THE DEVELOPMENT OF AUTONOMOUS RACING ROBOT BENDER

Stanislav Věchet*, Jiří Krejsa**, Vít Ondroušek*

Successful use of autonomous mobile robot in outdoor environment is nowadays still a challenge for developers. Navigation, localization and control issues are generally independent on the size of the robot, therefore smaller, low budget platforms can be successfully used for the development of appropriate algorithms. The paper describes the development of such a platform – autonomous racing wheel robot Bender, equipped only with odometry IRC sensors and camera, which won the Robotour 2006 competition. Both hardware issues and algorithms description including low and high level software layers are stated in the paper.

Key words : *autonomous robot, navigation, localization, image processing*

1. Introduction

Mobile robotics application recently received increasing attention thanks to rapidly developing advancements in this area. However, successful use of the robots in outdoor environment still represents a challenge for developers. Robot races bring one of the possibilities to compare and test robots ability to cope with real world tasks, while at the same time strictly defined rules of the race reduce the requirements to fully autonomous behavior of the robot to reasonable levels, thus keeping necessary cost of the hardware affordable.

DARPA Grand Challenge is a typical example of such event [4]. The essence of problems to be solved in order to reach the race goal (localization in the map, path planning, etc.) is in most cases independent on robots size. Small robots cost less, but algorithms developed for such robots can usually be transferred successfully to their 'bigger brothers'. This paper describes the development of such methods implemented on low-budget platform, aimed for the robotic competition Robotour 2006, held in Prague on October 14th, 2006. The goal of the competition was for the robot to autonomously drive on the Stromovka park path, approximately 800 m long, defined on park map which was provided by the competition organizers in advance.

The text consists of hardware description (chapter 2), and software description (chapter 3), including low level software (communication protocol, etc., chapter 3.1) and high level software (chapter 3.2) with description of image processing, navigation and localization.

* Ing. S. Věchet, Ph.D., Ing. V. Ondroušek, Brno University of Technology, Institute of Automation and Computer Science, Technická 2, 616 69 Brno
** Ing. J. Krejsa, Ph.D., Brno University of Technology, Institute of Solid Mechanics, Mechatronics and Biomechanics, Technická 2, 616 69 Brno

2. Hardware

Hardware of autonomous racing robot Bender consists of chassis, control electronics, vision sensor and control computer (notebook).

2.1. Chassis

The chassis is based on the original TAMYIA Overlander monocoque chassis made of polycarbonate with 4-wheel double wishbone suspension and rear 2WD drive. The chassis is 431 mm long and total weight with onboard computer is 3700 g. It was modified with control computer base, IRC sensors holders and vision sensor tower. Original battery pack was replaced with LithiumPolymer accumulator which enables approximately two hours of continuous ride, depending on the terrain. Nominal speed of the chassis is 7 km/h. The overall view of the robot can be seen on figure 1. Main hardware parameters are listed in table 1.

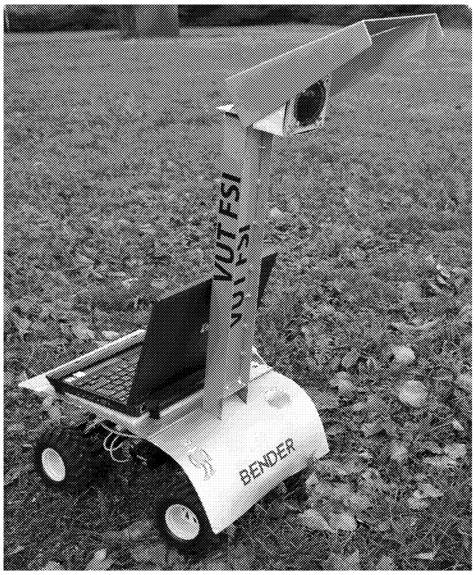


Fig.1: Overall view of the Bender robot

Device	Parameters
Power supply	Battery Li-Pol Racing Pack, 8.4 V, 4000 mAh (motor) Stabilized 5 V (onboard electronic)
IRC sensors	158 segments in 360° each Precision of measuring distance 2.6 mm
Steering servodrive	Pulse width control, operating voltage range 4.8–6.0 V Operating speed 0.21 sec/60°, Stall torque 3.3 kg.cm
Motor type	DC motor Mabuchi 540, operating voltage range 3.6–8.4 V, No-load rev. 22000 RPM at 7.2 V, current at max. efficiency 12 A
Speed controller	Speed controller JETI-JES 350, Maximum permanent current 35 A, Supply voltage 7.2–14.4V
Processor	ATMEGA8-16AU, 8 KB of programmable flash, 512 B EEPROM, 1 KB SRAM, Three flexible timers, Serial programmable USART – RS232, Communication speed 9.6 kHz (optional)

Tab.1: Hardware summary

2.2. Control electronics

The onboard electronics is based on microcontroller ATMEGA8. Basic structure of the electronics is shown on figure 2. The robot is equipped with odometry readings (two IRC sensors on front wheels) and camera connected with PC via USB port. Front wheels steering angle and drive motor revolutions are controlled by standard servomotor and speed controller. The hardware control unit forms the interface between sensors/actuators and the PC. For this hardware structure (designed in Laboratory of mechatronics) the possibility of unlimited extension with whatever other device is characteristic.

All uniboards (unified board) are of the same hardware design, the difference is in its software, which depends on sensor type connected to the board.

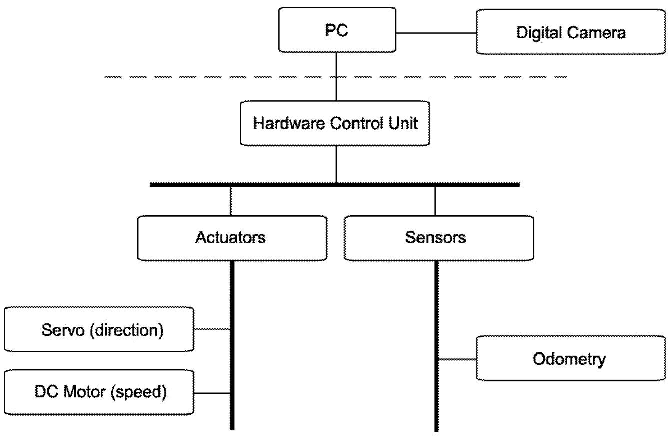


Fig.2: Electronic hardware structure

2.3. IRC sensor

The maximal speed of the robot and the accuracy in determination of robot position depend the number of segments on each wheel and the reading frequency used for pulses reading (generated by moving segments on IRC). Both front wheels have IRC sensor with 158 segments, therefore the precision is 2.3° and for given wheel diameter 130mm the position accuracy is 2.6 mm. The correct distance value obtained from IRC sensors is limited with the internal timer frequency used for reading the pulses from IRC. The relationship between timer frequency, numbers of pulses and robots velocity is $v = f/p$, where v is maximal velocity which can be measured [m/s], f is timer frequency, p is the number of pulses per meter. For timer with frequency 4.6 kHz and 387 pulses per meter (for wheel with diameter 130 mm and 158 segments) the maximal velocity which can be measured is 12 m/s. The device sends the traveled distance to the master in two bytes (see table 2), therefore the limit for maximal measured distance without reset can not exceed 65535 pulses (because the slave is unable to send higher number). The maximal distance is $s = n/p$, where s is maximal traveled distance without reset [m/s], n is data size, p is number of pulses per meter. For two bytes (data size 65535 pulses) and 387 pulses per meter the distance is 169.3 m.

2.4. Vision sensor

Bender is equipped with Logitech CCD camera with VGA chip of resolution 640×480 px, capable of 30 fps. In order to extend the view angle the vision system is equipped with dispersing lens with $DPT = -8$, positioned in 42 mm prior to the camera objective. The dispersing lens is covered by reflective layer, however, as the direct sun beams can degrade the image content the optics is also covered using protective cup. Vision system (camera + lens) is placed on the tower 810 mm above the surface. The camera is connected to the control computer via USB port.

2.5. Control computer

Control computer is standard subnotebook HP nc2400 equipped with Intel Core Solo processor U1400 1.2 GHz, 1 GB of SDRAM with Windows XP Pro. Computer is of compact size (25.0×282.0×213.0 mm), weights only 1.29 kg. The capacity of the 6-cell battery

(5100 mAh, 58 Wh) enables about 6 hours of processing. The computer is equipped with 2 USB ports, the first one connects the camera, the second one connects the onboard electronics through USB2RS232 adapter. The notebook runs the control software, described in detail in next chapter.

3. Software

Software consists of two layers – low level layer which controls the steering, main drive controller and IRC sensors, and high level performing camera image processing, localization in the map and navigation.

3.1. Low level software

The low level software is a layer between the hardware and the high level software. This layer is implemented as a software driver (DLL – dynamic link library). The driver enables the transfer of high level commands to the hardware.

3.1.1. Communication protocol

The communication protocol between PC and hardware is bidirectional. Standard CRC (Cyclic Redundance Check) is used for capturing errors in communication. The communication is based on master-slave protocol (PC as the master and robot's hardware parts as slaves). Each hardware part is connected as one slave, so there are four slaves: left and right IRC, steering controller and speed controller. Master sends 13-byte data packet which consists of start byte, slave ID, data and CRC. The slave answers immediately with the status message. If the status message contains no errors, it is followed with the data packet consisting of slave ID, requested data and CRC. The data packet is sent by IRC device only, as other devices (steering and speed controller) do not have feedback. The data packet sent from PC is shown on figure 3A.

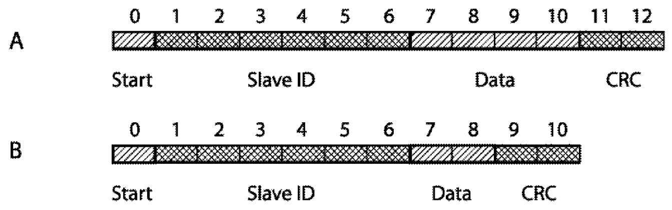


Fig.3: Master (A) and slave (B) data packet

Device	Data byte 7	Data byte 8	Data byte 9	Data byte 10
IRC	Action : 1-reset, 0-read distance	unused	unused	unused
Steering controller	Steering command value is in range [660–1460], (steering angle [25°, −25°])		0	unused
	Low byte	High byte		
Speed controller	0	0	Speed 1–255	unused

Tab.2: Master data message

Device	Message	Data byte 7	Data byte 8
All devices: status messages	CRC error	255	255
	Data packet without errors	254	255
Device feedback message: IRC only	Measured distance 0–65533 pulses	Data low	Data high

Tab.3: Slave data message

The start byte has value of 255, slave ID is a unique combination of six bytes used as a device address. The data bytes depend on the type of the connected device, see table 2. CRC bytes are used for errors detection and correction in the communication. The structure of the slave data packet is similar (see figure 3B). The only difference is in the data part. The data sent to the master contain two bytes only, consisting of requested data or error message. Table 3 shows the data part for the slave devices.

3.1.2. Communication interface

Communication between PC and hardware units is realized by the standard DLL libraries which use standard Win API interface for communication by RS232 protocol. We prepared one library which exports necessary functions for robot control. It's functions are used in high level software to transfer the necessary commands to the low level software.

3.2. High level software – control architecture

The highest level of the task to be solved (drive on a priory given path from the initial to the goal position) repeats two basic routines
- drive on road until the end of segment is reached,
- drive on the cross road until start of the following segment is reached.

The segment is defined as the road from the end of crossroad to the beginning of next crossroad on the path to be driven in given task.

3.2.1. Driving on segment

Driving on road segment consists of three basic substeps.
- determination of steering angle for current position and step move for given distance,
- localization of the robot in the map,
- determination of the end of the segment (cross road identification).

3.2.2. Keeping robot on the road

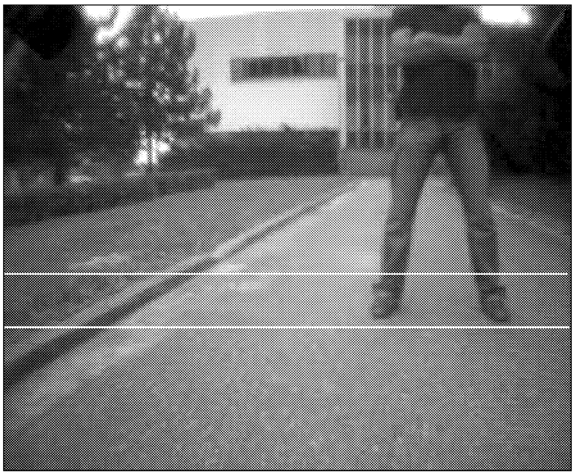
Driving on the road requires robot to know where the road is. Such information is then used for determining the steering angle. Camera image analysis was used for the estimate of road direction relatively to robot front. A number of algorithms were tested, the one described below is simple but showed the robustness required in real world task. Image processing routines return relative distances of road curbs. Steering angle is then determined as:

$$SA = SG\,[50 - CL - (CR - CL)/2] \tag{1}$$

where SA is steering angle, SG is steering gain (set to 0.015), CL and CR are camera left and right curb relative distances. To avoid the loss of the road orientation the steering angle is limited during road driving to ± 7 degrees.

3.2.3. Image processing

Camera image processing routines determine relative distances of road curbs. Such curb can be both the real road curb or the edge of obstacle on the road. A number of image processing algorithms were implemented and tested, the best results were reached using simple method described below.



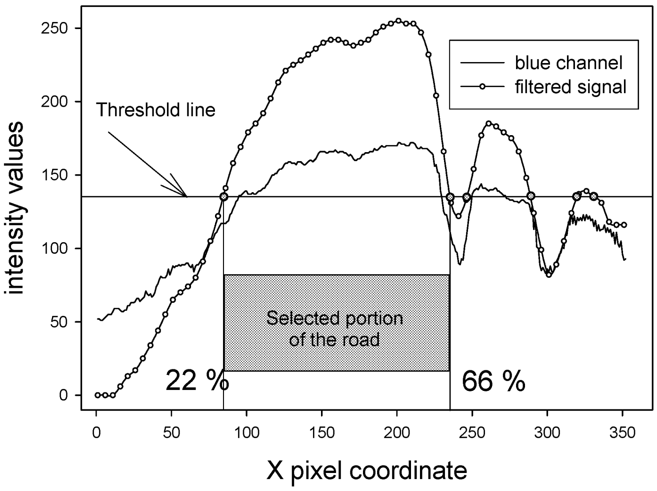


Fig.4: Sample image to be processed (top), image processing algorithm (bottom)

a) **Select image portion:** Camera image covers space in range of 1.2 m to infinity. Portion of the image corresponding to certain range only is selected for further processing. The range differs in relation to the task to be solved (2.2–3.2 m for steering angle determination for simple road drive, 2.2–5 m for cross road identification).

b) **Process blue channel signal:** Blue channel showed up the best contrast between road and its surroundings, therefore only blue channel signal from selected portion of the image is used for further processing. For each column of selected data the blue channel pixel values are summed up and resulting values are linearly transformed in the range of 0–255.

c) **Filtering:** Resulting signal is filtered using floating average filter. The length of the filter can vary in the range of 5–10 for particular camera image resolution.

d) **Thresholding:** Filtered signal is further processed using thresholding, [1]. Threshold value is set to half of the range (128).

e) **Selecting the part corresponding to obstacle free road:** The widest portion of thresholded signal is considered the obstacle free road. This solves the case of obstacle on the road when two (or more) possible ways are available.

Image processing algorithm is clearly shown on example in figure 4.

3.2.4. Localization in the map

While driving from the initial to the goal position the map of the path to be passed is available to the robot. GPS coordinates of the path are incorporated in the map. Consequently, latitude and longitude coordinates are converted into the Cartesian coordinate system the origin of which is in the initial point of the path, the Y-axis is concurrent with north direction [3]. For localization purposes the software implementation of odometry and of the Ackerman steering is used. The Ackerman steering principle [6] defines the geometry that is applied to the vehicle to enable the correct turning angle of the steering wheels to be generated when negotiating a curve. These two tasks had to be solved:

a) **Forward task:** The initial position $[P0]$ and the initial orientation, ω_{r0}, of the robot are known as well as the distances moved by the front left and front right wheel (information from IRC; $IRCl$, $IRCr$) and steering angle φ_{SA}. The objective is to calculate the target position $[P1]$ of the robot and its orientation, ω_{r1}, see figure 5.

– change of the orientation of the robot $\Delta\omega_{\mathrm{r}}$:

$$\Delta\omega_{\mathrm{r}} = \frac{(IRCl + IRCl)\,\tan\varphi_{\mathrm{SA}}}{2\,b}\ , \tag{2}$$

– change of the robot's position in the X axis direction Δx:

$$\Delta x = \frac{b}{\tan\varphi_{\mathrm{SA}}}\,(1-\cos\Delta\omega_{\mathrm{r}})\ , \tag{3}$$

– change of the robot's position in the Y axis direction Δy:

$$\Delta y = \frac{b}{\tan\varphi_{\mathrm{SA}}}\,\sin\Delta\omega_{\mathrm{r}}\ . \tag{4}$$

b) **Backward task:** The initial and target point $[P0]$, $[P1]$ of the robot position are known, as well as its orientation in the Cartesian coordinate system, ω_{r0}. The objective is to calculate steering angle φ_{SA} of the fore-axle wheels, so that the robot will get from the initial point into the target point. We also calculate the distance L_1 moved by the robot, see figure. 5.

– turning radius R:

$$R = \frac{\Delta x^2 + \Delta y^2}{2\Delta x}\ , \qquad \text{where} \quad \Delta x = |P_{0\mathrm{x}} - P_{1\mathrm{x}}|\ , \quad \Delta y = |P_{0\mathrm{y}} - P_{1\mathrm{y}}|\ , \tag{5}$$

– change of the orientation of the robot $\Delta\omega_{\mathrm{r}} = \omega_{\mathrm{r0}} - \omega_{\mathrm{r1}}$:

$$\Delta\omega_{\mathrm{r}} = \arctan\frac{2\,\Delta x\,\Delta y}{\Delta y^2 - \Delta x^2}\ , \tag{6}$$

– moved distance L_1 :

$$L_1 = \Delta\omega_\mathrm{r}\, R\ . \tag{7}$$

It is necessary to calculate the new position of the robot on the map (in the way described above) repeatedly each time the robot moves the selected number of meters, L. It is important to select an optimum value of L. If the value of L is set too high the robot is not able to react quickly enough to an obstacle or it can even drive off the road. On the contrary, selecting a too small value of L can lead not only to higher computational requirements, but also to a bigger disprepancy between the calculated and real positions of the robot on the map. This disprepancy is caused by the persistence of motion due to dynamic effects of the mass of the robot. The value of L has been experimentally set up to a half-meter.

After every 0.5 m segment moved by the robot its localization on the map is calculated. Further calculations of the robott́s position are based on the position calculated in the previous step. Every calculation has an error and thus errors accumulate and after certain distance traveled the calculated position on the map is completely different from reality. That is why after an experimentally defined segment an 'odometry reset' is carried out, which means that a point of the path is determined and taken as the actual robott́s position on the map.

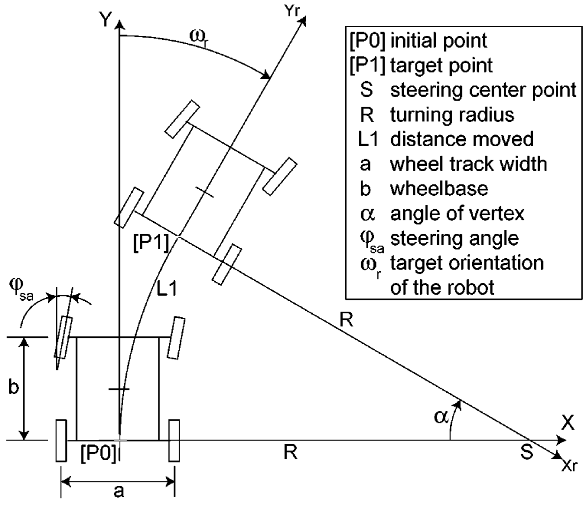


Fig.5: Calculating the position of the robot

The reset can be used if the following conditions are met :

a) the robot is at every moment of its movement on the road; this is ensured by image processing module.
b) there is sufficiently precise information about the moved distance of the robot from the last 'odometry reset'; information from IRC sensor.
c) the history of particular steering angles from the last 'odometry reset' is known as well as the history of changes of the road direction

New position of the robot in the map is calculated in the following way (see figure 6) :

a) Based on the known values of turning angles and road direction changes, the parameter ξ is determined.

$$\xi = \frac{\zeta\,(\sum|\varphi_\mathrm{rob}| - \sum|\varphi_\mathrm{map}|)}{L_\mathrm{s}} \tag{8}$$

where ζ is a coefficient experimentally determined to be equal to 0.3, φ_{rob} is robot's steering angle during single step move, φ_{map} is corresponding steering angle determined from the map and L_{s} is the traveled distance.

b) Moved distance of the robot on the map, L, is calculated:

$$L = (1 - \xi)\, IRC\text{distance} \tag{9}$$

where IRCdistance is the average value of distances moved by the front left and front right wheel.

c) New position of the robot in the map is determined using the full travel length L, estimated in previous step projected to the map path.

'Odometry reset' is carried out each 10 meters of the road and each time the robot is 4 meters prior to the cross of roads.

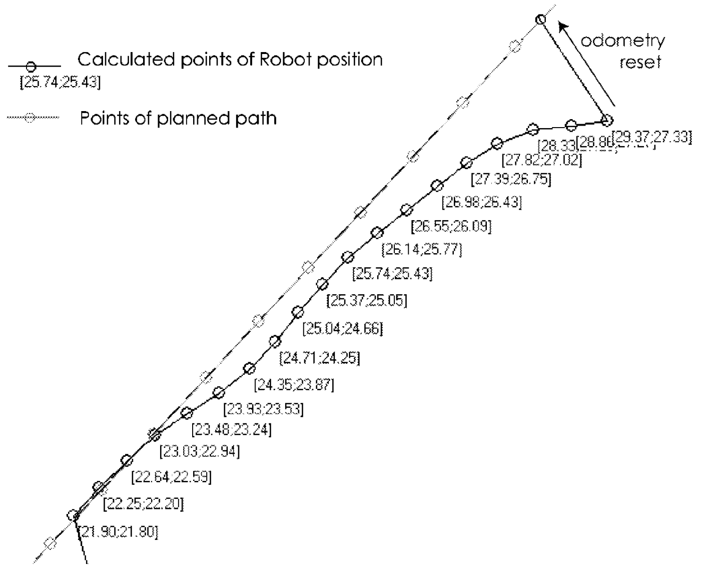


Fig.6: Reset robot's position each 10 meters of the path

3.2.5. Cross roads

When the end of the segment is reached, the position and orientation of the robot is estimated and robot turns on the crossroad. Turn angle is determined from robot current orientation and crossroad map information (difference in last segment end and next segment start orientation). Steering angle is determined for given step move, when steering angle exceeds robot capabilities the move length is prolonged according to Ackerman steering model. After the turn the robot moves towards the start of the next segment, checking its position using the image analysis with portion of the image selected to cover longer distance from the robot. This approach is only usable for 'reasonably' small cross roads – when cross road form open space the image analysis often fails.

3.2.6. Map module

Control application also contains the map module, which defines the map of the environment and path to be taken on given map. The map data can be imported from the file in RNDF (Road Network Definition File) format specified by DARPA. Map network is constituted from segments containing one or several lanes. The lane is represented by the list of points described by latitude and longitude in WGS84 system. Certain points in the lane can be marked as check points, used in the path definition. The path to be taken by the robot can be imported from the MDF file (Mission Definition file), which contains the sequence of check points. Path can be also input manually in the map module as the sequence of map segments.

3.3. Tests and competition

Stability of above mentioned algorithms was tested on several test fields, starting from a simple road, through short circular path to long roads with a number of crossroads in Lužánky park. We have found that robot in current state is a low cost platform ideal for further development of both hardware mainly in sensor area and software ů- image processing, high level algorithms, etc. Relatively small dimensions of the robot enable to use it in the indoor applications as well as the outdoor ones, however in outdoor the use of the robot is limited to relatively smooth surfaces.

Robotour 2006 competition was the final test of Bender's abilities. The task for the robots was to drive on the park pavements with the goal path definition given one day before the actual ride. The total length of the path was approximately 800 meters and contained 7 cross roads. Every robot had 5 attempts to reach the goal, with one attempt per hour. In the competition of 8 robots which met the homologation criteria Bender proved its flexibility when it was the only robot which successfully performed given task (reached the goal point) in its final 5th attempt. The total distance traveled during the competition was about 2.8 km.

4. Conclusion

Bender successful performance on the competition proved that the robot with limited sensor equipment (IRC odometry sensors, low resolution camera) is capable of autonomous motion in an outdoor environment with known map. We have found that simple algorithms bring higher robustness to the application in question compared to fine tuned complex algorithms which usually offer better performance, but are generally more sensitive to unexpected disturbances often presented in real world experiments.

However, there is a number of issues to be solved in order to enhance the robot performance. Among others we can mention better cross determination, better resistance against variable light conditions and improved way to overcome the open spaces. Apart from improving the current algorithms the additional sensor equipment might be necessary to solve mentioned issues.

Acknowledgement

Published results were acquired using the subsidization of the Ministry of Education, Youth and Sports of the Czech Republic, research plan MSM 0021630518 'Simulation modelling of mechatronic systems'.

References

[1] Heath D., Sarkar S., Sanocki T., Bowyer K.W.: Robust Visual Method for Assessing the Relative Performance of Edge-Detection Algorithms, Proceedings in IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1338–1359, December 1997

[2] Hu H., Gu G., Brady M.: Navigation and guidance of an intelligent mobile robot, Proceedings in Second Euromicro Workshop on Advanced Mobile Robots (EUROBOT '97), October 1997

[3] Schwarz T., Hoenle N., Grossmann M., Nicklas D.: A Library for Managing Spatial Context Using Arbitrary Coordinate Systems, Proceedings in Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, pp. 48, March 2004

[4] Seetharaman G., Lakhotia A., Blasch E.P.: Unmanned Vehicles Come of Age: The DARPA Grand Challenge, Proceedings in Computer, The flagship magazine of the IEEE Computer Society, December 2006

[5] Šolc F., Žalud L.: Robotika, Brno Faculty of Mechanical Engeneering, 2002

[6] Wu D., Zhang Q., Reid J.F.: Dynamic steering simulator of wheel type tractors, ASAE Paper 983115, ASAE, 1998

Received in editor's office: January 12, 2007
Approved for publishing: June 7, 2007