





# Control Strategies for Human–Robot–Environment Interaction in Assisted Gait with Smart Walkers

# 10

Sergio D. Sierra M. , Mario F. Jiménez, Anselmo Frizzera-Neto, Marcela Múnera, and Carlos A. Cifuentes 

## 10.1 Introduction

Recent advances and developments in rehabilitation engineering have been focused on the design and implementation of control strategies that allow natural, safe, and compliant interaction between users, the smart walker, the environment, and the healthcare professionals [1, 2]. In particular, multiple research projects have been oriented to develop innovative tools to assist older people, neurological patients, and people with cognitive impairments. Among these, the *AGoRA Smart Walker* [3], the *UFES Smart Walker* [4], the *GUIDO Smart Walker* [5], and the *MOBOT Platform* [6] are found.

These strategies have gained considerable popularity in rehabilitation and everyday scenarios, owing to their positive usability outcomes, proper users and medical acceptance, and the increasing cooperation between engineers, medical staff, and patients [7–9]. Specifically, a key issue in such interdisciplinary collaborations is related to the fact that they focus on generating solutions with a particular focus on the user, that is to say, strategies and prototypes centered on the users' requirements [10].

---

S. D. Sierra M. · M. Múnera · C. A. Cifuentes (✉)  
Biomedical Engineering, Department of the Colombian School of Engineering Julio Garavito,  
Bogotá D.C., Colombia  
e-mail: [sergio.sierra@escuelaing.edu.co](mailto:sergio.sierra@escuelaing.edu.co); [marcela.munera@escuelaing.edu.co](mailto:marcela.munera@escuelaing.edu.co);  
[carlos.cifuentes@escuelaing.edu.co](mailto:carlos.cifuentes@escuelaing.edu.co)

M. F. Jiménez  
School of Engineering, Science and Technology, Universidad del Rosario, Bogotá D.C.,  
Colombia  
e-mail: [mariof.jimenez@urosario.edu.co](mailto:mariof.jimenez@urosario.edu.co)

A. Frizzera-Neto  
Graduate Program in Electrical Engineering, Federal University of Espírito Santo, Vitória, Brazil  
e-mail: [frizzera@ieee.org](mailto:frizzera@ieee.org)

Considering the particular case of smart walkers, in the previous chapters, it has been stated that these interaction strategies require the rehabilitation device to count with appropriate sensory architectures, precise actuation interfaces, and sufficient communication interfaces with the user. In general, the selection of such components is aimed at providing three types of interaction in smart walkers: (1) Human–Robot Interaction (HRI), (2) Robot–Environment Interaction (REI), and (3) Human–Robot–Environment Interaction (HREI). In this sense, this chapter seeks to describe the most common architectures that can be used to provide these types of interaction during walker-assisted gait and demonstrate some case studies and provide insights into their implementation in real devices.

---

## 10.2 Design Considerations for Control Strategies

During the design process of an interaction strategy for walker-assisted gait, several milestones should be attained by researchers, healthcare professionals, and stakeholders. For instance, this process involves (1) identification of patients' requirements, (2) co-design of robotic solutions (involving engineers, clinicians, patients, and relatives), (3) implementation in healthy patients, (4) validation in clinical scenarios, and (5) analysis of effects [11, 12]. This is not a straightforward process but a continuous loop of development, integration, and testing. Moreover, depending on the type of interaction that is sought to be developed, there are several baseline design criteria, such as safety, compliance, and comfort, among others. Table 10.1 describes these underlying concepts for HRI, REI, and HREI.

As it can be inferred from Table 10.1, it is a common denominator in the design criteria that the smart walker behavior is safe, intuitive, compliant, and appropriate for the users' specific requirements [1]. Some of these criteria have been widely reported in literature reviews focused on smart walkers [2, 13, 14]. Additional constraints might also include (1) the smart walker motion should be smooth and only triggered by the users' intentions, (2) the control strategies should not induce hazardous situations neither for the users nor for the environment, (3) the healthcare professional should always be involved in the interaction loop, either for monitoring or for active participation, and (4) the smart walker should track and store the users' progress and session's performance [1].

Finally, the control architectures for walker-assisted gait should include several minimal modules to interact with users. Particularly, Fig. 10.1 illustrates a standard control diagram in a walker-assisted gait application, where the involvement of the user, the smart walker, and the clinician is showcased. Moreover, this architecture aims to implement the design criteria defined previously (see Table 10.1). Several smart walkers reported in the literature have already proposed control strategies that can be framed within this control diagram [3–6, 15–17]. This diagram offers a generalization of the control strategies, and thus some smart walkers might not include all of the proposed modules.

Multiple control strategies will be described in the following sections, categorizing them as strategies for HRI, REI, or HREI. Moreover, some of such strategies

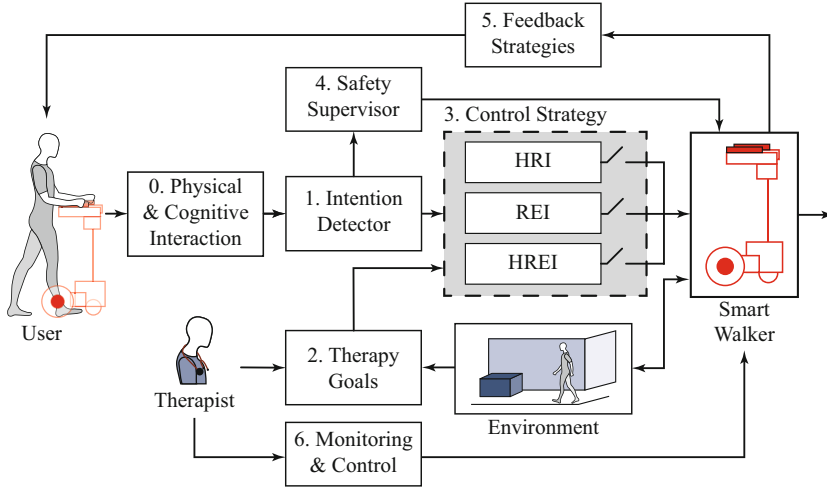
**Table 10.1** Description of general design criteria for the development of control strategies in walker-assisted gait

Interaction type	Design criteria	Example
Human–Robot Interaction (HRI)	Recognition of physical interaction with the user	Sensing of forces, torque, and pressure exerted by the user
	Recognition of cognitive interaction with the user	Voice processing modules. Gestures recognition
	Estimation of user’s navigation commands	Rule-based algorithms. Admittance controllers
	Estimation of user’s gait parameters	Sensing gait with IMUs, pressure insoles, ranging devices, etc.
	User monitoring	Hearth rate estimation
	Safety management	Detection of proper user’s support and posture. Emergency braking
	Implementation of compliant control strategies	Natural and intuitive interaction. Personalized behaviors
Robot–Environment Interaction (REI)	Smart walker motion control	Low-level controllers to generate desired velocity
	Implementation of autonomous navigation	Localization and mapping. Path planning. Obstacle detection. Guidance
	Social interaction ability	Detection of surrounding people. Motion adaption to avoid intruding into personal spaces
	Safety management	Redundant systems. Remote control
Human–Robot–Environment Interaction (HREI)	Implementation of shared control strategies	Adaptation of control authority. Modulation of user’s participation. The user triggers motion
	Clinician participation	Close accompanying teleoperation and remote monitoring
	Feedback of environment information to the user	Haptic, auditory, and visual feedback

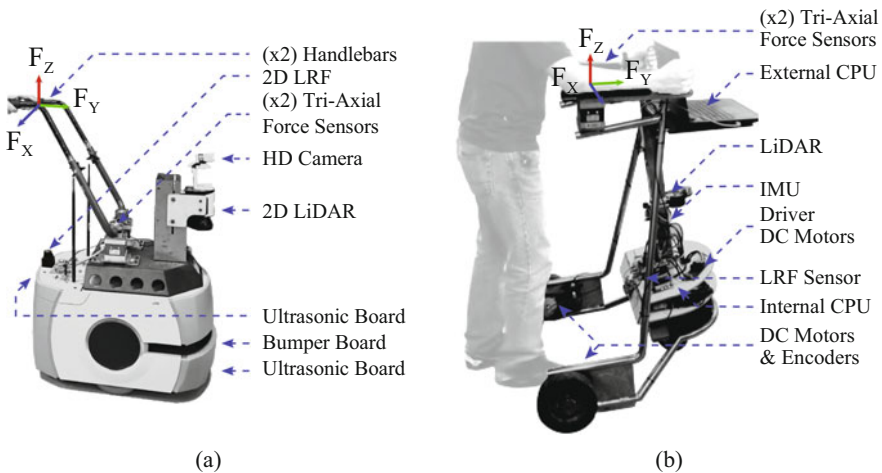
have already been validated with users, and thus, the following section presents two particular smart walkers used for these purposes: the *AGoRA Smart Walker* and the *UFES Smart Walker*.

### 10.3 Robotic Platforms

The *AGoRA Smart Walker* and the *UFES Smart Walker*, illustrated in Fig. 10.2, are two active robotic walkers that have been used to implement control strategies for HRI, REI, and HREI [3, 4]. In this way, their internal components and main characteristics are described as follows:



**Fig. 10.1** Standard control architecture for a walker-assisted gait application



**Fig. 10.2** Illustration of two standard robotic walkers for Human–Robot–Environment Interaction. (a) AGoRA Smart Walker. (b) UFES Smart Walker

### 10.3.1 AGoRA Smart Walker

The *AGoRA Smart Walker* is a robotic walker mounted on a commercial robot (Pioneer LX, Omron Adept, USA), emulating an assistive smart walker’s structural frame and functionality (see Fig. 10.2a).

The platform is equipped with (1) two motorized wheels and four caster wheels, (2) two encoders, one Inertial Measurement Unit (IMU), and two hall sensors to measure walker’s overall position and speed, (3) a 2D Light Detection

and Ranging (LiDAR) sensor (S300 Expert, SICK, Germany) for environment sensing, (4) two ultrasonic boards for detection of users and low-rise obstacles, (5) a bumper panel to stop the platform under collisions, (6) two tri-axial load cells (MTA400, FUTEK, USA) to estimate the user's navigation commands, (7) a camera (LifeCam Studio, Microsoft, USA) to sense people in the environment, and (8) a 2D laser range finder (LRF) (URG-04LX, Hokuyo, Japan) for user's gait parameters estimation [3].

The device's onboard CPU runs a Linux distribution to support the Robotic Operating System (ROS) framework and the software requirements [3]. Moreover, to ensure efficient processing resources, an external computer is used to off-load non-critical modules. The platform's Ethernet and WiFi modules allow communication with the external CPU [3].

### 10.3.2 UFES Smart Walker

The *UFES Smart Walker*, developed at the Federal University of Espírito Santo, Brazil, is an active three-wheeled walker that provides gait rehabilitation and assistance. The platform is depicted in Fig. 10.2b, as well as its sensory and actuation interfaces.

The smart walker is based on a differential drive configuration with one front caster wheel and two rear motorized wheels. The device is equipped with (1) an encoder at each motorized wheel (H1, US Digital, USA) to estimate the wheel's position and movement, (2) an Inertial Measurement Unit (BNO055, Adafruit, USA) to estimate the platform's orientation, (3) two tri-axial force sensors at each forearm support handlebar (MTA400, Futek, USA) to estimate physical interactions between the user and the platform, (4) a laser rangefinder (LRF) sensor (URG-04LX, Hokuyo, Japan) located in front of the user's legs to obtain user's gait spatiotemporal parameters and distance to the platform, and (5) a 2D Light Detection and Ranging (LiDAR) sensor (RPLIDAR A1, SLAMTEC, CHN) pointing towards the front for environment sensing.

Additionally, the platform is equipped with an onboard computer (PC/104-Plus Standard, 1.67 GHz Atom N450, 2GB RAM). This computer is configured to run a real-time architecture based on the *Matlab Simulink Real-Time xPC Target Toolbox*. An external computer is used for programming purposes of the onboard computer and for experimental data storage.

---

## 10.4 Control Strategies for HRI

One of the significant improvements that have brought the emergence of smart walkers is their ability to acquire and process physical and cognitive interaction with users. Considering that each user may have different health conditions, the smart walkers are often equipped with a wide range of sensors and actuators to meet the particular assistance requirements of the users (see Chap. 2). This section

describes several interaction strategies that have been proposed in the literature to provide natural and compliant HRI.

To follow the design criteria outlined in Table 10.1 and the control architecture illustrated in Fig. 10.1, the first module in an HRI strategy is related to estimating the user's intentions. This is a crucial issue, considering that the outputs of this module are in charge of triggering the smart walker motion, so that it is compliant with the user's motivational demands. A common source of this information is the physical interaction between the user and the smart walker. This interaction is often quantified employing force and pressure sensors on the device's forearm supports and handlebars (see Fig. 10.2). These sensors output force and torque signals that can be used to estimate the user's intentions.

### 10.4.1 Estimation of Physical Interaction

As shown in the *AGoRA Smart Walker* and the *UFES Smart Walker* sensory interfaces, the forces are acquired from the sensors placed on the left and right forearm supports. In particular, tri-axial force sensors can obtain magnitudes along the  $x$ ,  $y$ , and  $z$  axes. In this way, to compute the final exerted force and torque by the user, the following equations are used:

$$\mathbf{F}_Y = (\mathbf{F}_{LY} + \mathbf{F}_{RY}) * \frac{1}{2}, \quad (10.1)$$

$$\mathbf{F}_Z = (\mathbf{F}_{LZ} + \mathbf{F}_{RZ}) * \frac{1}{2},$$

$$\boldsymbol{\tau} = (\mathbf{F}_{LY} - \mathbf{F}_{RY}) * \frac{d}{2}. \quad (10.2)$$

In particular, Eq. 10.1 shows that the resulting impulse force  $\mathbf{F}_Y$  can be estimated by averaging the forces along the  $y$ -axis on both sensors, i.e.,  $\mathbf{F}_{LY}$  and  $\mathbf{F}_{RY}$ , and it provides information about the users' intention to start walking. Similarly, the support force  $\mathbf{F}_Z$  can be estimated using the forces along the  $z$ -axis on both sensors, i.e.,  $\mathbf{F}_{LZ}$  and  $\mathbf{F}_{RZ}$ . This support force is useful to detect the oscillatory components of gait and the posture of the user. Note that the force component along the  $x$ -axis is discarded, as it does not provide any additional or relevant information about the user's intentions.

Regarding the torque  $\boldsymbol{\tau}$ , it provides information about the turning intentions of the users. Equation 10.2 shows that it can be estimated using the difference between the forces along the  $y$ -axis, i.e.,  $\mathbf{F}_{LY}$  and  $\mathbf{F}_{RY}$ , and the sensors' separation distance  $d$ . In this case, the vertical forces are not used to calculate another torque signal, indicating the user's intention to roll the device about the  $x$ -axis.

### 10.4.2 Signals Processing

Current commercial force and torque sensors can extract clear signals, containing meaningful information about the user’s support, propulsion, and turning intentions. However these signals also contain information about the oscillatory patterns of the users’ gait and high-frequency noise related to vibrations produced by the floor [3, 18]. Therefore, before implementing a control strategy based on such interaction forces, a filtering and gait parameters extraction process is required. Consequently, the estimation of the user’s intentions of movement and the user’s navigation commands could be achieved with ease and fewer probabilities to misinterpretations [3].

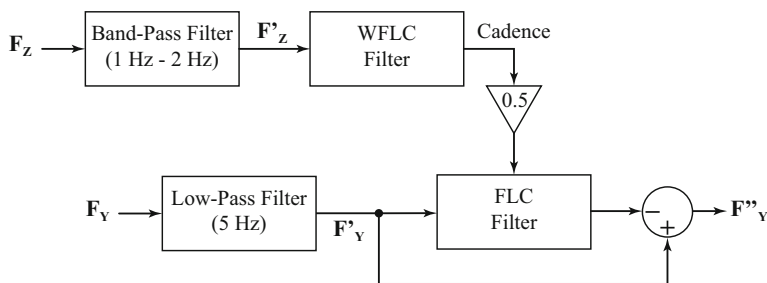
According to the above, there are several alternatives to achieve such a filtering process. Two of them are briefly introduced as follows:

1. **Low-Pass Filters:** These types of filters have been proposed to remove high-frequency noise components. Among these, Gaussian, Blackman, Moving Average, and multiple-pass filters are commonly found. The Moving Average filter is one of the most common techniques in digital signal processing and one of the simplest in terms of implementation and formulation. This type of filter acts as a low-pass filter, although it has poor ability to handle frequency-domain responses. The operation of this filter takes  $M$  input points, estimates the average of those points, and finally produces a single output point. In a force signal processing application, Eq. 10.3 describes the formulation of the Moving Average filter:

$$\mathbf{F}'_{\mathbf{Y}}[i] = \frac{1}{M} \sum_{k=0}^{M-1} \mathbf{F}_{\mathbf{Y}}[i + k]. \quad (10.3)$$

$\mathbf{F}'_{\mathbf{Y}}[i]$  is the filtered force signal, and  $M$  indicates the number of points to average [19]. Considering that this filter is intended to remove random noise, it might not remove oscillatory patterns related to gait. Moreover, given that the estimation of each filtered point requires  $M$  input samples, this filter induces an amount of delay that increases with the  $M$  value. Regarding Gaussian and Blackman variations of this type of filter, they have better stopband attenuation than the Moving Average filter itself. The Gaussian filter, for instance, sets smaller amplitudes near the ends of the averaging window, thus producing smoother results [19].

2. **Adaptive Filters Based on Gait Parameters:** It is well known that the gait pattern exhibits a natural oscillatory behavior, which is commonly related to movements of the human trunk and center of mass in the sagittal plane [3]. In walker-assisted gait applications, the force sensors also capture such movements of the users’ upper body [3]. Thus, the frequency of the gait components that contaminate the force signals is often related to the gait cadence [20].



**Fig. 10.3** Illustration of the adaptive filtering process using Weighted Fourier Linear Combiner (WFLC) and Fourier Linear Combiner (FLC)

In this way, as proposed in [20], an appropriate filtering process of the force signals requires estimating the gait cadence. This process relies on the implementation of adaptive filters, such as (i) the Weighted-Fourier Linear Combiner (WFLC) and (ii) the Fourier Linear Combiner (FLC), which allow the online tracking of quasi-periodic signals [20]. The mathematical formulation of these filters has been previously described in Chap. 5, and thus, their implementation for force signal filtering is outlined here.

As illustrated in Fig. 10.3, the filtering process consists of several steps. On the one hand, the resulting support force  $\mathbf{F}_Z$  (see Eq. 10.1) is passed through a band-pass filter with cutoff frequencies of 1 Hz and 2 Hz to remove the signal's offset and high-frequency noise [3]. Several studies have validated these frequencies in walker-assisted gait applications [3, 17, 20]. Afterward, the first harmonic of the filtered force signal  $\mathbf{F}'_Z$ , i.e., the gait cadence, is estimated by the WFLC.

On the other hand, the resulting impulse force  $\mathbf{F}_Y$  (see Eq. 10.1) is filtered by a fourth-order *Butterworth* low-pass filter with a cutoff frequency of 5 Hz. In parallel, the FLC is fed with the cadence from the WFLC and the filtered signal  $\mathbf{F}'_Y$ . Finally, the FLC outputs the cadence signal  $\mathbf{F}'_{Y\_CAD}$ , which is subtracted from the  $\mathbf{F}'_Y$  to get the final  $\mathbf{F}''_Y$  filtered signal.

The above-mentioned filtering processes are helpful to process and remove undesired components from the force signals acquired from the sensory interfaces of the smart walkers. Note that one can obtain a filtered torque signal if these processes are not carried out with the resulting force signals  $\mathbf{F}_Y$  and  $\mathbf{F}_Z$ , but with the independent signals of each sensor,  $\mathbf{F}_{RY}$ ,  $\mathbf{F}_{LY}$ ,  $\mathbf{F}_{RZ}$ , and  $\mathbf{F}_{LZ}$ .

At this point, it is still necessary to obtain the users' intentions of movement to set appropriate behaviors (i.e., velocities) on the smart walker. To this end, the following section describes one of the most common methods employed to extract velocities from force and torque signals.



### 10.4.3 Motion Intention Detector

As stated in the design criteria, the smart walkers should compliantly respond to users' motivational demands, to guarantee safety and acceptance [3,4]. In this sense, admittance controllers have been widely implemented in smart walkers, as they allow users to control the device by exerting forces and torques on the handlebars or supporting themselves on the devices' forearms [1, 3, 4]. The main idea with these controllers is that the users require less effort to handle the smart walker than to control the device in a passive configuration [1].

In general, the admittance controllers are dynamic models that generate linear and angular velocities from users' intentions [3, 4]. These controllers model the smart walker as two first-order *mass-damper* systems, whose inputs are the resulting impulse force  $\mathbf{F}_Y^1$  and the resulting torque  $\tau$ . The outputs of these controllers are the linear ( $v$ ) and angular ( $\omega$ ) velocities, meaning the user's navigation commands.

To estimate the linear velocity  $\mu(t)$  from the exerted force  $\mathbf{F}_Y(t)$ , the first-order system shown in Eq. 10.4 is used:

$$\mu(t) = \frac{\mathbf{F}_Y(t) - m\dot{\mu}(t)}{b_\mu}, \quad (10.4)$$

where  $m$  is a virtual mass and  $b_\mu$  is the damping constant. Similarly, this system can also be represented in terms of the following transfer function:

$$L(s) = \frac{\mu(s)}{\mathbf{F}_Y(s)} = \frac{\frac{1}{m}}{s + \frac{b_\mu}{m}}. \quad (10.5)$$

The torque  $\tau$  is used to obtain the angular velocity for the smart walker. Using the first-order *mass-damper* system, the angular velocity  $\omega(t)$  can be calculated as shown in the equation below:

$$\omega(t) = \frac{\tau(t) - J\dot{\omega}(t)}{b_\omega}, \quad (10.6)$$

where  $J$  represents virtual inertia and  $b_\omega$  is the damping constant. Similarly, this system can be represented in the frequency domain, as follows:

$$A(s) = \frac{\omega(s)}{\tau(s)} = \frac{\frac{1}{J}}{s + \frac{b_\omega}{J}}. \quad (10.7)$$

In addition to the above, the quality and type of interaction are strongly related to the values of the controllers' constants. In particular, during the selection of

<sup>1</sup>For simplicity, the filtered force is referred to as  $\mathbf{F}_Y$ .

these parameters, it is possible to provide different assistance levels by changing the general virtual stiffness of the platform [3,4]. On the one hand, experimental studies with the *AGoRA Smart Walker* reported that using  $m = 0.5$  kg,  $b_\mu = 4$  N.s/m,  $J = 2.1$  kg.m<sup>2</sup>/rad, and  $b_\omega = 2$  N.m.s/rad, the controllers provided the most effortless and lightest interaction.

On the other hand, experimental studies have also reported that it might be helpful to make the smart walker oppose the users' intentions, i.e., for muscular and gait training purposes. In patients in later stages of rehabilitation, it could be helpful to set the controllers' parameters to render a heavier and more challenging maneuvering experience [21].

To accomplish this, it is assumed that people with higher Body Mass Index (BMI) values can exert higher force and torque values on the device. Therefore, a unique set of parameters is not suitable. In this sense, to provide a resistive mode, the virtual mass should be at least ten times greater than the virtual mass of the previous configuration. The value of the virtual inertia remains unchanged to avoid increasing the risk for falls. An experimental study with the *AGoRA Smart Walker* reported the following values:  $m = 10$  kg,  $b_\mu = \beta$  N.s/m,  $J = 2.1$  kg.m<sup>2</sup>/rad, and  $b_\omega = 7$  N.m.s/rad. The calculation of the damping constant ( $\beta$ ) employs the user's weight, as follows:

$$\beta = 0.375 * weight - 12.5, \quad (10.8)$$

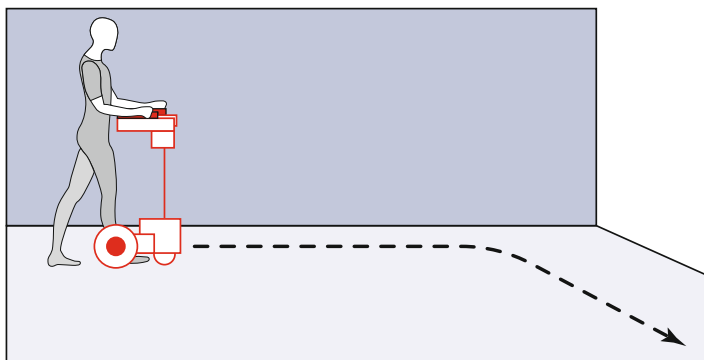
The values of the model presented in Eq. 10.8 were estimated empirically, in such a way that a subject with a maximum weight of 120 kg or a minimum weight of 55 kg could move the device with moderate resistance [21].

#### 10.4.4 HRI Strategy: Case of Study

As an illustration of the previously explained modules, a simple task is proposed. A user is asked to follow an L-shaped path, while the admittance controllers are in charge of generating linear and angular velocities from the force and torque signals (see Fig. 10.4). In this case, the user was asked to walk at the preferred speed, and the first set of constants was used (i.e.,  $m = 0.5$  kg,  $b_\mu = 4$  N.s/m,  $J = 2.1$  kg.m<sup>2</sup>/rad, and  $b_\omega = 2$  N.m.s/rad).

Moreover, for safety purposes, the motion of the smart walker is only allowed if the user is appropriately supporting on the device's forearm supports. This can be achieved by employing the information obtained from the support force  $\mathbf{F}_Z$  and setting a simple threshold. Similarly, another implementation of this safety constraint can be made by using ranging sensors pointing towards the user's legs and setting a distance threshold. If the distance threshold is exceeded, the smart walker stops.

Figure 10.5 illustrates the outcomes of the HRI case study. In particular, Fig. 10.5a shows the raw and filtered signals of the resulting impulse force and the resulting torque. In this case, the filtered force and torque signals in dark red



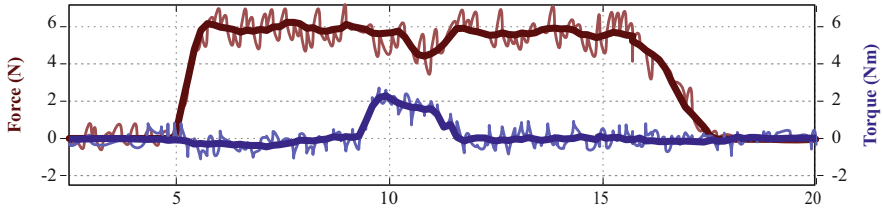
**Fig. 10.4** Description of a simple case of study with the control strategy for HRI

and dark blue, respectively, were obtained by filtering the independent force signals of each sensor. As it can be noted, the filtering process removes both the high-frequency noise and the cadence components. Regarding the generation of linear and angular velocities, Fig. 10.5b shows the obtained velocities from the filtered force and torque signals. These velocities were generated using the admittance controllers described in the previous section.

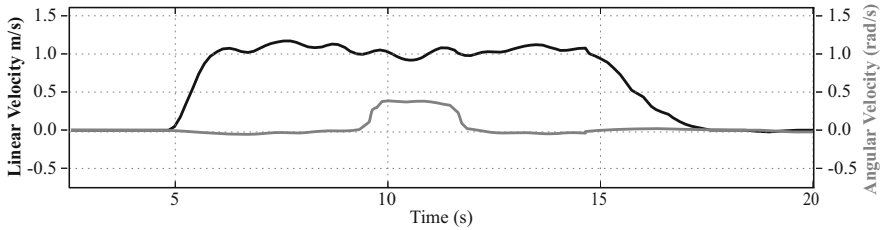
In general, these outcomes highlight how the admittance controllers can extract the users' intentions to move. Particularly, increases in the impulse force are commonly associated with increases in the linear velocity. Analogously, increases or decreases in the torque exerted by the users are associated with turning intentions. Moreover, the turning intentions are also accompanied by a slight decrease in the impulse force. This behavior is explained by the fact that users prefer to perform soft turns, rather than turns around their axis (i.e., 90 degree turns) [3,4].

## 10.5 Control Strategies for REI

Smart walkers are usually deployed in complex and dynamic environments, such as homes, hospitals, and rehabilitation centers. Likewise, smart walkers are often required to provide cognitive support to the user by assisting them in moving tasks. In this sense, the control strategies for REI are designed to provide guidance and path following capabilities. To this end, the smart walkers navigate autonomously and effectively while avoiding static and dynamic obstacles in the environment. According to the above, this section presents the following components for REI: (1) position control, (2) path following control, (3) autonomous navigation, and (4) low-level safety constraints.



(a) Force signals in red and torque signals in blue.



(b) Linear velocity in black and angular velocity in gray.

**Fig. 10.5** Outcomes of the HRI case study. (a) Illustration of raw (light colors) and filtered (dark colors) force and torque signals. (b) Illustration of the linear and angular velocities generated by the admittance controllers

### 10.5.1 Positioning Control

One of the simplest ways to interact with the environment is to use a positioning strategy. This type of strategy allows a robotic walker to be taken from one point to another without defining a particular trajectory. In this scenario, two strategies are outlined below.

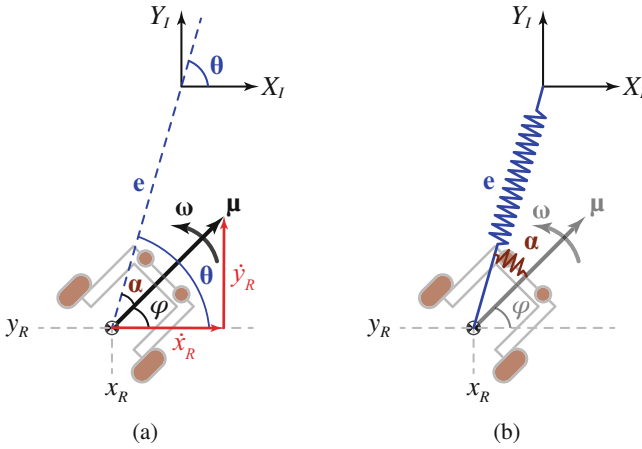
#### 10.5.1.1 Non-linear Position Controller

In a positioning strategy, the controller should safely and naturally take the smart walker to a desired point. Let us describe this problem as shown in Fig. 10.6a, where the kinematic unicycle model is presented in polar coordinates  $(e, \theta)$ . In this scenario, the smart walker is located at the position defined by  $(x_R, y_R)$ , and the desired point is the origin of the inertial reference frame  $(X_I, Y_I)$ .

As explained in Chap. 2, the unicycle kinematic model is described by Eq. 10.9, and the control variables of the robot are  $\mu$  and  $\omega$ :

$$\begin{bmatrix} \dot{x}_R \\ \dot{y}_R \end{bmatrix} = \begin{bmatrix} \mu \cos(\varphi) \\ \mu \sin(\varphi) \end{bmatrix} \tag{10.9}$$

The conversion to polar coordinates is described by Eq. 10.10, where  $e$  is the error distance between the smart walker and the goal, and  $\theta$  is the orientation with respect to the global reference frame:



**Fig. 10.6** (a) Polar coordinates for the unicycle model in positioning problem. The goal is located at the origin of the global reference frame. (b) Formulation of the positioning problem as a mechanical system

$$\begin{bmatrix} X_I \\ Y_I \end{bmatrix} = \begin{bmatrix} e \cos(\theta) \\ e \sin(\theta) \end{bmatrix} \tag{10.10}$$

Considering that  $e^2 = X_I^2 + Y_I^2$  and that the orientation error is defined by  $\alpha = \theta - \varphi$ , the kinematic model is replaced by

$$\begin{bmatrix} \dot{e} \\ \dot{\alpha} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} -\mu \cos(\alpha) \\ -\omega + \mu \frac{\sin(\alpha)}{e} \\ \mu \frac{\sin(\alpha)}{e} \end{bmatrix} \tag{10.11}$$

At this point, the problem is focused on finding a control law that guarantees that  $e \rightarrow 0$ ,  $\alpha \rightarrow 0$ , and  $\theta \rightarrow 0$ , asymptotically. To this end, as described in [22], the Lyapunov-based control method can be applied. In particular, the basic idea of Lyapunov’s direct method is to find the mathematical extension of a physical observation for a given system [23]. In general, this method states that if a mechanical or electrical system’s energy is continuously dissipated, it must eventually stabilize to an equilibrium point [23].

In this way, consider the alternative formulation of the positioning problem shown in Fig. 10.6b. Specifically, the unicycle model is described as a mechanical system that comprises two springs, which are in charge of taking the system to the desired goal. Thus, the candidate Lyapunov function for this system can be formulated by examining the total system energy, i.e., the sum of the energy of the springs, as shown in the equation below:

$$V(e, \alpha) = V_1 + V_2 = \frac{1}{2}e^2 + \frac{1}{2}\alpha^2. \quad (10.12)$$

Moreover, the rate of energy variation of the system is obtained by taking the time derivative of  $V(e, \alpha)$ , as expressed in Eq. 10.13. Physically, this implies that the system will stabilize at the natural length of the springs, i.e., at the desired goal [23].

$$\begin{aligned} \dot{V} &= \dot{V}_1 + \dot{V}_2 = e\dot{e} + \alpha\dot{\alpha} \\ \dot{V} &= e(-\mu \cos(\alpha)) + \alpha \left( -\omega + \mu \frac{\sin(\alpha)}{e} \right). \end{aligned} \quad (10.13)$$

The Lyapunov energy-like function  $V(e, \alpha)$  should be positive definite and should have a continuous first partial derivative to guarantee the stability of the system's equilibrium point stability. Similarly,  $\dot{V}$  should be negative semi-definite. Furthermore, if  $\dot{V}$  is locally negative definite, the stability is asymptotic. The rigorous formulation of this theorem can be found in [23].

According to this,  $\dot{V}_1$  can be made non-positive by choosing:

$$\mu = \lambda e \cos(\alpha), \quad \lambda > 0 \quad (10.14)$$

which yields that

$$\dot{V}_2 = \alpha(-\omega + \lambda \sin(\alpha) \cos(\alpha)), \quad (10.15)$$

and consequently,  $\dot{V}_2$  can also be made non-positive by choosing

$$\omega = k\alpha + \lambda \sin(\alpha) \cos(\alpha), \quad k > 0. \quad (10.16)$$

Then, replacing the equations from  $e$  and  $\alpha$ , it gives

$$\dot{V} = -\lambda e^2 \cos^2(\alpha) - k\alpha^2, \quad (10.17)$$

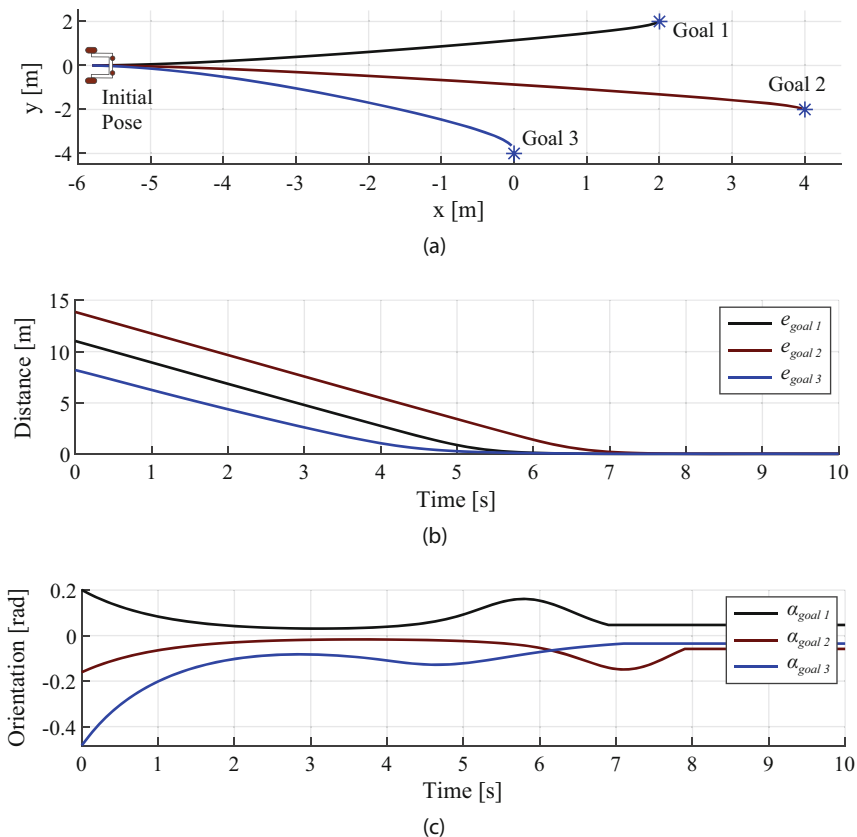
which finally implies that  $e(t), \alpha(t) \rightarrow 0$  with  $t \rightarrow \infty$ .

At this point, the equations of the position controller have been defined by Eqs. 10.14 and 10.16. However, to obtain a safe behavior for  $\mu$  and  $\omega$ , a saturation strategy can be added. Particularly, to avoid motor saturation, the linear and angular velocities can be truncated, by saturating the error  $e$  with the hyperbolic tangent. Thus, the controller is now defined by

$$\begin{bmatrix} \mu \\ \omega \end{bmatrix} = \begin{bmatrix} \lambda \tanh(e) \cos(\alpha) \\ k\alpha + \lambda \frac{\tanh(e)}{e} \sin(\alpha) \cos(\alpha) \end{bmatrix} \quad (10.18)$$

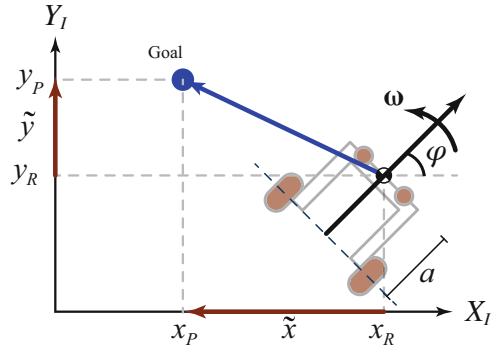
As an illustration of the behavior of this control strategy, Fig. 10.7 shows the outcomes of several positioning tasks with a healthy user. Remarkably, three positioning tasks from the same initial pose were executed (see Fig. 10.7a). Moreover, to demonstrate the asymptotic behavior of the distance error, Fig. 10.7b shows the distance error  $e$  for each goal. Finally, Fig. 10.7c presents the behavior of the steering error  $\alpha$  for each goal. In this case,  $\alpha$  did not exhibit a strict asymptotic behavior. However, it stabilized around an equilibrium point near 0.

It should be noted that this controller does not consider the users' intentions to make the smart walker move. Thus, to guarantee users' safety, this controller can be coupled with a motion triggering system, so that the smart walker only moves if the user is exerting a minimal impulse force.



**Fig. 10.7** Example of three positioning tasks. (a) describes the initial position of the smart walker, the performed trajectory, and the final position. (b) describes the behavior of the distance error  $e$  between the smart walker and each goal. (c) describes the behavior of the orientation error  $\alpha$

**Fig. 10.8** Simple formulation of the positioning problem for the displaced kinematic model



**10.5.1.2 Proportional Position Controller**

An alternative to the previously described non-linear controller is related to the formulation of a simple proportional controller. In this case, consider the position problem described in Fig. 10.8, where the smart walker is modeled with the displaced kinematic model presented in Chap. 2 (see Eq. 10.19).

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \mu \cos(\varphi) - a \omega \sin(\varphi) \\ \mu \sin(\varphi) + a \omega \cos(\varphi) \end{bmatrix} \tag{10.19}$$

This model can be expressed in terms of the kinematic matrix  $C$  (also referred to as Jacobian matrix  $\mathbf{J}$ ) as

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \cos(\varphi) & -a \sin(\varphi) \\ \sin(\varphi) & a \cos(\varphi) \end{bmatrix} \begin{bmatrix} \mu \\ \omega \end{bmatrix} \tag{10.20}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = C \begin{bmatrix} \mu \\ \omega \end{bmatrix}$$

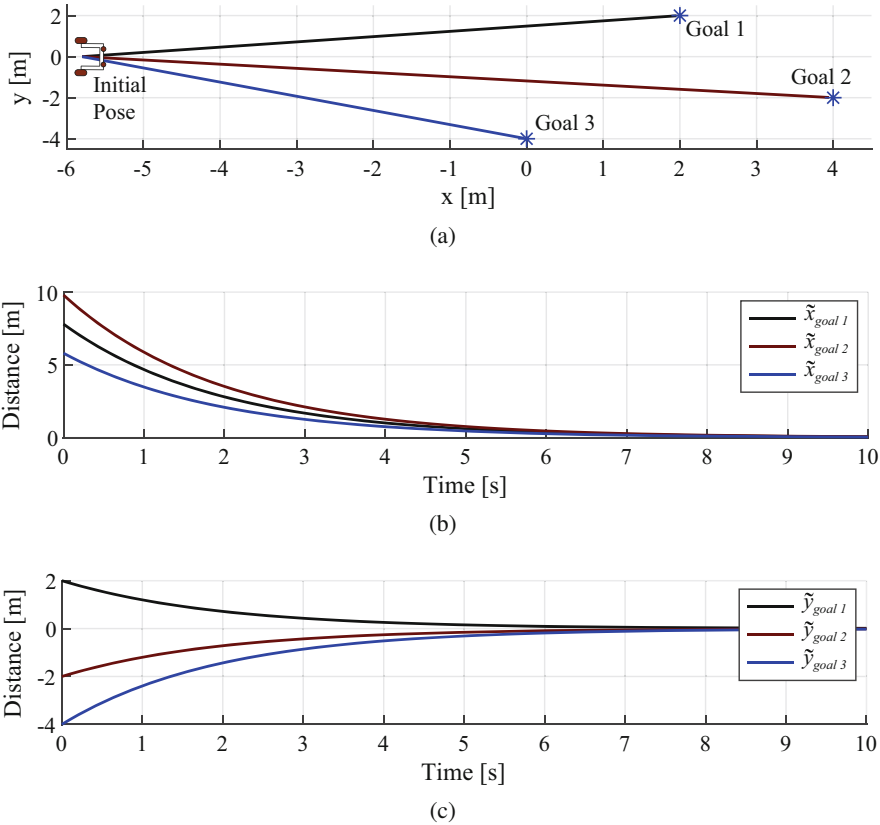
In this context, the formulation of a position controller should provide an expression for the required linear and angular velocities to reach the desired goal. Thus, it can be obtained from Eq. 10.20 that

$$\begin{bmatrix} \mu \\ \omega \end{bmatrix} = \begin{bmatrix} \cos(\varphi) & \sin(\varphi) \\ -\frac{1}{a} \sin(\varphi) & \frac{1}{a} \cos(\varphi) \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} \tag{10.21}$$

$$\begin{bmatrix} \mu \\ \omega \end{bmatrix} = C^{-1} \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}$$

where  $C^{-1}$  is the inverse kinematic (or Jacobian) matrix. At this point, the equations of a proportional controller can be used to define  $[\dot{x}, \dot{y}]^T$ , as follows:





**Fig. 10.9** Example of three positioning tasks with the proportional controller. (a) describes the initial position of the smart walker, the performed trajectory, and the final position. (b) describes the behavior of the error between the  $x$  coordinates of the smart walker and the desired goal. (c) describes the behavior of the error between the  $y$  coordinates of the smart walker and the desired goal

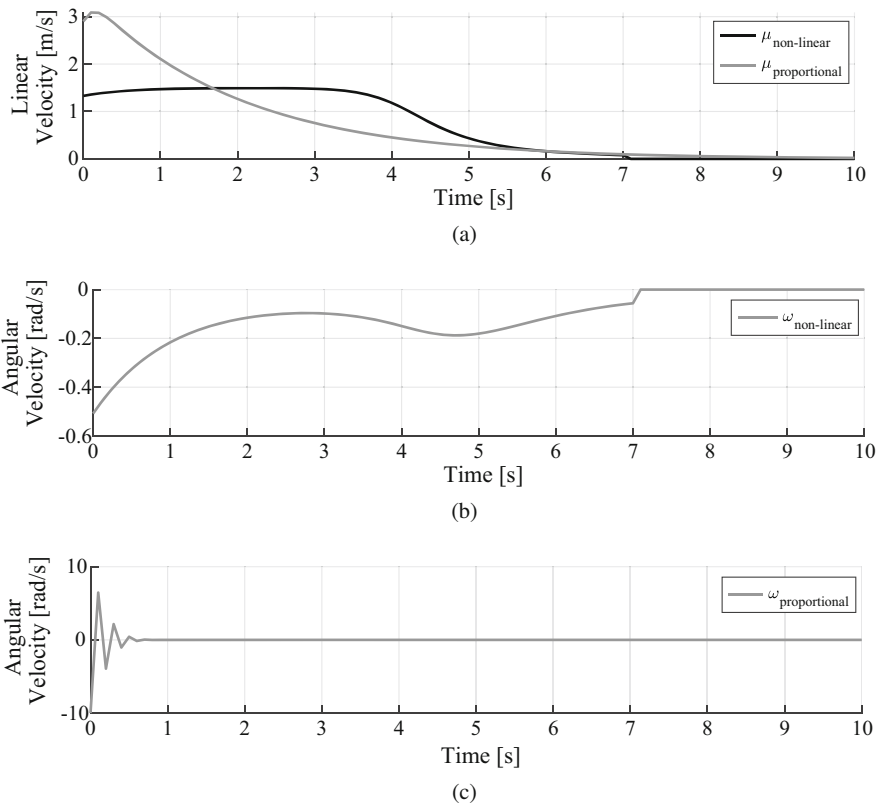
$$\begin{bmatrix} \dot{\tilde{x}} \\ \dot{\tilde{y}} \end{bmatrix} = \begin{bmatrix} K \tilde{x} \\ K \tilde{y} \end{bmatrix} \tag{10.22}$$

$K$  is a proportional gain, while  $\tilde{x}$  and  $\tilde{y}$  are the distance errors between the  $x$  and  $y$  coordinates of the smart walker and the desired goal, respectively. An illustration of the behavior of this controller with a healthy user is shown in Fig. 10.9. The same three goals were used to compare this controller with the previous non-linear strategy (see Fig. 10.9a). Moreover, Fig. 10.9b and c shows the behavior of  $\tilde{x}$  and  $\tilde{y}$ , respectively.

For this controller, it should also be noted that the users' intentions are not taken into account; thus, it is recommended to integrate a motion triggering system. Additionally, the results presented in Fig. 10.9 might suggest that the proportional

controller exhibits a faster and more asymptotic response than the outcomes presented in Fig. 10.7 for the non-linear controller.

However, let us analyze the behavior of the linear and angular velocities generated by these two controllers, for one of the proposed goals (see Fig. 10.10). Regarding the linear velocities illustrated in Fig. 10.10a, the proportional controller generates larger velocity magnitudes, as it does not integrate any saturation strategy. This behavior might lead to sudden unsafe movements of the smart walker in an actual application. The angular velocity generated by the non-linear controller (see Fig. 10.10b) exhibits a soft behavior that does not impose considerable effort on the robot's actuators. Nevertheless, the angular velocity generated by the proportional controller exhibits more aggressive behavior that could induce unsafe movements in the smart walker (see Fig. 10.10c). Furthermore, this controller saturates the robot's actuators.



**Fig. 10.10** Comparison of linear and angular velocities generated by the two positioning strategies. (a) Linear velocities generated by the two controllers. (b) Angular velocity generated by the non-linear controller. (c) Angular velocity generated by the proportional controller

### 10.5.2 Path Following Control

As stated in the previous chapters, a valuable functionality of smart walkers is related to guiding users with cognitive and physical impairments. In general, a guiding task aims at taking the user through the desired route, consisting of several predetermined poses or goals. A standard solution for path following in *wheeled mobile robots* consists of proposing a controller that generates the required linear and angular velocities to achieve the desired route.

#### 10.5.2.1 Non-linear Path Following Controller

Let us consider the path following problem described in Fig. 10.11, where the smart walker is also modeled with the displaced kinematic model, previously described in Eq. 10.19.

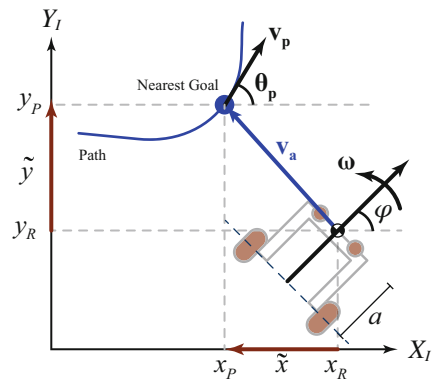
This model can also be expressed in terms of the kinematic matrix  $C$ , as shown in Eq. 10.20. In this context, the inverse kinematic matrix  $C^{-1}$  is again used to obtain expressions for the required linear and angular velocities to follow a particular path (see Eq. 10.21).

Thus, finding the equations of the path following controller relies on defining proper expressions for  $[\dot{x}, \dot{y}]^T$ . In particular, Andaluz *et al.* proposed a set of equations for this formulation, which have been widely used in mobile robotics applications [24]. For every path pose, the closed-loop equation of the controller proposed in [24] is represented as follows:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = v_p + v_a, \tag{10.23}$$

where  $v_p$  is the desired velocity vector on the path and  $v_a$  is an attraction vector to the path. In this sense, Eq. 10.23 can be further expressed as

**Fig. 10.11** Illustration of the path following problem, where the smart walker should reach a desired point on the route



$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} v_r \cos(\theta_p) + l_x \tanh\left(\frac{k_x}{l_x} \tilde{x}\right) \\ v_r \sin(\theta_p) + l_y \tanh\left(\frac{k_y}{l_y} \tilde{y}\right) \end{bmatrix}, \quad (10.24)$$

where  $v_r$  is the magnitude of the desired velocity on the path;  $\theta_p$  is the reference orientation of the path, defined by the tangent of the nearest point to the path;  $l_x$  and  $l_y$  determine the saturation limits of the position error;  $k_x$  and  $k_y$  are constant gains that establish the linear zone of the position error; and  $\tilde{x}$  and  $\tilde{y}$  are the position errors of the smart walker with respect to the path [4, 24].

With these equations, it is possible to implement a path following strategy in any wheeled mobile robot that can be modeled as shown in Eq. 10.19. Note that in the case of walker-assisted gait applications, this solution does not consider the users' intentions, and thus it assumes that the user will follow the smart walker's motion. However, to avoid unsafe situations, this strategy can be coupled with a simple motion triggering system, so that the motion of the smart walker is only enabled if the user exerts a minimum impulse. Such impulse force will indicate that the user is ready to start walking with the device.

As an illustration of this control strategy, Fig. 10.12 describes the outcomes of a path following task with a healthy user, where the desired route was configured as a lemniscate curve. Figure 10.12a also shows the initial pose of the smart walker and the pose after 5 s ( $t_5$ ). The blue asterisk indicates the desired goal on the route at  $t_5$ . Moreover, Fig. 10.12b shows the distance errors  $\tilde{x}$  and  $\tilde{y}$  during the execution of the task.

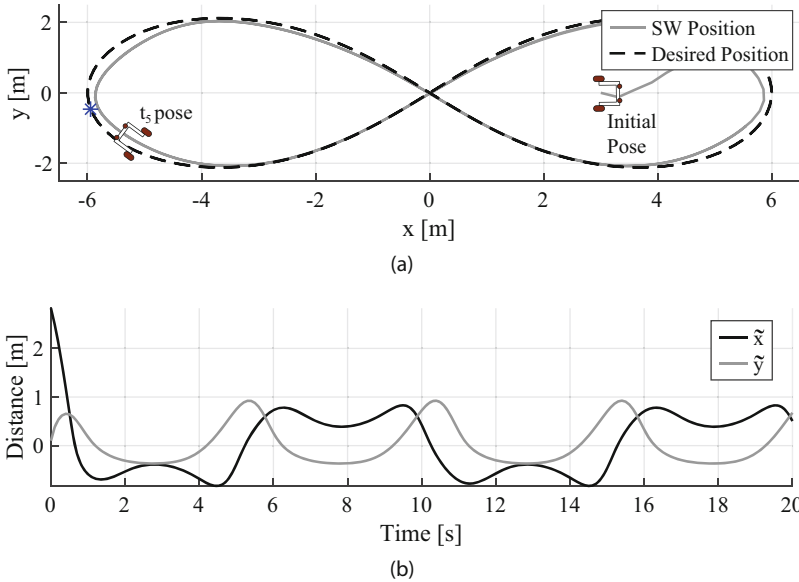
### 10.5.2.2 Proportional Path Following Controller

Another simple yet valuable solution to following a desired route is based on the formulation of a proportional controller. Let us assume the same formulation shown in Fig. 10.11, modeling the smart walker with the displaced kinematic model described by Eq. 10.20. Once again, the equations for the linear and angular velocities can be obtained from Eq. 10.21, and it is required to formulate an expression for  $[\dot{x}, \dot{y}]^T$ .

In this case, the closed-loop equation of the controller can be defined as

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \Delta x + K \tilde{x} \\ \Delta y + K \tilde{y} \end{bmatrix}, \quad (10.25)$$

where  $\Delta x$  and  $\Delta y$  correspond to the difference of the  $x$  and  $y$  coordinates between the current and the next desired point on the route, respectively. Likewise,  $\tilde{x}$  and  $\tilde{y}$  are the position errors between the smart walker and the desired goal on the path, and  $K$  is a proportional constant. Similar to the previous controller, this formulation does not consider the users' intention to move. Therefore, a motion triggering system is required only to make the smart walker move, when the user exerts a minimum impulse force. An illustration of this control strategy is shown in Fig. 10.13.



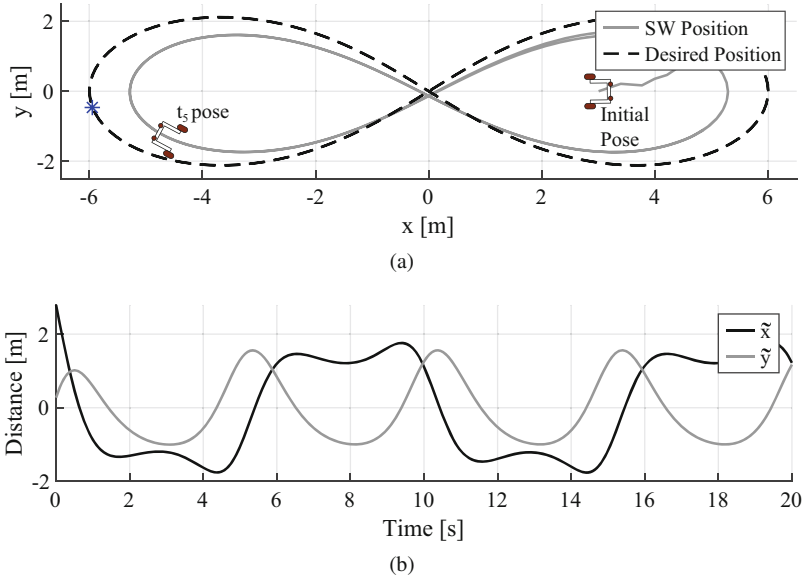
**Fig. 10.12** Example of a path following task using the non-linear path controller proposed by Andaluz et al. [24] (a) describes the position of the smart walker during the task. (b) shows the distance errors during the task

Figure 10.13a compares the desired route and the position of the smart walker during the task. It also shows the initial pose of the smart walker and the pose after 5 s ( $t_5$ ). The blue asterisk indicates the desired goal on the route at  $t_5$ . Moreover, Fig. 10.13b shows the distance errors  $\tilde{x}$  and  $\tilde{y}$  during the execution of the task.

In this case, the errors between the smart walker and the route are larger, and thus the smart walker does not follow the exact desired route. Moreover, considering that the controller is based on a proportional equation, it might generate large velocities when the error is not zero. This can cause saturation on the smart walker’s motor and abrupt movement of the device. In this way, this controller could be helpful to provide path following; however, it should be softened with saturation prevention strategies. Likewise, the formulation of this controller can be extended to versions including integral and derivative gains.

### 10.5.3 Autonomous Navigation

Navigation during walker-assisted gait is mainly focused on safety provision while guiding the users through different environments. Such guiding might respond to the users’ intentions, e.g., “*I want to go to my room,*” or be used in rehabilitation scenarios to perform circuit-based tasks. The concept of autonomous navigation often differs from the previous position and path following controllers, as these controllers



**Fig. 10.13** Example of a path following task using a simple proportional controller a. The desired route and the position of the smart walker during the task b. The distance errors during the execution of the task

often require a static environment with controlled conditions. Moreover, one of the significant benefits of autonomous navigation is related to the online estimation and adaptation of routes based on the environment's dynamic characteristics. Overall, the fundamentals of autonomous navigation are based on four concepts [25].

1. **Perception:** This refers to the ability of the robotic device to acquire information from the environment through sensory interfaces and extract meaningful information. This is often accomplished by ranging sensors (e.g., ultrasonic sensors and laser rangefinders), cameras, and depth cameras, among others.
2. **Localization:** This concept states that the robot must determine and track its position (i.e., odometry) in known and unknown environments. Commonly, this concept is also related to the robot's ability to create and update maps of its environment.
3. **Decision-Making:** Given a set of mission commands or goals, the navigation system must decide how to achieve such objectives. Similarly, this concept also states that the robot should determine if a particular goal is feasible or not and to determine when to perform recovery strategies, e.g., the localization is distorted. Moreover, this also refers to making plans and updating them if the environment conditions change.

4. **Motion Control:** Finally, with a particular plan or path to follow, the robot must determine the appropriate motor commands to achieve such a plan. This concept is very similar to the path following and position controllers described previously.

In this context, the navigation systems in smart walkers employ the same navigation's concepts for wheeled mobile robots. It comprises map building, autonomous localization, obstacle avoidance, and path following strategies [3]. Moreover, for simplicity, these concepts will be introduced in a high-level context, and their implementation will be based on the functionalities of the Robot Operating System (ROS) navigation stack.

#### 10.5.3.1 Requirements for Navigation

Several hardware and software considerations should be met so that a wheeled mobile robot (or a smart walker) can integrate a navigation system, such as the one provided by ROS.

First, the robot should have a **transform configuration** system that provides information about the relationships between the coordinate frames of the robot. Typically, every sensor, the robot's decks and the robot's base have a coordinate frame. Thus, the transform configuration defines offsets in translation and rotation between different coordinate frames [26]. Second, the device must implement a reliable **odometry system** to estimate its position, orientation, and velocity [27]. Third, the robot must equip and properly acquire information from **range sensors**. Preferably, laser rangefinders or LiDARs are recommended as they provide sufficient information about the surrounding obstacles [28]. Finally, the robot must integrate a low-level **base controller** in charge of converting the linear and angular velocities generated by the navigation system to independent motors' commands [29].

The following sections describe the main functional components required for an autonomous navigation system, focused on the ROS navigation stack.

#### 10.5.3.2 Localization and Map Building

The problem of robot's localization has been an active research area in the last decades, since several methodologies for both indoors and outdoors applications have been proposed [25]. In general, every localization strategy requires an environment map that could be made by hand or automatically with ranging and odometry sensors.

For mapping, the ROS navigation stack offers an implementation of a 2D map building algorithm based on the Simultaneous Localization and Mapping (SLAM) technique [3]. This strategy has emerged considering that robots often equip sensors with limited ranges. Therefore the robots are obligated to create maps while exploring and localizing themselves in an unknown environment [25]. Specifically, the *GMapping* ROS package aims at creating a static map of the complete interaction environment [30]. The static map is made offline and focuses on defining the main constraints and characteristics of the environment [3]. For instance, in a walker-assisted gait application in clinical environments, the map of the environment should be built before any interactions with users or patients.

Once the desired map is obtained, it is also used for the robot's online localization. To this end, the navigation stack implements the Adaptive Monte Carlo Localization Approach (AMCL) [31]. This is a probabilistic strategy that uses a particle filter to estimate all the possible poses of the robot within the map. With the robot's motion, the filter starts to converge to a single localization [32].

A common problem related to these strategies is often associated with zones such as stairs, elevator entrances, corridor railings, and glass walls. These zones are defined as non-interaction unsafe zones, and the ranging sensors such as LiDARs cannot completely sense their physical surface. Moreover, these zones are also restricted to the robot, mainly due to the risk of collisions and falls [1]. In this sense, and considering that a gray-scale image often represents the environment's map, these restrictions are achieved by editing the resulting static map [3].

It is worth mentioning that robots can also perform autonomous navigation without having a map of the environment. In this case, the robot can only plan and execute motion tasks that are limited to the field of view of its ranging sensors.

### 10.5.3.3 Path Planners and Cost Maps

One of the major functionalities of a navigation system is to receive a mission command or a desired goal and plan an optimal route to attain such an objective. In this sense, to perform path planning, the ROS navigation system implements three main concepts: (1) cost maps, (2) a global planner, and (3) a local planner.

A cost map consists of an occupancy grid, where every detected obstacle is represented as a cost, and it is elaborated from the previous edited map. To define the numerical value of an obstacle's cost, several aspects are considered. For instance, the distance that the robot is allowed to approach the obstacles, and this process is called obstacles inflation [3]. There is a global cost map, as well as a local cost map. The global cost map is generated by inflating the obstacles on the edited map. The local cost map is generated online by inflating obstacles detected by the field of view of the robot's sensors. These cost maps are also capable of semantically separating the obstacles in several layers [33]. Frequently, a navigation system employs a *static map layer*, an *obstacle layer*, a *sonar layer*, and an *inflation layer* [33]. Moreover, if the robot can detect people in the environment, a *social layer* can also be used to integrate proxemics and social zones.

Regarding the global planner, it executes two tasks. First, the global planner checks if the desired goal is feasible; if not, a near feasible goal is automatically proposed. Second, it seeks to find a global route with a minimum cost between the start point (i.e., current robot's position) and the endpoint (i.e., desired goal). To this end, ROS provides implementations of path planning algorithms such as Dijkstra's and A\* algorithms [32].

Regarding the local planner, it provides a controller that drives the robot in the environment. Using the local cost map, the local planner creates a kinematic trajectory for the robot to get from a start point to a goal location [3]. Specifically, the Trajectory Rollout and the Dynamic Window Approach (DWA) plan local paths based on environmental data and sensory readings [32, 34]. To determine the required linear and angular velocities to execute the local plan, the DWA performs



forward simulations from the robot's current state to predict possible velocities and trajectories [32]. Each simulation is scored by metrics, such as proximity to obstacles, proximity to the goal, proximity to the global path, and speed [32]. The trajectories that collide with obstacles are considered illegal and thus discarded. Finally, the highest-scoring trajectory is chosen and the associated velocity is sent to the robot's motion controller [32].

#### 10.5.3.4 Considerations for Smart Walkers

Accordingly, a navigation system is of great relevance in walker-assisted gait applications for guiding and path following tasks [1, 3]. However, as in the previous controllers, this system does not consider the user's movement intentions. In this sense, for the navigation system to be safe and intuitive, the walker must move only when the user wants it.

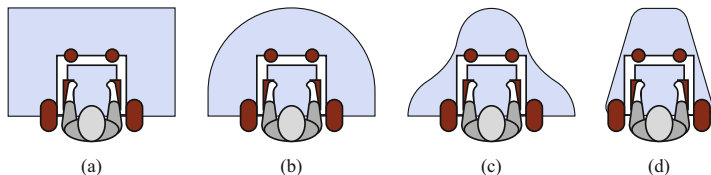
There are several alternatives to solve this. One option is to leave control of the smart walker's linear speed to the user through an admittance controller, while the navigation system controls the angular speed. This allows the users to follow the planned route at their speed. Another option is to implement a triggering system, as mentioned in the previous controllers. In this case, the movement of the smart walker is only allowed if the user is exerting a minimum impulse force. Also, for this, a maximum navigation speed must be configured to be comfortable for the user.

In addition to the above, it is essential to configure the navigation system so that the robotic walker does not make turns on its axis. In other words, a minimum turning radius must be established to prevent users from stumbling and thus reduce the risk of falls.

#### 10.5.4 Low-Level Safety Supervisor

This module is in charge of guaranteeing users' safety in case of malfunctioning of the above-described control strategies. In particular, the *AGoRA Smart Walker* and the *UFES Smart Walker* have reported several safety rules that constraint the walker's movement when hazardous situations are detected [3, 4]. As proposed in Fig. 10.1, the safety supervisor should be implemented to override the control strategies, if required. In general, the supervisor monitors two main safety conditions.

1. **User condition:** In this case, the device movement is only allowed if the user supports himself/herself on the walker handlebars and properly stands behind it. This information can be obtained from the force or pressure sensors on the forearm supports and from ranging sensors pointing towards the user.
2. **Warning zone condition:** Using the information gathered from the ranging sensors mounted on the device, the walker's speed can be constrained when surrounding obstacles are detected. In this sense, an area of interest must be defined around the robotic walker, for which obstacles will be taken into account. This is known as a warning zone. There is no definitive warning zone since, depending on the application, the context, or the user's requirements, one zone or



**Fig. 10.14** Illustration of possible warning zones to detect surrounding obstacles and constraint the smart walker's motion. **(a)** Square zone. **(b)** Semi-circular zone. **(c)** Gaussian zone. **(d)** Conic zone

another may be chosen. In particular, Fig. 10.14 shows some applicable warning zones. Regardless of the shape of the warning zone, the speed limitation goes as follows [3]:

- a. The readings from ranging sensors are processed to detect surrounding obstacles. Clustering algorithms can be helpful for such processing.
- b. Only the obstacles in the warning zone are taken into account.
- c. The smart walker's velocity is constrained proportionally to the distance between the smart walker and the obstacle.
- d. A stopping distance is defined, so that if the obstacle is at this distance or closer, the robot comes to a complete stop. In this case, only angular velocities are allowed, to let the user avoid the obstacle.

On the other hand, smart walkers are constantly monitored by healthcare professionals or researchers. In this sense, if a device malfunctioning occurs, they can remotely disable, fix, or stop the device. The supervisor's safety restrictions should always be redundant. That is to say, they are executed from the onboard computer, as well as from external computers. In case of communication loss with the external computer, the device can continue running the safety supervisor autonomously.

## 10.6 Conclusions

The ability of robotic walkers to interact with the user or the environment is due to their actuators, sensory interface, and the implementation of control strategies. These strategies allow obtaining specific behaviors such as responding to the user's movement intentions, guiding a user between two points or along a trajectory.

In this sense, this chapter presented some of the control strategies that are most commonly used in robotic walkers to ensure Human–Robot interaction (HRI), Robot–Environment interaction (REI), and Human–Robot–Environment Interaction. It is essential to clarify that some of these strategies have already been implemented in wheeled mobile robots, which are not necessarily related to rehabilitation. Thus, in these sections, concepts of applying these strategies in walker-assisted gait applications were given.

## References

1. S. Sierra, L. Arciniegas, F. Ballen-Moreno, D. Gomez-Vargas, M. Munera, C.A. Cifuentes, Adaptable robotic platform for gait rehabilitation and assistance: design concepts and applications, in *Exoskeleton Robots for Rehabilitation and Healthcare Devices* (Springer, 2020), pp. 67–93
2. T. Mikolajczyk, I. Ciobanu, D.I. Badea, A. Iliescu, S. Pizzamiglio, T. Schauer, T. Seel, P.L. Seiciu, D.L. Turner, M. Berceanu, Advanced technology for gait rehabilitation: An overview. *Adv. Mech. Eng.* **10**, 1–19 (2018)
3. S.D. Sierra M., M. Garzón, M. Múnera, C.A. Cifuentes, Human–Robot–environment interaction interface for smart walker assisted gait: AGoRA walker. *Sensors* **19**, 2897 (2019)
4. M.F. Jiménez, M. Monllor, A. Frizera, T. Bastos, F. Roberti, R. Carelli, Admittance controller with spatial modulation for assisted locomotion using a smart walker. *J. Intell. Robot. Syst.*, 1 (2018)
5. G.J. Lacey, D. Rodriguez-Losada, The evolution of guido. *IEEE Robot. Autom. Mag.* **15**(4), 75–83 (2008)
6. E. Efthimiou, S.-E. Fotinea, T. Goulas, A.-L. Dimou, M. Koutsombogera, V. Pitsikalis, P. Maragos, C. Tzafestas, The MOBOT platform – Showcasing multimodality in human-assistive robot interaction, in *Universal Access in Human-Computer Interaction. Interaction Techniques and Environments* (Springer, 2016), pp. 382–391
7. S. Sierra, M. Jimenez, M. Munera, T. Bastos, A. Frizera-Neto, C. Cifuentes, A therapist helping hand for walker-assisted gait rehabilitation: A pre-clinical assessment, in *4th IEEE Colombian Conference on Automatic Control: Automatic Control as Key Support of Industrial Productivity, CCAC 2019 - Proceedings* (2019)
8. P. Boissy, S. Briere, H. Corriveau, A. Grant, M. Lauria, F. Michaud, Usability testing of a mobile robotic system for in-home telerehabilitation, in *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society* (IEEE, 2011), pp. 1839–1842
9. Y. Koumpouros, A systematic review on existing measures for the subjective assessment of rehabilitation and assistive robot devices. *J. Healthcare Eng.* **2016**, 1–10 (2016)
10. J. Varela, R.J. Saltaren, L.J. Puglisi, J. López, M. Alvarez, J.C. Rodríguez, User centred design of rehabilitation robots, in *Advances in Automation and Robotics Research in Latin America. Lecture Notes in Networks and Systems* (Springer, Cham, 2017), pp. 97–109
11. A.A. Ramírez-Duque, L.F. Aycardi, A. Villa, M. Munera, T. Bastos, T. Belpaeme, A. Frizera-Neto, C.A. Cifuentes, Collaborative and inclusive process with the autism community: A case study in Colombia about social robot design. *Int. J. Soc. Robot.* **13**, 153–167 (2021)
12. D. Casas-Bocanegra, D. Gomez-Vargas, M.J. Pinto-Bernal, J. Maldonado, M. Munera, A. Villa-Moreno, M.F. Stoelen, T. Belpaeme, C.A. Cifuentes, An open-source social robot based on compliant soft robotics for therapy with children with ASD. *Actuators* **9**, 91 (2020)
13. M.M. Martins, C.P. Santos, A. Frizera-Neto, R. Ceres, Assistive mobility devices focusing on Smart Walkers: Classification and review. *Robot. Auton. Syst.* **60**(4), 548–562 (2012)
14. M. Martins, C. Santos, A. Frizera, R. Ceres, A review of the functionalities of smart walkers. *Med. Eng. Phys.* **37**, 917–928 (2015)
15. M.F. Jiménez, R.C. Mello, T. Bastos, A. Frizera, Assistive locomotion device with haptic feedback for guiding visually impaired people. *Med. Eng. Phys.* (2020)
16. A. Wachaja, P. Agarwal, M. Zink, M.R. Adame, K. Möller, W. Burgard, Navigating blind people with walking impairments using a smart walker. *Autonomous Robots* **41**, 555–573 (2017)
17. C.A. Cifuentes, A. Frizera, *Human-Robot Interaction Strategies for Walker-Assisted Locomotion*, vol. 115 of *Springer Tracts in Advanced Robotics* (Springer International Publishing, Cham, 2016)
18. M.A.D. Brodie, T.R. Beijer, C.G. Canning, S.R. Lord, Head and pelvis stride-to-stride oscillations in gait: validation and interpretation of measurements from wearable accelerometers. *Physiological Measurement* **36**, 857–872 (2015)

19. S.W. Smith, Moving average filters, in *The Scientist and Engineer's Guide to Digital Signal Processing*, ch. 15, 2nd edn. (California Technical Publishing, San Diego, 1999), pp. 277–284
20. A. Frizera, J. Gallego, E. Rocon de Lima, A. Abellanas, J. Pons, R. Ceres, Online cadence estimation through force interaction in walker assisted gait, in *ISSNIP Biosignals and Biorobotics Conference 2010* (Vitória, 2010), pp. 1–5
21. S.D. Sierra M., M. Munera, T. Provot, M. Bourgain, C.A. Cifuentes, Evaluation of physical interaction during walker-assisted gait with the AGoRA walker: Strategies based on virtual mechanical stiffness. *Sensors* (In revision) (2021)
22. S.G. Tzafestas, Mobile robot control I, in *Introduction to Mobile Robot Control* (Elsevier, 2014), pp. 137–183
23. J. Slotine, J. Slotine, W. Li, *Applied Nonlinear Control* (Prentice Hall, 1991)
24. V.H. Andaluz, F. Roberti, J.M. Toibero, R. Carelli, B. Wagner, Adaptive dynamic path following control of an unicycle like mobile robot, in *Intelligent Robotics and Applications*, ch. 56 (Springer Berlin Heidelberg, 2011), pp. 563–574
25. R. Siegwart, I.R. Nourbakhsh, D. Scaramuzza, *Introduction to Autonomous Mobile Robots*, 2nd edn. (MIT Press, 2011)
26. ROS, Setting up your robot using tf (2021)
27. ROS, Publishing odometry information over ROS (2019)
28. ROS, Publishing sensor streams over ROS (2012)
29. ROS, Setup and configuration of the navigation stack on a robot (2018)
30. G. Grisetti, C. Stachniss, W. Burgard, Improved techniques for grid mapping With Rao-Blackwellized particle filters. *IEEE Trans. Robot.* **23**, 34–46 (2007)
31. D. Fox, W. Burgard, F. Dellaert, S. Thrun, Monte Carlo localization: efficient position estimation for mobile robots, in *AAAI-99*, no. Handschin, vol. 1970, pp. 343–349 (1999)
32. K. Zheng, ROS navigation tuning guide (2016)
33. D.V. Lu, D. Hershberger, W.D. Smart, Layered costmaps for context-sensitive navigation, in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IEEE, 2014)*, pp. 709–715
34. D. Fox, W. Burgard, S. Thrun, The dynamic window approach to collision avoidance. *Robot. Autom. Mag.*, 1–23 (1997)