Control of a Mobile Robot and Collision Avoidance Using Navigation Function - Experimental Verification

Wojciech Kowalczyk[1] Mateusz Przybyła[1] and Krzysztof Kozłowski[1]

***Abstract*— In this paper experimental verification of a mobile robot control and colision avoidance is presented. The control is based on the navigation function that was proposed in [2]. Navigation function agregates position, orientation and colision avoidance terms. Experimental results show that both position and orientation lead to desired values very quickly and reach them with small errors.**

I. Introduction

Artificial potential functions (APF) are widely used in robotics since 1986 when they were proposed by Khatib [1]. The idea of repulsive and attractive interactions used to avoid collisions and move to desired position in the original version had some limitations. The main one was possibility of local minima when the APF was not designed carefully.

In the beginning of '90 Rimon and Koditschek presented concept of the navigation function. At first a sphere world version was introduced [3] that assumes that the obstacles are bounded with spheres in three dimensional spaces or with circles in planar case. Then the method was expanded to more complex environments [4], [5] and [6]. All these algorithms assumed a point-like robot without constraints.

In 2004 Urakubo [2] expanded the method introducing navigation function that takes into account nonholonomic constraints of the mobile robot. The orientation of the robot can reach desired value just as both position coordinates. Convergence proof was included in the paper.

In the papers [11], [12] and [13] navigation function was used to control multiple mobile robots. They adress the problem of collision avoidance in multiagent robotic systems. In the first extension of the navigation function called multi-robot navigation functions (MRNFs) was applied. Second and third use prioritization to solve conflicts in case of concurrent goals of the agents. In work [7] authors propose a local potential function to control formation of mobile robots in an environment with obstacles.

In this paper experimental results for the algorithm presented in [2] are shown. In Section II the model of the system is presented. In Section III the control algorithm is described. In Section IV the test-bed used in the experiments is described shortly. Sections V and VI provide brief information on state estimation and obstacle detection. In Section VII experimental results for the method are given. In the final section we include concluding remarks.

*This work was supported by NCBiR grant PBS1/A3/8/2012 (RobREx)

[1]Authors are with Chair of Control and Systems Engineering, Faculty of Computing, Poznan University of Technology, Piotrowo 3A, Poznan, Poland wojciech.kowalczyk@put.poznan.pl mateusz.przybyla@doctorate.put.poznan.pl krzysztof.kozlowski@put.poznan.pl

II. Model of the system

Although the experiments were conducted on a four wheeled, skid steering mobile platform, it is treated as a differentially driven mobile robot with a kinematic model given by the following equation:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = B \begin{bmatrix} v \\ \omega \end{bmatrix}, \tag{1}$$

where x, y and θ are position coordinates and orientation of the robot, respectively, $\vec{u} = \begin{bmatrix} v & \omega \end{bmatrix}^\top$ is the control vector with v denoting linear velocity and ω denoting angular velocity of the platform and

$$B = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix}. \tag{2}$$

The task space is assumed to be bounded by a circle. The definition of the obstacle function [4] (the boundary of the task space is the obstacle number zero) is $\beta_0 = \rho_0^2 - ||\vec{r} - \vec{p}_0||^2$, where ρ_0 is radius of the task space, $\vec{r} = [x \quad y]^\top$ is the current position of the robot and $\vec{p}_0$ is the center of the task space, $\beta_i = ||\vec{r} - \vec{p}_i||^2 - \rho_i^2$ where $\vec{p}_i$ is the position of the center of the i-th obstacle, ρ_i is the radius of the i-th obstacle and $i = 1, ..., M$, M - the number of the internal obstacles. For simplicity the goal position is the origin of the coordination frame with the orientation equal to zero.

III. Control

Control proposed in [2] is given by the equation:

$$\begin{bmatrix} \dot{v} \\ \dot{w} \end{bmatrix} = -\left\{ a \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + b \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \right\} B^\top \nabla V, \tag{3}$$

where

$$b = -\bar{b} \frac{L^\top \nabla V}{h(g)}, \tag{4}$$

$L = [\sin\theta \quad -\cos\theta \quad 0]^\top$, $h(g) = g^2 + \epsilon_g \sqrt{g}$, $g = ||B^\top \nabla V||$, a, $\bar{b}$ are positive constants and ϵ_g is small positive constant.

The ∇V is the gradient of the navigation function:

$$V = \frac{C}{(C^\kappa + \beta)^{\frac{1}{\kappa}}}, \tag{5}$$

where $C = ||\vec{r}||^2 + w\theta^2$, $w = \frac{k_w}{k_w + ||\vec{r}||^2}$ and k_w is positive constant that allows to tune the influence of the orientation term on the navigation function depending on the euclidean distance to the goal. As noted in [4] all the undesired local minima disappear as the parameter κ increases. The equation

978-1-4799-7043-8/15/$31.00 ©2015 IEEE

(5) is a navigation function as the parameter κ exceeds a certain function of the geometric data (configuration of the obstacles in the environment). An algorithm for automatically tuning analytic navigation functions for sphere worlds was presented in [10]. 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 sufficenly high value of the κ parameter navigation function (5) has a critical point associated with each isolated obstacle, the saddle point. V has no other critical point other then these points. Saddle points are unstable equilibrium points. In [2] special control procedure for saddle point avoidance is described. It uses time varying trigonometric function to push the robot away from the unstable equilibrium point.

The obstacle function:

$$\beta = \prod_{i=0}^{M} \beta_i \tag{6}$$

is used as collision avoidance term in (5).

IV. Experimental test-bed

The platform on which the experiments were conducted is a four-wheeled, skid-steering mobile robot (Fig. 1). It is approximately 50 cm wide and 60 cm long, and weights about 30 kg. The slippage of the wheels, that occurs during the motion, introduce unmodeled disturbances which result in a more challenging control task. During the experiments, only the rear wheels were directly actuated, whereas the front wheels were actuated with the use of passive transmission belts. The platform velocities were scaled so that it did not exceed the speed of 0.5 m/s.

The platform is equipped with a computer running a Robot Operating System (ROS, version Hydromedusa), a pair of Hokuyo URG-04LX-UG01 laser scaners and an digital camera. The laser scanners are positioned at the top of the robot (the scan plane lays 34 cm above the ground) with a separation between axes of mirrors rotation equal to 45 cm. The digital camera points upwards in order to recognize the artificial landmarks placed at the ceiling with the use of ArUco library. The robot uses Wi-Fi to connect with the user interface on a stationary PC but all of the processing is done onboard.

V. State Estimation

The absolute pose of the robot is obtained from the ArUco vision system which works with the rate of 25 Hz. Both scaners work with the rate of 10 Hz, and the ROS main cycle works with the rate of 100 Hz. Two separate Kalman filters are presented in order to enchance the estimate of the pose. For the sake of smooth state estimation, the system was augmented with additional state variables and divided into two parts:

$$\vec{q} = \begin{bmatrix} \vec{q}_a \\ \vec{q}_l \end{bmatrix} = \begin{bmatrix} [\varphi & r & \dot{r}] & [x & u_f & \dot{u_f} & y & u_s & \dot{u_s}] \end{bmatrix}^\top, \tag{7}$$

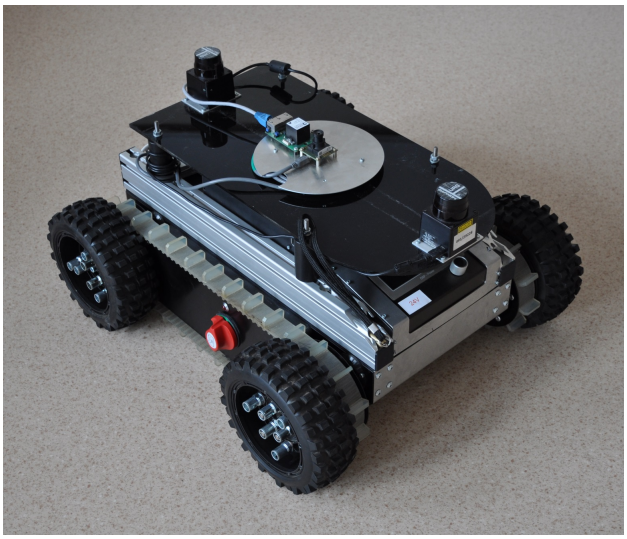

Fig. 1. Robot used for experiments

where u_f and u_s denote forward and sideways linear velocities. The angular subsystem $\vec{q}_a$ is a LTI (linear, time-invariant) system and the linear subsystem $\vec{q}_l$ is a LTV (linear, time-variant) system. The angular subsystem can be described with a following state space representation:

$$\begin{aligned} \dot{\vec{q}}_a &= A_a \vec{q}_a = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \vec{q}_a, \\ \vec{y}_a &= C_a \vec{q}_a = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix} \vec{q}_a. \end{aligned} \tag{8}$$

The lack of input matrix results from the fact that the input ω is in fact one of the state variables $\dot{\theta}$. The output matrix C_a represent: orientation measured by the vision system, orientation measured by the scan matching, angular velocity measured by the odometry and the control signal ω itself. The rank of matrix:

$$O_a = \begin{bmatrix} C_a \\ C_a A_a \end{bmatrix} \tag{9}$$

is full, so the system is observable. The system was discretized with the form $A_{da} = e^{A_a T_p}$ and $C_{da} = C_a$, where $T_p = 0.01$ s denotes the sampling time. As a result one obtained:

$$\begin{aligned} \vec{q}_a[k+1] &= A_{da} \vec{q}_a[k] = \begin{bmatrix} 1 & T_p & \frac{1}{2} T_p^2 \\ 0 & 1 & T_p \\ 0 & 0 & 1 \end{bmatrix} \vec{q}_a[k], \\ \vec{y}_a[k] &= C_{da} \vec{q}_a[k]. \end{aligned} \tag{10}$$

Similar reasoning was used to obtain the linear subsystem,

which continuous state space form can be represented as:

$$\dot{\vec{q}}_l = A_l \vec{q}_l = \begin{bmatrix} 0 & \cos\varphi & 0 & 0 & -\sin\varphi & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sin\varphi & 0 & 0 & \cos\varphi & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \vec{q}_l, \tag{11}$$

$$\vec{y}_l = C_l \vec{q}_l = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \vec{q}_l.$$

The measurements described by the output matrix C_l are position x and y obtained from both vision system and scan matching, as well as forward velocity obtained from optical encoders, and control input v itself. The matrix

$$O_l = \begin{bmatrix} C_l \\ C_l A_l \\ C_l A_l^2 \end{bmatrix} \tag{12}$$

has a full rank, so the linear subsystem is also observable. The discretization leads to the following system equations:

$$\vec{q}_l[k+1] = A_{dl}[k]\vec{q}_l[k] =$$

$$\begin{bmatrix} 1 & T_p\mathrm{c}\varphi[\mathrm{k}] & \frac{1}{2}T_p^2\mathrm{c}\varphi[\mathrm{k}] & 0 & -T_p\mathrm{s}\varphi[\mathrm{k}] & -\frac{1}{2}T_p^2\mathrm{s}\varphi[\mathrm{k}] \\ 0 & 1 & T_p & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & T_p\mathrm{s}\varphi[\mathrm{k}] & \frac{1}{2}T_p^2\mathrm{s}\varphi[\mathrm{k}] & 1 & T_p\mathrm{c}\varphi[\mathrm{k}] & \frac{1}{2}T_p^2\mathrm{c}\varphi[\mathrm{k}] \\ 0 & 0 & 0 & 0 & 1 & T_p \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \vec{q}_l[k],$$

$$\vec{y}_l[k] = C_{dl}[k]\vec{q}_l[k], \tag{13}$$

where s and c denote $\sin$ and $\cos$, respectively.

The variances of measurements were obtained from observations (with the exception of control signals). For the angular subsystem they are as in the Table I.

TABLE I

ANGULAR SUBSYSTEM MEASUREMENT VARIANCES.

θ_{cam}	θ_{scn}	$\dot{\theta}_{enc}$	ω
1.13e-006	1.25e-006	4.60e-007	2.10e-006

The linear measurement variances were observed to be as in the Table II. The process variances were tunned

TABLE II

LINEAR SUBSYSTEM MEASUREMENT VARIANCES.

x_{cam}	x_{scn}	y_{cam}
1.13e-006	1.25e-006	2.00e-005
y_{scn}	u_{f-enc}	v
4.60e-007	2.10e-006	1.00e-007

empirically. Because the scan matching measurements are prone to incrementing errors, the measurement covariance matrix for both the angular and the linear subsystems were multiplied by a forgetting factor which enlarges the appropriate variances over time. This ensures that the scan matching measurements are less important in the data fusion as the time elapses. When the appropriate variances grow significantly enough, the scan matching measurements are reset with the use of last pose estimate.

VI. OBSTACLE DETECTION

The range data obtained from the laser scanners is used to detect local obstacles around the robot. Because two laser scanners are used, their output is firstly merged into a single laser scan of angular range equal to $360°$. This step is performed with the use of a ROS `ira_laser_tools` package, which eliminates the occlusion effect. Secondly, the set of all data points (transformed from range data into local cartesian coordinates) is split into smaller subsets which pretend to circumscribe a *continuous* figure. The fact that the set of all data points is ordered is here exploited. A point is assumed to lay in the previous subset if the following inequality is satisfied.

$$||\vec{p}_i - \vec{p}_{i-1}|| < D_0 + D_1||\vec{p}_i||, \quad \text{for } i \in 2, 3, ..., n, \tag{14}$$

where $||\vec{p}_i||$ denotes the i-th data point and D_0, D_1 denote parameters used for algorithm refinement. If the first and last point of the total set of points satisfy the above condition, the first and last subsets are merged.

Next, each of the subsets of points is treated with a *split-and-merge* algorithm (more precisely: an *iterative-end-point-fit* algorithm), which additionally splits them into smaller groups pretending to represent a basic geometric shape (segment of a line or arc of a circle). The algorithm starts with creation of a leading line based on first and last points of the set, where the line is described with the general equation, and searches for a point (from the same set) that has maximal distance to this line. If this distance does not satisfy a criterion (14) (with different parameters values) the set is split into two at that point. Both new subsets are recursively treated with the same procedure. The *split-and-merge* was chosen based on comparision given in work [14].

Once the subsets of points are no longer divisible, each of them is used for data fitting. Firstly, the points are fitted with a circle equation. If its radius is too large (the size is provided arbitrarily) or its center lays closer to the robot than the mean of points (we assume only convex obstacles), then the linear fitting is used. For both fitting procedures the total-least-squares fitting is used.

The parametric equation for a line is:

$$Ax + By + C = 0, \tag{15}$$

where $C = -1$ is set constant in order to find an unambiguous solution of the A, B parameters. The parametric equation for circle is [8]:

$$(x - x_0)^2 + (y - y_0)^2 - r_0^2 = x^2 + y^2 + a_1 x + a_2 y + a_3 = 0, \tag{16}$$

where $(x_0,\ y_0)$ and r_0 are center of a circle and its radius, respectively. One can obtain the values from the following set of equations:

$$x_0 = -\frac{a_1}{2} \quad y_0 = -\frac{a_2}{2} \quad r_0 = \sqrt{\frac{a_1^2+a_2^2}{4} - a_3}. \tag{17}$$

The basic shapes fitting was implemented with the help of Armadillo library for handling matrices in C++ [9].

An example of detected obstacles (here only linear) is depicted in Fig. 2.

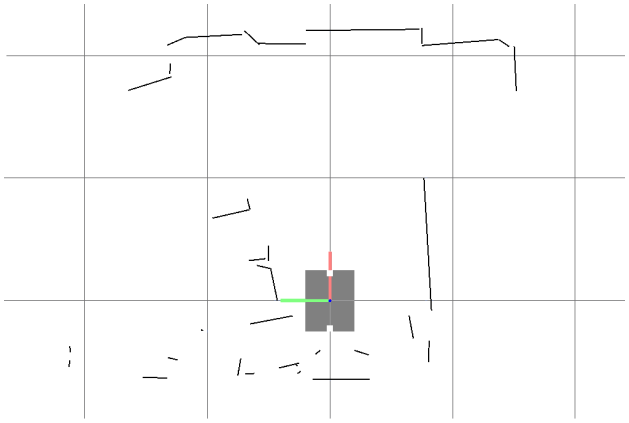

Fig. 2. Obstacles detected from laser scanners range data

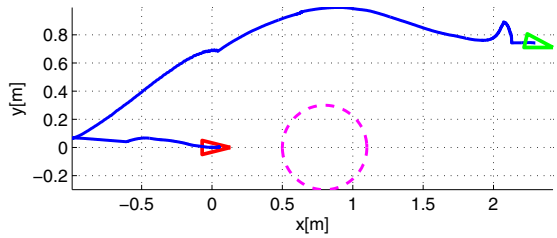

Fig. 3. Robot's path in XY plane during first experiment

VII. Experimental results

Experiments were performed for a circle shaped world with a centre in $\vec{p}_0 = (0, 0)$ and radius assumed to be $\rho_0 = 5$ m. In both presented cases the desired position was $(x_d, y_d) = (0, 0)$ m and orientation $\theta_d = 0$ rad. During experiments the position of the obstacles was known to the controller a priori. Algorithm constants used during the experiments were as follows: $a = 1$, $\bar{b} = 5$, $\epsilon_g = 10^{-6}$, $k_w = 1$ and $\kappa = 2$.

In the first experiment there was a single internal, circular obstacle with centre in $\vec{p}_1 = (0.8, 0.0)$ m and radius $\rho_1 = 0.3$ m. Robots initial position and orientation coordinates were

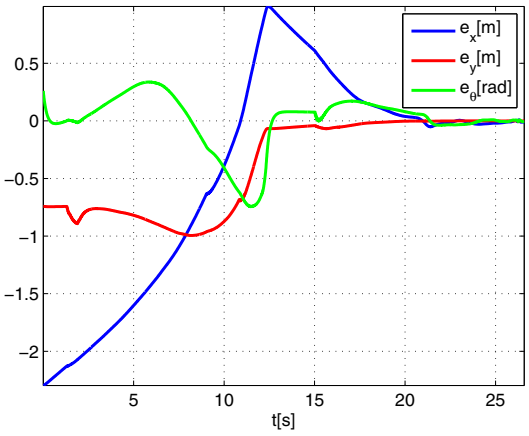

Fig. 4. Position and orientation errors during first experiment

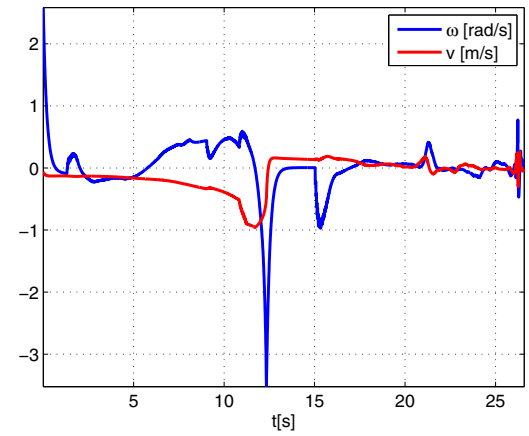

Fig. 5. Control signals during first experiment

as follows: $x_0 = 2.30$ m, $y_0 = 0.74$ m, $\theta_0 = -0.28$ rad. In Fig. 3 robot's path in XY plane is presented. Figures 4 and 5 depict, respectively, position and orientation errors as well as robot's control signals during the task execution. They reach near zero value after the transient state (practical stability). Final position oscilates back and forth slowly about the origin due to measurement noises.

In the second experiment three obstacles were placed in the vicinity of the goal: $\vec{p}_1 = (0.8, 0.0)$ with $\rho_1 = 0.3$ m, $\vec{p}_2 = (1.5, -1.0)$ with $\rho_2 = 0.5$ m and $\vec{p}_3 = (0.0, 2.5)$ with $\rho_3 = 0.5$ m. The robot started its motion with a configuration of $x_0 = 2.30$ m, $y_0 = 0.50$ m, $\theta_0 = -1.19$ rad. Figure 6 depicts robot's path in XY plane. Figures 7 and 8 depict errors and control signals, respectively.

VIII. Conclusion

In this paper experimental verification of the nonholonomic mobile robot control and collision avoidance algorithm was presented. Both position errors and orientation error are quickly reduced to near zero values. Future research will focus on the more complex environments, including star shaped obstacles and tree of stars obstacles. Authors plan to

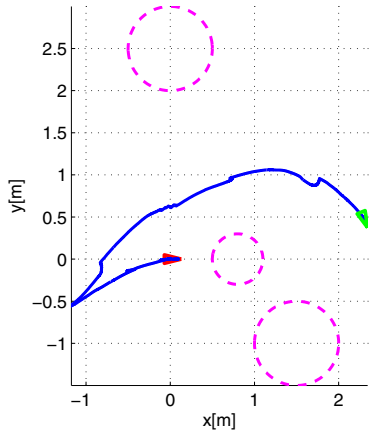

Fig. 6. Robot's path in XY plane during second experiment

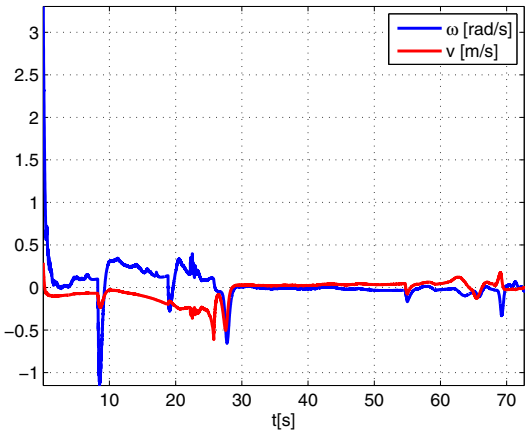

Fig. 8. Control signals during second experiment

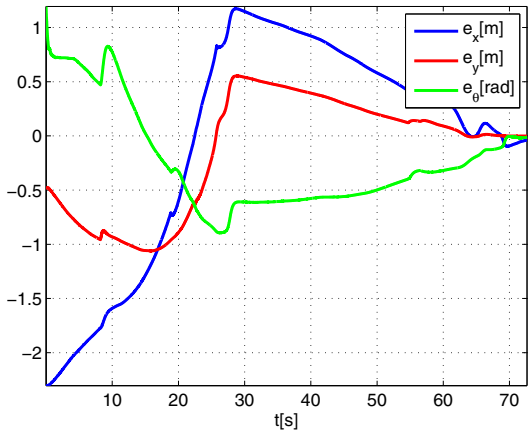

Fig. 7. Position and orientation errors during second experiment

expand the algorithm to adopt it to multiple nonholonomic robot control and dynamic environment. The saddle point avoidance procedure described in [2] will be also tested in the near future.

REFERENCES

[1] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, The International Journal of Robotics Research, Vol 5, No. 1, pp. 90–98, 1986.

[2] T. Urakubo, K. Okuma and Y. Tada, Feedback control of a two wheeled mobile robot with obstacle avoidance using potential functions, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vol. 3, pp. 2428–2433, 2004.

[3] E. Rimon and D.E. Koditschek, Exact robot navigation using cost functions: the case of distinct spherical boundaries, IEEE International Conference on Robotics and Automation, Vol. 3, pp. 1791–1796, 1988.

[4] E. Rimon and D. Koditschek, Exact Robot Navigation Using Artificial Potential Functions, IEEE Transactions on Robotics and Automation, Vol 8, No 5, pp. 501–518, 1992.

[5] E. Rimon and D. E. Koditschek, The construction of analytic diffeomorphisms for exact robot navigation on star worlds, Transaction of the American Mathematical Society, Vol. 327, pp. 71–116, 1991.

[6] E. Rimon and D.E. Koditschek, Exact robot navigation using artificial potential fields, IEEE Transactions on Robotics and Automation, Vol. 8, No. 5, pp. 501–518, 1992.

[7] W. Kowalczyk, K. Kozlowski, and J.K. Tar, Trajectory tracking for multiple unicycles in the environment with obstacles, International Conference on Robotics in Alpe-Adria-Danube Region (RAAD), pp. 451–456, 2010.

[8] I. Kasa, A circle fitting procedure and its error analysis, IEEE Transactions on Instrumentation and Measurement, pp. 8–14, 1976.

[9] C. Sanderson, Armadillo: An Open Source C++ Linear Algebra Library for Fast Prototyping and Computationally Intensive Experiments, Technical Report, NICTA, Australia, 2010.

[10] I. Filippidis and K.J. Kyriakopoulos, Adjustable navigation functions for unknown sphere worlds, IEEE Conference on Decision and Control and European Control Conference (CDC-ECC), pp. 4276–4281, 2011.

[11] D.V. Dimarogonasa, S.G. Loizoua, K.J. Kyriakopoulos and M.M. Zavlanosb, A feedback stabilization and collision avoidance scheme for multiple independent non-point agents, Automatica, Vol. 42, No. 2, pp. 229–243, 2005.

[12] G. Roussos, K.J. Kyriakopoulos, Decentralized and Prioritized Navigation and Collision Avoidance for Multiple Mobile Robots, Distributed Autonomous Robotic Systems - Springer Tracts in Advanced Robotics, Vol. 83, pp. 189–202, 2013.

[13] G. Roussos, K.J. Kyriakopoulos, Completely Decentralised Navigation of Multiple Unicycle Agents with Prioritisation and Fault Tolerance, IEEE Conference on Decision and Control (CDC), pp. 1372–1377, 2010.

[14] V. Nguyen, A. Martinelli, N. Tomatis and R. Siegwart, A comparison of line extraction algorithms using 2D laser rangefinder for indoor mobile robotics, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1929-1934, 2005.