LOCAL AND GLOBAL ARTIFICIAL POTENTIAL FUNCTIONS IN THE CONTROL OF MOBILE ROBOTS

K. KOZŁOWSKI* and W. KOWALCZYK*

Faculty of Computing, Poznań University of Technology, Poznań, 60-965, Poland
** E-mail: {krzysztof.kozlowski, wojciech.kowalczyk}@put.poznan.pl*
control.put.poznan.pl

The article presents overview of authors' results concerning mobile robot control algorithms that use local artificial potential functions (APF) to avoid collisions and global artificial potential functions, named also navigation functions (NF) used to both collision avoidance and driving robot to a desired goal. All included algorithms assume that the mobile platform is differentially driven mobile robot with nonholonomic constraints. Effectiveness of presented methods is illustrated by simulation and experimental results. Experimental setup used to demonstrate control algorithms is presented.

Keywords: mobile robot; collision avoidance; artificial potential function; navigation function.

1. Introduction

The number of applications in which mobile robots are used to solve some practical problems is rapidly increasing. Collision avoidance is one of the basic problems in this kind of systems. During the task execution robot performs motion to the goal or along the desired trajectory simultaneously avoiding collisions.

In this paper two alternative approaches to collision avoidance are reviewed. The former are APFs originally proposed by[1] in 1986. A large number of methods use this approach. Its main advantages are conceptual simplicity and ease of implementation. On the other hand this methodology has an important limitation. If the obstacles' areas of repulsion overlap local minima may occur.

This problem was solved in the turn of 1980s and 1990s by Rimon and

This work is supported by statutory grant 09/93/DSPB/0811

H. Montes et al. (eds.), *CLAWAR 2018: 21st International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines*,
Robotics Transforming the Future

Koditschek,[2,3,4] They proposed NF method, called also global potential function that guarantees that local minima will not appear. The proposed method assumed that the robot has no nonholonomic constraints. In 2004 Urakubo proposed an extension to a two-wheeled mobile robot.[5]

This paper presents the overview of the control methods that use APFs and NF to control differentially driven mobile platform. Depending on the method the goal is to track desired trajectory or convergence to the desired fixed position and orientation.

Section 2 presents model of the mobile robot. Section 3 introduces the concept of APF and methods based on it. In Subsection 3.1 the linear control algorithm is described. It uses linearized model of the robot. Subsection 3.2 presents persistent excitation method. Persistent excitation block is responsible for the convergence of the position in the direction transverse to the main axis of the robot. In Subsection 3.3 vector field orientation method is introduced. Section 4 presents two NF control methods. In Subsection 4.1 control algorithm for sphere worlds is described. In Subsection 4.2 its extension to more complex star worlds is presented. Section 5 describes experimental setup used to verify effectiveness of the proposed methods. In the last Section concluding remarks are given.

2. Model of the robot

The kinematic model of the differentially-driven mobile robot is given by the following equation:

$$\dot{q} = \tilde{B}u = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} u \tag{1}$$

where x, y, θ are position and orientation coordinates of the robot with respect to a global, fixed coordinate frame; v - linear control velocity, ω - angular control velocity.

3. APF methods

Collision avoidance behavior is based on the artificial potential functions (APF). This concept originally was proposed in.[1] All robots are surrounded by APFs that raise to infinity near objects border r_j ($j = 1, ..., M$, M - number of the obstacles) and decreases to zero at some distance R_j, $R_j > r_j$.

One can introduce the following function:[6]

$$B_{aj}(l_j) = \begin{cases} 0 & for \quad l_j < r_j \\ e^{\frac{l_j - r_j}{l_j - R_j}} & for\ r_j \leq l_j < R_j \\ 0 & for \quad l_j \geq R_j \end{cases}, \tag{2}$$

that gives output $B_{aj}(l_j) \in \langle 0, 1)$. Euclidean distance between the robot and the j-th obstacle is as follows: $l_j = \left\| [x_j\ y_j]^\top - [x\ y]^\top \right\|$. Note that for $l_j < r_j$ an arbitrary value of the function $B_{aj}(l_j)$ can be set assuming that the robot does not get into this area. In the presented algorithms this is guaranteed.

Scaling function (2) within the range $\langle 0, \infty)$ can be obtained as follows:

$$V_{aj}(l_j) = \frac{B_{aj}(l_j)}{1 - B_{aj}(l_j)}, \tag{3}$$

that is used later to avoid collisions. Note that $V_{aj}(l_j)$ and its spatial derivatives are bounded for $l_j > r_j$.

3.1. *Linear control method*

In this section trajectory tracking controller proposed in[7] is extended by collision avoidance behavior.

The goal of the control is to drive the formation along the desired trajectory avoiding collisions with the static obstacles. The assumption is made that the planner generates desired trajectory that does not intersects APFs of the obstacles. Trajectory tracking is equivalent to bringing the following quantities to zero:

$$\begin{aligned} p_x &= x_d - x \\ p_y &= y_d - y \\ p_\theta &= \theta_d - \theta, \end{aligned} \tag{4}$$

where x_d and y_d are desired position coordinates and θ_d is desired orientation. The system error expressed with respect to the coordinate frame fixed to the robot is described below:

$$\begin{bmatrix} e_x \\ e_y \\ e_\theta \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_\theta \end{bmatrix}. \tag{5}$$

Using above equations and nonholonomic constraint $\dot{y}\cos(\theta) - \dot{x}\sin(\theta) =$

0 the error dynamics is as follows:

$$\begin{aligned} \dot{e}_x &= e_y\omega - v + v_d \cos e_\theta \\ \dot{e}_y &= -e_x\omega + v_d \sin e_\theta \\ \dot{e}_\theta &= \omega_d - \omega \end{aligned} . \tag{6}$$

One can introduce the position correction variables that consist of position errors and collision avoidance terms:

$$\begin{aligned} P_x &= p_x - \textstyle\sum_{j=1}^{M} \frac{\partial V_{aj}}{\partial x} \\ P_y &= p_y - \textstyle\sum_{j=1}^{M} \frac{\partial V_{aj}}{\partial y} \end{aligned} . \tag{7}$$

V_{aj} depends on x and y according to equation (3). The correction variables can be transformed to the local coordinate frame fixed in the mass center of the robot:

$$\begin{bmatrix} E_x \\ E_y \\ e_\theta \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ p_\theta \end{bmatrix} . \tag{8}$$

Above equation can be transformed to the following form:[8]

$$\begin{bmatrix} \frac{\partial V_{aj}}{\partial e_x} \\ \frac{\partial V_{aj}}{\partial e_y} \end{bmatrix} = \begin{bmatrix} -\cos\theta & -\sin\theta \\ \sin\theta & -\cos\theta \end{bmatrix} \begin{bmatrix} \frac{\partial V_{aj}}{\partial x} \\ \frac{\partial V_{aj}}{\partial y} \end{bmatrix} . \tag{9}$$

Finally, correction variables expressed with respect to the local coordinate frame are as follows:

$$\begin{aligned} E_x &= e_x + \textstyle\sum_{j=1}^{M} \frac{\partial V_{aj}}{\partial e_x} \\ E_y &= e_y + \textstyle\sum_{j=1}^{M} \frac{\partial V_{aj}}{\partial e_y} \end{aligned} . \tag{10}$$

For $l_j > R_j$ components of the gradient of the APF vanish: $\frac{\partial V_{aj}}{\partial e_x} = 0$ and $\frac{\partial V_{aj}}{\partial e_y} = 0$. It leads to the conclusion that in this case $E_x = e_x$ and $E_y = e_y$.

The algorithm presented in[7] extended by the collision avoidance functionality is as follows:

$$\begin{aligned} v &= v_d + k_1 E_x \\ \omega &= \omega_d + k_2 \mathrm{sign}(v_r) E_y + k_3 e_\theta \end{aligned} . \tag{11}$$

Substituting (11) into (6) one can express error dynamics as follows:

$$\begin{aligned} \dot{e}_x &= e_y\omega - k_1 E_x + v_d(\cos e_\theta - 1) \\ \dot{e}_y &= -e_x\omega + v_d \sin e_\theta \\ \dot{e}_\theta &= -k_2 \mathrm{sign}(v_r) E_y - k_3 e_\theta \end{aligned} . \tag{12}$$

When the robot detects the obstacle its reference trajectory is temporarily frozen, reference signals: v_d and ω_d are set to zero. The tracking process is temporarily suspended because collision avoidance has a higher priority. Once the robot is outside the collision detection region, it updates the reference to the new values.

Error dynamics for $v_d = 0$ and $\omega_d = 0$ becomes:

$$\begin{aligned} \dot{e}_x &= k_3 e_y e_\theta - k_1 E_x \\ \dot{e}_y &= -k_3 e_\theta e_x \\ \dot{e}_\theta &= -k_3 e_\theta \end{aligned} \quad . \tag{13}$$

Fig. 1 shows path of the robot in xy-plane. Time graph of position and orientation error is presented in Fig. 2. Notice that despite of the fact that the algorithm is based on the linearization (first Lyapunov method) it ensures quick convergence even if the initial position is far from the equilibrium point.

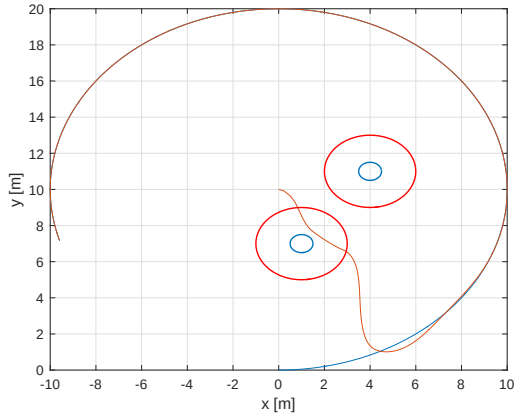

Fig. 1. Path of the robot in xy-plane (simulation results)

3.2. *Control with persistent excitation*

In this subsection the trajectory tracking control algorithm proposed in[9] is extended by collision avoidance. Equations (4) - (10) introduced in Section 3.1 remain in force in further computations.

Following reference[9] control signals of the robot in the case of collision

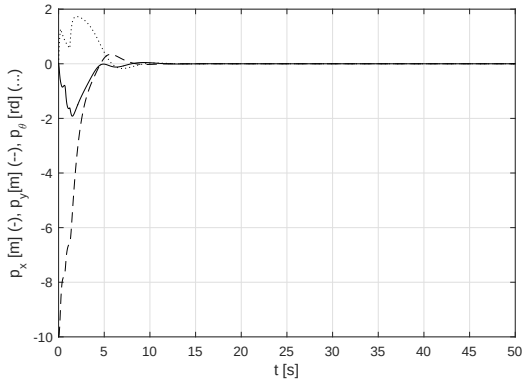

Fig. 2. Robot position and orientation errors (simulation results)

avoidance are proposed as follows:

$$\begin{aligned} v &= v_d + c_2 E_x \\ \omega &= \omega_d + h(t, E_y) + c_1 e_\theta \end{aligned} \tag{14}$$

where $h(t, E_y)$ is bounded, depends linearly on E_y, and continuously differentiable function. It must be properly chosen to ensure persistent excitation of the reference angular velocity.[10]

Substituting Eq. (14) into Eq. (6) one can express error dynamics as follows:

$$\begin{aligned} \dot{e}_x &= e_y \omega - c_2 E_x + v_d(\cos e_\theta - 1) \\ \dot{e}_y &= -e_x \omega + v_d \sin e_\theta \\ \dot{e}_\theta &= -h(t, E_y) - c_1 e_\theta \end{aligned} \quad . \tag{15}$$

When the robot is in the interaction area its reference trajectory is temporarily frozen, reference signals: v_d and ω_d are set to zero.

Error dynamics become:

$$\begin{aligned} \dot{e}_x &= h(t, E_y) e_y + c_1 e_y e_\theta - c_2 E_x \\ \dot{e}_y &= -h(t, E_y) e_x - c_1 e_\theta e_x \\ \dot{e}_\theta &= -h(t, E_y) - c_1 e_\theta \end{aligned} \quad . \tag{16}$$

Fig. 3 presents path of the robot in xy-plane. Time graph of position and orientation errors is shown in Fig. 4.

3.3. *Vector-Field-Orientation method*

This subsection presents VFO trajectory tracking control algorithm[11] extended by the collision avoidance behavior.[12]

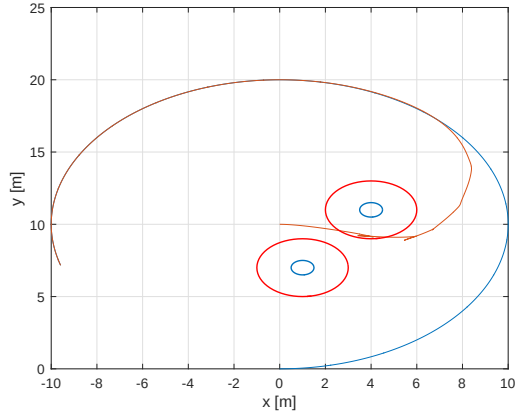

Fig. 3. Path of the robot in xy-plane (simulation results)

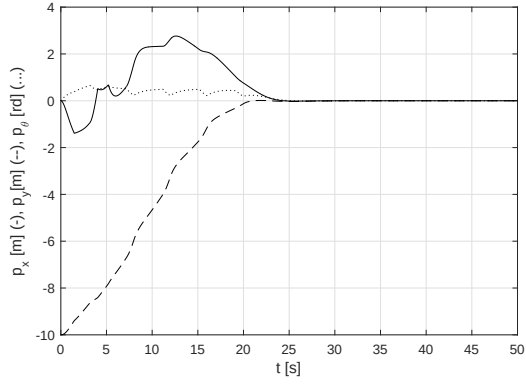

Fig. 4. Robot position and orientation errors (simulation results)

One can introduce the convergence vector:

$$\mathbf{h} = \begin{bmatrix} h_x \\ h_y \\ h_\theta \end{bmatrix} = \begin{bmatrix} k_p P_x + \dot{x}_d \\ k_p P_y + \dot{y}_d \\ k_\theta e_a + \dot{\theta}_a \end{bmatrix}, \tag{17}$$

where k_p, $k_\theta > 0$ are control gains of position and orientation errors, respectively; P_x, P_y are correction variables given by Eq. (4), $e_a = \theta_a - \theta$ is auxiliary orientation error. Auxiliary orientation variable θ_a is defined as follows: $\theta_a = atan2c(h_y, h_x)$ ($atan2c(\cdot,\cdot)$[11] is continuous version of the

function $Atan(\cdot)$). Proposed control law for the robot are the following:

$$\begin{array}{c} u_v = h_x \cos\theta + h_y \sin\theta \\ u_\omega = h_\theta \end{array}. \tag{18}$$

The following assumptions are imposed.

Assumption 3.1. *Desired trajectories do not intersect APF areas of obstacles and robots do not interact when tracking is executed perfectly.*

Assumption 3.2. *If robot position is in the repel area then reference trajectory is frozen:*

$$\mathbf{q}_d(t) = \mathbf{q}_d(t^-), \tag{19}$$

where t^- is the time value before robot gets to the repel area. Higher derivatives of $q_d(t)$ are kept zero until robot leaves the repel area.[13]

Assumption 3.3. *When $e_a \in (\frac{\pi}{2} + \pi d - \delta, \frac{\pi}{2} + \pi d + \delta)$, where δ is a small positive value, $d = 0, \pm 1, \pm 2, ...$, then auxiliary orientation variable θ_a is replaced by $\tilde{\theta}_a = \theta_a + sgn\left(e_a - \left(\frac{\pi}{2} + \pi d\right)\right)\varepsilon$, where ε is a small value that fulfills condition $\varepsilon > 0$ and $sgn(\cdot)$ denotes the signum function.*

Assumption 3.4. *When robot reaches a saddle point reference trajectory is disturbed to drive robot out of local equilibrium point. In the saddle point the following condition is fulfilled:*

$$\|\mathbf{h}^*\| = 0, \tag{20}$$

where $\mathbf{h}^* = \begin{bmatrix} h_x & h_y \end{bmatrix}^T$. In this case $\theta_a(t)$ is frozen: $\theta_a(t) = \theta_a(t^-)$.

Fig. 5 presents path of the robot in xy-plane.

4. NF methods

In the turn of 1980s and 1990s Rimon and Koditschek presented concept of the navigation function. At first a sphere world version was introduced[2] that assumes that the obstacles are bounded with spheres in three dimensional space or with circles in planar case. Then the method was expanded to more complex environments,[3,4] All these algorithms assumed a point-like robot without constraints.

In 2004 Urakubo[5] expanded the method introducing navigation function that takes into account nonholonomic constraints of the mobile robot. In his method the orientation of the robot can reach desired value just as both position coordinates. Convergence proof was included in the paper.

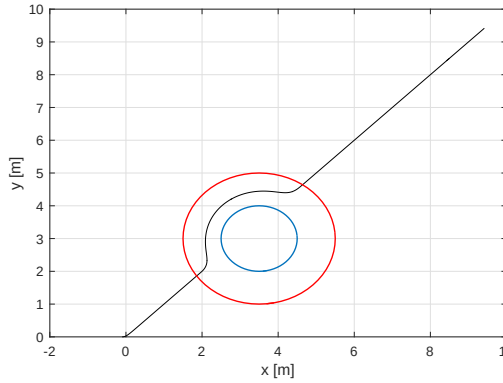

Fig. 5. Path of the robot in xy-plane (simulation results)

In the papers,[14],[1516] navigation function was used to control multiple mobile robots. Authors of these publications adress the problem of collision avoidance in multiagent robotic systems. In the first of the mentioned papers extension of the navigation function called multi-robot navigation functions (MRNFs) was applied. Second and third of these papers propose the use of prioritization to solve conflicts in case of concurrent goals of the agents. Examples of navigation function used for collision avoidance in 3D space were presented among others in work[17] and.[16]

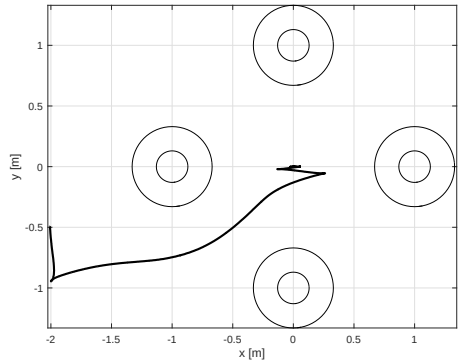

Fig. 6. Path of the robot in xy-plane (experimental results)

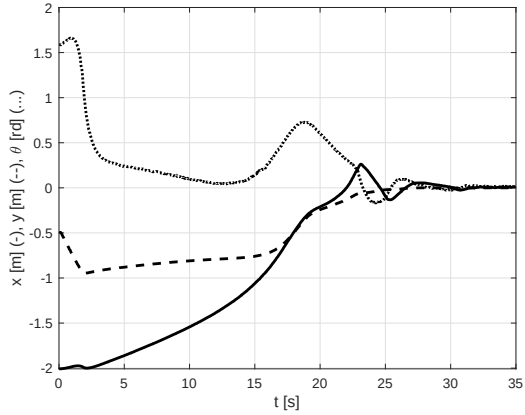

Fig. 7. Time graphs of x, y, θ variables (experimental results)

4.1. *NF for sphere worlds*

The control algorithm requires the robot task space to be bounded by a circle. It is the obstacle number zero described by the following obstacle function:[4]

$$\beta_0 \triangleq \rho_0^2 - ||r - p_0||^2, \tag{21}$$

where ρ_0 is the radius of the task space, $r = [x \;\; y]^\top$ is the current position of the robot and p_0 is the center of the task space. The controller design assumes the obstacles to be circular-shaped objects of radii ρ_i ($i = 1, ..., N$, N being the number of obstacles) with their centers located at positions p_i. The definition of repelling potential function for i-th obstacle is:

$$\beta_i \triangleq ||r - p_i||^2 - \rho_i^2. \tag{22}$$

Given an environment with N obstacles and a task of stabilizing the robot in its origin the total navigation potential is defined as:

$$V \triangleq \frac{C}{(C^\kappa + \beta)^{\frac{1}{\kappa}}}, \tag{23}$$

where κ is a positive, constant design parameter and

$$C \triangleq ||r||^2 + \theta^2 \frac{k_w}{k_w + ||r||^2}. \tag{24}$$

Symbol k_w in (24) denotes a positive, constant design parameter that allows to tune the influence of the orientation term on the navigation function depending on the Euclidean distance to the goal. The aggregation of repelling

potentials happens in term β which is defined as:

$$\beta \triangleq \prod_{i=0}^{N} \beta_i. \tag{25}$$

One must note that the iteration starts from zero, which means the inclusion of task space boundary potential.

The control algorithm proposed in work[18] is defined as:

$$u \triangleq -\left\{ a \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + b \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \right\} \tilde{B}^{\top} \nabla V, \tag{26}$$

where a is a positive, constant design parameter and

$$b \triangleq -\bar{b} \frac{L^{\top} \nabla V}{h(g)}. \tag{27}$$

Symbol $\bar{b}$ in (27) denotes another positive, constant design parameter and $L^{\top} \triangleq [\sin\theta \quad -\cos\theta \quad 0]$. Function $h(g)$ is defined as:

$$h(g) \triangleq g^2 + \epsilon\sqrt{g}, \tag{28}$$

where $g \triangleq ||\tilde{B}^{\top} \nabla V||$ and ϵ is a small positive constant. Finally, ∇V denotes the gradient of the navigation function with respect to variables x, y and θ. Regardless of number of obstacles, the gradient can be obtained in analytical form as:

$$\nabla V = \frac{\nabla C (C^{\kappa} + \beta)^{\frac{1}{\kappa}} - \frac{C}{\kappa}(C^{\kappa} + \beta)^{(\frac{1}{\kappa} - 1)} (\kappa C^{\kappa - 1} \nabla C + \nabla \beta)}{(C^{\kappa} + \beta)^{(\frac{2}{\kappa})}}, \tag{29}$$

where

$$\nabla C = \begin{bmatrix} \frac{\partial C}{\partial x} \\ \\ \frac{\partial C}{\partial y} \\ \\ \frac{\partial C}{\partial \theta} \end{bmatrix} = \begin{bmatrix} 2x(1 - \frac{k_w \theta^2}{(k_w + ||r||^2)^2}) \\ \\ 2y(1 - \frac{k_w \theta^2}{(k_w + ||r||^2)^2}) \\ \\ 2\theta \frac{k_w}{k_w + ||r||^2} \end{bmatrix} \tag{30}$$

and

$$\nabla \beta = \sum_{i=0}^{N} \left\{ \frac{\partial \beta_i}{\partial q} \prod_{j=0, j \neq i}^{N} \beta_j \right\}. \tag{31}$$

As noted in[4] all the undesired local minima of navigation function (23) disappear as the parameter κ increases. An algorithm for automatically tuning analytic navigation functions for sphere worlds was presented in.[19]

The tuning parameter must satisfy a lower bound to ensure convergence to the desired value. In this paper navigation functions have been manually tuned to assure convergence. For the sufficiently high value of the κ parameter navigation function (23) has a critical point associated with each isolated obstacle, the saddle point. V has no other critical point other than these points. Saddle points are unstable equilibrium points. In[18] special control procedure for saddle point avoidance is described. It uses time varying function to push the robot away from the unstable equilibrium point.

4.2. *NF for star worlds*

NF algorithm for the star world is an extension of the method described in subsection 4.1. In this approach the position of the robot is transformed to auxiliary sphere world:

$$\hat{r} = \left(1 - \sum_{i=1}^{M} s_i(r)\right) r + \sum_{i=1}^{M} s_i(r) T_i(r) \tag{32}$$

where

$$s_i(r) = \frac{||r||^2 \prod_{j=1, j\neq i}^{M} \beta_j(r)}{||r||^2 \prod_{j=1, j\neq i}^{M} \beta_j(r) + \lambda_s \beta_i(r)} \tag{33}$$

and

$$T_i(r) = \frac{\rho_i(1 + \beta_i(r))}{||r - q_i||}(r - q_i) + p_i, \tag{34}$$

$$T_0(r) = \frac{\rho_0(1 - \beta_i(r))}{||r - q_i||}(r - q_i) + p_i. \tag{35}$$

In the above equations p_0 and p_i are centers of the spheres to which original obstacles are transformed, ρ_0 and ρ_i are their radii (they should fall entirely in their original star obstacles), q_0 and q_i are are centers of the stars (points from which all the rays cross the boundary of the obstacle once and only once), β_0 and β_i are star obstacle functions.

Navigation function is given by the following equation:

$$V \triangleq \frac{\hat{C}}{(\hat{C}^{\kappa} + \hat{\beta})^{\frac{1}{\kappa}}}, \tag{36}$$

where

$$\hat{C} \triangleq ||\hat{r}||^2 + \theta^2 \frac{k_w}{k_w + ||\hat{r}||^2} \tag{37}$$

and

$$\hat{\beta}_0 \triangleq \rho_0^2 - ||\hat{r} - p_0||^2, \tag{38}$$

$$\hat{\beta}_i \triangleq ||\hat{r} - p_i||^2 - \rho_i^2. \tag{39}$$

The form of the control law does not change in comparison to the sphere world algorithm (26).

Fig. 8 presents intersection of the potential field for $\theta = 0$. In Fig. 9 path of the robot is shown. Fig. 10 presents time graphs of the robot coordinates.

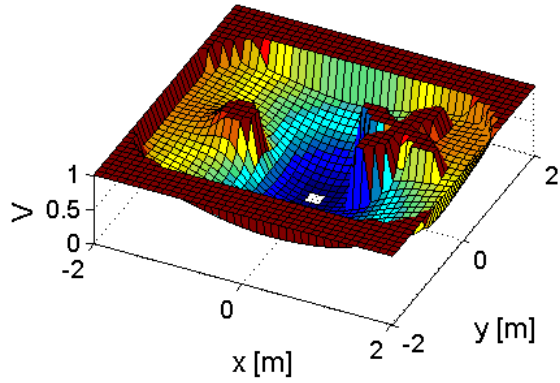

Fig. 8. Intersection of the potential field for $\theta = 0$ (notice that NF is a function of three variables: x, y and θ; it can not be shown on a flat drawing)

5. Experimental setup

The mobile platform that was used in the presented experiments is the differentially driven MTracker robot. It wast designed at Poznan University of Technology, Institute of Automation and Robotics. It is controlled by a two-level hardware controller: the low-level motion controller uses the signal processor TMS320F28335 150MHz and the high-level one is a single-core Intel Atom 1.2GHz board, equipped with WiFi radio used for remote management, task setting and communication with the external localization system. Depending on the requirements the high-level controller works under the Linux Ubuntu or Windows XP operating systems. The MTracker robot is a small platform: its diameter is $0.14m$, its height is $0.13m$, its

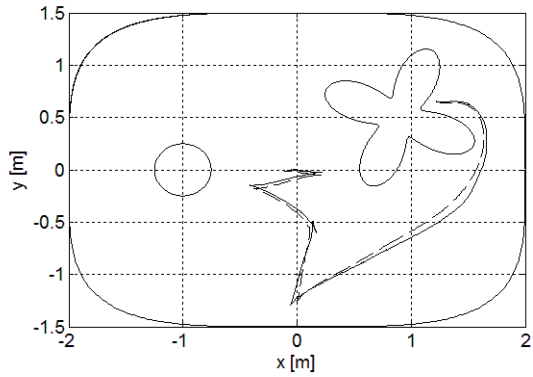

Fig. 9. Path of the robot in xy-plane (solid line - experimental data, dashed line - numerical simulation) (experimental results)

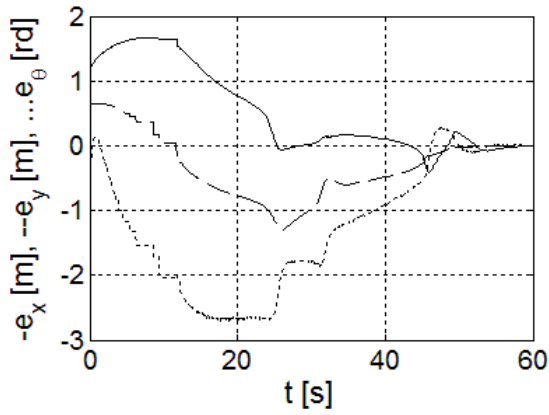

Fig. 10. Time graphs of error variables (experimental results)

weight is $1.4kg$ and its wheels have a diameter of $0.05m$. The on-board power supply is LiIo $3.7Ah$ battery that allows two hour active operation.

During the test the robot is localized by the OptiTrack motion capture system. On the top of the robot four infra-red reflecting markers were mounted.

Wheel velocity control signals were scaled down when their value(s) exceeded the limit. This limit was set to $12rd/s$, while the physical limitation of actuators is $24rd/s$. The lower value of the limit prevented robot wheels from longitudinal slip. The obstacles were known *a priori* to prevent the

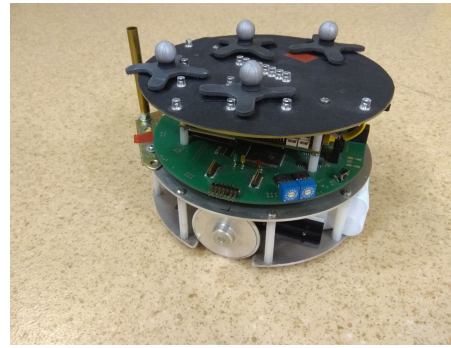

Fig. 11. MTracker robot (Poznan University of Technology)

influence of measurement inaccuracies on the experimental results.

A special scaling procedure is applied to the wheel controls. The desired wheel velocities are scaled down when at least one of the wheels exceeds the assumed limitation. The scaled control signal $\vec{u}_s$ is calculated as follows:

$$\vec{u}_s = s\vec{u}, \tag{40}$$

where

$$s = \begin{cases} \frac{\omega_{max}}{\omega_o} & \text{if} \quad \omega_o > \omega_{max} \\ 1 & \text{otherwise} \end{cases}, \tag{41}$$

and

$$\omega_o = \max\{|\omega_r|, |\omega_l|\}, \tag{42}$$

where ω_r, ω_l denote right and left wheel angular velocity, ω_{max} is the predefined maximal allowed angular velocity for each wheel. This scaling procedure preserves the direction of motion of the mobile platform.

6. Conclusion

In this article the overview of the control methods that use APFs and NFs is presented. Effectiveness of presented methods was illustrated by simulation results and experiments conducted using MTracker differentially-driven mobile robot. Description of the experimental setup was included.

References

1. O. Khatib, *The International Journal of Robotics Research* **5**, 90 (1986).

2. E. Rimon and D. Koditschek, Exact robot navigation using cost functions: the case of distinct spherical boundaries, in *IEEE International Conference on Robotics and Automation*, 1988.
3. E. Rimon and D. Koditschek, *Transaction of the American Mathematical Society* **327**, 71 (1991).
4. E. Rimon and D. Koditschek, *IEEE Transactions on Robotics and Automation* **8**, 501 (1992).
5. T. Urakubo, K. Okuma and Y. Tada, Feedback control of a two wheeled mobile robot with obstacle avoidance using potential functions, in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2004.
6. W. Kowalczyk, M. Michałek and K. Kozłowski, *Bulletin of the Polish Academy of Sciences Technical Sciences* **60**, 537 (2012).
7. C. C. de Wit, H. Khennouf, C. Samson and O. J. Sordalen, *Nonlinear Control Design for Mobile Robots* 1994.
8. W. Kowalczyk and K. Kozłowski, Leader-follower control and collision avoidance for the formation of differentially-driven mobile robots, in *23nd International Conference on Methods and Models in Automation and Robotics (MMAR)*, August 2018. accepted for publication in conference proceedings.
9. A. Loria, J. Dasdemir and N. A. Jarquin, *IEEE Transactions on Control Systems Technology* **24**, 727 (2016).
10. A. Loria, E. Panteley and A. Teel, *IEEE Trans. Autom. Control* **46**, 1363 (2001).
11. M. Michałek and K. Kozłowski, *IEEE Transactions on Control Systems Technology* **18**, 45 (2010).
12. W. Kowalczyk, K. Kozłowski and J. Tar, Trajectory tracking for multiple unicycles in the environment with obstacles, in *International Conference on Robotics in Alpe-Adria-Danube Region (RAAD)*, 2010.
13. S. Mastellone, D. Stipanovic, C. Graunke, K. Intlekofer and M. Spong, *The International Journal of Robotics Research* **27**, 107 (2008).
14. D. Dimarogonasa, S. Loizoua, K. Kyriakopoulos and M. Zavlanosb, *Automatica* **42**, 229 (2005).
15. G. Roussos and K. Kyriakopoulos, *Distributed Autonomous Robotic Systems - Springer Tracts in Advanced Robotics* **83**, 189 (2013).
16. G. Roussos and K. Kyriakopoulos, Completely decentralised navigation of multiple unicycle agents with prioritisation and fault tolerance, in *IEEE Conference on Decision and Control (CDC)*, 2010.
17. G. Roussos and K. Kyriakopoulos, *IEEE Transactions on Control Systems Technology* **20**, 1622 (2012).
18. T. Urakubo, *Nonlinear Dynamics* **81**, 1475 (2015).
19. I. Filippidis and K. Kyriakopoulos, Adjustable navigation functions for unknown sphere worlds, in *IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, 2011.