or direction of left and right wheels rotation. Each driven wheel has one engine Pololu1106 [4] (in czech). The motors are controlled by the driver Sabertooth 2X5. The heart of the control subsystem is Kontron pITX-SP 1.6 GHz [5]. It is computer that runs Windows XP Embedded operating system. This device is connected to any of sensor subsystems. These include ultrasonic sensor SRF08, optical rangefinders GP2Y0A02. In addition, the robot has a 2D laser sensor HOKYUO URG04-LX [6]. The robot direction can be detected by the electronic compass CMPS10. Robot also includes front video camera Logitech. Absolute position is determined by GPS module. Manual control is enabled by connecting a wireless gamepad. Two powerful Li-pol batteries are used for power supply. Both batteries used voltage 14.4 V and capacity 4 Ah and 5 Ah.

Manuscript received July 2nd, 2013; revised August 1st, 2013. This work was supported by project SP2013/168, Methods of Acquisition and Transmission of Data in Distributed Systems of Student Grant System, VSB-TU Ostrava.

All authors are with the Department of Cybernetics and Biomedical Engineering, VSB-Technical University of Ostrava, 17. listopadu 2172/15, 70833 Ostrava, Czech Republic, Europe, e-mail: jaromir.konecny@vsb.cz.

III. Map design for the robot

This chapter describes the procedure for drawing maps for robot. Robot is able to work with maps, which must be provided by operator. Map data are in XML format and they contains information about the environment in which the robot will move. The maps are also recorded information about possible trajectories which the robot can safely move. The vector graphics application is used to create map surface, which allows the conversion of the drawing to XML file is suitable for the robot. As software for the maps preparation is used AutoCAD 2012. AutoCAD 2012 is able to cooperate with specialized GIS Software to professional maps creating. AutoCAD 2012 is also essentially an opened platform that allows custom extensions that extend the functionality of AutoCAD 2012. These extensions can be created either in the Visual LISP programming language, use the .NET libraries, which are then loaded by command `netload`. Connection with .NET provides full program control. The extension was written for making maps, which is capable directly generate map background from the drawing opened in AutoCAD 2012. Drawing must meet several conditions: Entity placed in the drawing must be in UTM coordinates. This is only a partial coordinates `x` and `y`. Zone and hemisphere are entered in the Settings dialog.

Drawing is subdivided into different layers. The suitable map for the robot has several layers. The most important are the layers which constitute the safety trajectory lines network. Safety trajectory segment provides the way the robot can move from the start point to the end point. The effort corresponds to the length of the line multiplied by a weight factor of the layer.

Trajectories are formed to three layers. The road, sidewalk, footpath. Obviously, the effort of movement along the road is less than the effort of movement along the footpath. The weight coefficients were selected: Road `k = 1`, sidewalk `k = 2`, footpath `k = 3`. Map data does not reflect a priori

ISBN: 978-988-19252-3-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

terrain bumps, and therefore it is necessary to enter this manually during the map creation. Unsurprisingly, the road down the hill for the robot will be less demanding than the way up the hill on the same segment. In order to take into account the ascent and descent, it is possible to add a special dimensions to the map. This dimensions modifies the effort form start point to end point.

Other layers in the map symbolize walls, forest, concrete pillars, metal structures, etc. It is important to distinguish from other metal structures, because of the proximity of metal structures can affect the accuracy of the electronic compass. The layer content is set in a special dialog that is part of the extension of AutoCAD 2012. The data from those dialog are stored into the dictionary, which is part of the drawing. There is no need to carry another drawing file containing those information. In addition, dictionary is hidden to a normal user and it can not be accidentally damage by the user. After setting those additional information it is possible to process the map generation.

IV. Control application

This chapter describes an application that allows complex robot control and full control over the hardware components of the robot. The application is written in C# language under .NET framework 2.0. Older version framework .NET 2.0 was chosen for compatibility with Windows Embedded operating system. Advantage of this solution is that the development of application, including debugging and stepping can be performed on a standard PC running Windows and then no changes are required for the Windows Embedded operating system device. The only problem may be the speed of the program. When deploying the application, it is necessary to think about the fact that the target device does not have so much hardware resources as a regular personal computer or notebook.

A. Application structure

The application itself is divided into several parts. Furthermore, the application functionality is divided into several threads, which are mutually synchronized. The main thread is a application thread and graphical user environment. The application starts only in system icons area, in order to the final deployment forms and graphic components do not consume unnecessary hardware resources. If it is desired, control the application forms can be shown any time by clicking onto mentioned icon.

The second thread is the `guardian`. It is the thread that handles the connection of all critical hardware components. If the guardian determines that e.g. laser sensor does not communicate, it tries to reboot it and reconnect it. Class guardian also allows you to get comprehensive information about the state of hardware.

Other threads are carrying for the individual hardware. Each complex sensor has own thread serving function. In particular, I2C, laser sensor, GPS, camera, gamepad, motor unit are controlled by separate threads. There a chance of some sensor failure during the movement of the robot. In this case failures only one appropriate thread. Another functionality of the robot is not compromised. In addition `guardian` thread restart it and the sensor will likely run properly again.

Executive thread is navigation control thread. See fig. 1. This thread takes care of collecting data from other treads, and navigation. This thread cyclically performs the specified action. At first the robot state is checked. The state of the sensors and main hardware components necessary for independent movement are also checked. Furthermore, it is verified whether the robot is in the manual control mode. Note that in the absence of main hardware components, it is also not possible to control the robot manually. This is determined for safety reasons. The control system controls obstacles and it not permits a collision with an obstacle during the manual mode control. Block `navigation`, which is a powerful navigation block, will be described in more detail below. The following block is obstacle detection, which may reduce the speed of the robot, so the final value of speed are written to the output as the last step.

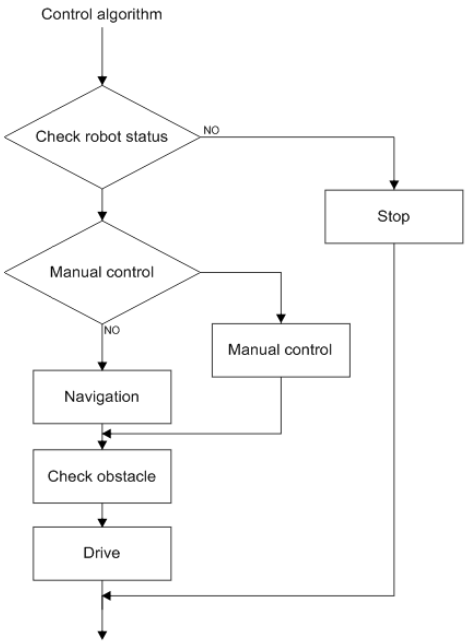

Fig. 1. Robot control flowchart

B. User interface

An important and indispensable feature during developing time is the user interface. C# allows to create a very comfortable user interface which is one of its advantage. User interface of control application implements full visualization and robot control. The application clearly shows the status of connected sensors, current sensor values. Furthermore it shows the robot surroundings measured by laser sensor and also shows current camera frame. The application also includes a powerful component that is able to represent and display the map background. See fig. 2. In this picture you can see the map of the VSB - Technical University area.

The maps user control can show the map, furthermore it allows basic map operations. Certainly there is a shift, zoom using the mouse and display and hide individual map layers. Map can get GPS or UTM coordinates at a certain position, it is able to perform basic measurements and it can find a

ISBN: 978-988-19252-3-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Fig. 2. Map representation in control application

suitable route from point A to point B. At the same time the robot is able to enter the target position. The user interface is shown fig. 3. The figure shows one of the application tab. On the top is the status panel with status icons. This panel is visible all the time. When any icon appears, it means the appropriate system does not work properly. On the rest of the screen is tab area. There are situated any other required controls. The first tab is the basic parameter tab which includes basic sensor information. The graphic representation of URG sensor, optical and ultrasonic sensors are shown on SRF and URG sensor area. The connection status area shows detailed information about connected subsystems. The engine status area shows load of the each motor. The control application includes several another tabs. Camera tabs provides full camera control, process values shows the values of the sensors, map tab provides map control. The next tab is features tab. It allows action stack and action queue control, goals control and manual action settings. The tab parameters allows set any constant and finally the log tab shows log messages.

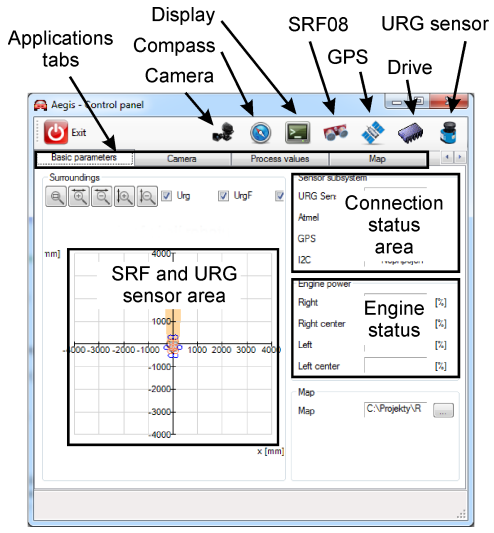

Fig. 3. Control application user interface

C. Object oriented approach to implementing navigation

To navigate the robot the navigation block is used, which has been already mentioned above. This block cyclically performs actions, that are used to control the robot movement. Object-oriented approach is used for implementation. The action is generally implemented as an interface. This interface contains generally defined methods and properties that must implement each action. The most important of them are the method `DoAction()` and `Terminated` property. Method `DoAction()` is called cyclically, and it controls the robot. If the action is done `Terminated` property is set to `True`, and the navigation algorithm knows which action has to be performed. The advantage of this approach is a generalization of the action and to implement new action is not necessary to modify any executive navigation function.

V. Robot Navigation

Now the navigation block will be described and strategy robot navigation will be mentioned. The goal navigation is similar to car navigation. It is calculated list of points to be achieved to reach the goal. The following paragraphs will describe the algorithm scheduling checkpoints and perform a sequence of actions that will lead to the goal.

A. Route planing

The route planning will be briefly discussed. The robot moves in the known environment, so it has a map. There are recorded information about possible trajectories of movement in this map. The aim is to design a trajectory from point A to point B. For finding the optimal trajectory Dijkstra algorithm algorithm is used. This algorithm finds the best route according to the criteria. In this case, the criterion of segment weight. Weight is the product of the length of the segment and its coefficient of performance. See equation (1). In a sense, we can say that it is the shortest route. The Dijkstra algorithm is presented in a well arranged way in reference [9].

$$w = k \cdot \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{1}$$

The base unit of the map is the vertex. Vertex is general object, which should be comparable. In this case the vertex is Cartesian point. The another base object is connecting rod. This object involves start point, end point, weight start to end and end to start respectively. See equation (2). s is start point, e is end point, w_1 is start to end weight and w_2 is end to start weight. The input parameters for the Dijkstra algorithm are start point, end point, and list of connecting rods. The list of connecting rods is extracted from the map. The output of Dijkstra algorithm is list of vertexes, which heads to goal.

$$r = \{[s_x, s_y], [e_x, e_y], w_1, w_2\} \tag{2}$$

Obtained list of vertexes corresponds with actions. Each vertex from list is one partial goal, which should be reached. Thats is path-planning strategy.

ISBN: 978-988-19252-3-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

B. Actions execution

The preceding paragraph described the principle of obtaining the actions sequence that need to be done. Now, the principles will described how actions are performed. Scheduled actions are stored in the queue. See fig. 4. There are now stored actions `A4` - `A7` in the queue. Most recent actions will therefore perform last. If new action `A8` is inserted, it is put in a queue behind action `A7`. The queue is large enough and it does not overflow. `A8` action will thus be added at the end of the queue. On the other side of the queue are actions fetched and executed. Assume that the stack of events is now empty and there is no need for replace. In this case, the actions are sequentially fetched from the queue and executed.

Suppose that there is a requirement that is necessary to execute the action `A9`, which was not planned and should be performed first. Actions `A3` in this case is stored to the stack and action `A9` replaces the current event. Once complete, `A9`, the algorithm first looks into the stack. If not empty, instead of the queue it pick up the action from the stack. Thus it is possible to heap unplanned actions to the stack. Example of this is wrong robot direction. Suppose the queue planned route. In the simplest case, the scheduled actions are a sequence of checkpoints that must be achieved. Achieving one point will be done so that the robot will move along the azimuth direction of the checkpoint. However if desired azimuth significantly different, it is necessary rotate robot first. This rotation is different action so it is necessary to replace the operation and current action push into stack. The new action must be execute immediately, therefore it can not be push at the end of the queue.

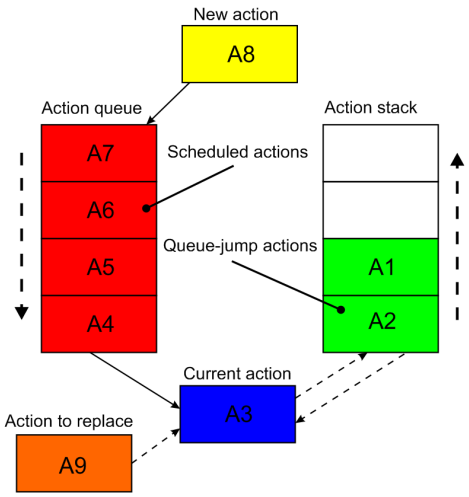

Fig. 4. Executing action scheme

C. Avoiding obstacles

This section will briefly explain the principle by which the robot avoid obstacles. See fig. 5. The figure shows the known and unknown obstacle. The planned route (dashed line) heads towards goal in an expected ground without obstacles. However, the sensory subsystem detects an obstacle that should be avoided. Sensory subsystem evaluate whether it is better to avoid an obstacle on the left or right and suggests an alternative route via intermediate point `C`. The same way as described in previous chapter, the overcoming obstacles action is inserted instead of current action. So first robot reaches point `C`, and then point `B`. When it reaches the point C or detection of another obstacles can be re-generate the optimal route to reach the destination. This ensures that the robot is not trying to reach points that are irrelevant. This approach addresses the situation where the obstacle is located directly on navigating point. The obstacle avoidance takes several steps. First step is obstacle detection. The obstacle detection checks the region, which define the controlled area in immediate proximity of robot. See equation (3).

$$R = \{[r_{1x}, r_{1y}], [r_{2x}, r_{2y}], ..., [r_{6x}, r_{6y}]\} \quad (3)$$

The laser scan region is defined in Cartesian coordinates by equation (4) and in polar coordinates (5).

$$U = \{[u_{1x}, u_{1y}], [u_{2x}, u_{2y}], ..., [u_{nx}, u_{ny}]\} \quad (4)$$

$$U = \{[u_{1\alpha}, u_{1ABS}], [u_{2\alpha}, u_{2ABS}], ..., [u_{n\alpha}, u_{nABS}]]\} \quad (5)$$

Obstacle detection step checks if one or more points from region U is inside polygon R. If it is true, then the obstacle is detected and escape point is calculated. The demanded angle for obstacle avoidance shows equation (6). W is robot width and l is obstacle distance. Now the escape angle is calculated. The laser scan region (5) is checked, whether demanded angle (6) in certain direction is obstacle free and that angle is escape angle. The navigation azimuth is adjusted and the robot runs new azimuth during certain time and likely avoid obstacle.

$$\alpha = \arccos\left(1 - \frac{W^2}{2l^2}\right) \quad (6)$$

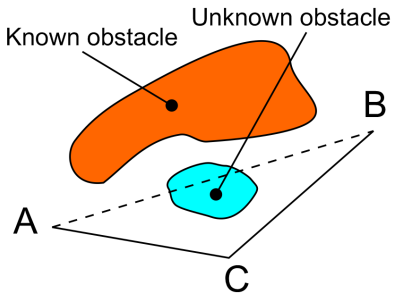

Fig. 5. Avoiding obstacles

VI. Results and Testing

The robot prototype was built to demonstrate navigation functionality. See fig. 6. The GPS antenna and electronic compass is situated on the top. On the front are fitted front optical and ultrasonic sensors and also laser sensor URG. Camera is also situated on the front, between the front SRF08 sensors. Robot was tested on the university area. Fig. 7 shows robot testing. It was inserted on the random start position and the goal was also set. The green point is start position, the red point is the goal. The red arrow is the robot position on the map during the movement. The robot automatically calculated the trajectory and started navigate to

ISBN: 978-988-19252-3-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

goal immediately. Figure shows, that the robot is navigated to each segment on the road towards goal. The violet curve represents calculated trajectory.

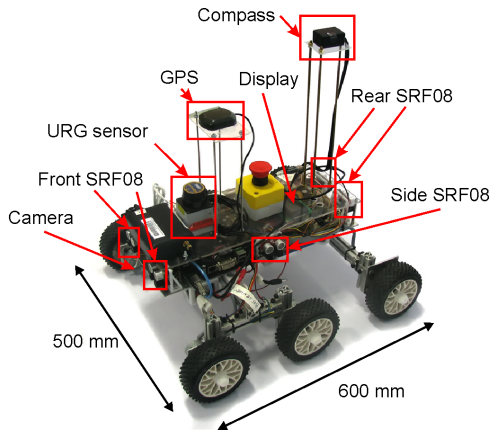

Fig. 6. Robot Aegis

During the tests several issues appears. The robot was built for roboorienteering competition primary [3]. The area of the university has several differences instead of competition area. One of the issue is parked cars. Parked cars are higher than robot sensor subsystem and therefore robot does not detect it. Second issue are bollards. Bollard is instead of car to low to detect. When the robot meet low bollard, it tries to climb it. Not every time it is possible. In future work we are expecting sensor subsystem modification. Likely we adjust the ultrasonic and optic sensors to trace cars and bollards. Despite obstacle issues the robot is able to move automatically. When GPS position is available with sufficient precision, the robot goes on the roads by the map and reach the goals. When the robot has position with deviation, it still expect that it is on the road. In actual fact robot is next to road and likely there are mentioned issues like parked cars, bollard, trees or high amount of another obstacles. The situation is better when the bollard is higher or further it is a wall. In this case the robot detect it and it adjust the azimuth and follows the wall.

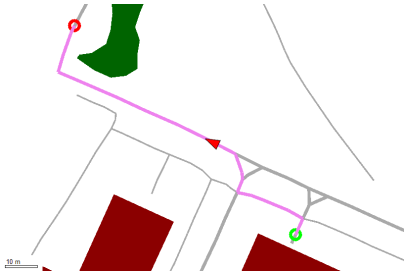

Fig. 7. Testing in university area

One of the advantage of map concept is complex route planning. See fig. 8. The start point and the destination point is relatively close. There is no suitable way to reach the goal direct. There is forbidden area. It is not important way is the area forbidden, there might be ravine, or hard terrain or something else. The basic impact is the robot choose another way to reach the goal, which is likely safer.

Fig. 8. Route calculation

VII. Conclusion

We were making the robot prototype with GPS navigation on the map and it was tested on the university area. For next year we will the research goes on and we will take part the roboorienteering competition. The main advantage of map concept is the ability of calculate complex trajectory. As it is shown in fig. 8, the route to goal is much different than straight direction. Another advantage is knowledge of obstacles, especially steel construction. Steel construction may affect the compass and GPS precision. In case the safety roads in the map are created with respect to steel construction, the robot do not move close. The main disadvantage is mentioned dependency on GPS signal. For the future work is expected, that we solve the issues with GPS signal. For the future work we will implement one of the known method such as fuzzy logic [7].

References

[1] J. Kotzian, J. Konecny, H. Prokop, T. Lippa, M. Kuruc, “Autonomous explorative mobile robot navigation and construction,” in *Roedunet International Conference (RoEduNet)*, 2010 9th , vol., no., pp.49,54, 24-26 June 2010

[2] J. Konecny, M. Kelnar and M. Prauzek, “Advanced Waste Rock Exploring by Mobile Robot.” *Applied Mechanics and Materials.* 2013, 313-314, s. 913-917. DOI: 10.4028/www.scientific.net/AMM.313-314.913.

[3] Roboorienteering. *Roboorienteering*, 2013 URL: http://www.vosrk.cz/roboorienteering/

[4] *Snail Instruments*, 2013 [cit. 2013-07-02]. URL: http://www.snailshop.cz/

[5] News from Kontron.com. *Kontron*, 2013 [cit. 2013-07-02]. URL: http://emea.kontron.com/products/boards+and+mezzanines/embedded +sbc/pitx+25+sbc/pitxsp.html

[6] *HOKUYO AUTOMATIC CO.,LTD.: Photoelectric switch, FA sensor, Parallel I/O, HMD/CMD, Laser distance sensor and for Robots*, 2013 [cit. 2013-07-02]. URL: http://www.hokuyo-aut.jp/

[7] Seraji, H.; Howard, A., ”Behavior-based robot navigation on challenging terrain: A fuzzy logic approach,” *Robotics and Automation, IEEE Transactions on* , vol.18, no.3, pp.308,321, Jun 2002 doi: 10.1109/TRA.2002.1019461

[8] Lixiao Guo; Qiang Yang; Wenjun Yan, ”Intelligent path planning for automated guided vehicles system based on topological map,” *Control, Systems & Industrial Informatics (ICCSII), 2012 IEEE Conference on* , vol., no., pp.69,74, 23-26 Sept. 2012 doi: 10.1109/CCSII.2012.6470476

[9] Huijuan Wang; Yuan Yu; Yuan, Q., ”Application of Dijkstra algorithm in robot path-planning,” *Mechanic Automation and Control Engineering (MACE), 2011 Second International Conference on* , vol., no., pp.1067,1069, 15-17 July 2011 doi: 10.1109/MACE.2011.5987118

[10] Kuhnert, K., ”Software architecture of the Autonomous Mobile Outdoor Robot AMOR,” *Intelligent Vehicles Symposium, 2008 IEEE* , vol., no., pp.889,894, 4-6 June 2008 doi: 10.1109/IVS.2008.4621234

[11] Ohno, K.; Tsubouchi, T.; Shigematsu, B.; Maeyama, S.; Yuta, S., ”Outdoor navigation of a mobile robot between buildings based on DGPS and odometry data fusion,” *Robotics and Automation, 2003. Proceedings. ICRA ’03. IEEE International Conference on* , vol.2, no., pp.1978,1984 vol.2, 14-19 Sept. 2003 doi: 10.1109/ROBOT.2003.1241884

ISBN: 978-988-19252-3-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)