Sliding-Mode Control and Sonnar Based Bubble Rebound Obstacle Avoidance for a WMR

Adrian Filipescu, Bogdan Dumitrascu,
Adriana Filipescu, George Ciubucciu
Department of Automation and Electrical Engineering
"Dunarea de Jos" University of Galati, Romania
Adrian.Filipescu@ugal.ro

Eugenia Minca
Department of Automation, Computer Science and Electrical Engineering
"Valahia" University of Targoviste, Romania
eugenia.minca@gmail.com

Alina Voda
Grenoble Image Parole Signal Automatique (GIPSA-lab),
University Joseph Fourier Grenoble 1/CNRS,
UMR 5216, B.P. 46, F-38402 St Martin d'Heres, France
alina.voda@gipsa-lab.grenoble-inp.fr

***Abstract*— In this paper an algorithm for trajectory-tracking and obstacle avoidance for wheeled mobile robots (WMR) is presented. The algorithm creates a trajectory composed of a global trajectory generated off-line and local obstacle avoidance trajectories that are created when an obstacle is detected by the sonar sensors. Only one discrete-time sliding-mode controller is required to track the resulting trajectory and it does not require separate controllers for following the intended trajectory and avoiding the obstacle. The local avoidance trajectories are generated using Quintic equations to generate a path for the robot and assigning calculating the velocity, acceleration, angular velocity and angular acceleration needed by the discrete-time sliding-mode controller.**

***Keywords*— sliding-mode, obstacle avoidance, WMR.**

I. INTRODUCTION

WMRs have been used in a multitude of applications, especially for industrial applications. Some of these applications require the robots to track a set trajectory. Occasionally temporary obstacles, like a broken robot, or fallen debris, can appear and the robots will be unable to follow the desired trajectory leading to a full stop of the entire work. A solution to this problem is to include an obstacle avoidance module that will allow the robot to avoid the blocked part of the trajectory and to return to the desired trajectory. This solution will add a delay to the process but will prevent a full stop of the process. Sliding-mode control (SMC) is a robust approach for various applications and it can be used. The SMC methodology has been presented in [1]. Sliding-mode has been used in many control systems, such as: [2]-[16]. The discrete-time sliding-mode control was chosen to solve the trajectory-tracking problem because it promises great results. One of the simplest obstacle avoidance algorithm is "the bug", proposed by in [20]. This algorithm requires the robot to circle the obstacle before determining the best point to continue towards its goal. Other obstacle avoidance algorithms use virtual potential fields, like in [18] to determine the path the robot has to take in order to avoid the obstacle. The algorithms consider that the robot is subjected to forces that pull the robot towards the goal and rejecting forces that pull the robot away from the obstacles. The robot's trajectory results from the combination of these forces. A vector field histogram is used in [17] for obstacle avoidance, that uses a histogram to reduce the errors from the sensors. A bubble rebound algorithm is used in [19] to avoid obstacles. The algorithm uses a sensitivity bubble to determine if an obstacle is blocking the path. If an obstacle is detected the algorithm calculates the path that has the minimum density of obstacles and moves in that direction until the goal is visible or a new obstacle is detected. An algorithm that uses capable of tracking a desired trajectory and avoids obstacles that block the desired path by generating local trajectories and following them until the initial trajectory can be resumed is proposed in this paper. The global and local trajectories will be merged and form a new trajectory that can be tracked using only one discrete-time sliding-mode controller. The robot used to test the algorithm is the two driving wheels/two free wheels (2DW/2FW), PowerBot (Fig. 1). The WMR PowerBot is a high-payload differential-drive robot capable of moving at speeds up $1.6m/s$ and carrying up to 100 kg. The obstacles are detected using the sonar sensors of the robot and the sensitivity bubble from [19].

The rest of the paper is organized as follows: discrete-time, sliding-mode control based on kinematic model of the WMR is presented in Section II; bubble rebound obstacle avoidance algorithm for the 2DW/2FW WMR PowerBot is presented in Section III; simulation results of obstacle avoidance, trajectory tracking control are presented in Section IV; some final remarks can be found in Section V.

II. DISCRETE-TIME SLIDING-MODE CONTROL

Consider the model of a wheeled mobile robot, presented in Fig. 2. The model takes into account the two diametrically opposed drive wheels of radius R, the distance between the

Fig. 1. PoweBot avoing an obstacle

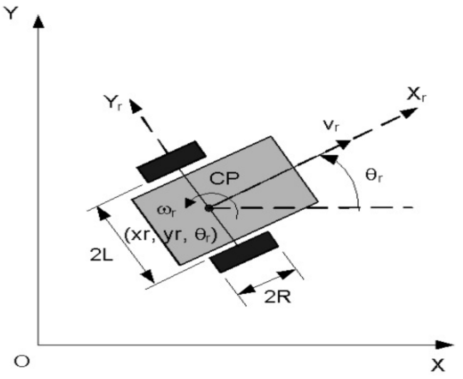


Fig. 2. Kinetic model of a differential drive robot

wheels ($2L$), the angular speeds of the drive wheels (ω_L, ω_R), the center point (CP) of the robot.

The localization of the robot is given by $P=(x_r, y_r, \theta_r)$, where x_r represents the position on the x axis, y_r the position on the y axis and θ_r the heading of the robot, v_r the linear velocity while ω_r represents the angular velocity of the robot. Considering a sample interval T$_s$ and a zero-order hold, the kinematic model in discrete-time is

$$\begin{cases} x_r[k+1]=x_r[k]+v_r[k]\cdot\cos\theta_r[k]\cdot T_s \\ y_r[k+1]=y_r[k]+v_r[k]\cdot\sin\theta_r[k]\cdot T_s \\ \theta_r[k+1]=\theta_r[k]+\omega_r[k]\cdot T_s \end{cases}. \quad (1)$$

The trajectory tracking problem is how to design a controller capable of tracking a desired trajectory. For this purpose a virtual robot, with the desired trajectory $q_d[k]=[x_d(k)\ \ y_d(k)\ \ \theta_d(k)]^T$, is considered and the following kinematic model of the virtual robot is obtained:

$$\begin{cases} x_d[k+1]=x_d[k]+v_d[k]\cdot\cos\theta_d[k]\cdot T_s \\ y_d[k+1]=y_d[k]+v_d[k]\cdot\sin\theta_d[k]\cdot T_s \\ \theta_d[k+1]=\theta_d[k]+\omega_d[k]\cdot T_s \end{cases}, \quad (2)$$

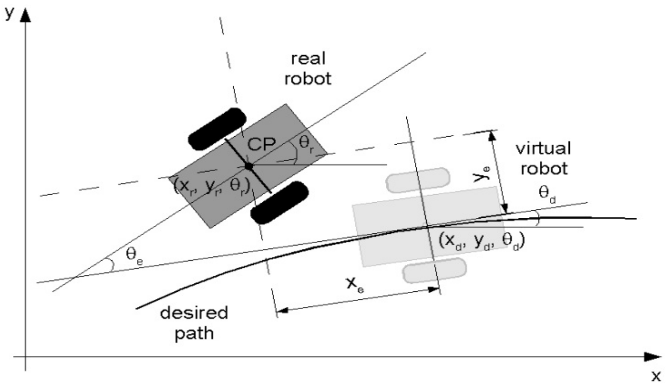


Fig. 3. Trajectory tracking errors

where $P_d=(x_d, y_d, \theta_d)$ represents the desired pose, v_d the desired linear velocity, ω_d the desired angular velocity. The trajectory-tracking errors are shown in Fig.3, and can be expressed as follow:

$$\begin{cases} x_e[k]=x_{rd}[k]\cdot\cos\theta_e[k]+y_{rd}[k]\cdot\sin\theta_e[k] \\ y_e[k]=-x_{rd}[k]\cdot\sin\theta_e[k]+y_{rd}[k]\cdot\cos\theta_e[k] \\ \theta_e[k]=\theta_r[k]-\theta_d[k] \end{cases} \quad (3)$$

$$x_{rd}[k]=x_r[k]-x_d[k] \quad (4)$$

$$y_{rd}[k]=y_r[k]-y_d[k] \quad (5)$$

The error dynamics for trajectory tracking are defined as

$$\begin{aligned} x_e[k+1]&=x_e[k]+(v_1-v_d[k]+y_e[k]\cdot\omega_d[k])\cdot T_s \\ y_e[k+1]&=y_e[k]+(v_2-x_e[k]\cdot\omega_d[k])\cdot T_s \\ \theta_e[k+1]&=\theta_e[k]+(\omega_r[k]-\omega_d[k])\cdot T_s \end{aligned}, \quad (6)$$

where:

$$v_1=v_r[k]\cdot\cos\theta_e[k], \quad (7)$$

$$v_2=v_r[k]\cdot\sin\theta_e[k]. \quad (8)$$

are the linear speed along x-axis and y-axis, respectively. Corresponding to [4], the discrete-time sliding mode occurs if the following attractiveness condition is satisfied:

$$s[k]\cdot(s[k+1]-s[k])<0. \quad (9)$$

with s sliding surface. The reaching law proposed in [15] is:

$$s[k+1]=(1-q\cdot T_s)\cdot s[k]-\varepsilon\cdot T_s\cdot\operatorname{sgn}(s[k]), \quad (10)$$

$$0<1-q\cdot T<1, \quad (11)$$

$$0<\varepsilon\cdot T_s<1, \quad (12)$$

where $T_s>0$ is the sampling period, $\varepsilon>0$ is the reaching velocity, $q>0$ is the converging exponential. The magnitude of the resulting chattering is limited within the quasi-sliding mode bandwidth:

$$2\Delta = 2 \cdot \frac{\varepsilon \cdot T_s}{1 - q \cdot T_s} \tag{13}$$

The sliding surfaces are defined as follow:

$$\begin{cases} s_1[k] = x_e[k+1] + k_1 \cdot x_e[k] \\ s_2[k] = y_e[k+1] + k_2 \cdot y_e[k] + k_0 \cdot \mathrm{sgn}[k] \cdot \theta_e[k] \end{cases}, \tag{14}$$

$$\mathrm{sgn}[k] = \mathrm{sgn}(y_e[k]), \tag{15}$$

where k_0, k_1, k_2 are positive constant parameters, x_e, y_e and θ_2 are trajectory tracking errors. From (10) and (14), the sliding surfaces can be expressed as follow:

$$\begin{aligned} s_1[k+1] = x_e[k+2] + k_1 \cdot x_e[k+1] = (1 - q \cdot T_s) \cdot s_1[k] - \\ \varepsilon \cdot T_s \cdot \mathrm{sgn}(s_1[k]) \end{aligned}, \tag{16}$$

$$\begin{aligned} s_2[k+1] = y_e[k+2] + k_2 \cdot y_e[k+1] + k_0 \cdot \mathrm{sgn}(y_e[k]) \cdot \theta_e[k] = \\ (1 - q \cdot T_s) \cdot s_1[k] - \varepsilon \cdot T_s \cdot \mathrm{sgn}(s_1[k]) \end{aligned}. \tag{17}$$

The discrete-time sliding-mode controller (the linear and angular speeds) is obtained from (10), (14), (16), (17):

$$\begin{aligned} v[k+1] = \frac{1}{\cos\theta_e[k] \cdot T_s} \cdot [-(1 - q_1 \cdot T_s) \cdot s_1[k] + \varepsilon_1 \cdot \\ \cdot T_s \cdot \mathrm{sgn}(s_1[k] - x_e[k+1] \cdot (1 + k_1) - (v_d[k+1] - \\ v_r[k] \cdot \theta_e[k+1] \cdot \sin\theta_e[k] - \omega_d[k+1] \cdot y_e[k] - \\ \omega_d[k] \cdot y_e[k+1]) \cdot T_s] \end{aligned}, \tag{18}$$

$$\begin{aligned} \omega[k] = \frac{1}{v_r[k] \cdot \cos\theta_e + k_0 \cdot \mathrm{sgn}\, y_e[k+1]} \cdot [-(1 - \\ q_2 \cdot T_s) \cdot s_2[k] + \varepsilon_2 \cdot T_s \cdot \mathrm{sgn}(s_2[k]) - \theta_e[k] - \\ y_e[k+1] \cdot (k_2 + 1) - (v_r[k+1] \cdot \sin\theta_e[k] + \\ \omega_d[k+1] \cdot x_e[k] - \omega_d[k] \cdot x_e[k+1]) \cdot T_s)] + w_d[k] \end{aligned} \tag{19}$$

III. Sonnar Based Obstacle Avoidance Algorithm

The robot is intended to track a given trajectory as long as no obstacle is detected. If an obstacle is detected a local trajectory for the avoidance of the obstacle is calculated and the robot will track this new trajectory until the obstacle has been avoided and afterwards will return to tracking the given trajectory. The robot uses the same discrete-time sliding-mode control for both the desired trajectory and the local avoidance trajectory. The WMR PowerBot uses 14 sonar sensors in order to detect the obstacles. The sensors cover an area of 180 degrees and allow the detection of any obstacle that can block the robot's path. Fig.4 presents the configuration of the sonar sensors of the PowerBot. The obstacle detection is achieved using the sensitivity bubble from [19], Fig.5. This bubble is dependent on the robot's speed and allows for a good detection of the obstacles. Using safety coefficients and the robot's speed a safety area is defined inside the sensibility

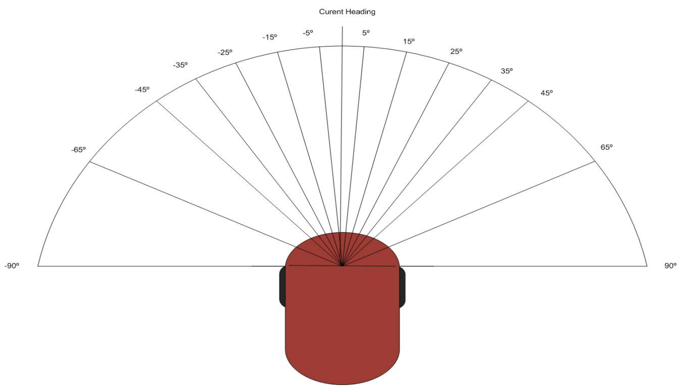


Fig. 4. The sonar configuration of the Powerbot

bubble and any obstacle detected inside this area will trigger the obstacle avoidance procedure. The components for the sensitivity bubble are calculated using

$$b_i = k_i \cdot v \cdot T_s, \tag{20}$$

where: i is the number of the sonar; b_i is the component of the sensitivity bubble corresponding to sonar i; k_i is the safety coefficient of the bubble corresponding to sonar i; v is the velocity of the robot; T_s is the sample time. Fig. 6 presents an example of an obstacle that was detected by the sensitivity bubble. An obstacle is detected when the distance to the obstacle measured by any of the sonar sensors is lower than the component of the sensitivity bubble corresponding to that sonnar

$$Range(i) < b_i. \tag{21}$$

First a global desired trajectory is generated off-line using a program that calculates the desired velocity, acceleration, angular velocity and angular acceleration for each time sample. A program for generating the global trajectory is presented in [6]. Before imposing the trajectory to the robot, the distances measured by the sonar sensors are collected and the sensitivity bubble is calculated. Using this information the algorithm determines if an obstacle was detected. As long as no obstacle is detected the desired trajectory is fed to the controller and the robot tracks this trajectory to its destination. If an obstacle is detected the obstacle the obstacle avoidance procedure is triggered. An obstacle detected using the sensitivity bubble is presented in Fig. 5. In order to avoid the obstacle, a local trajectory is generated. The local trajectory for the obstacle from Fig.5 is presented in Fig. 6. Define two search areas for the new trajectory bounded by the sonar arrays. The first seven sonar sensors determine the left area for avoidance and the last seven the right area for avoidance. The search will begin in the area opposite to the area with the closest obstacle. A new search will be performed in the other area if a solution is not found. First the coordinates of the end point of the trajectory that allows the obstacle to be avoided is calculated. This point is situated on the line perpendicular to the direction of travel from the point where the closest obstacle was detected.

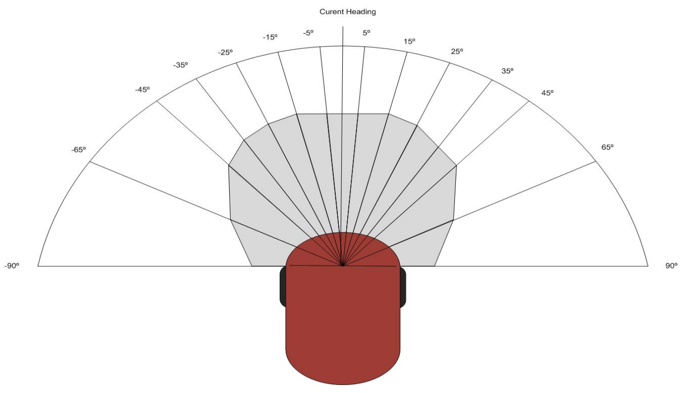


Fig. 5. The sensitivity buble of the PowerBot

Consider $h_{\min}$ the width of the PowerBot plus a small safety margin, as the minimum width of an area that allows the passage of the robot. Once the search area was selected, will search for the closest sonar to the direction of travel that does not detect an obstacle. The algorithm determines the points where the sonar beams intersect the perpendicular from the point where the obstacle was detected to the direction of travel. The algorithm tries to find a segment determined by the intersection points, starting from the intersection point of the selected sonar, which is at least as long as $h_{\min}$. The desired point is located at the middle of this segment.

The segment is searched using

$$h = m\_range \cdot \tan(angle(j)) - Ze \, , \tag{22}$$

where: h is the length of the segment, m_range is the distance to the closest obstacle projected on the direction of travel, $angle(j)$ is the angle formed by the sonar, $Ze = m\_range \cdot \tan(angle(i))$ is the exclusion zone, i and j are the sonar sensors used at the current step. The coordinates are determined using

$$\begin{cases} x = x_r + m\_range \cdot \cos(\theta_r) - (\frac{h}{2} + Ze) \cdot \sin(\theta_r) \\ y = y_r - m\_range \cdot \cos(\theta_r) - (\frac{h}{2} + Ze) \cdot \sin(\theta_r) \end{cases} \tag{23}$$

The path from the current robot's position to $P(x, y,)$ is determined using Quintic equations [6]. The formula of the Quintic G^2-splines is

$$p_{i,i+1}(u) = \begin{bmatrix} \alpha_{i0} + \alpha_{i1} \cdot u + \alpha_{i2} \cdot u^2 + \ldots + \alpha_{i5} \cdot u^5 \\ \beta_{i0} + \beta_{i1} \cdot u + \beta_{i2} \cdot u^2 + \ldots + \beta_{i5} \cdot u^5 \\ \theta_i(u) \end{bmatrix}, \tag{24}$$

$$\alpha_{i0} = x_i(0) \, , \tag{25}$$

$$\alpha_{i1} = g_1 \cdot \cos(\theta_i(0)) \, , \tag{26}$$

$$\alpha_{i2} = \frac{1}{2}\left(g_3 \cdot \cos(\theta_i(0)) - g_1^2 \cdot k_i \cdot \sin(\theta_i(0))\right), \tag{27}$$

$$\begin{aligned} \alpha_{i3} &= 10 \cdot (x_{i+1} - x_i) - \left(6 \cdot g_1 + \frac{3}{2} \cdot g_3\right)\cos(\theta(0)) - \\ &\left(4 \cdot g_2 - \frac{1}{2} g_4\right)\cos(\theta_{i+1}(0)) + \frac{3}{2} \cdot g_1^2 \cdot k_i \cdot \sin(\theta_i(0)) \, , \\ &-\frac{1}{2} g_2^2 \cdot k_{i+1} \cdot \sin(\theta_{i+1}(0)) \end{aligned} \tag{28}$$

$$\begin{aligned} \alpha_{i4} &= -15 \cdot (x_{i+1} - x_i) + \left(8 \cdot g_1 + \frac{3}{2} \cdot g_3\right)\cos(\theta(0)) + \\ &(7 \cdot g_2 - g_4)\cos(\theta_{i+1}(0)) - \frac{3}{2} \cdot g_1^2 \cdot k_i \cdot \sin(\theta_i(0)) + \, , \\ &g_2^2 \cdot k_{i+1} \cdot \sin(\theta_{i+1}(0)) \end{aligned} \tag{29}$$

$$\begin{aligned} \alpha_{i5} &= 6 \cdot (x_{i+1} - x_i) - \left(3 \cdot g_1 + \frac{1}{2} \cdot g_3\right)\cos(\theta(0)) - \\ &\left(3 \cdot g_2 - \frac{1}{2} g_4\right)\cos(\theta_{i+1}(0)) + \frac{1}{2} \cdot g_1^2 \cdot k_i \cdot \sin(\theta_i(0)) \, , \\ &-\frac{1}{2} g_2^2 \cdot k_{i+1} \cdot \sin(\theta_{i+1}(0)) \end{aligned} \tag{30}$$

$$\beta_{i0} = y_i(0) \, , \tag{31}$$

$$\beta_{i2} = \frac{1}{2}\left(g_3 \cdot \sin(\theta_i(0)) - g_1^2 \cdot k_i \cdot \cos(\theta_i(0))\right) , \tag{32}$$

$$\begin{aligned} \beta_{i3} &= 10 \cdot (y_{i+1} - y_i) - \left(6 \cdot g_1 + \frac{3}{2} \cdot g_3\right)\sin(\theta(0)) - \\ &\left(4 \cdot g_2 - \frac{1}{2} g_4\right)\sin(\theta_{i+1}(0)) + \frac{3}{2} \cdot g_1^2 \cdot k_i \cdot \cos(\theta_i(0)) - \, , \\ &\frac{1}{2} g_2^2 \cdot k_{i+1} \cdot \cos(\theta_{i+1}(0)) \end{aligned} \tag{33}$$

$$\begin{aligned} \beta_{i4} &= -15 \cdot (y_{i+1} - y_i) + \left(8 \cdot g_1 + \frac{3}{2} \cdot g_3\right)\sin(\theta(0)) \\ &+ (7 \cdot g_2 - g_4)\sin(\theta_{i+1}(0)) - \frac{3}{2} \cdot g_1^2 \cdot k_i \cdot \cos(\theta_i(0)) \, , \\ &+ g_2^2 \cdot k_{i+1} \cdot \cos(\theta_{i+1}(0)) \end{aligned} \tag{34}$$

$$\begin{aligned} \beta_{i5} &= 6 \cdot (y_{i+1} - y_i) - \left(3 \cdot g_1 + \frac{1}{2} \cdot g_3\right)\sin(\theta(0)) - \\ &\left(3 \cdot g_2 - \frac{1}{2} g_4\right)\sin(\theta_{i+1}(0)) + \frac{1}{2} \cdot g_1^2 \cdot k_i \cdot \cos(\theta_i(0)) \, , \\ &-\frac{1}{2} g_2^2 \cdot k_{i+1} \cdot \cos(\theta_{i+1}(0)) \end{aligned} \tag{35}$$

where k_i, k_{i+1} are scalar curvatures and can be arbitrary set, g_1, g_2, g_3, g_4 are the parameters that determine the shape of the curve. For each element of this path the velocity ($v[n]$), angular velocity ($\omega[n]$), acceleration ($a[n]$) and angular acceleration ($a\omega$) are calculated using:

$$l[n] = \sqrt{(x[n]-x[n-1])^2 + (y[n]-y[n-1])^2}\ ; \tag{36}$$

$$v[n] = l[n]/T_s\ ; \tag{37}$$

$$\omega[n] = a\tan 2(y[n]-y[n-1], x[n]-x[n-1])/T_s\ ; \tag{38}$$

$$a[n] = v[n]/Ts\ ; \tag{39}$$

$$a\omega[n] = \omega[n]/Ts\ ; \tag{40}$$

The desired trajectory for obstacle avoidance is presented in Fig. 7 with a blue line, the blocked trajectory is marked with a black line.This new trajectory is fed to the controller and when the robot reaches the end point of the trajectory it will move forward in a straight line with constant velocity until the obstacle has been cleared. Once the obstacle has been cleared a return trajectory is calculated and the initial trajectory is resumed (see the procedure from Fig.8).

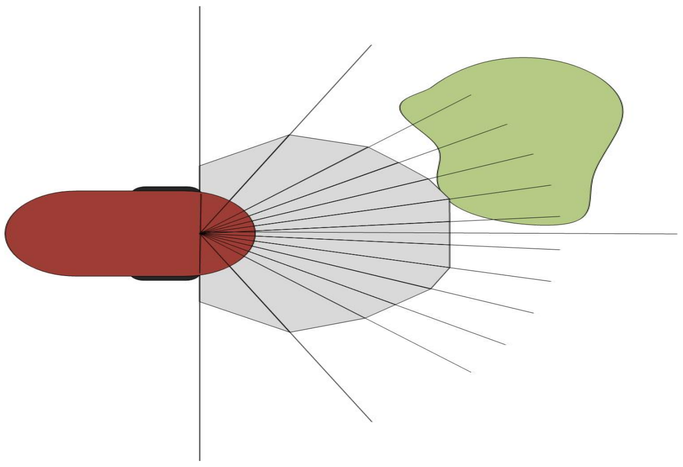


Fig. 6. Example of a detected obstacle

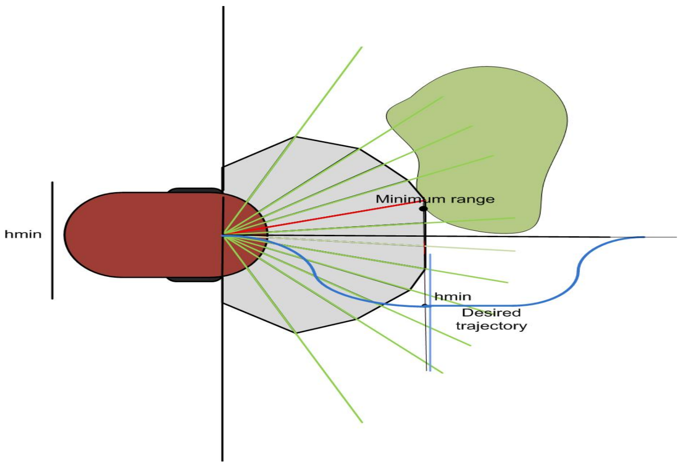


Fig. 7. The desired trajectory for obstacle avoidance

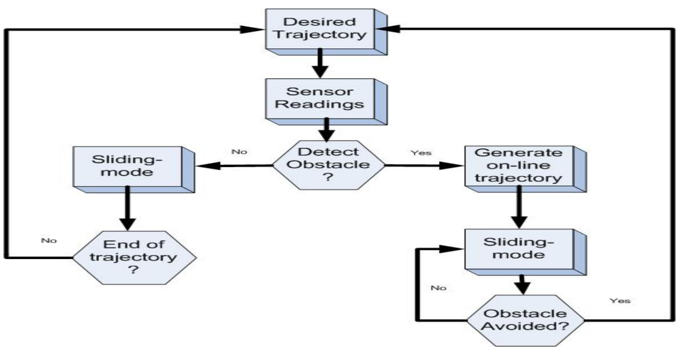


Fig. 8. The schematic of the obstacle avoidance algorithm

IV. Simulation and Real-Time Control

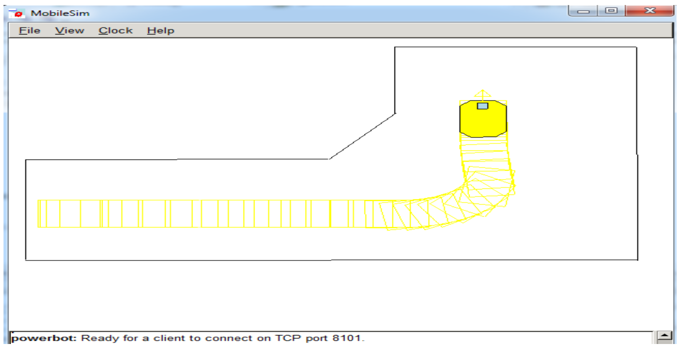
Simulation results are presented in order to validate the proposed algorithm for obstacle avoidance. The control program is written in C++ and run on a PC, with sampling time set to T_s=100ms. All the simulations were made using MobileSim. MobileSim is a software designed to simulate the behavior of robots produced by Adept Mobile Robots Inc. A map with the simulated obstacles was created using Maper3Basic and loaded into Mobilesim. The desired trajectory is presented in Fig. 9. The trajectory tracking was first tested in the absence of any obstacle and the robot followed the intended trajectory with small tracking errors. The obstacle avoidance performance of the algorithm is tested using the trajectory from Fig. 9 as the global trajectory needed to be tracked and 2 obstacles blocking this path. Fig. 10 presents the MobileSim simulated trajectory of the robot obtained after avoiding the obstacles. It can be seen that the algorithm allows the robot to avoid the obstacles and continue on its initial trajectory when the path is clear. Fig. 11 presents the simulated trajectory and the desired trajectory resulted from the initial trajectory and the local avoidance trajectories. From this figures it can be determined that the algorithm is capable of tracking a desired trajectory and avoid obstacles, that block the intended path, and resume the initial trajectory afterwards. The switch between the initial trajectory and the local avoidance trajectory and back increase the trajectory-tracking errors. One of the causes for the larger tracking errors can be attributed to the delay caused by the time required to obtain the sonar readings and process the information, which leads to higher delays in the command inputs as opposed to the case when only the sliding-mode control is used and the sonars are disabled.

V. Conclusions

An algorithm for discrete-time sliding-mode control and obstacle avoidance for wheeled mobile robots is presented in this paper. The effectiveness of this algorithm is proven by simulation results. The robot tracks a global trajectory and, if an obstacle is detected, a local trajectory is generated and followed until the obstacle is cleared and the initial trajectory is resumed. Both trajectories are followed using the same discrete-time sliding mode controller. An increase of the errors in the trajectory tracking is caused by switching from the global and local trajectories, but the robot can still follow the trajectory with satisfactory precision.

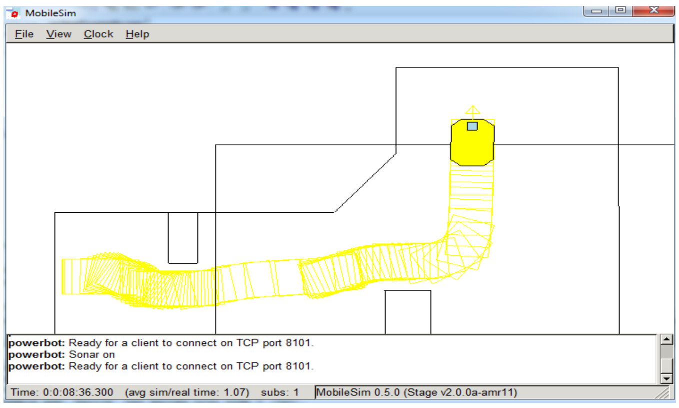
ACKNOWLEDGMENT

This work was supported by the Romanian Executive Unit of Funding Higher Education, Research, Development and Innovation (UEFISCDI), project number PN-II-ID-PCE-2011-3-0641, project title: *Advanced control of reversible manufacturing systems of assembling and disassembling using wheeled mobile robots equipped with robotic manipulators* and by the project number PN-II-PT-PCCA-2013-4-0686, project title: *Prototypes of autonomous robotic systems for medical/social assistance and servicing of manufacturing processes in metallurgy, ceramics, glass and automotive.*



```
powerbot: Ready for a client to connect on TCP port 8101.
```

Fig. 9. The MobileSim simulated trajectory without obstacles



```
powerbot: Ready for a client to connect on TCP port 8101.
powerbot: Sonar on
powerbot: Ready for a client to connect on TCP port 8101.
Time: 0:0:08:36.300   (avg sim/real time: 1.07)   subs: 1   MobileSim 0.5.0 (Stage v2.0.0a-amr11)
```

Fig. 10. The MobileSim simulated trajectory with 2 obstacles blocking the intended trajectory

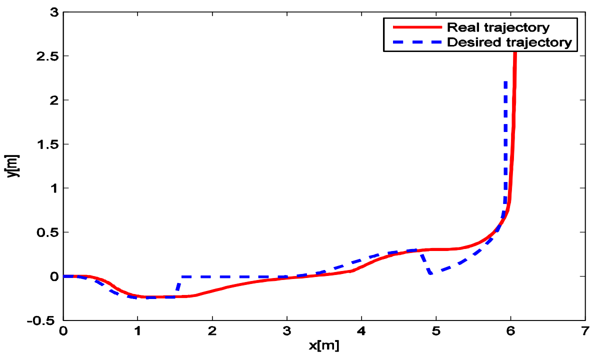


Fig. 11. The simulated trajectory and the desired trajectory

REFERENCES

[1] V.I. Utkin, Sliding modes in optimization and control. New York: Springer-Verlag, 1992.

[2] V.I. Utkin, J. Shi, Sliding mode control in electromechanical systems. London: Taylor&Francis, 1999.

[3] W. Gao, J. C. Hung, "Variable structure control on nonlinear systems: A new approach," *IEEE Transactions on Industrial Electronics*, vol. 40, pp. 45–55, 1993.

[4] W. Gao, Y. Wang and A. Homaifa, ""Discrete-time variable structure control systems," *IEEE Transactions on Industrial Electronics*, vol. 42, pp. 117–122, April 1995.

[5] R. Solea, A. Filipescu and G. Stamatescu, "Sliding-mode real-time mobile platform control in the presence of uncertainties," in *Proc. 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference,* Shanghai, pp. 7747-7752, 2009.

[6] R. Solea, "Sliding mode control applied in trajectory-tracking of WMRs and autonomous vehicles," Ph.D. Thesis Dept. of Electrical and Computer Engineering, University of Coimbra, Portugal, 2009.

[7] R. Solea, A. Filipescu, U. Nunes, "Sliding-mode control for trajectory-tracking of a wheeled mobile robot in presence of uncertainties," in *Proc. 7th Asian Control Conference,* Hong Kong, pp. 1701-1706, 2009.

[8] R. Solea, A. Filipescu, S. Filipescu and B. Dumitrascu, "Sliding-mode controller for four-wheel-steering vehicle: Trajectory-tracking problem," in *Proc. 8th World Congress on Inteligent Control and Automation,* Jinan, pp. 1185-1190, 2010.

[9] R. Solea, A. Filipescu, V. Minzu and S. Filipescu, "Sliding-mode trajectory-tracking control for a four," in *Proc. 8th IEEE International Conference on Control and Automation,* Xiamen, pp. 382-387, 2010.

[10] K. Furuta and Y. Pan, "Discrete-time variable structure control," Lecture Notes in Control and Information Sciences, vol. 274, pp 57-81, Berlin: Springer, 2006.

[11] B. Bandyopadhyay and S. Janardhanan, "Discrete-time sliding mode control: A multirate output feedback approach," Lecture Notes in Control and Information Sciences, vol. 323, Berlin: Springer-Verlag, 2005.

[12] B. Dumitrascu and A. Filipescu, "Discrete-time sliding-mode controller for wheeled mobile robots," ," in *Proc. 18th International Conference on Control Systems and Computer Science,* vol.1, Bucharest, pp. 397-403, may, 2011.

[13] B. Dumitrascu and A. Filipescu, "Sliding mode control of lateral motion for four driving-steering wheels autonomous vehicle," Annals of the University of Craiova, vol(7), pp. 20-25, 2010.

[14] B. Dumitrascu and A. Filipescu, A. Radaschin, A. Filipescu Jr., E. Minca, "Discrete-time sliding mode control of four driving/steering wheels mobile platform," in *Proc. 8th Asian Control Conference (ASCC), Kaohiung, Taiwan, pp 771-776, 2011.*

[15] A. Filipescu, V. Minzu, B. Dumitrascu, A. Filipescu, "Trajectory-tracking and discrete time sliding-mode control of wheeled mobile robots, *The 2011 IEEE International Conference on Information and Automation*, Shenzhen, China, pp. 27-32, 2011.

[16] B. Dumitrascu, "Contributions to control, navigation and obstacle avoidance for mobile robots and autonomous vehicles," Ph.D. Thesis Dept. Department of Automation and Electrical Engineering, University Dunarea de Jos of Galati, Romania, 2012.

[17] J. Borenstein, Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE Journal of Robotics and Automation,* vol. 7, pp 278-288, 1991.

[18] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots, *IEEE International Conference on Robotics and Automation*, pp 500-505, 1985.

[19] Susnea I, A. Filipescu, G. Vasiliu, G. Coman, A. Radaschin, ""The buble rebound obstacle avoidance algorithm for mobile robots," in *Proc. 8th International Conference on Control and Automation, Xiamen, pp. 540-545, 2010.*

[20] Lumelsky, V., Skewis, T., "Incorporating Range Sensing in the Robot Navigation Function." *IEEE Transactions on Systems, Man, and Cybernetics*, 20:1990, pp. 1058–1068.