An Automaton and Super-Twisting Sliding-Mode Control for Wall Following

Edgar Martinez * **Rafael Murrieta-Cid**[1] ** **Hector M. Becerra** ***

* *Tecnológico Nacional de México / ITS de Guanajuato, Guanajuato, México (e-mail: emartinez@itesg.edu.mx).*
** *Centro de Investigación en Matemáticas (CIMAT), C.P 36023 Guanajuato, Mexico (e-mail: murrieta@cimat.mx)*
*** *Centro de Investigación en Matemáticas (CIMAT), C.P 36023 Guanajuato, Mexico (e-mail: hector.becerra@cimat.mx)*

[1]Corresponding Author: Rafael Murrieta-Cid

Abstract: This work proposes an approach for wall following with a differential drive robot in a polygonal environment using laser range finder measurements. We propose an automaton which manages the robot observations and selects the control set-points according to the state in the automaton. We also propose a super-twisting sliding-mode control which rapidly reaches the control set-points. The approach allows the robot to continuously move without the need to stop when obstacles get in the path of the robot. This approach allows the robot to follow the walls fast and robustly. The wall following capability can be used to explore an unknown environment or to find objects in the environment.

Keywords: Wall following, Automaton, Super-twisting Control, Nonholonomic constraints, Obstacles Avoidance

1. INTRODUCTION

This paper proposes an approach for wall following with a differential drive robot (DDR) in a polygonal environment. A DDR is a robot with two independent wheels, each wheel is controlled with a different motor. This system is nonholonomic (Laumond, 1998), which means that it has motion constraints, namely the system cannot move instantaneously perpendicular to the wheels direction. We propose to use an automaton which manages the robot observations and selects the control set-points of a single controller according to the state in the automaton. This is a novel use of automatons, since typically an automaton switches controllers (Martinez et. al, 2019; Toibero et. al, 2009, 2011). We introduce a super-twisting sliding-mode control which rapidly reaches the control set-points, this is due to the property of this type of control of finite time convergence (Levant and Fridman, 2002). The approach allows the robot to continuously move without the need to stop when corner obstacles get in the path of the robot. Thus, this approach allows the robot to follow the walls fast and robustly. The resulting wall following capability can be used in the task of exploring an unknown environment (Gonzalez and Latombe, 2002; Laguna et. al, 2014; Martinez et. al, 2019) or to find objects in the environment (Tovar et. al, 2007; Sarmiento et. al, 2009; Katsev et. al, 2011). The approach proposed in this work *avoids the estimation of the robot's state*, which is a typical strategy for robot's control. Our strategy provides a mapping that directly relate an observation to control inputs.

We use two discs for modelling the desired robot trajectories. See Fig. 1, where the black solid disc represents the physical robot. The disc centred in the robot centre is used as a virtual circular robot to track the boundary of the environment touching it or lying close to such boundary. Another larger disc is used to travel trajectories corresponding to arcs of circle on the boundary of the larger disc. The main idea of following the robot boundary is to make the robot to execute three motion primitives according to the structure of the environment boundary. If the robot is following a line segment, then the robot travels in straight line; if the robot is moving around a convex corner, then the robot executes a clockwise turn around the corner and; if robot encounters a concave corner, then the robot executes a counterclockwise turn.

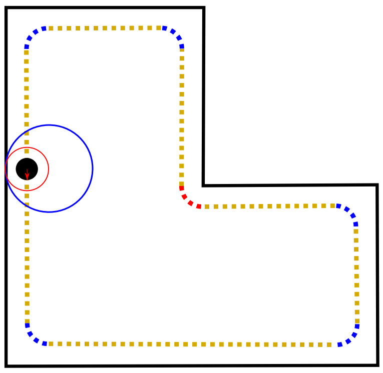


Fig. 1. Robot trajectory for following the environment boundary in continuous motion.

1.1 Related work

This work is about controlling a differential drive robot to follow walls. Several works exist to control a robot to follow a wall using information obtained with a range sensor (Bicho, 2000; Toibero et. al, 2009, 2011). The work in (Bicho, 2000) focuses on sensing to achieve a correct modelling of walls. The control scheme in (Toibero et. al, 2006, 2009) was proposed to avoid obstacles reactively following the border of the obstacles. The control method uses distance to the obstacles and odometry, the goal is avoiding saturation of the angular velocity originated by discontinuous contours in the environment. In the work presented in (De and Koditschek, 2013), the sensing and control of a robot are considered simultaneously for following walls. In (Lamperski et. al, 2005) a bio-inspired antennae is used for wall following. In (Juang and Hsu, 2009), the authors propose a reinforcement fuzzy learning technique for controlling wall following with a mobile robot. In (Pasteau et. al, 2013), the authors present an approach for navigation of a wheelchair using a single camera controlled with visual servo control. They focus on following a corridor where no prior information of the environment is needed.

Our work is also related to robot motion planning for avoiding collision with obstacles (Khatib, 1986; Borenstein and Koren, 1989; Minguez and Montano, 2004; Minguez et. al, 2008; Fraichard and Kuffner, 2012), and particularly with nonholonomic robots (Laumond et. al, 1994; Bicchi et. al, 1996; Hayet et. al , 2014; Cherubini and Chaumette, 2012; Agarwal et. al, 2012; Martinez et. al, 2019). In (Cherubini and Chaumette, 2012), the authors propose a method for collision avoidance with obstacles in a visual navigation task. Visual navigation corresponds to sense an ordered set of images. The visual navigation is solved by using image based visual servoing and to avoid collision the obstacles are detected with a laser. In (Reyes and Murrieta-Cid, 2019) an approach is proposed, which integrates image based visual servoing and planning for road following and to avoid collision with moving obstacles. One objective of that work is to represent a motion strategy of the robot with an automaton or finite state machine. This work is somehow related to the work in (Reyes and Murrieta-Cid, 2019) since an automaton is also used. However, in (Reyes and Murrieta-Cid, 2019) visual servo control is used to command the system while in this work the robot is commanded using a super-twisting sliding-mode controller. Furthermore, the main tasks are different, in this work the task is to follow the walls without colliding with them, while in (Reyes and Murrieta-Cid, 2019) the goal is to keep the robot within a road lane and avoid moving obstacles. In (Tovar et. al, 2007) an approach for exploring an environment and to navigate optimally in the sense of the Euclidean distance traveled by the system has been proposed. That method proposes a motion strategy based directly on measurements obtained with sensors, the motion strategy in (Tovar et. al, 2007) requires the capability of following the environment boundary. In (Katsev et. al, 2011), a wall following method is proposed to explore an environment with a point robot. That work proposes a data structure called cut ordering. Once the date structure is built the authors use it to deal with a pursuit/evasion problem. In (Laguna et. al, 2014), the authors proposed an exploration strategy for a differential drive robot and the motion strategy guarantees that the robot will discover as much as possible of the environment; the exploration strategy proposed in (Laguna et. al, 2014) is based on wall following. However, the works in (Tovar et. al, 2007; Katsev et. al, 2011; Laguna et. al, 2014) *did not provide* a control approach to accurately follow the walls, in this work we provide that control approach. The work presented in (Martinez et. al, 2019) deals with an exploration problem for a planar and polygonal environment, the theoretical conditions ensuring that the robot discovers the largest possible region of the environment are provided. The exploration strategy proposed in that work is based on a wall following navigation. A finite state machine is proposed, the machine filters spurious observations and activates feedback-based controllers. The control technique selects controllers according to observations obtained from sensors. This work improves the navigation component in (Martinez et. al, 2019) in the following manner. In (Martinez et. al, 2019) each time that the robot encounters a concave or convex corner the robot must stop, in contrast, in this work the robot continuously moves regardless whether or not it encounters corners while following the walls. Other important differences between the work in (Martinez et. al, 2019) and the work presented in this paper are described in detail in the next section.

1.2 Main contributions

Several approaches for navigation avoiding collision with obstacles and particularly for following walls exist, but to the best of our knowledge, the previous work most closely related to our approach is presented in (Martinez et. al, 2019). Nevertheless, it is important to stress that there are several differences between the work in (Martinez et. al, 2019) and this work, The main differences are the following:

(1) In this work the robot is commanded with a super-twisting sliding-mode controller while in (Martinez et. al, 2019) PD controllers were used. The super-twisting sliding-mode controller reaches the control set-points faster than the PD controllers, providing more reactivity to the robot.
(2) In (Martinez et. al, 2019), in each state of the finite state machine different controllers are used to generate the angular and linear velocity of the robot, in contrast, in the work presented in this paper a different approach is proposed, namely, a single controller is used to compute the linear and angular velocities in all the states in the automaton, only the control set-points are changed. This simplifies the system and makes easier to obtain smoother motions.
(3) The proposed control scheme allows the robot moving without stopping, keeping the continuity in the angular and linear velocities of the robot. Indeed, this is a novel aspect with respect to all existing wall following strategies.
(4) The time to travel the environment boundary is significantly reduced compared with the time needed in the approach presented in (Martinez et. al, 2019).

1.3 Problem statement

A differential drive robot equipped with an omnidirectional laser sensor used to find features in a polygonal and simply connected environment and avoid collision with the obstacles, is given the task of following the environment boundary traveling to desired nominal linear velocity even when obstacles get in the robot trajectory. To fulfill the assumption of availability of the omnidirectional sensor, in the experiments, this type of sensor is engineered using two laser range finders with limited field of view, oriented in opposite directions.

We also assume that the environment where the robot moves is a structured polygonal environment (such as an office), that is possible to be represented with line segments and corners which are modeled from points obtained with a laser range finder. This type of conditions appears frequently in indoors environments.

2. AUTOMATON

An automaton or finite-state machine (FSM) is a mathematical model of computation; it is an abstract machine that can be in one of a finite number of states (Hopcroft et. al, 2000). Fig. 2 shows a graphical representation of the automaton proposed in this work.

The automaton has 3 states: SL (straight line), CWT (clockwise turn) and CCWT (counterclockwise turn), these states correspond to the motion primitives that the robots executes, assuming, without loss of generality, that the boundary of the environment is to the right of the robot heading. The symmetric case is when the boundary of the environment is to the left of the robot heading. In that case, the sensor must be also placed to the left of the robot heading and all the modeling is equivalent for the symmetric case. The automaton is at state SL when the robot is following a line segment, the automaton is at state CCWT when the robot is executing a counterclockwise turn, this typically happens when the robot encounters a concave corner, and the automaton is at state CWT when the robot is executing a clockwise turn around a convex corner.

Sensor measurements obtained with the laser are used in two different ways: 1) They are used as feedback information in the controller of the robot's velocities (see Sections 2.5 and 3). 2) They are used to respond binary questions, which allows one to change or not from a state to another; those elemental decisions are called "observations", since they are directly based on sensor measurements. The observations are labeled by a y_i in Fig 2. They are described in detail in Section 2.4.

2.1 Robot's sensor and discs

The differential drive robot is modeled as a disc of radius r and it has a defined forward heading. The extrema right side robot's point is called rp. The robot is equipped with an omnidirectional laser range finder, this sensor is used to measure distances and angles to features in the environment, and it is located at point rp.

We use two discs for modelling the robot trajectories and helping defining the transitions between states in the

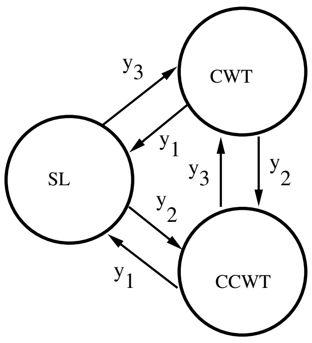


Fig. 2. Automaton.

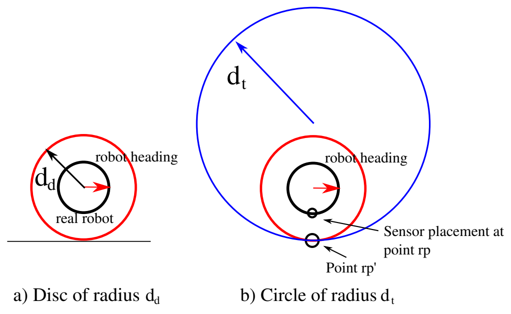


Fig. 3. Two discs.

automaton. Refer to Fig. 3. A disc centred at the robot centre, this disc has a radius $d_d > r$, the extrema right side point over this disc is called rp'. This disc of radius d_d is used as a virtual circular robot to track the boundary of the environment touching it or lying close to such boundary. Another larger disc is used to travel trajectories corresponding to arcs of circle on the boundary of this disc. This second disc has radius d_t (with $d_t > d_d$), it is centred at other point different to the robot centre but sharing point rp' with the disc of radius d_d. The line segment passing over points rp and rp' is called line $rp - rp'$. The control algorithm requires that the user defines the value of d_d and d_t, see below.

2.2 Feature detection

To detect convex and concave corners that delimit walls, a local and simple line fitting technique is used. First, the closest laser point from the laser sensor is detected. Then, the angles between the ray from the laser sensor to the closest point obstacle and the rays between the closest point and the next 10 sensed points, in counterclockwise sense, are measured. Those ten angles are averaged and the resulting angle is called reference angle. Then, the angle between the ray from the laser sensor to the closest point obstacle and the ray between the closest point and a given sensed point is computed, that angle is called angle of the point. Finally, the following conditions are used to detect corners. A concave corner is detected if the angle of the point is smaller than the reference angle plus a given threshold. Similarly, a convex corner is detected if this angle of the point is larger than the reference angle plus a given threshold. There are other well known algorithms, which fit lines based on points and it can be used for more complex environments (Gonzalez and Latombe, 2002; Press et. al, 1994).

2.3 Sensor measurements: angles and distance

Refer to Fig. 4. The ray that points to the closest point obstacle over the line segment that the robot is following is called r_{min}. Let d_1 be the smallest distance from the centre of the robot to the line segment that the robot follows. Let θ_1 be the angle from line $rp - rp'$ to the ray r_{min} (in counterclockwise sense). Angle θ_1 and distance d_1 are used, at state SL, as feedback information (see Section 2.5).

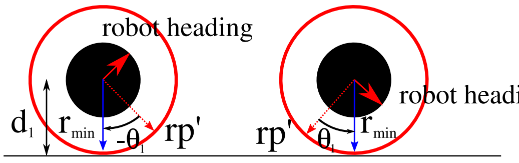


Fig. 4. Angle θ_1 and ray r_{min}.

Refer to Fig. 5. Let θ_2 be the angle between the line $rp - rp'$ and the ray d_{min} that points to the closest point in the last polygonal segment (in counterclockwise sense) of the boundary of the obstacle, that intersects the circle of radius d_t.

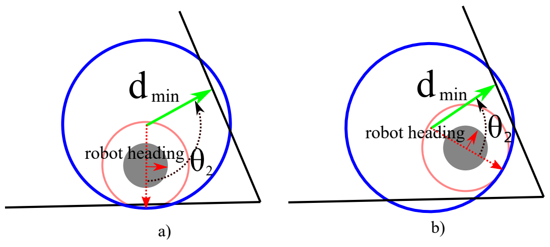


Fig. 5. Angle θ_2.

Refer to Fig. 6. The distance between the centre of the disc of radius $d_t = a + d_d$ and the concave corner is called h. At the moment when a bicontact between the disc of radius d_t and the obstacles border occurs, the distance from the centre of the robot to the concave corner (distance d_{corner}) is measured. Using distance d_t, distance d_{corner}, and the angle γ (which can also be measured with the laser), distance h is computed using the law of cosines. The first time that this distance h is computed, it is called h_0. Later during the robot motion the distance h is computed again at each iteration of the control cycle. Angle θ_2 and distance h are used as feedback information at state CCWT (see Section 2.5).

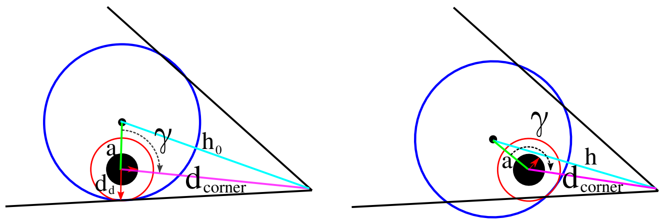


Fig. 6. Distance h

Refer to Fig. 7. The ray from the robot centre that points to the convex corner is called r_{corner}. Two distances are used to compute angle θ_3, the distance to an obstacle in the direction of the line $rp - rp'$, called d_w and the distance from the centre of robot to the corner d_{corner}. The cosines law is used to compute an angle called A, angle $\theta_3 = \frac{\pi}{2} - A$. Line $rp - rp'$ and ray r_{corner} are not necessarily colinear. Note that sometimes the robot aligns its heading to a virtual line segment (called s_v), see Figs. 7 a) and b). However, as the line passing over point rp and rp' gets perpendicular to that line segment, the correct segment will be sensed and considered, see Fig. 7 c). The motion terminates as soon as the angle θ_3 is smaller than a threshold ϵ_2, this is the same that having the robot heading aligned with the segment after the corner (in counterclockwise sense), see Fig. 7 d).

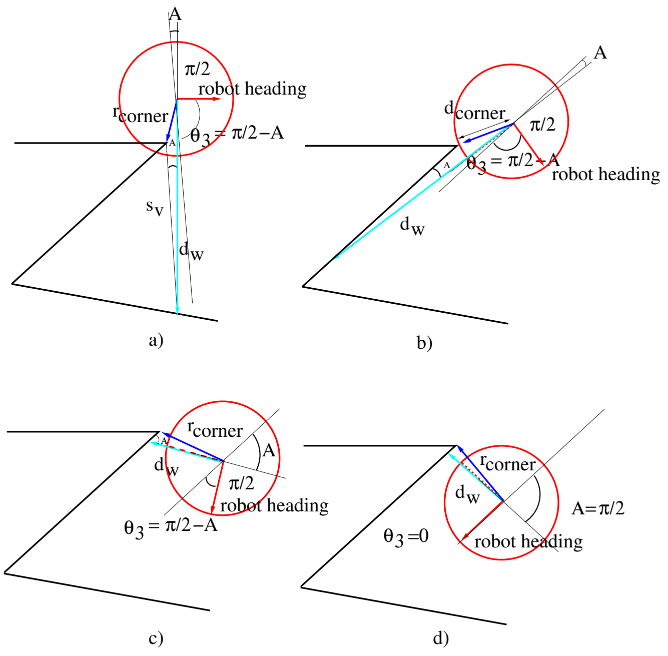


Fig. 7. Angle θ_3

If there exist obstacles points within the disc of radius d_t then angle θ_4 is calculated to determine what angle the robot has to rotate, to avoid a collision with the obstacles. An obstacle point is a point that does not belong to the segments sharing the convex corner.

To compute θ_4, two distances are used: d_{cc} and d_{cdo}, (refer to Fig. 8): d_{cc} is the distance from the centre of the disc of radius d_t to the convex corner, d_{cdo} is the distance from the centre of the disc of radius d_t to the closest point belonging to an obstacle. The cosines law is used for computing distance d_{co}, it is the distance between the convex corner and the point at distance d_{cdo} from the centre of the disc of radius d_t. To find the point that collides with the closest obstacle, from the centre of the disc of radius d_t, a circle of radius d_{co} centred at the convex corner is used. This circle and the circle of radius d_t are intersected. The intersection point that is closer to the closest obstacle, from the centre of the disc of radius d_t, is denoted point IC. θ_4 is the angle between the ray from the corner to the point IC and the ray from the corner to the point obstacle that lies closest to the centre of the disc of radius d_t. Fig. 8 a) shows the case when the obstacle is a corner and Fig. 8 b) the case when the obstacle is a segment. The angle that the robot has to rotate around the corner is $\min\{\theta_3, \theta_4\}$. Thus, the smallest angle between θ_3 and θ_4 is selected at every time instance. Angles θ_3 or θ_4 and distance d_{corner} are feedback information at state CWT (see Section 2.5).

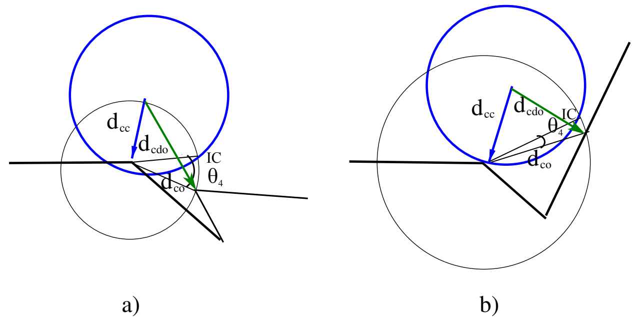


Fig. 8. Angle θ_4

2.4 Observations and bits of the observations

In this work, algebraic Boolean operations of "and" among several binary answers (bits values) trigger transitions between states in the automaton. These logic operations of "and" between different bits are called observations, since they are established directly based on sensors' measurements. The bits of the observations represent the answer: yes or not of a binary question. These answers are deduced from the laser measurements.

The 4 bits defining an observation are the following:

- rp': If there is a sensed point belonging to an obstacle point closer to rp' than a given threshold ϵ_1 then this bit is true, otherwise it is false. This bit is only relevant to transit to state CWT.
- bc: If two different circular sectors of the disc of radius d_t intersect the obstacle region then this bit is true, otherwise it is false. This bit is relevant to transit to states SL and CCWT.
- $rp'-e$: If a convex corner is closer to the point rp' than a given threshold ϵ_1 then this bit is true, otherwise it is false. This bit is only relevant to transit to state CWT.
- *aligned*: If $|\theta_1| < \epsilon_2$ or $|\theta_2| < \epsilon_2$ or $|\theta_3| < \epsilon_2$ or $|\theta_4| < \epsilon_2$ then this bit is true, otherwise it is false (the different angles are related to different states in the automaton). This bit is relevant to transit to the 3 states SL, CCWT and CWT.

Table 1 shows the 3 observations that establish the transitions between states in the finite state machine. In that table, X denotes "any binary value".

$y_i =$	$(rp'$,	bc,	$rp'-e$,	*aligned*)	STATE
$y_1 =$	(X,	0,	X,	1)	SL
$y_2 =$	(X,	1,	X,	0)	CCWT
$y_3 =$	(1	X	1,	0)	CWT

Table 1. Observations y_i.

Note that the proposed approach allows some approximations, 2 thresholds are used. Threshold ϵ_1 is the radius of the circle that models point rp' determining whether or not a convex corner touches point rp' or whether or not the robot is touching a wall with point rp'. Threshold ϵ_2 corresponds to an angular threshold determining whether or not the robot is aligned, the same threshold is used for the four angles: θ_1, θ_2, θ_3 and θ_4. The tuning of these parameters represents a compromise between precise control action and robustness against imperfect sensing measurements.

2.5 Control set-points according to the state in the automaton

The control law is in charge of carrying the robot orientation from an initial value θ_i to a final value θ_f. In this paper, the control set-point corresponding to the final orientation θ_f changes according to the state in the automaton. We have also an error related to a desired distance, the control set-point corresponding to a distance also changes according to the state in the automaton. Below we specify both the control set point related to the angle and the distance.

Traveling in straight line SL. In this state $\theta_f = \theta_1 = 0$ since we want that the robot is aligned with the line segment that is following.

Regarding the control set point related to the distance, in the state, one wants $d_1 = d_d$.

Counterclockwise turn CCWT. In this state, one wants to finish a counterclockwise robot turn CCWT to get the robot aligned with the last polygonal segment in counterclockwise sense (after a concave corner) of the obstacle region boundary that intersects the disc of radius d_t, thus: $\theta_f = \theta_2 = 0$. The control set point related to the distance is $h = h_0$.

Clockwise turn CWT. In this state, the robot's task is to finish a clockwise robot turn CWT around a convex corner to get the robot aligned with the line segment after (in clockwise sense) that corner, thus: $\theta_f = \min\{\theta_3, \theta_4\} = 0$. In this state, one wants $d_{corner} = d_d$.

Table 2 summarizes the important bits in the observations that change from a state to other in the automaton, and the control set points for each state.

3. CONTROL SCHEME

In this section, we will describe the proposed control scheme, yielding the linear and angular velocities, respectively called v and ω.

3.1 Control of linear velocity

The linear velocity of the robot is give by the following equation:

$$v = \frac{v_n}{2}(1 + \tanh(\alpha(t - \beta))) \quad (1)$$

β is a time shifting and α is a scale factor. These parameters establish the duration of the transition from being motionless to v_n, the desired nominal robot's velocity. The parameters α and β are calculated such that the maximum robot acceleration is not exceeded. The velocity profile in Eqn. 1 allows one to increase smoothly the robot velocity from 0 to the nominal robot velocity v_n.

3.2 Control of angular velocity

The angular velocity is used to correct deviations from the control set-points in both angles and distances.

A super-twisting sliding-mode controller as in (Rivera et. al, 2011) is used to obtain the angular velocity of the robot.

Transition between states	Control set points	Init. state key bits	Final state key bits
SL $\rightarrow$ CCWT	$\theta_1 = 0,\ d_1 = d_d$	$bc = 0,\ aligned = 1$	$bc = 1,\ aligned = 0$
SL $\rightarrow$ CWT	$\theta_1 = 0,\ d_1 = d_d$	$bc = 0,\ aligned = 1$	$rp' = 1,\ rp' - e = 1,\ aligned = 0$
CCWT $\rightarrow$ SL	$\theta_2 = 0,\ h = h_0$	$bc = 1,\ aligned = 0$	$bc = 0,\ aligned = 1$
CCWT $\rightarrow$ CWT	$\theta_2 = 0,\ h = h_0$	$bc = 1,\ aligned = 0$	$rp' = 1,\ rp' - e = 1,\ aligned = 0$
CWT $\rightarrow$ SL	$\min\{\theta_3, \theta_4\} = 0,\ d_{corner} = d_d$	$rp' = 1,\ rp' - e = 1,\ aligned = 0$	$bc = 0,\ aligned = 1$
CWT $\rightarrow$ CCWT	$\min\{\theta_3, \theta_4\} = 0,\ d_{corner} = d_d$	$rp' = 1,\ rp' - e = 1,\ aligned = 0$	$bc = 1,\ aligned = 0$

Table 2. Transitions between states, bits that determine the transition and control set points for each state.

The advantage of this type of control is its convergence property in finite-time of the error to the sliding surface, which rapidly reaches the control set-points (Moreno and Osorio, 2012; Utkin et. al, 2013). In this work, the sliding surface is a linear combination of the distance and angles set points that must be driven to zero. The angular robot velocity ω_s is calculated as follows:

$$\omega_s = -k_4 |s|^{\frac{1}{2}} \text{sgn}(s) + \sigma \qquad (2)$$

where

$$\text{sgn}(s) = \begin{cases} -1 & if\ \ s < 0 \\ 1 & if\ \ s \geq 0 \end{cases} \qquad (3)$$

$\dot{\sigma}$ is given by:

$$\dot{\sigma} = -k_3 \text{sgn}(s) \qquad (4)$$

and the sliding surface s is given by the following equation:

$$s = k_1 e_d + k_2 e_\theta \qquad (5)$$

k_1, k_2, k_3 and k_4 are control gains.

Finally, the control errors e_θ and e_d related respectively to the robot orientation and the distance between the robot and the obstacles are given by the two following equations:

$$e_\theta = \begin{cases} \theta_1 & if\ \ \text{SL} \\ \theta_d - \theta_2 & if\ \ \text{CCWT} \\ \theta_d - \min\{\theta_3, \theta_4\} & if\ \ \text{CWT} \end{cases} \qquad (6)$$

$$e_d = \begin{cases} d_d - d_1 & if\ \ \text{SL} \\ h_0 - h & if\ \ \text{CCWT} \\ d_d - d_{corner} & if\ \ \text{CWT} \end{cases} \qquad (7)$$

where θ_d follows a profile to avoid undesired discontinuities in the controls. This profile is described in the next section.

3.3 Changing smoothly the task specifications

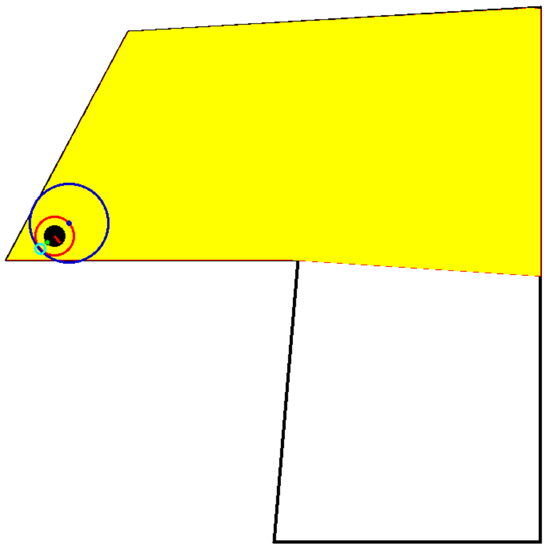
From a state to another, the desired orientation of the robot w.r.t a local robot's reference frame might change abruptly, then the error in the orientation might also change from a small to a large value. This will generate undesired jumps in the controls. To avoid this discontinuities in the control errors, we shall enforce that the desired θ_d varies smoothly from an initial θ_i to a final $\theta_f = 0$ value in a predetermined time (τ). This profile is given by the next equation:

$$\theta_d = \frac{\theta_{i\in\{2,3,4\}|t=0}}{2} \left(1 + \cos\left(\frac{\pi t}{\tau}\right)\right) \qquad (8)$$

4. SIMULATIONS AND EXPERIMENTS IN A REAL ROBOT

4.1 Simulation results

Our simulation software run on a 2.2GHz Intel Core i7-2670QM quad-core processor PC, equipped with 8 GB of RAM, executing Linux, and are programmed in C++ using the computational geometry library LEDA (Zaroliagis , 2008), which provides implementations of several algorithms for graph theory and computational geometry.


(a) Robot is executing a counterclockwise turn at a concave corner

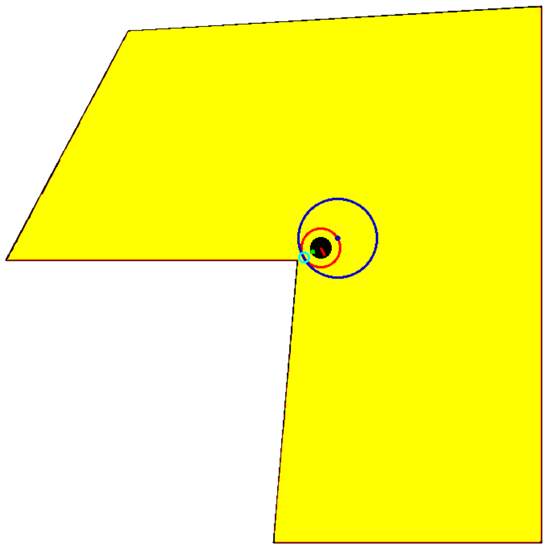

(b) Robot is executing a clockwise turn around a convex corner

Fig. 9. Simulations: The real robot is depicted with a black disc, a red circle represents a virtual robot that is moving in contact with the environment boundary. The blue circle represents the trajectories that the robot executes when it encounters a concave corner.

A laser range finder is simulated to obtain sensed points. Based on these points a current local representation (local map) of the environment is obtained, a very simple line fitting technique (see Section 2.2) is used to obtain the segments and corners in the environment and measure the distance and orientation from these features. Those distances and angles are used as feedback information in the robot's controller.

The line fitting technique assumes that the environment is such that is possible to easily detect their segments and corners. However, for more complex environments, other well known algorithms exist to fit lines based on points (see for instance (Press et. al, 1994).)

Additionally, in our software implementation, both in the numerical simulation and in the experiments in the physical robot, we have emulated the finite state machine presented in Fig. 2, that determines the control set-points of the robot's controller.

Fig. 9 shows two snapshots of a simulation experiment. In that figure the line segments delimiting the environment are shown in black, the real robot is depicted with a black disc, its heading is shown with a red arrow, a red circle represents a virtual robot that is moving in contact with the environment boundary, the blue circle represents the trajectories that the robot executes when it encounters a concave corner. The green point represents the robot's sensor that is placed at point rp. A cyan circle represents distance's tolerance of the point rp', if there is an obstacle inside this circle then it is said that the point rp' is touching an obstacle. The yellow region represents the robot's visibility region. The dashed red lines depict distance discontinuities generated by convex corners. Fig. 9(a) shows the robot executing a counterclockwise turn at a concave corner, and Fig. 9(b) shows the robot executing a clockwise turn around a convex corner.

4.2 Experiments with a physical robot

In all the experiments, we used a robot Pioneer P3-DX, which is a differential drive system. A disc of radius 0.2 m is used to model the robot; this robot has a maximum translational velocity of 1.2 m/s and a maximum rotational velocity of 5.236 rad/s. The nominal linear velocity v_n, used in the experiments, was set to 0.35 m/s. The distance d_d between the environment's boundary and the robot's centre was set to 0.4 m. Hence, the distance between the environment and the robot's boundary is regulated to 0.2 m.

Our algorithms run directly on the robot computer in all the experiments; this computer is a Pentium M at 1.8 GHz with 1 GB of RAM. Linux is the operating system and some ROS functionalities are used, the control cycle runs to 12.5 Hz. The software is programmed in C++. The omnidirectional sensor is engineered using two laser range finders Hokuyo model URG-04LX, those lasers are mounted on the robot in opposite directions, see Fig. 10.

The used control gains are tuned experimentally, one starts with small positive gains and progressively the values are increased to obtain a faster converge of the errors to zero. This process is repeated while no large oscillation in the robot's trajectory appears.

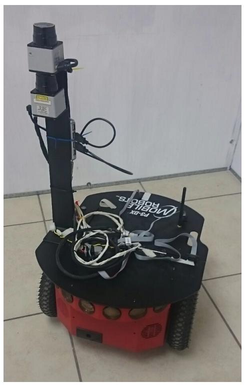
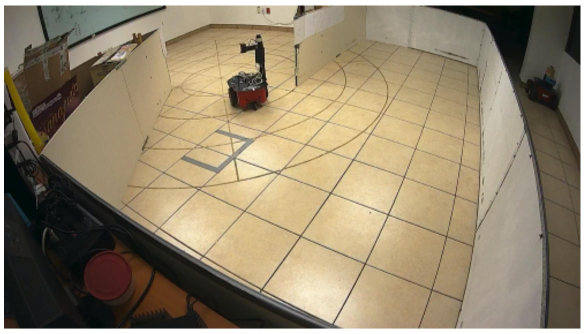
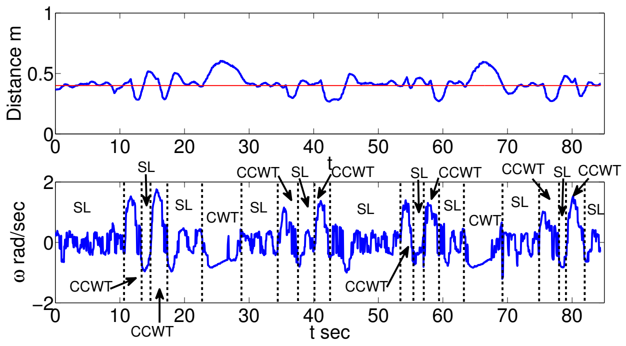


Fig. 10. The robot and the lasers.

Two different environments were used to perform the experiments: a robotics lab and an office, see Figs. 11 and 12. First, we describe the results in the robotics lab.



(a) Robot turning around a convex corner

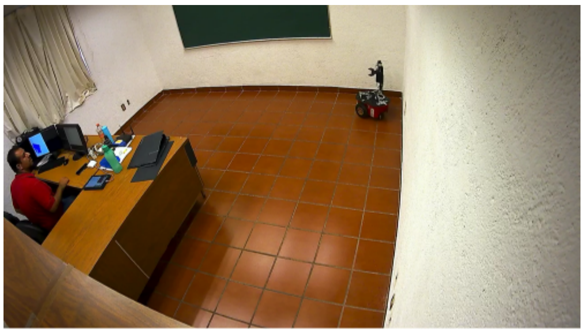


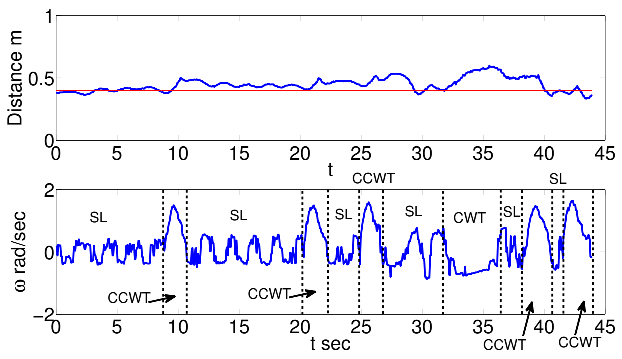
(b) Distance to the wall (top) and angular velocity (bottom)

Fig. 11. Experiment in a robotics lab.

Experiment in a laboratory. Table 3 shows the statistics over 3 laps in a robotics laboratory. The perimeter of lab is approximately 27.4 meters. The method for wall following proposed in (Martinez et. al, 2019) is compared with the method proposed in this paper. The reported performance metrics are: the average distance between the robot centre and the environment boundary, the corresponding standard deviation and the maximum and minimum distance to the wall per lap. All these values are given in meters. Finally, it is reported the time per lap in seconds. All the performance metric are almost the same for both methods with the exception of the time needed to complete a lap. The average time required to complete a lap is 162.136 sec for the method in (Martinez et. al, 2019). In contrast, the average time per lap obtained with the method reported in this work is 83.62 sec, this represents a reduction of approximately 78.5 sec., that is a reduction of 48.5%. This reduction is explained because in the method presented in (Martinez et. al, 2019) the robot totally stops when it reaches a convex or concave corner while in the method proposed in this work, the robot does not stop when it encounters a corner.

Fig. 11(a) shows the robot turning around a convex corner, Fig. 11(b) (top) shows the distance between the centre of the robot and the wall while the robot is following the environment boundary. Fig. 11(b) (bottom) shows the robot's angular velocity, the evolution of states in the automaton as time elapses are labeled in the figure. One can observe that discontinuities in the angular velocity do not happen.


(a) Robot executing a counterclockwise turn at a concave corner



(b) Changing to right lane

Fig. 12. Experiment in an office

Experiment in an office. Table 4 shows the statistics over 3 laps in an office. The perimeter of the office is approximately 16 meters. Again the method for wall following proposed in (Martinez et. al, 2019) is compared with the method proposed in this paper. The reported performance metrics are the same that in the previous experiment. Again, all the performance metric are almost the same for both methods with the exception of the time to complete a lap. The average time to complete a lap is 59.616 sec with the method reported in (Martinez et. al, 2019). The method proposed in this paper yields an average time per lap of 43.109 sec; this is a reduction of approximately 16.5 sec. equivalent to 27.7%

Fig. 12(a) shows the robot executing a counterclockwise turn at a concave corner. Fig. 12(b) (top) shows the distance between the centre of the robot and the wall while the robot is following the environment boundary. Fig. 12(b) (bottom) shows the robot's angular velocity. Again discontinuities in the angular velocity do not appear.

In the experiments, we observed that spurious corner detection sometimes instantaneously appeared. To alleviate this problem, a way is to filter the raw data. However, this issue has never prevented the robot to terminate the task. Nevertheless, for dealing with more difficult environments, our implementation of the corners detection must be improved either using robust line fitting methods to detect convex corners (Press et. al, 1994) or well known filtering techniques (Gonzalez and Latombe, 2002) on the raw laser data, making it more robust.

Based on the experiments, one can conclude that the approach proposed in this paper allows the robot to continuously move without the need to stop when obstacles get in the path of the robot. Our experiments show that the robot's velocities did not present discontinuities, and that the approach proposed in this paper allows the robot to use a single controller in all the states in the automaton, which makes the implementation and the tuning of the controller gains easier. This approach also allows the robot to circumnavigate faster the environment compared with the method proposed in (Martinez et. al, 2019).

5. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed an approach for wall following with a nonholonomic differential drive robot in a polygonal environment. An automaton manages the robot observations and select the control set-points according to the state in the automaton. We have also proposed a super-twisting sliding-mode control which rapidly reaches the control set-points. In all the states in the automaton the same controller is used to compute the linear and angular velocities, only the control set-points are changed. The approach allows the robot to continuously move without the need to stop when obstacles get in the path of the robot. This approach allows the robot to follow the walls fast and robustly. The wall following capability can be used to explore an unknown environment or to find objects in the environment. The control law and the algorithms to get the observations have been implemented; simulation results and experiments with a physical robot are included and discussed.

As a future work, we would like to explore another control strategy, in which the robot reduces the linear velocity when it approaches a corner without totally stopping the robot. We think that this strategy can further reduce the time to complete a lap.

ACKNOWLEDGEMENTS

This work was supported in part by CIMAT.

REFERENCES

P. K. Agarwal, T. Biedl, S. Lazard, S. Robbins, S. Suri and S. Whitesides. Curvature-Constrained Shortest Paths in a Convex Polygon. *SIAM J. Comput.*, 31(6):1814-1851, 2012.

Experiments in a robotics lab.					
The robot stops at the corners, method proposed in (Martinez et. al, 2019)					
Lap Number	Average Distance [m]	Std Dev [m]	Max Distance [m]	Min Distance [m]	Lap Time [sec]
1	0.41	0.066	0.813	0.285	163.488
2	0.418	0.076	0.811	0.271	162.032
3	0.412	0.064	0.81	0.292	160.888
Method proposed in this paper					
Lap Number	Average Distance [m]	Std Dev [m]	Max Distance [m]	Min Distance [m]	Lap Time [sec]
1	0.416	0.07	0.604	0.267	84.42
2	0.436	0.078	0.665	0.238	82.92
3	0.412	0.08	0.632	0.234	83.52

Table 3. Comparison of statistics of the method proposed in (Martinez et. al, 2019) vs the method proposed in this paper, for the experiment in a robotics lab.

Experiments in an office					
The robot stops at the corners, method proposed in (Martinez et. al, 2019)					
Lap Number	Average Distance [m]	Std Dev [m]	Max Distance [m]	Min Distance [m]	Lap Time [sec]
1	0.364	0.034	0.447	0.291	59.04
2	0.359	0.038	0.441	0.293	58.368
3	0.358	0.041	0.431	0.27	61.44
Method proposed in this paper					
Lap Number	Average Distance [m]	Std Dev [m]	Max Distance [m]	Min Distance [m]	Lap Time [sec]
1	0.43	0.06	0.598	0.293	43.102
2	0.444	0.073	0.69	0.286	42.568
3	0.463	0.109	0.818	0.251	43.658

Table 4. Comparison of statistics of the method proposed in (Martinez et. al, 2019) vs the method proposed in this paper, for the experiment in an office.

A. Bicchi, G. Casalino, and C. Santilli. Planning shortest bounded-curvature paths for a class of nonholonomic vehicles among obstacles. *J. of Intelligent Robots Systems*, 16(4):387–405, 1996.

E. Bicho. Detecting, representing and following walls based on low-level distance sensors. In *Proc. of the Int. Symposium on Neural Computation*, Berlin, Germany, 2000.

J. Borenstein and Y. Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5):1179–1187, 1989.

A. Cherubini and F. Chaumette, Visual Navigation of a Mobile Robot with Laser-based Collision Avoidance, *The International Journal of Robotics Research*, 32(2):189-205, Sep. 2012.

A. De and D. E. Koditschek. Toward dynamical sensor management for reactive wall-following. In *Proc. of IEEE Int. Conf. on Robotics and Automation, ICRA 2013*, pages 2400–2406, Karlsruhe, Germany, 2013.

T. Fraichard, and J. Kuffner Jr., Guaranteeing motion safety for robots. *Auton. Robots* 32(3): 173-175 2012.

H. González-Banos and J.-C. Latombe. Navigation strategies for exploring indoor environments. *International Journal of Robotics Research*, 21(10-11):829–848, 2002.

J.-B.. Hayet, H. Carlos, C. Esteves, and R. Murrieta-Cid. Motion planning for maintaining landmarks visibility with a differential drive robot. *Robotics and Autonomous Systems*, 4(62):456–473, 2014.

J. Hopcroft, R. Motwani, and J. Ullman. Introduction to Automata Th.eory, Languages, and Computation. *Pearson Education*, 2000.

C.-F. Juang and C.-H. Hsu. Reinforcement Ant Optimized Fuzzy Controller for Mobile-Robot Wall-Following Control. *IEEE Transactions on Industrial Electronics* 56(10):3931-3940, 2009.

O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90–98, 1986.

M. Katsev, A. Yershova, B. Tovar, R. Ghrist, and S. M. LaValle. Mapping and pursuit-evasion strategies for a simple wall-following robot. *IEEE Transactions on Robotics*, 27(1):113–128, 2011.

G. Laguna, R. Murrieta-Cid, H.M. Becerra, R. Lopez-Padilla, and S.M. LaValle. Exploration of an unknown environment with a differential drive disc robot. In *Proc of IEEE Int. Conf. on Robotics and Automation*, pages 2527–2533, 2014.

A. G. Lamperski, O. Y. Loh, B. L. Kutscher, and N. J. Cowan. Dynamical wall following for a wheeled robot using a passive tactile sensor. In *Proc. of IEEE Int. Conf. on Robotics and Automation, ICRA 2005*, pages 3838–3843, Barcelona, Spain, 2005.

J.-P. Laumond, P.E. Jacobs, M. Taïx, and R.M. Murray. A motion planner for nonholonomic mobile robots. *IEEE Trans. on Robotics and Automation*, 10(5):577–593, 1994.

J.-P. Laumond, Robot Motion Planning and Control, *Springer*, 1998.

A. Levant, L. Fridman. Higher order sliding modess, W. Perruqueti, J. Barbot (Eds.). *Sliding Mode Control in Engineering*, pp. 53-101, Marcel Dekker, NY, USA 2002.

E. Martinez, G. Laguna, R. Murrieta-Cid, H. M. Becerra, R. Lopez-Padilla and S. M. LaValle, A Motion Strategy for Exploration Driven by an Automaton Activating Feedback-based Controllers, *Autonomous Robots*, pp. 1-25, 2019.

J. Minguez and L. Montano. Nearness diagram (nd) navigation: collision avoidance in troublesome scenarios. *IEEE Transactions on Robotics and Automation*, 20(1):45–59, 2004.

J. Minguez, F. Lamiraux and J. P. Laumond. Motion planning and obstacle avoidance. In Siciliano B and Khatib O (eds.), *Springer Handbook of Robotics.* Berlin: Springer, pp. 827-852, 2008.

J. A. Moreno, M. Osorio, Strict Lyapunov functions for the super-twisting algorithm. *IEEE Trans. Autom. Control*, vol. 57, no. 4, pp. 1035-1040, Apr. 2012.

F. Pasteau, M. Babel, R. Sekkal. Corridor following wheelchair by visual servoing. *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2013*, pages 590-595, Tokyo, Japan, 2013.

W. H. Press, S. A. Teukolsky, W. T. Vettering, and B. P. Flannery. Numerical Recipes in C. *Cambridge University Press*, 1994.

R. Reyes and R. Murrieta-Cid, An Approach Integrating Planning and Image Based Visual Servo Control for Road Following and Moving Obstacles Avoidance. *International Journal of Control*, pp. 1-15, 2019.

J. Rivera, L. Garcia, Ch. Mora, J. Raygoza and S. Ortega. Super-Twisting Sliding Mode in Motion Control Systems. *Sliding Mode Control*, Chap. 13, InTech, 2011.

A. Sarmiento, R. Murrieta-Cid, and S. Hutchinson. An efficient motion strategy to compute expected-time locally optimal continuous search paths in known environments. *Advanced Robotics*, 23(12-13):1533–1560, 2009.

M. Toibero, R. Carelli and B. Kuchen. Stable Switching Contour-Following Controller for Wheeled Mobile Robots. In *Proc. of IEEE Int. Conf. on Robotics and Automation, ICRA 2005*, pages 3724–3729, Orlando, Florida, 2006.

M. Toibero, F. Roberti, and R. Carelli. Stable contour-following control of wheeled mobile robots. *Robotica*, 27(1):1–12, 2009.

M. Toibero, F. Roberti, R. Carelli, and P. Fiorini. Switching control approach for stable navigation of mobile robots in unknown environments. *Robotics and Computer-Integrated Manufacturing*, 27(2):558–568, 2011.

B. Tovar, R. Murrieta-Cid, and S. M. LaValle. Distance-optimal navigation in an unknown environment without sensing distances. *IEEE Transactions on Robotics*, 23(3):506–518, 2007.

V. Utkin. On convergence time and disturbance rejection of super-twisting control. *IEEE Trans. Autom. Control*, vol. 58, no. 8, pp. 2013-2017, Aug. 2013.

C. Zaroliagis. LEDA: a Library of Efficient Algorithms. In: Kao, MY. (eds) *Encyclopedia of Algorithms.* Springer, Boston, MA. 2008.