APPLICATION OF PATH PLANNING BASED ON ARTIFICIAL POTENTIAL FIELDS TO MOBILE ROBOTS WITH DISTRIBUTED PROCESSING

Kamilla Pereira Peixoto, kmillapp@hotmail.com[1]
Pedro Henrique Silva Coutinho, phcoutinho@ufmg.br[2]
Allan Cavalcante Araújo, allancavalcante@gmail.com[1,3]
Berguem Paula Santos Almeida, berguempaula@gmail.com[1]
Filipe Hamdan Sampaio Souto, filipehamdan@gmail.com[1]
Thiago Pereira das Chagas, tpchagas@uesc.br[1]
Leizer Schnitman, leizer@ufba.br[3]

[1]Universidade Estadual de Santa Cruz, Campus Soane Nazaré de Andrade, Rodovia Jorge Amado, km 16, Salobrinho 45662-900. Ilhéus, BA, Brasil
[2]Universidade Federal de Minas Gerais, Av. Antônio Carlos 6627, 31270-901, Belo Horizonte, MG, Brasil
[3]Universidade Federal da Bahia, Rua Prof. Aristides Novis, 02, Federação, 40210-630, Salvador, BA, Brasil

Abstract: *This paper presents an autonomous nonholonomic differential drive mobile robot through processing entirely embedded. For this purpose, an Artificial Potential Field (APF) algorithm was used for local path planning and trajectory controller. The proposed APF algorithm uses vortex field to avoid null resultant potential field points and predicts future positions using odometry as an alternative to compensate processing times and serial communication lags. Furthermore, the robot has an isolated and decentralized processing architecture using Arduino Mega and Raspberry PI as embedded systems. Simulated and real experiments strongly suggest the effectiveness of the proposed path planning algorithm for real-time applications.*

Keywords: *Nonholonomic Robot, Autonomous Robot, Embedded Processing, Path Control.*

1. INTRODUCTION

Autonomous wheeled robots have been applied in different areas such as manufacturing, logistics, aerospace and others since 1970 (Tzafestas, 2013). Recently, due to applications at autonomous cars and industry 4.0, this theme is subjected to several advances, such as path planning, trajectory control and embedded processing. An autonomous robot requires some capabilities of path planning, sensors for ambient and self-perceptions, decision and trajectory control and some of them are explored in the present work with simulated and real experiments.

One class of wheeled robots is the Differential Drive Mobile Robot (DDMR) which has two parallel powered and independently driven wheels and, possibly, castor wheels for balance and stability. The motion of this type of mobile robot is subject to nonholonomic constraints since it is not able to move independently parallel to the powered wheels axis and requires a set of maneuvers to achieve a given configuration (Corke, 2011). These maneuvers shall be defined by the path planner and executed by the trajectory controller, both proposed for nonholonomic robots. Examples of this type of robot are motorized wheelchairs and automated guided vehicles in factories.

The Artificial Potential Field (APF) is a well-known method for path planning and obstacle avoidance (Krogh, 1984; Khatib, 1986). The application of the APF results in a trajectory from the actual robot position to the goal position considering it is a particle subjected to an attractive field (produced by the goal position) and repulsive fields (created by obstacles). The interaction among the fields results in a global attractive field with a global minimum at the goal position, however, zero potential field may arise at some specific points of the space causing the robot stagnation. One known solution to avoid local zero potential is using a vortex repulsive field around the obstacles, forcing the robot to contour them (DeMedio and Oriolo, 1991). Other techniques to avoid such local minima include, e.g, evolutionary algorithms (Vadakkepat *et al.*, 2000), modification on the structure of potential fields (Shi and Zhao, 2009) and, more recently, chaos optimization (Chen *et al.*, 2015) and fuzzy theory (Abdalla *et al.*, 2017).

Another limitation of the classical APF is considering robots as particles whose movement is not constrained by nonholonomic constraints. De Luca and Oriolo (1994) proposed a strategy for applying the APF to nonholonomic mobile robots based on kinematics model. Bemporad *et al.* (1996) applied the APF based on vortex field with local, incremental and on-line actualization of trajectories allowed to a trajectory controller to provide a solution to the problem of guiding the robot to the desired position avoiding obstacles.

The main contribution of this this work is applying the methodology of Bemporad *et al.* (1996) into a real experiment using a low cost DDMR completely autonomous, in the sense of do not require communication with external computers, based on an embedded distributed processing. APF and trajectory controller are executed in a Raspberry Pi using the software Scilab, while wheel speed measurement (based on encoders), wheel speed controllers (based on PI controllers) and robot localization (based on odometry) are executed in an Arduino Mega. Different processing times and serial communication lags demand a new contribution to the APF, which is applied here based on an estimation of future positions predicted by using the odometry. This results in an APF suitable for real-time applications based on distributed processing.

The following sections are organized as follows. Section 2 presents the robot kinematic model and defines odometry. Sections 3 and 4 describe the path planning based on APF and the trajectory controller, respectively. The application methodology is detailed in Section 4. Results of numerical and real experiments are presented and discussed in Section 5. And Section 6 summarizes a concluding remark.

2. ROBOT KINEMATIC MODEL AND ODOMETRY

Kinematic modelling considers the robot as a point whose mass does not influence motion, so perturbations such as friction and inertia are disregarded. In this section, a DDMR, whose structure is depicted in Fig. 1, is modelled. It has two powered wheels with radius R distanced by $2L$.

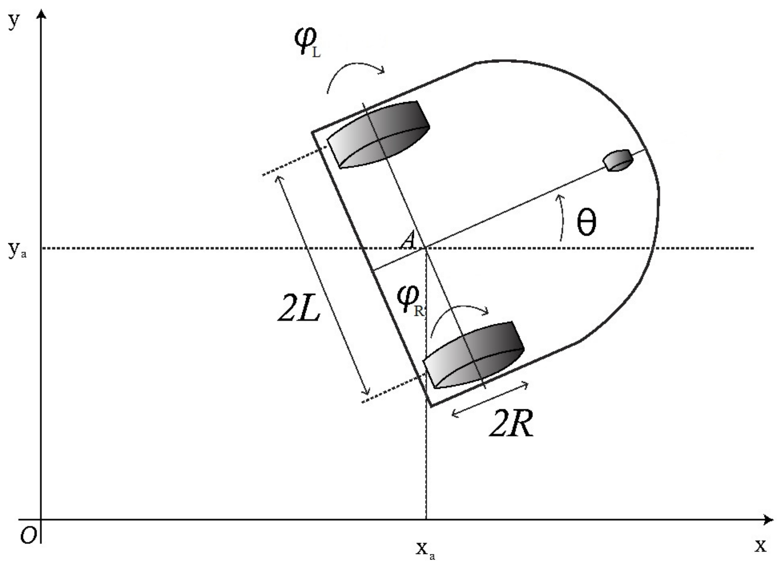


Figure 1: Differential Drive Mobile Robot schematic model. Adapted from Dhaouadi and Hatab (2013)

The DDMR's kinematic model is based on its geometry and considers two nonholonomic motion constraints: the robot cannot move sideways and the wheels cannot slip nor skid (Dhaouadi and Hatab, 2013). Thus, the robot's motion can be described by Eq. (1), where $\varphi_L(t)$ and $\varphi_R(t)$ are the left and right angular wheel speeds, respectively. Dhaouadi and Hatab (2013) presents details about this formulation.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \frac{R}{2} \begin{bmatrix} cos\theta & cos\theta \\ sin\theta & sin\theta \\ \frac{1}{L} & -\frac{1}{L} \end{bmatrix} \begin{bmatrix} \varphi_R \\ \varphi_L \end{bmatrix} \tag{1}$$

This model is used by the odometry technique. This technique consists of obtaining the current position and orientation from the discrete kinematic model and instantaneous wheel speed measurements (Chong and Kleeman, 1997). In this work, optical encoders mounted on both powered wheels provide the inputs needed to increment the robot's current location according to the discrete model obtained by applying the Euler method on Eq. (1), resulting in Eq. (2), where k denotes the time sample and T the sampling time.

$$\begin{aligned} x(k+1) &= x(k) + \frac{RT}{2} cos\theta(k)(\varphi_R(k) + \varphi_L(k)) \\ y(k+1) &= y(k) + \frac{RT}{2} sin\theta(k)(\varphi_R(k) + \varphi_L(k)) \\ \theta(k+1) &= \theta(k) + \frac{RT}{2L}(\varphi_R - \varphi_L) \end{aligned} \tag{2}$$

Odometry is a straightforward and inexpensive technique, however, it accumulates errors (Borenstein and Feng, 1996) associated with the conversion of angular wheel speed into linear displacement relative to the floor (Abbas *et al.*, 2006). Therefore, its isolated use may be impracticable for large routes.

3. PATH PLANNING

The (APF) is a path planning method in global and local planning. It computes an incremental trajectory considering the robot as a particle subject to attractive and repulsive fields associated to the goal position of the displacement and the known obstacles, respectively.

In this work, potential fields are defined in the workspace $\mathcal{W} = \mathbb{R}^2$. The attractive field $U_a(x_a, y_a)$ is defined as the Euclidean distance in Eq. (3), where $\xi \in \mathbb{R}_+^*$ is the factor of attraction of the field, (x_a, y_a) is the configuration of the current robot position in the workspace and (x_g, y_g) is the configuration of the goal position. Note that $U_a(x_a, y_a)$ is parabolic and has a minimum point at (x_g, y_g), so its force converges rapidly to zero as the robot approaches the goal position, what provides high stability (Barraquand and Latombe, 1991).

$$U_a(x_a, y_a) = \frac{1}{2}\xi||(x_a, y_a) - (x_g, y_g)||^2 \tag{3}$$

Furthermore, each i-th obstacle, ($i = 1, 2, 3 \ldots, r$, $r \in \mathbb{N}$) produces a repulsive field U_{ri} with radius of influence limited by a distance $\rho_0 \in \mathbb{R}_+^*$. Defining the repulsive field only in the neighborhood bounded by ρ_o prevents oscillations in the movement of the robot when it is distant from the obstacle (Khatib, 1986). The considered repulsive field used has a hyperbolic profile, as described in Eq. (4), where $\eta \in \mathbb{R}_+^*$ is the factor of repulsion of the field, $\rho_i(x_a, y_a)$ is the distance between the robot and the obstacle i, and $\gamma > 1$ is the potential decay rate (Bemporad *et al.*, 1996).

$$U_{ri}(x_a, y_a) = \begin{cases} \dfrac{1}{\gamma}\eta\left(\dfrac{1}{\rho_i(x_a, y_a)} - \dfrac{1}{\rho_0}\right)^{\gamma} & \text{, if } \rho_i(x_a, y_a) \leq \rho_0 \\ 0 & \text{, if } \rho_i(x_a, y_a) > \rho_0 \end{cases} \tag{4}$$

The resulting artificial potential field $U(x_a, y_a)$ is equivalent to the sum of all fields acting on that point and is defined at each iteration throughout the workspace (Tzafestas, 2013):

$$U(x_a, y_a) = U_a(x_a, y_a) + U_r(x_a, y_a), \tag{5}$$

where

$$U_r(x_a, y_a) = \sum_{i=1}^{r} U_{ri}(x_a, y_a). \tag{6}$$

Then, the resulting force applied to the robot $F(x_a, y_a)$ points to the goal position and is defined as the negative gradient of the resulting artificial potential field, i.e.

$$F(x_a, y_a) = -\nabla U(x_a, y_a) = -\nabla U_a(x_a, y_a) - \nabla U_r(x_a, y_a) = F_a(x_a, y_a) + F_r(x_a, y_a) \tag{7}$$

A vortex field is also included in order to force the robot to contour the obstacle instead of staying stuck at null resultant potential field points that are not the goal position (DeMedio and Oriolo, 1991). The force resulting from the vortex field $F_v(x_a, y_a)$ can be obtained in Eq.(8) and the resulting force becomes Eq. (10).

$$F_v(x_a, y_a) = \delta(x_a, y_a)\begin{bmatrix} \dfrac{\partial U_r(x_a, y_a)}{\partial y_a} & -\dfrac{\partial U_r(x_a, y_a)}{\partial x_a} \end{bmatrix}^T \tag{8}$$

The variable $\delta(x_a, y_a)$ specifies whether the vortex turns counterclockwise ($\delta(x_a, y_a) > 0$) or clockwise ($\delta(x_a, y_a) < 0$) and is defined as:

$$\delta(x_a, y_a) = sign\{\sin(\angle(x_g - x_o, y_g - y_o) - \angle(x - x_g, y - y_g))\}, \tag{9}$$

where (x_o, y_o) are the coordinates of the closest obstacle to be bypassed. Thus, the total force is given by Eq. (10). $F_v(x_a, y_a)$ is disregarded once the robot had bypassed the obstacle, i.e., when $\beta = \cos(\angle F_a - \angle F_r) > 0$.

$$F(x_a, y_a) = F_a(x_a, y_a) + F_v(x_a, y_a) \tag{10}$$

Therefore:

$$F(x_a, y_a) = \begin{cases} F_a(x_a, y_a) + F_v(x_a, y_a) & \text{, if } \beta \leq 0 \\ F_a(x_a, y_a) + F_r(x_a, y_a) & \text{, if } \beta > 0. \end{cases} \tag{11}$$

4. TRAJECTORY CONTROLLER

A closed loop control system is implemented since the APF compensates the error by computing a new trajectory at each iteration. In addition, the force $F(x_a, y_a)$ that the robot is subjected at each iteration is translated to angular speed of the wheels according to Eq.(12), where K_R and K_L are right and left constants, respectively.

$$\begin{aligned}\varphi_R(k) &= \frac{1}{2}K_R\left(F_x\cos\theta + F_y\sin\theta + 2L\angle F - \theta\right)\\ \varphi_L(k) &= \frac{1}{2}K_L\left(F_x\cos\theta + F_y\sin\theta - 2L\angle F - \theta\right)\end{aligned} \tag{12}$$

The robot's actuators have physical constraints, so they are not able to reach all speeds required by the path planning. Then the maximum force that will be demanded to the robot must be limited to avoid the motor's saturation zone. In this work, a strategy similar to *Force Control* proposed by Tilove (1990) is used. The main difference between both algorithms is that *Force Control* limits directly the acceleration and velocity of the robot, whereas the proposed algorithm limits the maximum magnitude of the force (F_{max}) according to:

$$F(x_a, y_a) = \begin{cases}\dfrac{F(x_a, y_a)}{||F(x_a, y_a)||}F_{max} & \text{, if } ||F(x_a, y_a)|| > F_{max}\\[2em] F_a(x_a, y_a) & \text{, if } ||F(x_a, y_a)|| \le F_{max}.\end{cases} \tag{13}$$

5. EXPERIMENTAL PROCEDURE

Once the robot's initial and final positions and the obstacles locations are defined, the APF and the path controller, both implemented on Scilab, compute, respectively, the force and wheel speed required to achieve the next position. Those speed values are transmitted to an Arduino board through serial communication, where they work as setpoints to PI controllers. The PI outputs are sent to a motor shield which controls the voltage applied to DC motors. The robot speed is measured by encoders coupled directly on the motor shaft instead of on the wheels, thus it avoids the gear reducer used to increase the motor torque. This configuration makes the encoder spin 48 times faster than the robot wheels, what increases considerably the encoder resolution.

The measured data is used by the odometry algorithm to compute the robot current position and to predict the position the robot will be one cycle later based on the current wheel speed setpoint applied to the PI, as detailed in the next subsection. This predicted position is sent back to the Raspberry PI, what closes a cycle. This cycle time is limited by the Raspberry Pi processor capacity and communication lags.

The communication between the Arduino board and the Raspberry Pi is sequential, characterizing an Universal Asynchronous Serial Communication (UART) with no parity, 8 bits packages and a baud rate of 115200 bits per second. For that, the standard Arduino library and the Scilab Serial Communication toolbox were used.

This methodology allows an autonomous and independent robot capable of achieving its targets without communicating with an external computer. Figure 2 shows the parts of the project as well as the platforms each one is implemented in.

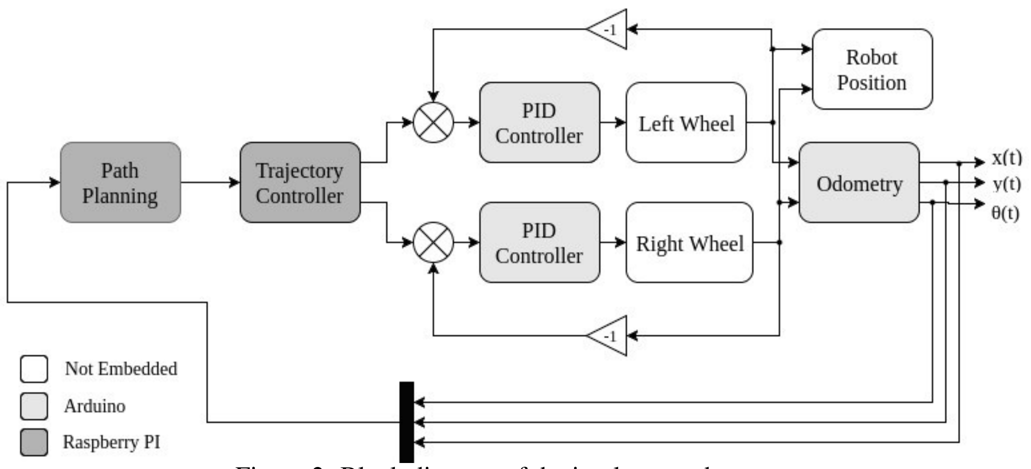


Figure 2: Block diagram of the implemented system

The experimental issues encountered have three main causes: use of an small resolution encoder disk coupled on the robot wheel, use of DC motors with a considerably dead zone, also, experiments made on floors with different coefficient of friction requires new PI parameters adjustment.

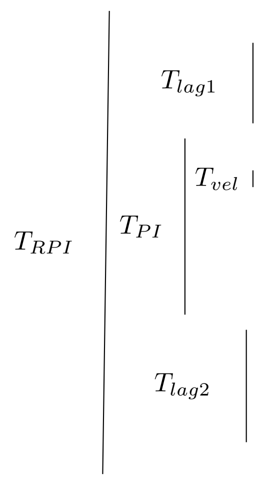
5.1 Arduino Algorithm

Figure 3 depicts the algorithm implemented in the Arduino Mega and its loop runtimes. The faster the loops, the better the robot's ability to identify obstacles and avoid them. However, the main loop must last enough to the Raspberry PI compute the new velocity setpoints and communicate with the Arduino.

One Raspberry PI loop runtime $T_{RPI} = 0.480s$ is enough to the PI loop $T_{PI} = 0.033s$ runs 13 times. The PI loop duration is defined based on two limitations. First, it has to last enough to count the encoder's pulses without meaningful loss of information. Second, it has to be fast enough to run several times and then make the PI transient response negligible.

The robot's current position is computed at each iteration of the PI loop using Eq.(2), where $R = 0.034\ m$, $2L = 0.147\ m$, $T = T_{PI}$ and φ_R and φ_L are the measured wheel speeds. This method ignores the time spent on serial communication and in other process, then, it increases the odometry systematic errors. A higher odometry precision is reached by also updating the robot current position at each main loop iteration considering $T = T_{RPI} - 13 * T_{PI} = T_{lag1} + T_{lag2}$, what represents the time that has not been included on each cycle. The PI setpoints are substituted on φ_R and φ_L because they represent well the average wheel velocities in most cases. However, if there are unexpected conditions, such as blocked wheels, the setpoints would not be a good approximation for the real robot movement, so a necessary condition to this update is the velocities being different from zero.

Considering $10rad/s$ as the maximum speed the robot's DC motors can achieve, it results in a $0.16m$ displacement in one Raspberry cycle ($0.480s$). That means when the robot receives the new speed setpoints they are already outdated. This delay can cause collisions with obstacles or make the robot keep moving after achieving the goal position. To compensate this lag, the APF input becomes the robot's next position which is predicted by adding to the robot's actual position a displacement relative to the current speed setpoints during $0.480s$ provided by the odometry.



```
while current position $\neq$ goal position do
    Compute estimated next position using odometry for $T_{RPI}$ based
on current speed setpoints
    Serial communication (send)
    for 13 times do
        Count pulses
        Compute velocity
        Apply PID control signal
        Update odometry based on measured speeds during $T_{PI}$
    end for
    Serial communication (receive)
    if velocity $\neq 0$ then
        Update odometry based on setpoints and $T_{lag1} + T_{lag2}$
    end if
end while
```

Figure 3: Arduino Algorithm. Left bars represent the duration of each loop

6. RESULTS AND DISCUSSIONS

Three experiments were performed in order to verify the effectiveness of the used methodology in making the robot avoid previously known obstacles and reach the goal position. For that, three obstacles are placed in such a way to produce a point with zero or very small resultant force in the robot's expected path when the vortex field is not considered. In Fig. 4, red marks represent obstacles at points $(0.73, 2.14)$, $(1.48, 1.44)$ and $(2.14, 0.73)$; the yellow mark represents the goal position $(2.5, 2.5)$ and the blue point the robot's initial position $(0, 0)$. The performed experiments are described below:

- Experiment 1: The main objective of Experiment 1 is to verify the robot's ability in completing the described task using only attractive and repulsive fields. So, the vortex field is not included in the APF algorithm. The trajectory controller is computed conform Eq.(12) and the PI and dometry algorithms are performed according to Fig. (3).

- Experiment 2: The main purpose of Experiment 2 is to analyze the vortex field influence. Then, the Experiment 1 is repeated including the vortex field, i.e., the robot operates with all capabilities proposed on this paper.

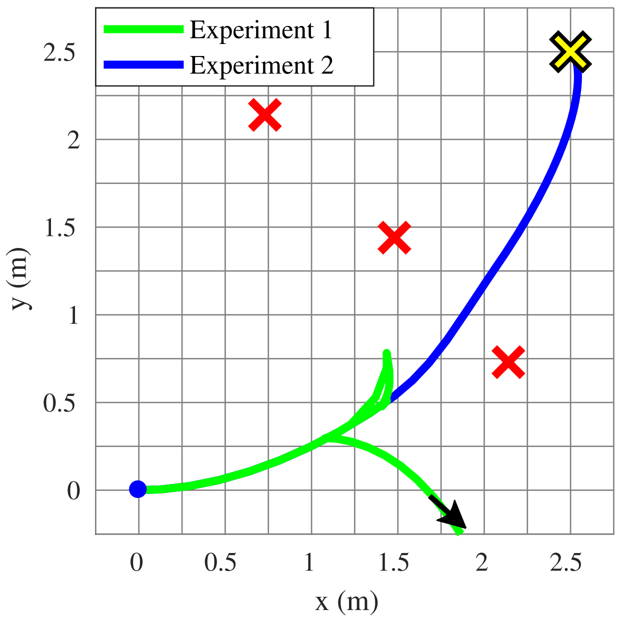
- Experiment 3: Experiment 3 aims at testify the proposed APF contributions. For that, the experiment 2 is repeated ignoring the estimated positions, i.e., the robot's current position works as APF input and the compensation of $T_{lag1} + T_{lag2}$ on the odometry computation is not included.

Experiments 1 and 2 are executed using simulations and real robot application, Experiment 3 is executed only to the real robot, since there is no real movement during processing time on simulations. Tab. 4 shows the APF, trajectory controller and PI parameters used. Note that K_p and K_i are the proportional and integral PI constants, and the subscripts $_R$ and $_L$ stand for right and left, respectively.

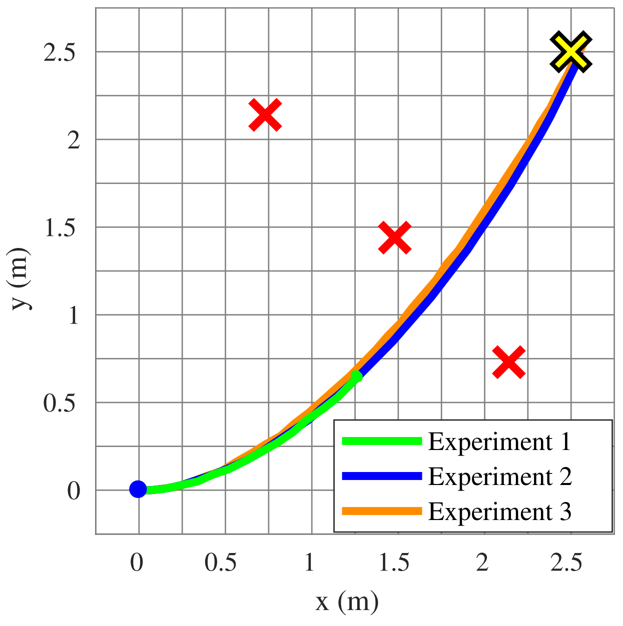
Table 1: **APF, Trajectory Controller and PI parameters**

ξ	γ	η	ρ_0 (m)	F_{max}	K_R	K_L	K_{pR}	K_{pL}	K_{IR}	K_{IL}
2	3	5	1	2.5	6	6	25.64	30.32	756.00	667.93

The robot's trajectories resulting from the three experiments are presented in: Fig. 4a, for simulation results and Fig. 4b for the odometry measurement of the real robot application. Green represents Experiment 1 (without vortex field), blue represents Experiment 2 (with all robot's features) and the orange one is the Experiment 3 (with no predicted movement). Figure 5 presents pictures composed by frames of the real robot executing those paths.



(a)



(b)

Figure 4: Robot's path on Experiments 1, 2 and 3. a) simulated, b) computed using real speed measurements. Yellow and red marks represent goal position and obstacles, respectively. The red arrow indicates the direction the robot keep moving.

It is possible to notice the green path does not succeed in getting to the goal position in both simulated and real experiments. The red arrow in Fig. 4a indicates the robot's trajectory continue in that direction, so it moves away from the goal position. In the real experiment, once the robot gets stuck at a low resultant potential field point, it becomes able to move only around its own axis. This considerable difference between the simulated path and the real robot path in the Experiment 1 is a result of the motor's dead zone, which does not allow robot movements in small velocities. Thus, if the DC motors are not able to follow the desired setpoints, the PI integral action increments the control effort until it moves. The motor response has overshoots and the motor's deadzones are different form each other what makes the robot settle in an unexpected position. The APF computes a force in the opposite direction to compensate that error, then the robot keep moving side to side, without completing the task.

After including the vortex field, the robot could contour the obstacles and complete its trajectory, as observed on Experiment 2 trajectory results. In addition, as shown in figures 6 and 7, the vortex field makes speed variations softer because distant obstacles do not have influence on the robot's trajectory. It also avoids low resultant fields that may set setpoints on the motor's deadzone, what makes the robot turn on and turn off several times. At each time the motor turn on, a current peak is required. Successive current peaks may decrease the battery life and make the battery discharge faster. Motor's deadzones are also responsible for not allowing the angular wheel speed to have a parabolic decay as simulated, instead there is a turbulent region. Note, the speed graphs on Figures 6b and 7b are filled with zero values to facilitate comparisons among them.

Figure 8 highlights the disparity between experiments 2 and 3 and includes a precision circle. That circle represents the area considered acceptable as the final position. This area is defined in the APF algorithm and is useful because it avoids the robot keep making extra maneuvers to get the perfect location. Comparing those results, it is possible to notice the contribution here proposed to the APF algorithm ensures the robot will remain inside the precision circle, so it decreases the error, i.e. the difference between the final robot position and the goal position. This difference could be crucial for applications that require a higher precision, such as robotic manipulators and robots used in environments as shop floor. Furthermore, in robots that include online detection of obstacles, it could avoid collisions with obstacles detected on the robot expected path. Lastly, for application with an external control loop, it could make the robot stabilize with errors because of the motor's dead zone.

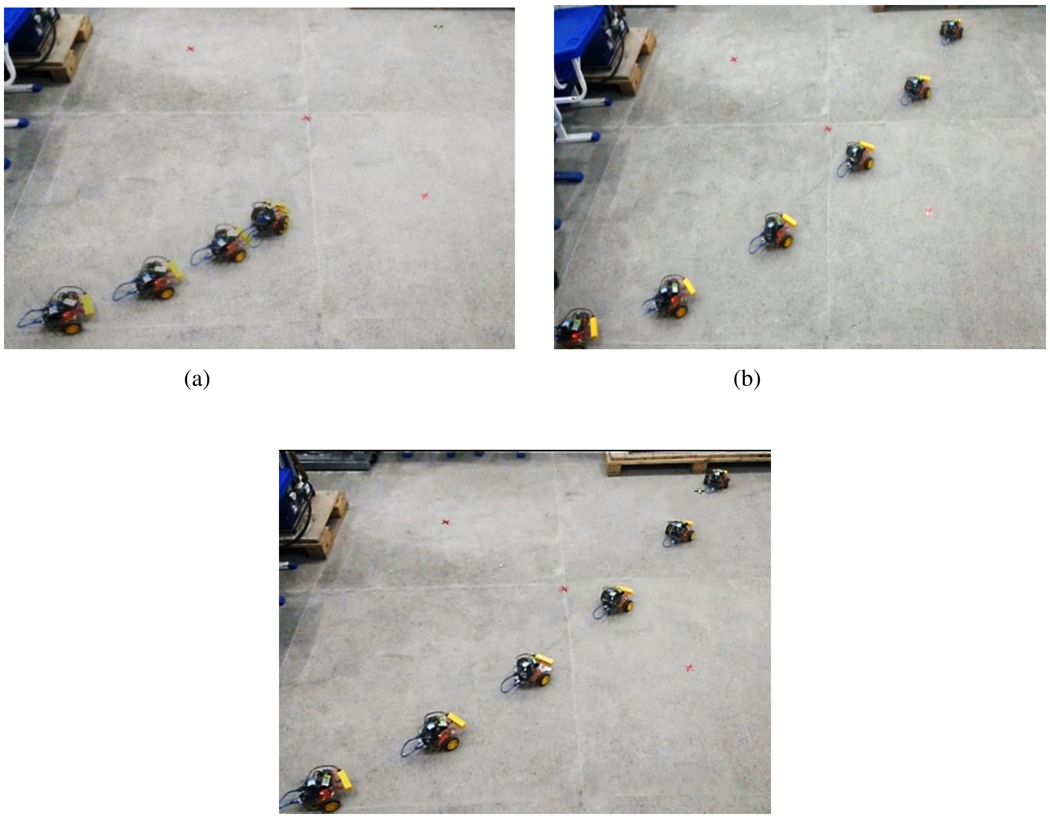
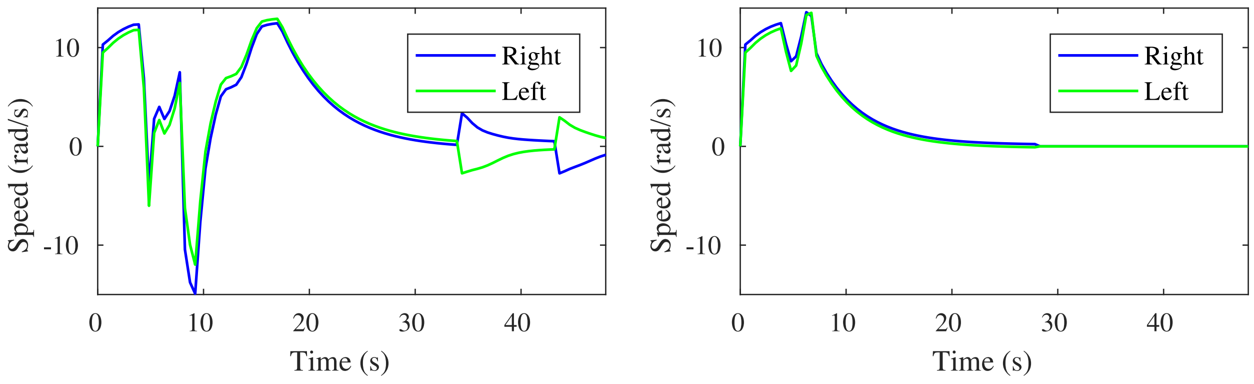
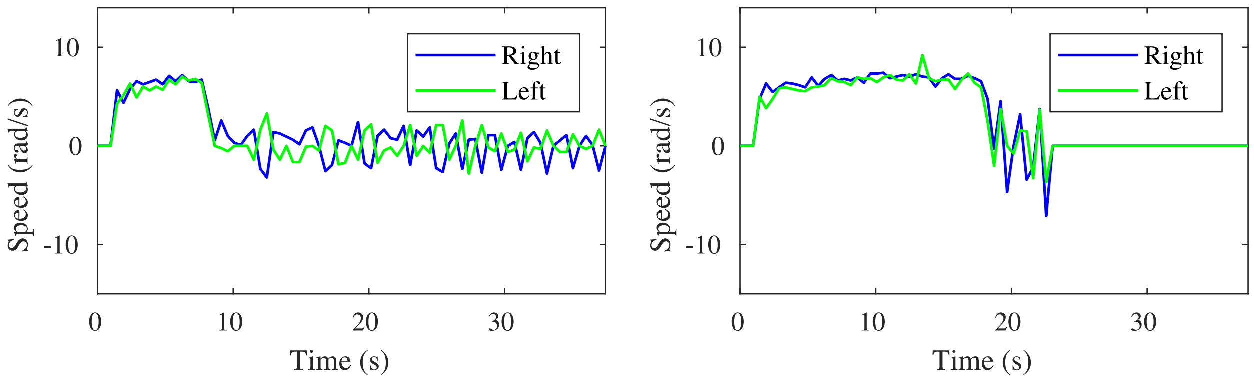

(a) (b) (c)

Figure 5: Frames of the real robot executing a) Experiment 1, b) Experiment 2 and c) Experiment 3.



(a) (b)

Figure 6: Simulated angular wheel speeds resulting from a) Experiment 1 b) Experiment 2.



(a) (b)

Figure 7: Measured angular wheel speeds resulting from a) Experiment 1 b) Experiment 2.

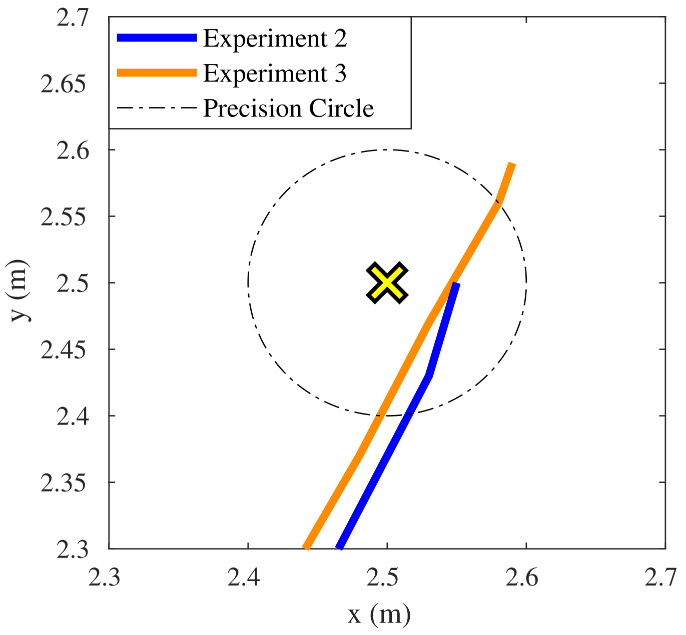


Figure 8: Goal position region of robot's path.

7. CONCLUSIONS

This work presented an autonomous nonholonomic differential drive mobile robot with embedded processing with path planning based on local incremental Artificial Potential Field method. Simulation and experimental results showed that the proposed methodology was effective to make the robot achieve the goal position while avoiding static and previously known obstacles. In addition, the considered vortex repulsive field allowed the robot to contour obstacles while avoiding low resultant potential fields. It also proposed a smoother set of speed setpoint. Furthermore, the proposed APF based on the estimated next position appears as a more suitable choice for the practical application because it compensates processing lags mainly caused by serial communication. Thus, its application leads to a smaller error. Future works include application of optimal controllers associated to APF and sensors to detect mobile and previously unknown obstacles.

8. ACKNOWLEDGEMENTS

This work was supported by FAPESB through project RED0004/2014, Aplicações de Controle em Sistemas Robóticos Não-Holonômicos

9. REFERENCES

Abbas, T., Arif, M. and Ahmed, W., 2006. "Measurement and correction of systematic odometry errors caused by kinematics imperfections in mobile robots". In *SICE-ICASE, 2006. International Joint Conference*. IEEE, pp. 2073–2078.

Abdalla, T.Y., Abed, A.A. and Ahmed, A.A., 2017. "Mobile robot navigation using pso-optimized fuzzy artificial potential field with fuzzy control". *Journal of Intelligent & Fuzzy Systems*, Vol. 32, No. 6, pp. 3893–3908.

Barraquand, J. and Latombe, J.C., 1991. "Robot motion planning: A distributed representation approach". *The International Journal of Robotics Research*, Vol. 10, No. 6, pp. 628–649.

Bemporad, A., De Luca, A. and Oriolo, G., 1996. "Local incremental planning for a car-like robot navigating among obstacles". In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*. IEEE, Vol. 2, pp. 1205–1211.

Borenstein, J. and Feng, L., 1996. "Measurement and correction of systematic odometry errors in mobile robots". *IEEE Transactions on robotics and automation*, Vol. 12, No. 6, pp. 869–880.

Chen, W., Wu, X. and Lu, Y., 2015. "An improved path planning method based on artificial potential field for a mobile robot". *Cybernetics and Information Technologies*, Vol. 15, No. 2, pp. 181–191.

Chong, K.S. and Kleeman, L., 1997. "Accurate odometry and error modelling for a mobile robot". In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*. IEEE, Vol. 4, pp. 2783–2788.

Corke, P., 2011. *Robotics, Vision and Control*. Springer Berlin Heidelberg.

De Luca, A. and Oriolo, G., 1994. "Local incremental planning for nonholonomic mobile robots". In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*. IEEE, pp. 104–110.

DeMedio, C. and Oriolo, G., 1991. "Robot obstacle avoidance using vortex fields". In *Advances in robot kinematics: with emphasis on symbolic computation*. Linz, Austria, pp. 227–235.

Dhaouadi, R. and Hatab, A.A., 2013. "Dynamic modelling of differential-drive mobile robots using lagrange and newton-euler methodologies: A unified framework". *Advances in Robotics & Automation*, Vol. 2, No. 2, pp. 1–7.

Khatib, O., 1986. "Real-time obstacle avoidance for manipulators and mobile robots". *The international journal of robotics research*, Vol. 5, No. 1, pp. 90–98.

Krogh, B.H., 1984. *A Generalized Potential Field Approach to Obstacle Avoidance Control*. Creative manufacturing engineering program. RI/SME.

Shi, P. and Zhao, Y., 2009. "An efficient path planning algorithm for mobile robot using improved potential field". In *Robotics and Biomimetics (ROBIO), 2009 IEEE International Conference on*. IEEE, pp. 1704–1708.

Tilove, R.B., 1990. "Local obstacle avoidance for mobile robots based on the method of artificial potentials". In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*. IEEE, pp. 566–571.

Tzafestas, S.G., 2013. *Introduction to mobile robot control*. Science Publishing Co Inc, 1st edition.

Vadakkepat, P., Tan, K.C. and Ming-Liang, W., 2000. "Evolutionary artificial potential fields and their application in real time robot path planning". In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*. IEEE, Vol. 1, pp. 256–263.

10. RESPONSABILIDADE AUTORAL

Os autores são os únicos responsáveis pelo conteúdo deste trabalho.