International Journal of Advanced Robotic Systems

Reactive Robot Navigation Utilizing Nonlinear Control

Regular Paper

Lei Ting[1], Chris J.B. Macnab[1,*] and Sebastian Magierowski[2]

1 University of Calgary, Calgary, Canada
2 York University, Toronto, Canada
* Corresponding author E-mail: cmacnab@ucalgary.ca

Received 30 Aug 2013; Accepted 22 Apr 2014

DOI: 10.5772/58705

Abstract In this paper, we propose a computationally efficient heuristic solution to choosing a path around obstacles in the face of limited sensor information. Specifically, we propose a navigation algorithm for a mobile robot that reaches a measured target position while avoiding obstacles, making decisions in real-time (without stopping) and relying strictly on information obtained from limited and noisy robot-mounted sensors to determine the location and severity of obstacles. The solution utilizes fuzzy processing to encode the environment - the fuzzy encoding is used both in deciding on an intermediate target direction and in a collision-avoidance strategy. A closed-loop nonlinear feedback control provides a smooth motion with stability guarantees. Simulation results in a corridor environment demonstrate expected collision-free trajectories.

Keywords Mobile Robot Navigation, Reactive Control, BEAM Robotics, Lyapunov Backstepping Control

1. Introduction

Unlike classic mathematical methods, heuristic algorithms often produce practical - although sub-optimal - solutions to mobile robot navigation problems in real-time. Masehian and Sedighizadeh's survey [1] shows that the proportion of heuristic methods to classic methods has been approaching unity for the past 30 years. However, even heuristic methods often rely on accurate sensor information and a large amount of computational power.

Different architectures for reactive mobile-robot navigation include fuzzy-logic, feedback control and BEAM (for Biology, Electronics, Aesthetics and Mechanics) strategies. The navigation system proposed here combines these three methods in a novel fashion, causing the robot to go around obstacles in a way that appears insect-like; it reacts to obstacles that it encounters along the way to the target rather than planning a global or optimal path. The fuzzy encoding of the environment includes the target, currently-faced obstacles and recently-passed obstacles. From this information, a fuzzy inference produces an intermediate target position. The closed-loop nonlinear feedback control, with a short-term memory element, uses the error between the current position and an intermediate target to produce commanded translational and rotational velocities. A BEAM-style control signal, referred to here as a 'virtual force', gets added directly to the commanded velocities as if it were a disturbance in the feedback loop. In this case, it is introduced as a deliberate and helpful disturbance. The virtual control serves as a collision-avoidance system to ensure that the robot does not brush against obstacles.

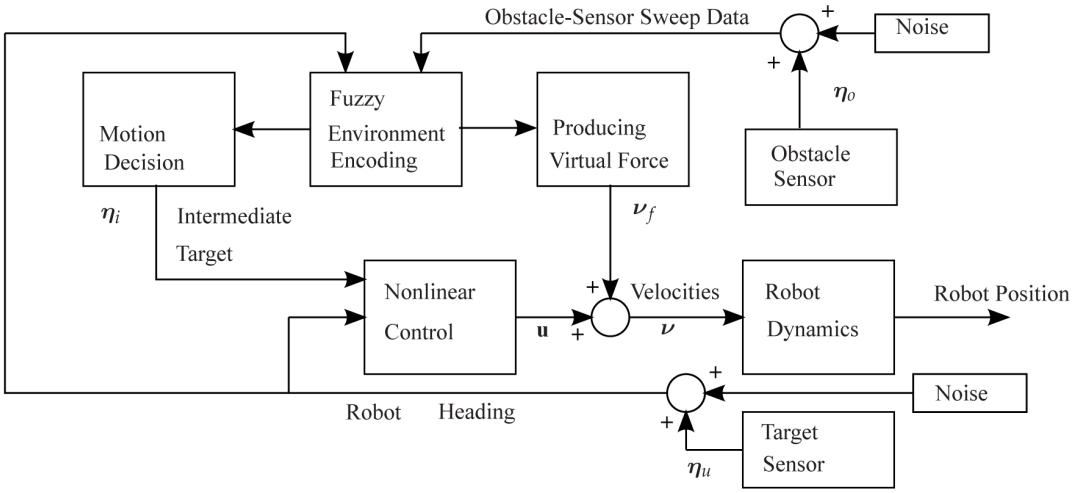


Figure 1. Navigation strategy

The very simplest control strategy stems from the concept of BEAM. Many hobbyists and students have followed this approach when building for fun or for robot games competitions, and the basic principles are outlined in [2]. Under this design approach, one connects the sensors directly to the actuators, which bypasses the need for processing the information and results in purely reactive motions. This strategy has proved very robust and reliable for simple movements, especially for avoiding imminent collisions. However, one normally desires more efficient and directed motions in real applications.

In order to avoid obstacles in an efficient - although sub-optimal - manner while seeking a target, a *reactive* control strategy appears to make the most sense. A reactive strategy does not rely on formal path planning algorithms requiring precise and/or global knowledge of the environment. Many researchers have designed fuzzy-rule bases for choosing paths based on obstacle distance/direction, which is a an approach that typically results in reasonable real-time decisions; for a recent example, see [3] where additional physical constraints are imposed on the fuzzy rules to reduce reliance on heuristics. An automatic algorithm modifies the fuzzy rules in real-time in response to the environment in [4]. A neural-network structure encodes the fuzzy rules in order to provide adaptation (or learning) for static obstacles in [5] and dynamic obstacles in [6]. In order to achieve more complex navigation behaviours, maze-navigation and/or extraction from dead-ends, the fuzzy rules must be augmented with a memory of the environment, for example [7]. However, heuristic fuzzy techniques lack mathematical guarantees of success and/or stability, which is an important issue to address in this field of research.

The theory of nonlinear feedback control holds the potential for mathematical guarantees in navigation problems in the face of uncertainty. Usually, nonlinear feedback controls simply track a pre-determined path or trajectory, as in [8],[9],[10] and [11]. The method of artificial potential fields is similar to feedback control and decides the path itself; modifications allow the use of only local sensor information to implement this popular technique in [12]. Given good knowledge of the environment and a series of pre-designed local feedback controls that funnel into each other in a hybrid control approach called 'sequential composition', it is possible to successfully steer the robot around obstacles, as in [13] and [14]. This is also the case for the similar-in-concept vector-field method, such as in [15] and [16]. Using only local sensor information, a Lyapunov-stable control for unicycle navigation in [17] uses a heuristic algorithm to switch between stable target-tracking and stable obstacle avoidance feedback modes. Similarly, a border-following feedback control law allows general obstacle avoidance, with straight movements towards the target otherwise, as in [18]. In [19], a direct connection between the output and control signal achieves BEAM-style obstacle avoidance, while a linear-filter memory element used inside a Lyapunov-stable backstepping scheme provides smooth, rounded motions around obstacles while guaranteeing target-reaching in the absence of obstacles.

Our work follows on from that proposed in [19] due its simplicity, mathematical stability analysis, robust characteristics in the face of sensor noise, and reliability in avoiding collisions with obstacles. However, we find the method insufficient to navigate some environments successfully. Rather than rely on the BEAM-style control to avoid obstacles, we modify it by adding some small amount of fuzzy processing and call it a *virtual force*, which slows the robot and chooses a logical turning direction only in the event of imminent collision. Rather than rely on the memory with Lyapunov backstepping control - with the target position as the only input - to choose the direction of travel, we introduce some fuzzy processing to pick an intermediate target for the nonlinear control that avoids the closest obstacles. The addition of these two fuzzy processing levels creates a new, novel control method which results in improved trajectories and obstacle avoidance compared to [19]. However, the scheme is still computationally simple and remains robust in the face of limited and noisy sensor data. The robot ends up behaving in an insect-like manner, in the sense that it simply heads for gaps between obstacles that are close to the direction of the target. The algorithm allows the robot

to head directly towards a sensed obstacle; for example, it may head towards a further obstacle at the other end of a gap between two closer obstacles.

Simulation results demonstrate the expected behaviour of the system in a corridor environment relying on a noisy, sweeping proximity sensor. We also try the control of [19] in the simulations, whereby we find that it fails to navigate this environment. Moreover, simulations reveal that eliminating any one of the three elements in the proposed control (fuzzy motion-planning, nonlinear control or virtual force) results in a failure to navigate this environment.

2. Overview

The proposed navigation strategy uses fuzzy logic to decide on both an intermediate target location provided as a reference input to a closed-loop control as well as a virtual obstacle force provided as a disturbance to the closed-loop control (Figure 1). First, a fuzzy processing operation outputs an intermediate target position $\boldsymbol{\eta}_i$ (a vector with a distance and a relative angle) that creates a logical heading for the robot to follow, taking into account the target position, measured obstacle positions and obstacle positions in the immediate past.The nonlinear control then takes this information and provides commanded translational and rotational velocities, in vector $\mathbf{u}$, based on its current position $\boldsymbol{\eta}_u$ (relative to the ultimate target) and $\boldsymbol{\eta}_i$. Another fuzzy logic operation decides on the virtual force term, $\boldsymbol{\nu}_f$ added directly to $\mathbf{u}$, that reacts quickly to obstacles that might cause a collision if forward motion continues in the same direction. Note that the addition of the control-system and virtual force outputs constitute the real commanded velocities of the robot. The resulting behaviour consists of:

- heading for a gap between obstacles, or towards an obstacle that is relatively far away, in a direction that is close to the target direction,
- making wide turns around obstacles so as to avoid collisions,
- suddenly slowing and turning if a collision appears imminent.

2.1. Robot Dynamics

The robot makes angle measurements to the target, θ_u, and obstacles relative to its own heading (Figure 2). Note that the information ends up stored in the fuzzy encoding using an absolute coordinate system denoted by ϕ with $\phi = 0$ aligned to a Cartesian X coordinate. The robot also measures the distance to the ultimate target, d_u, and the vector describing the target position becomes:

$$\boldsymbol{\eta}_u = \left[d_u \ \theta_u\right]^T. \tag{1}$$

The fuzzy logic system will pick a direction that avoids immediate obstacles (but possibly heading towards a distant obstacle) by choosing an intermediate target position defined by:

$$\boldsymbol{\eta}_i = \left[d_i \ \theta_i\right]^T. \tag{2}$$

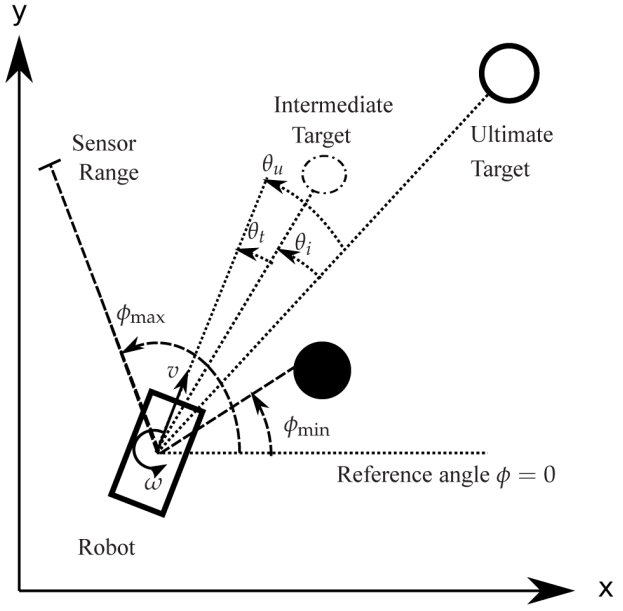


Figure 2. Coordinates: Dashed lines indicate the scanning range of the proximity (obstacle) sensor, another sensor provides a relative target location θ_u, and fuzzy motion-planning decides on an intermediate target relative angle θ_t

The relative coordinate is thus:

$$\boldsymbol{\eta} = \left[d_t \ \theta_t\right]^T, \tag{3}$$

where $d_t = d_i$ and:

$$\theta_t = \theta_u - \theta_i. \tag{4}$$

Since the desired value of $\boldsymbol{\eta}$ must be $[0 \ \ 0]^T$, the error becomes:

$$\mathbf{e} = \mathbf{0} - (\boldsymbol{\eta} - \mathbf{N}(t)) = -\boldsymbol{\eta} + \mathbf{N}(t) \tag{5}$$

where $\mathbf{N}$ is the sensor noise from measurements of $\boldsymbol{\eta}$. The control system outputs the desired (commanded) translational velocity v and rotational velocity ω:

$$\mathbf{u} = \left[v \ \omega\right]^T. \tag{6}$$

The final commanded velocities include the virtual force term $\boldsymbol{\nu}_f$, directly added to the control signal:

$$\boldsymbol{\nu} = \mathbf{u} + \boldsymbol{\nu}_f. \tag{7}$$

We assume that the internal motor controllers achieve the velocities in $\boldsymbol{\nu}$ in sufficient time such that the errors between the commanded and actual velocities remain insignificant; they are ignored in the navigation/control scheme developed here.Thus, in the computer simulations the only *dynamics* consist of:

$$\dot{\boldsymbol{\eta}} = \mathbf{T}(\boldsymbol{\eta})\boldsymbol{\nu}, \tag{8}$$

where $\mathbf{T}(\boldsymbol{\nu}) = \begin{bmatrix} -\cos(\theta_t) & 0 \\ \frac{\sin(\theta_t)}{d_t} & 1 \end{bmatrix}$.

2.2. Nonlinear Control

In the nonlinear control strategy, we first define a linear filter memory element that has error for the input. The key to achieving smooth motions is in designing a desired error that will cause the filter states to gradually go to zero. This is done with a Lyapunov backstepping procedure where the desired error is a virtual control in the first step of backstepping. A second step of backstepping results in the design of the velocities (the controls) to move the error towards its virtual control desired value.

2.2.1. Linear filter memory element

In order to reject noise and to have the robot move in a wide, rounded path, we use a linear time-invariant low pass filter to deal with the state errors e as an analogue memory element. Using standard state space-form for the filter gives:

$$\dot{\boldsymbol{\zeta}}(t) = \mathbf{A}\boldsymbol{\zeta}(t) + \mathbf{B}\mathbf{e}(t) \tag{9}$$

where the vector $\boldsymbol{\zeta}$ contains the states of the filter, with the first state being the output $y = \zeta_1$. The control achieves smooth and wide trajectories by trying to reduce filter states $\boldsymbol{\zeta}$ to zero as opposed to trying to reduce $\mathbf{e}(t)$ directly.

2.2.2. Lyapunov backstepping, step 1

Since the state equations appear in strict-feedback form:

$$\dot{\boldsymbol{\zeta}}(t) = \mathbf{A}\boldsymbol{\zeta}(t) + \mathbf{B}[-\boldsymbol{\eta}(t) + \mathbf{N}(t)], \tag{10}$$

$$\dot{\boldsymbol{\eta}}(t) = \mathbf{T}(\boldsymbol{\eta})\mathbf{u}, \tag{11}$$

the Lyapunov backstepping technique provides a suitable nonlinear control strategy. The first step of backstepping uses the positive-definite function:

$$V_1 = \boldsymbol{\zeta}^T\mathbf{P}\boldsymbol{\zeta}, \tag{12}$$

where V_1 is a scalar-valued function and $\mathbf{P}$ is a symmetric positive definite matrix which satisfies $\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} = -\mathbf{Q}$ where $\mathbf{Q}$ is a positive-definite matrix. Taking the time-derivative results in:

$$\dot{V}_1 = -\boldsymbol{\zeta}^T\mathbf{Q}\boldsymbol{\zeta} + 2\boldsymbol{\zeta}^T\mathbf{P}\mathbf{B}(-\boldsymbol{\eta} + \mathbf{N}). \tag{13}$$

The backstepping error is:

$$\mathbf{z} = \boldsymbol{\eta} - \boldsymbol{\eta}_d, \tag{14}$$

where $\boldsymbol{\eta}_d$ is the virtual control, a desired value of $\boldsymbol{\eta}$. Thus:

$$\dot{V}_1 = -\boldsymbol{\zeta}^T\mathbf{Q}\boldsymbol{\zeta} + 2\boldsymbol{\zeta}^T\mathbf{P}\mathbf{B}(-\mathbf{z} - \boldsymbol{\eta}_d + \mathbf{N}), \tag{15}$$

leads to a choice of virtual control:

$$\boldsymbol{\eta}_d = \mathbf{B}^T\mathbf{P}\mathbf{K}_1\boldsymbol{\zeta} + \mathbf{R}\tanh(\mathbf{S}\mathbf{B}^T\mathbf{P}\boldsymbol{\zeta}), \tag{16}$$

where $\mathbf{K}_1$, $\mathbf{R}$ and $\mathbf{S}$ are all diagonal matrices with positive constant terms on the diagonals (note $\mathbf{P}^T = \mathbf{P}$). The function $\tanh() \in \mathcal{R}^2$ performs an element-by-element hyperbolic tangent operation and provides robustness to noise when choosing $R_{i,i} > |N_i|_{\max}$ for $i = 1, 2$. Positive constants in $\mathbf{S}$ determine the *steepness* of the hyperbolic tangent, used in place of a discontinuous robust control so that the signal may be differentiated. Therefore, (15) becomes:

$$\begin{aligned}\dot{V}_1 = &-\boldsymbol{\zeta}^T\mathbf{Q}\boldsymbol{\zeta} - 2\boldsymbol{\zeta}^T\mathbf{P}\mathbf{B}\mathbf{B}^T\mathbf{P}\mathbf{K}_1\boldsymbol{\zeta} - 2\boldsymbol{\zeta}^T\mathbf{P}\mathbf{B}\mathbf{z} \\ &+ 2\boldsymbol{\zeta}^T\mathbf{P}\mathbf{B}\left[-\mathbf{R}\tanh(\mathbf{S}\mathbf{B}^T\mathbf{P}\boldsymbol{\zeta}) + \mathbf{N}\right].\end{aligned} \tag{17}$$

2.2.3. Lyapunov backstepping, step 2

In the second step of backstepping, the control Lyapunov function becomes:

$$V = V_1 + \mathbf{z}^T\mathbf{z}. \tag{18}$$

Differentiating (18) with respect to time gives:

$$\dot{V} = \dot{V}_1 + 2\mathbf{z}^T\dot{\mathbf{z}} \tag{19}$$

$$= \dot{V}_1 + 2\mathbf{z}^T(\dot{\boldsymbol{\eta}} - \dot{\boldsymbol{\eta}}_d) \tag{20}$$

$$= \dot{V}_1 + 2\mathbf{z}^T(\mathbf{T}(\boldsymbol{\eta})\mathbf{u} - \dot{\boldsymbol{\eta}}_d) \tag{21}$$

$$\begin{aligned}= &-\boldsymbol{\zeta}^T\mathbf{Q}\boldsymbol{\zeta} - 2\boldsymbol{\zeta}^T\mathbf{P}\mathbf{B}\mathbf{B}^T\mathbf{P}\mathbf{K}_1\boldsymbol{\zeta} - 2\boldsymbol{\zeta}^T\mathbf{P}\mathbf{B}\mathbf{z} + \\ &2\boldsymbol{\zeta}^T\mathbf{P}\mathbf{B}\left[-\mathbf{R}\tanh(\mathbf{S}\mathbf{B}^T\mathbf{P}\boldsymbol{\zeta}) + \mathbf{N}\right] + \\ &2\mathbf{z}^T(\mathbf{T}(\boldsymbol{\eta})\mathbf{u} - \dot{\boldsymbol{\eta}}_d).\end{aligned} \tag{22}$$

A nonlinear control that would ensure a negative-definite $\dot{V}$ under noise-free conditions is:

$$\mathbf{u} = \begin{bmatrix} v \\ \omega \end{bmatrix} = \mathbf{T}(\boldsymbol{\eta})^{-1}(\mathbf{B}^T\mathbf{P}\boldsymbol{\zeta} + \dot{\boldsymbol{\eta}}_d - \mathbf{K}_2\mathbf{z}), \tag{23}$$

where $\mathbf{K}_2$ has positive constant control gains on the diagonals. Note that $\dot{\boldsymbol{\eta}}_d$ must be calculated analytically:

$$\dot{\boldsymbol{\eta}}_d = \frac{\partial \boldsymbol{\eta}_d}{\partial \boldsymbol{\zeta}}\dot{\boldsymbol{\zeta}} \tag{24}$$

$$= \boldsymbol{g}(\mathbf{K}_1\mathbf{B}^T\mathbf{P} + \mathbf{R}\mathbf{S}\mathbf{B}^T\mathbf{P}[\mathbf{1} - \tanh^2(\mathbf{S}\mathbf{B}^T\mathbf{P}\boldsymbol{\zeta})]\boldsymbol{g})\dot{\boldsymbol{\zeta}}. \tag{25}$$

Substituting (23) into (22), the derivative of the control Lyapunov function becomes:

$$\begin{aligned}\dot{V} = &-\boldsymbol{\zeta}^T\mathbf{Q}\boldsymbol{\zeta} - 2\mathbf{K}_1\boldsymbol{\zeta}^T\mathbf{P}\mathbf{B}\mathbf{B}^T\mathbf{P}\mathbf{K}_1\boldsymbol{\zeta} \\ &2\boldsymbol{\zeta}^T\mathbf{P}\mathbf{B}[-\mathbf{R}\tanh(\mathbf{S}\mathbf{B}^T\mathbf{P}\boldsymbol{\zeta}) + \mathbf{N}] - 2\mathbf{z}^T\mathbf{K}_2\mathbf{z}.\end{aligned} \tag{26}$$

To analyse stability, first bind the time derivative:

$$\begin{aligned}\dot{V} \leq &-\lambda_{\min}(\mathbf{Q})||\boldsymbol{\zeta}||^2 - 2\lambda_{\min}(\mathbf{K}_2)||\mathbf{z}||^2 \\ &+2||\boldsymbol{\zeta}^T\mathbf{P}\mathbf{B}||[-\|\mathbf{R}\tanh(\mathbf{S}\mathbf{B}^T\mathbf{P}\boldsymbol{\zeta})|| + \|\mathbf{N}\|],\end{aligned} \tag{27}$$

where $\lambda_{\min}()$ denotes the minimum eigenvalues of the symmetric matrices. Thus, $\dot{V} < 0$ when:

$$|\zeta_i| > \frac{\tanh^{-1}(N_i/R_{i,i})}{(\mathbf{S}\mathbf{B}^T\mathbf{P})_i} \text{ for both } i = 1, 2, \tag{28}$$

and by standard arguments all signals are uniformly ultimately bounded. When there is no noise, $\mathbf{N}(t) = 0$, the time derivative is negative definite and the origin is asymptotically stable ($\zeta \to 0$ as $t \to \infty$).

2.3. Fuzzy Environment Encoding

The fuzzy encoding of the environment utilizes an absolute polar coordinate system, requiring conversion of the relative angle and distance measurements from the sensors. The proximity sensor outputs (after coordinate transformation) M equally-spaced angles $\phi_i, \quad i = 1 \ldots M$ in the current sweeping arc of the sensor between $\phi_{\min}(t)$ and $\phi_{\max}(t)$, each with a distance reading $d_i(\phi_i)$ (or equivalently $\boldsymbol{\eta}_i = [d_i\ \phi_i]^T$). We provide the fuzzy encoding system with a normalized distance $\bar{d}_i = d_i/d_{\max}$, where $d_{\max}$ is the maximum sensor range, interpreting each $\bar{d}_i$ as

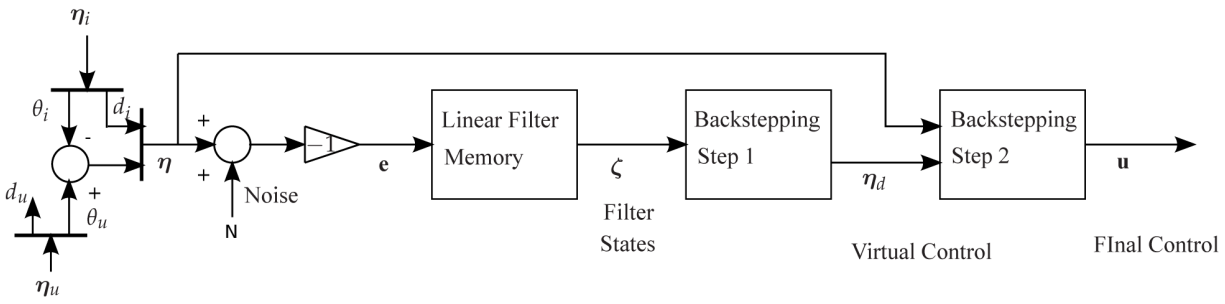


Figure 3. Inner structure of the control system: first is a linear filter used as memory, followed by nonlinear backstepping control

a fuzzy truth-value of FAR="The obstacle is far". When no obstacle is detected at ϕ_i, then $\bar{d}_i = 1$. Obtaining a continuous, filtered output defining FAR for all angles between 0 and 360°, we encode the information using m weighted Gaussian fuzzy membership functions of the form:

$$\Gamma_j(\phi_i) = \exp((-\phi_i - c_j)^2/\sigma^2) \text{ for } j = 1 \dots m \qquad (29)$$

where σ is a positive constant. The centres are evenly distributed with $c_j = 360j/m$. Expressing the truth-value of "The obstacle is far" as a function of an input ϕ_i, we write:

$$\text{FAR} = \sum_{j=1}^{m} w_j \Gamma_j(\phi_i). \qquad (30)$$

The algorithm continuously updates, in real-time, each weighting w_j by sweeping ϕ from 0 to 360° in M steps, and at each step applying the update $w_j = w_j + \Delta w_j$ where:

$$\Delta w_j = \begin{cases} \beta(\bar{d}_i - \text{FAR}) & \text{if } \phi_{\min}(t) < \phi_i < \phi_{\max}(t), \\ -\gamma w_j & \text{otherwise,} \end{cases} \qquad (31)$$

and where β is a positive learning rate and γ is a positive forgetting factor. Inside the sensor range, the Gaussian functions serve to filter the proximity-sensor data. Obstacles previously in sensor range remain in memory for some time in order that they can be taken into account in the motion planning algorithm. Once enough time has passed, these disappear from the memory such that γ should depend upon the speed of the robot.

The fuzzy encoding of the target heading utilizes a Gaussian function, with a value of 1 where the sensors locate the target at a given moment and a value approaching 0 at ±360°.

2.4. Decision on the Intermediate Target

The choice of desired heading depends, first, on the peaks (maxima) of the fuzzy obstacle encoding, FAR, being identified. The peaks then represent possible headings for the robot, i.e., the angular component of possible intermediate targets. A value greater than 1 at the peak (FAR > 1) means that no obstacle is within sensor range in that direction, while FAR < 1 indicates the relative distance to the obstacle in that direction. FAR = 0 would imply contact with the obstacle. Note that even though all values less than 1 represent possible obstacles, the choice of heading does not directly take obstacles into account (although these obstacles will be used in producing virtual forces). Possible headings might be in the same direction as obstacles but directed towards an obstacle relatively far away compared to other obstacles. The choice of intermediate target direction utilizes the fuzzy rule:

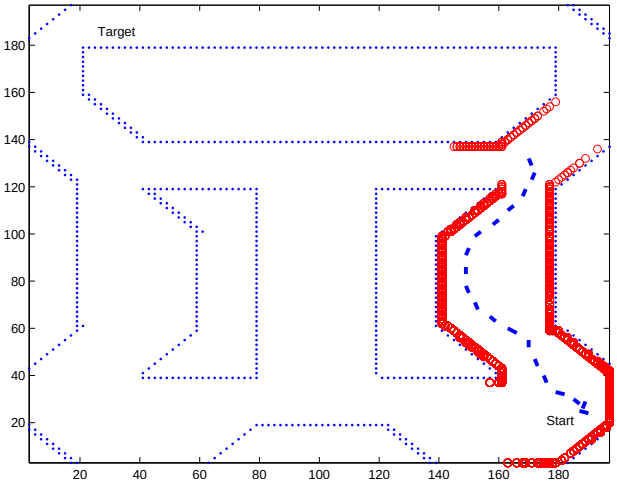


Figure 4. Cartesian path and robot orientation for 19 seconds. The red circles indicate current and past obstacle-sensor readings (the sensor range can be inferred.

- If the possible heading is CLOSE to the target direction AND an obstacle in that direction is FAR, then in it is DESIRED

which is computed at the kth peak of FAR as:

$$\text{DESIRED}_k = |\phi_k - \phi_{\text{target}}| \times \text{FAR}(\phi_k). \qquad (32)$$

The new desired heading becomes the one with the largest value of DESIRED. Note that the intermediate target depends upon time for both the angle and the distance. The angle changes continuously with the robot's motion as obstacles are perceived at different angles, and the angle may change discontinuously at any time if a different peak moves closer to the target. The distance to the intermediate target simply becomes the maximum sensor range when no obstacles are detected that way or else a fraction of the way to the obstacle given an obstacle in that direction. Specifically, in the simulations, when FAR< 1 in the direction of the desired heading, the vehicle is commanded to go 80% of the way to the obstacle.

2.5. Producing Virtual Force

The virtual force term directly augments the commanded velocities from the control, appearing as a disturbance in

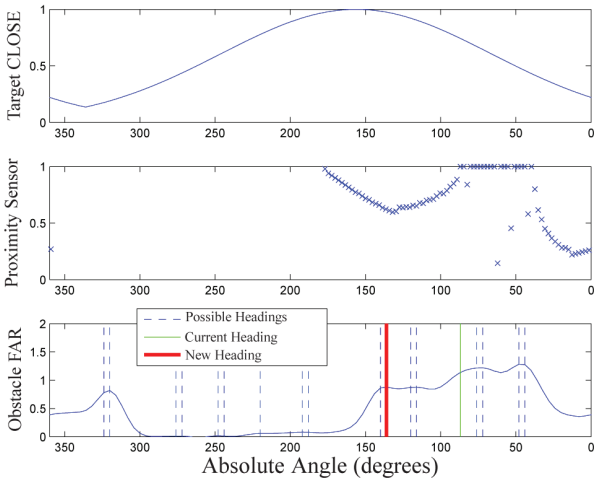


Figure 5. Encoding and decision-making at $t = 19$ - made when the robot heading is $\phi = 90°$

the feedback control loop. In this way, the robot reacts directly to an impending obstacle analogous to the way in which BEAM strategies manoeuvre The fuzzy encoding of the environment provides a convenient way to instantly decide whether the robot should move left or right, and by how much, in order to avoid the obstacle. It also decides how much it should slow down. The fuzzy rules for producing the virtual force in the direction are made by looking at the value of FAR. The algorithm looks to the left and right of the current heading in FAR and identifies the nearest obstacle (where FAR$<$ 1) to both the left and the right. The fuzzy rule base is:

1. If an obstacle is NEAR, then the translational velocity moves SLOWER,
2. If an obstacle is to the LEFT AND NEAR, then the angular velocity moves NEGATIVE,
3. If an obstacle is to the RIGHT AND NEAR, then the angular velocity moves POSITIVE,

where:

$$\text{NEAR} = \begin{cases} 1 - \text{FAR} & \text{if FAR} < 1 \\ 0 & \text{otherwise} \end{cases} \tag{33}$$

The fuzzy logic utilizes constants s and c to determine SLOWER, NEGATIVE and POSITIVE, calculated by:

$$\boldsymbol{\nu}_f = \begin{bmatrix} \Delta v \\ \Delta\omega \end{bmatrix} = \begin{bmatrix} -s & 0 & 0 \\ 0 & -c & c \end{bmatrix} \begin{bmatrix} \text{NEAR} \\ |\phi_h - \phi_{\text{left}}| \times \text{NEAR} \\ |\phi_h - \phi_{\text{right}}| \times \text{NEAR} \end{bmatrix} \tag{34}$$

where ϕ_h is the current heading and $\phi_{\text{left}}, \phi_{\text{right}}$ are the closest angles, in positive and negative angle directions, respectively, where FAR$<$ 1.

3. Simulation Results

In the simulation, the robot attempts to traverse a 200 cm by 200 cm environment with corridors starting in the bottom-right corner with the ultimate target at the top left (Figure 4). The robot knows where the target is in relation to its heading and has a (noisy) view of obstacles $\pm 90°$ from its heading up to a range of 25 cm in the Cartesian coordinates. The nominal (maximum) speed of the robot is 7 cm/s (1 unit/s in the graphs). The thick dashes on the Cartesian environment plot represent the position and orientation of the robot at each second.

Noise appears in both the obstacle (proximity) sensor and target sensor signals. The fuzzy encoding memory filters the proximity sensor noise while the linear filter and the robust nonlinear control handle the target sensor noise. The obstacle sensor model obtains a distance to the obstacle, $d_{measured}$ [at angle $\phi(t)$], which is a random number for a certain proportion of the time and the correct reading for other times. Formally:

$$d_{\text{measured}}(\phi(t)) = \begin{cases} d_{\max}(\phi(t))(1 - \text{rand}(1)) & \text{if rand}(1) < r \\ d_{\text{actual}}(\phi(t)) & \text{otherwise,} \end{cases} \tag{35}$$

where $d_{\max}$ is the maximum sensor range and function rand(1) produces a random number between 0 and 1, and where a positive constant r between 0 and 1 determines the noise level. For the simulations, we use a nominal value of $r = 0.2$. The target sensor output displays significant noise, giving angle readings within $\pm 40°$ of the real target angle:

$$\theta_{u,\text{measured}} = \theta_{u,\text{actual}} + 80 \cdot [0.5 - \text{rand}(1)] \quad \text{[deg]}. \tag{36}$$

We assume that the maximum orientation error is known and use $N_{\max} = 40°$ in the robust control, choosing the following parameters:

$$\mathbf{Q} = \text{diag}(1,1),$$
$$\mathbf{K}_1 = \text{diag}(0.65,5), \qquad \mathbf{K}_2 = \text{diag}(0.65,5),$$
$$\mathbf{R} = \text{diag}(0.1,50), \qquad \mathbf{S} = \text{diag}(1,1).$$

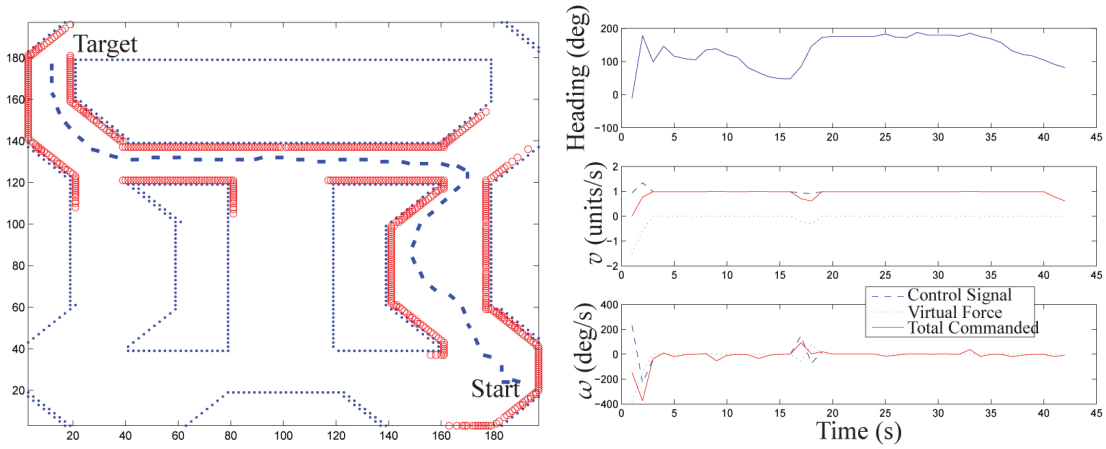
Note that as the noise appears in the angle measurement only, it is only the second diagonal element of $\mathbf{R}$ that provides robustness to noise. In our work, we found $c = 500$ deg/s and $s = 2$ m/s to be suitable values in (34). For the linear filter memory, we use a simple low pass filter:

$$\frac{\zeta_1(s)}{e(s)} = \frac{\omega_f}{s^2 + 2\zeta_f\omega_f s + \omega_f^2} \tag{37}$$

with $\omega_f = 1.6$ Hz and $\zeta_f = 0.7$.

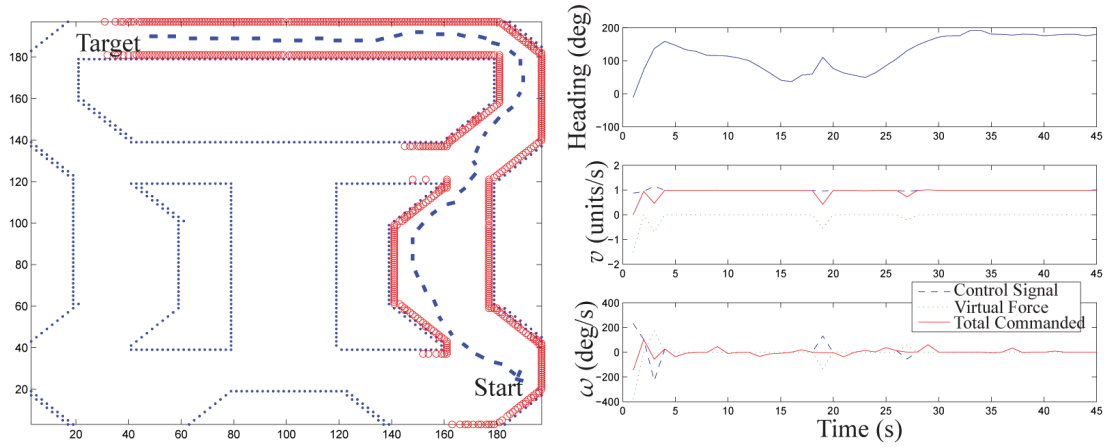
The robot starts out at an initial heading of 0°, directly facing the right-hand wall and facing away from the target. For the first two seconds, the robot turns back and forth as it attempts to find an obstacle-free heading (Figure 4 bottom right). Inspecting the simulation 19 seconds after the start (Figure 5) reveals typical robot decision-making during the manoeuvre. At this point in time, the robot faces 90° in absolute coordinates (Figure 5 bottom graph, green line), while the ultimate target appears at 155° absolute angle (Figure 5 top graph), thus making the relative target heading (error) 65°. The range-finder sensor sweeps 90° to either side of the heading, 0° to 180° in absolute coordinates, with noise clearly visible in the sensor readings (Figure 5 middle graph).

At 19 seconds, the robot also senses the wall directly ahead of it and makes a decision to put the intermediate target at 140° (Figure 5 bottom graph, red line), thus making a left turn of +50° from the current heading.



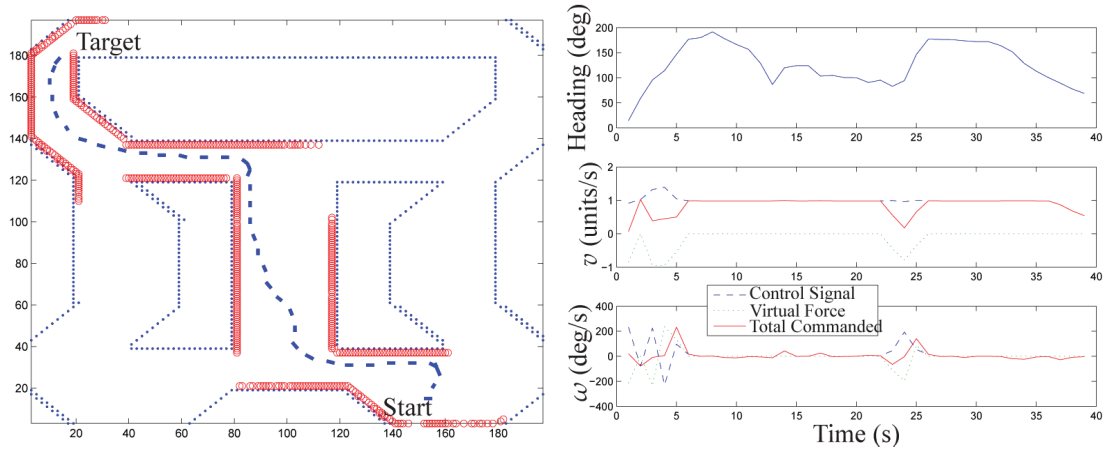
(a) 2nd simulation: Cartesian path

(b) 2nd simulation: contributions of control and virtual force



(c) 3rd simulation: Cartesian path

(d) 3rd simulation: contributions of control and virtual force



(e) 4th simulation: Cartesian path

(f) 4th simulation: contributions of control and virtual force

Figure 6. Complete simulations

Even though the robot sees an obstacle-free direction to its right between 85° to 35° in absolute coordinates (Figure 5 bottom graph, solid curve), it chooses a heading to its left since the obstacle remains relatively far away and this heading appears closer to the ultimate target position.

In another simulation, the random noise affects decision-making and results in a slightly different path (Figure 6a). However, at approximately the same point in space where the previous simulation ended, it makes the same decision - in this case, to turn left.

Tracking the commanded velocities from the motion-planning and the velocities from the virtual force term (Figure 6b) reveals that at the 17 second mark, when faced with a similar decision as in Figure 4, the virtual force command for a negative angular velocity (a right turn) was "overruled" by the motion-planning. The virtual force slows the robot and tries to direct the robot to the obstacle-free direction (to the right), but the fuzzy motion planning has a stronger command with a larger commanded positive angular velocity (to the left), thereby seeing a possible heading more aligned with the ultimate target direction.

In a third simulation, when faced with a similar decision, the robot - this time - moved to the right (Figure 6c). The effects of random sensor noise on the system have slightly altered the particular circumstances compared to the previous occasion. This time, the robot virtual force term slightly outweighs the fuzzy motion planning at the 19 second mark (Figure 6d) and the robot turns to the right. After turning, the fuzzy motion-planning identifies a new possible heading and does not try to turn around. The virtual force term again interferes in the motion at the 27 second mark, slowing and turning the robot away from the wall on the right when the motion planning appears to command a turn towards the wall.

In a fourth simulation, the robot starts at a slightly different location - to the left - and it ends up taking a completely different path (Figure 6e). In this simulation, the virtual force term helps the robot avoid a corner at the 24 second mark (Figure 6f). It causes the robot to turn to the right to avoid the corner, even though the motion planning commands a left turn. In a real robot, this type of behaviour should eliminate the chance that the machine would try to move too close to a corner and thus make contact with the extrusion due to the robot's width.

Sim.	Min. Distance (cm)	Time (s)	Min. Curvature (cm)
2nd	5.4	43	4.3
3rd	4.2	45	6.4
4th	4.4	38	5.3

Table 1. Summary of results

In all cases, the nonlinear control provides a smooth motion except for when the virtual force terms affect the system and provide quick responses. Table 1 summarizes the results in terms of the minimum distance to a wall, the total time and the minimum radius of the curvature of the trajectory. Note that the nonlinear control provides most of the changes in heading; for example, as seen in Figure 6f, between nine and 13 seconds. Here, the intermediate target position does not change significantly, remaining at approximately 140°, yet the nonlinear control causes a smooth, rounded right-turn around the corner to the right (Figure 6e first corner), with no input from the virtual force term needed.

To show the necessity of the each component in each of the controls, we test three different alternative methods:

1. Using the same virtual force and backstepping control, but no motion planning - only aiming for the ultimate target which reflects the control proposed in [19] (green dash, nearly hitting wall on left, Figure 7),
2. Using the same motion planning and memory/backstepping control, but without any virtual force term (red dash, hitting wall, Figure 7),
3. Using the same motion planning and virtual force, but without memory/backstepping control - instead using $\mathbf{u} = \mathbf{T}^{-1}(\boldsymbol{\eta})(-\mathbf{e})$ (black dash, nearly hitting wall on right, Figure 7.)

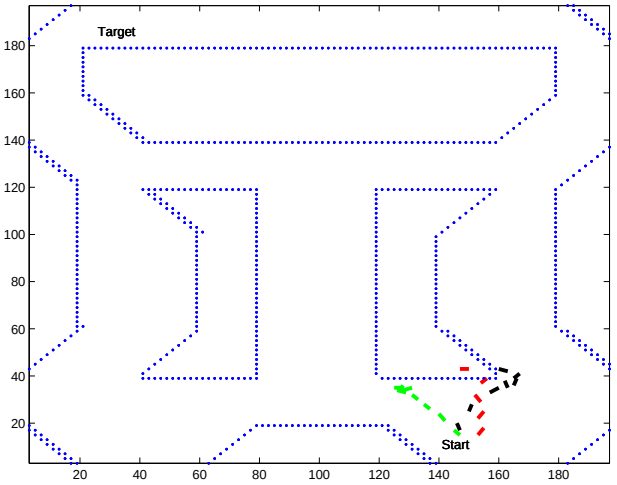


Figure 7. Other controls: Green (on left) Control 1, Red (crashes into wall) Control 2, Black (on right) Control 3

Although the virtual force term keeps the system from actually touching the wall in Controls 1 and 3, the robot does come very close and ends up having to reverse direction. The previous method proposed in the literature - Control 1, which uses just the ultimate target without planning any intermediate targets - relies too heavily on the BEAM-like virtual force to make an effective navigation strategy. Control 2, without virtual force, ends up driving directly into a wall. Control 3, without backstepping, does not give the corner enough room to get by safely.

In order to test the necessity of the robust virtual control term [with tanh()], another control, Control 4, eliminates the robust term (as the only change):

$$\boldsymbol{\eta}_d = \mathbf{K}_1 \mathbf{B}^T \mathbf{P} \boldsymbol{\zeta}. \tag{38}$$

This affects only the angle-control in the simulation, since the robust term accounts only for noisy θ_t (not noise-free d_t). The results show that the robot is easily confused by obstacles and cannot always find the obstacle-free path (Figure 8 top right), and also that it tends to wander back and forth in its movements when it should be following a nearly-straight line in a long corridor.

Finally, we test the robustness to noise by varying the noise-level parameter r from (35). The simulation was attempted eight times at each level of r, and r was changed by 0.05 each time. The simulations had 100% success rate, as defined in terms of reaching the target without hitting a wall, when using a noise level below $r < 0.5$. The results appear in Table 2.

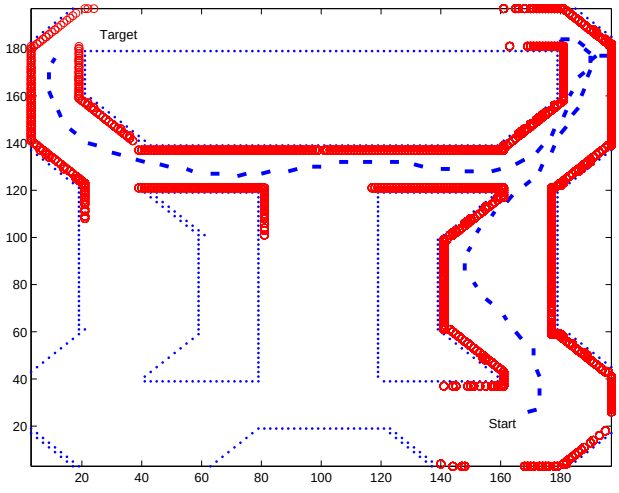


Figure 8. Control without tanh() as the robust control term (Control 4)

r	<0.5	0.5	0.55	>0.55
Success rate	100%	75%	25%	0%

Table 2. Robustness to noise

4. Conclusion

This paper proposes a reactive-navigation strategy for a mobile robot, suitable for implementation on a robot with limited computational power. The strategy relies on knowledge of the ultimate target location and local knowledge of obstacles provided by a sweeping, forward-looking proximity sensor. A fuzzy motion-planning algorithm plans an intermediate target between obstacles for the control system to track. A Gaussian fuzzy-encoding of the environment serves to filter sensor noise and provide a basis for choosing a heading. A nonlinear Lyapunov backstepping control system with a linear filter element provides robustness to noise in the control system itself, resulting in a wide path around obstacles. A collision avoidance strategy, implemented with a BEAM-style virtual force term, directly augments the commanded velocities in order to react immediately to any impending obstacles - both slowing and turning in a logical direction.

The advantages over computationally-complex approaches, include suitability for small mobile robots, immediate decision-making, continuous motion in a complex environment, reasonable choices in the face of environmental uncertainty from limited sensors, smooth motion in the face of noisy sensors, and immediate reactions in the face of imminent collisions. Unlike as with computationally simple BEAM control, the robot follows a smooth path when possible and can immediately head for a space between sensed obstacles.

To evaluate the performance of the control system, computer simulations in a corridor environment were conducted with noisy sensor data. The results show that the control system provides robustness to noise and can successfully give obstacles/walls a wide clearance while maintaining a reasonable speed. The robot navigates an unknown static environment (lacking dead-ends, traps and moving obstacles) while avoiding obstacles/walls and arriving at the desired location successfully.

5. References

[1] Ellips Masehian and Davoud Sedighizadeh. Classic and heuristic approaches in robot motion planning - A chronological review. In *Proc. World Academy of Science, Engineering and Technology*, pages 101–106, 2007.

[2] M.W. Tilden. The evolution of functional robo-ecologies. *ARS Electronica*, 93:195–200, 1993.

[3] I. Baturone and A.A. Gersnoviez. A simple neuro-fuzzy controller for car-like robot navigation avoiding obstacles. In *IEEE Intl. Fuzzy Systems Conf.*, pages 1–6, London, UK, 2007.

[4] Anmin Zhu and Simon Yang Simon. *An Adaptive Neuro-fuzzy Controller for Robot Navigation*, pages 277–307. Springer London, 2009.

[5] K. Al Mutib and E. Mattar. Neuro-fuzzy controlled autonomous mobile robotics system. In *Intl. Conf. Computer Modelling and Simulation*, pages 1–7, 2011.

[6] P. Kondaxakis, H. Baltzakis, and P. Trahanias. Learning moving objects in a multi-target tracking scenario for mobile robots that use laser range measurements. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 1667 –1672, Oct. 2009.

[7] Meng Wang and James N.K. Liu. Fuzzy logic-based real-time robot navigation in unknown environment with dead ends. *Robotics and Autonomous*, 56:625–643, 2008.

[8] Shuli Sun. Designing approach on trajectory-tracking control of mobile robot. *Robotics and Computer-Integrated Manufacturing*, 21:81–85, 2005.

[9] Jun Ye. Adaptive control of nonlinear pid-based analog neural networks for a nonholonomic mobile robot. *Neurocomputing*, 71:1561–1565, 2008.

[10] Tamoghna Das and Indra Narayan Kar. Design and implementation of an adaptive fuzzy logic-based controller for wheeled mobile robots. *IEEE Trans. Contr. Syst. Technol.*, 14(3):501–510, 2006.

[11] An-Min Zou, Long Cheng, and Min Tan. Adaptive control of an electrically driven nonholonomic mobile robot via backstepping and fuzzy approach. *IEEE Trans. Contr. Syst. Technol.*, 17(4):803–815, 2009.

[12] Yu Yu Lwin, D. Takahashi, and Y. Yamamoto. Local/global frame switching in sensor-based outdoor navigation in a wheeled mobile robot. In *Intl. Conf. Modeling, Simulation and Applied Optimization*, pages 1–6, 2011.

[13] David C. Conner, Howie Choset, and Alfred A. Rizzi. Integrating planning and control for single-bodied wheeled mobile robots. *Autonomous Robots*, 30:243–264, 2011.

[14] Vinutha Kallem, Adam T. Komoroski, and Vijay Kumar. Sequential composition for navigating a nonholonomic cart in the presence of obstacles. *IEEE Trans. Rel.*, 27(6):1152–1159, 2011.

[15] Stephen R. Lindemann, Islam I. Hussein, and Steven M. LaValle. Real time feedback control for nonholonomic mobile robots with obstacles. In *IEEE Conf. Dec. Contrl.*, pages 2406–2411, San Diego, 2006.

[16] Maciej Michalek and Krzysztof Kozlowski. Vector-field-orientation feedback control method for a differentially driven vehicle. *IEEE Trans. Contr. Syst. Technol.*, 18(1):45–65, 2010.

[17] Jiangmin Chunyu, Zhihua Qu, Eytan Pollak, and Mark Falash. Reactive target-tracking control with obstacle avoidance of unicycle-type mobile robots in a dynamic environment. In *Amer. Cntrl. Conf.*, pages 1190–1195, Baltimore, 2010.

[18] Alexey S. Matveev, Chao Wang, and Andrey V. Savkin. Real-time navigation of mobile robots in problems of border patrolling and avoiding collisions with moving and deforming obstacles. *Robotics and Autonomous Systems*, 60:769–788, 2012.

[19] T. Zourntos and N.J. Mathai. A beam-inspired lyapunov-based strategy for obstacle avoidance and target-seeking. In *American Control Conference, 2007. ACC '07*, pages 5302 –5309, July 2007.