A Real-Time 3D Reconstruction of Staircases for Rehabilitative Exoskeletons

Marina Raineri, *Member, IEEE*, Riccardo Monica, *Member, IEEE*, and Corrado Guarino Lo Bianco, *Senior Member, IEEE*

***Abstract*—In medical contexts, the use of assistive exoskeletons for the rehabilitation of people with impaired mobility represents a common practice. Recent advances suggest that, soon, such mechatronic systems will also be used to assist people in their everyday life. In order to reach such target, exoskeletons must become able to perceive the environment. To this purpose, a system for the parametric identification of a staircase is proposed in this article. More precisely, given a staircase of unknown geometry, the system identifies its 3D shape. Furthermore, it also estimates the reciprocal orientation and distance between the exoskeleton and the staircase. Differently from other approaches, this result is achieved by means of low cost devices: an inertial measurement unit, two ranging sensors, and an Arm-Cortex processor. Starting from the ranging sensors acquisitions, the staircase model is identified in real time, during the execution of a step. The proposed procedure is based on an extended recursive total least squares strategy, in order to fully exploit the computational capabilities of the Arm processor, and it is characterized by execution times smaller than 10^{-3} s. The estimation algorithm has been tested on an actual exoskeleton and the resulting experimental outcomes are compared with the results obtained through alternative methods.**

***Index Terms*—Staircases 3D reconstruction, ranging sensor, lower-limb exoskeleton.**

I. Introduction

Exoskeletons are assistive devices which can be used in different contexts. In particular, they can be adopted to reduce the physical efforts of workers but, more frequently, they represent powerful instruments for rehabilitation of injured people or the assistance of patients with limited motion capabilities. An exhaustive review on the actual therapeutic uses and performances of lower limb exoskeletons is proposed in [1]. The Authors underline that almost all the studies in the field neglect the system dynamics. This is probably motivated by the fact that commercial exoskeletons are currently not equipped with self-balancing systems. Apart from possible technical reasons which could motivate such choice, another, practical one, can be emphasized. At the moment, commercial exoskeletons are not directly driven by patients but, conversely, they are operated by specialized physiotherapists who activate, depending on the therapy, proper pre-programmed gaits. The motion is balanced by the patient, assisted by the operator, through the use of crutches. Consequently, at a control level, only kinematics problems need to be handled.

Large part of the papers mentioned in [1] focus on straight walking movements, while other human activities are poorly covered. In particular, the ability to ascend and descend staircases is addressed in a few works and, due to the previously mentioned reasons, it is handled through the adoption of repetitive movements. Such choice, which does not pose particular problems on flat surfaces, complicates the staircases climbing motion. In fact, steps geometry changes depending on the architectonic context and, moreover, the approach to the first tread must be adapted depending on the initial distance between the exoskeleton and the staircase: with commercial devices, such uncertainties are handled, through relevant physical efforts, by means of the crutches.

This considerations suggest equipping the exoskeleton with sensors, so as to detect the surrounding environment and, consequently, to adapt its gait. In recent years, a branch of the research activity in this field focused on the gait identification problem, managed through the use of different sensors like, for example, Inertial Measurement Units (IMU) and force-sensitive resistors. On the basis of the acquired data the Authors of [2], [3] were able to determine if the exoskeleton is walking on a straight ground, on a ramp, or on a staircase. In [4], the actual gait phase was identified. The common aim of these studies was to compute, depending on the obstacle, the most appropriate exoskeleton reaction. For example, in [5]–[7] different methodologies have been developed in order to help people to climb staircases of known geometry.

When the environment is unknown, two alternative strategies can be adopted. In the first, the perception capabilities of the person who wears the exoskeleton – acquired for example by means of electrophysiological measurements sensors, such as electromyography or electroencephalography sensors – are used to directly drive the unit [8]. In the second, the environment must be perceived through devices which are typically based on Time-of-Flight (ToF) technologies. Laser scanners are the most frequently used devices. They allow one obtaining, with a single sweep, a complete point cloud of the environment. Such point cloud needs to be post-elaborated in order to convert it into parametric surfaces.

Manuscript received October 11, 2020; revised December 8, 2020; accepted January 5, 2021. Date of publication January 11, 2021; date of current version February 22, 2021. This article was recommended for publication by Associate Editor M. Munih and Editor P. Dario upon evaluation of the reviewers' comments. *(Corresponding author: Corrado Guarino Lo Bianco.)*

The authors are with the Dipartimento di Ingegneria e Architettura, University of Parma, 43124 Parma, Italy (e-mail: marina.raineri@unipr.it; riccardo.monica@unipr.it; guarino@ce.unipr.it).

Digital Object Identifier 10.1109/TMRB.2021.3050561

The resulting elaboration represents a challenging task, which requires specific computational resources.

Early works concerning the plane fitting problem appeared in [9], [10]. The first paper estimates both the best fitting plane and its reliability, while the second one proposes a model which also considers the noise in range data. In the subsequent years, alternative works addressed the same problem by using data obtained from different laser sensors. In [11], 3D georeferenced data were used and two algorithms, based on the planar regression and the moment of inertia, were proposed; a SICK laser scanner was used in [12] and a probabilistic plane fitting concept was suggested for the point cloud elaboration. An alternative method, which still requires a high computational burden, appeared in [13]. It was based on the recognition of common primitives, such as cuboids or cylinders, and on a neighboring technique. A cheaper solution was adopted in [14]. In that paper a 2D laser scanner, driven by a stepper motor so as obtain a 3D reconstruction, was used for the acquisition of the point cloud. In a recent work, focused on the estimation of the environmental features for an exoskeleton application [15], an IMU and a depth camera were used to generate a 3D point cloud. Such point cloud was then processed through a neural network, so as to classify the surrounding objects. A survey concerning the known plane fitting methods can be found in [16]. Such paper underlines that, in practical applications, a trade-off between speed and quality must be reached.

Other works in the literature specifically address the staircase identification problem. A visual system was implemented in [17] for a humanoid robot, while in [18] a NAO robot was additionally equipped with a laser range finder. A stepper motor and a 2D laser scanner were mounted in [19] on a hexapod robot in order to acquire 3D point cloud of a staircase. Finally, an exoskeleton application was considered in [20]: the staircase was identified by means of a depth camera mounted on the patient chest.

All the mentioned techniques can be divided into two main categories: those based on visual systems and those adopting laser scanners. The use of such identification systems in commercial exoskeletons is prevented, in part, by cost reasons. For, example, laser scanners are typically expensive – the cheapest ones cost some hundred dollars – and their dimensions are inadequate for the slim structures of the exoskeletons: the dimension of the smallest Sick sensor is equal to $(6.0 \times 6.0 \times 8.6) \cdot 10^{-2}$ m. Approaches based on visual systems may admit reduced sensor costs, but they still assume the acquisition of a point cloud, which is later elaborated by means of dedicated boards with sufficient computational capabilities.

This article proposes a solution to the plane fitting problem for an exoskeleton, based on low cost devices. In particular, the target is represented by the parametric identification of a staircase posed in front of the exoskeleton. The identification methods proposed in the literature – see, for example, the above mentioned ones – hypothesize that (a) a point cloud is acquired, (b) the point cloud is partitioned, so as to associate points belonging to the same surface to a single cluster, and (c) an appropriate identification algorithm is used on each cluster in order to identify the parameters of the associated surface. Such working schema can not be adopted in a low cost architecture with limited computational capabilities and, for this reason, in this article an alternative strategy is proposed. It fuses the three previously mentioned phases into a single one, which is recursively executed at each sampling time. Practically, the point cloud is obtained during the motion of the exoskeleton, by acquiring a sequence of single points and by immediately processing them in real time through a recursive method characterized by a limited computational burden.

The surfaces points are acquired by means of a VL53L1X ranging sensor whose current cost is close to 4 dollars. Such ranging sensor has already been used in a collaborative environment. Tests made in [21], [22] confirm that it can be successfully used in real-time contexts. It is a solid state miniaturized sensor, which emits a single laser beam. Every time a new point is acquired, it is tentatively associated to one of the possible surfaces, whose parameters are immediately updated through a Recursive Total Least Squares (RTLS) approach [23], [24]. The resulting staircase identification algorithm is characterized by a limited computational burden, so that it has been implemented, as a task, in the exoskeleton control board: the additional hardware, required for the staircase identification, is limited to 2 ranging sensors.

The paper is organized as follows. Section II describes the exoskeleton and the proposed acquisition system. Section III reports the equations of the adopted RTLS, while Section IV proposes the recursive algorithm used to identify the staircase surfaces. Experimental results obtained with an actual exoskeleton are presented in Section V. In the same section, comparisons are proposed with the outcomes of alternative methods. Final conclusion are drawn in Section VI.

II. The System

The perceptual system developed in this work was conceived for a commercial exoskeleton. Consequently, much attention was paid to the cost of the acquisition system. Three-dimensional environments are typically acquired by means of stereoscopic cameras or lidar scanners but, for economic reasons, in this work a totally different approach is followed and staircases are detected by means of an economic ranging sensor named VL53L1X produced by *STMicroelectronics*. Such sensor has reduced dimensions $[(4.9 \times 2.5 \times 1.56) \cdot 10^{-3}$ m$]$ and can acquire distances up to 4 m at a frequency of 50 Hz. While lidar scanners and camera sensors instantly acquire an entire point cloud, ranging sensors can only measure a single distance at each sampling instant. Necessarily, the point cloud is not acquired as a whole, but it grows during the motion. In particular, when the exoskeleton faces a staircase, it begins hinting a step and, at the same time, it progressively estimate the geometry of the staircase itself. When the next touch down tread is identified, the leg trajectory is modified in order to correctly conclude the step.

The exoskeleton considered in this work has a structure like the one shown in Fig. 1. It has two degrees of freedom for each leg, which are used for the movement of the hip and the knee, respectively. An IMU is mounted on the torso and measures the exoskeleton orientation w.r.t. a world reference

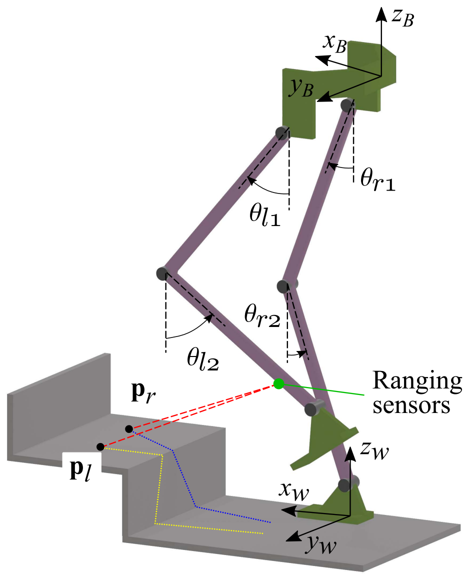

Fig. 1. Schematic representation of the exoskeleton. Two ranging sensors measure the distance (red dashed lines) between the calf and the staircase surfaces. Positions of $\mathbf{p}_l$ and $\mathbf{p}_r$ w.r.t. inertial frame $\{W\}$ are obtained by solving a forward kinematic problem.

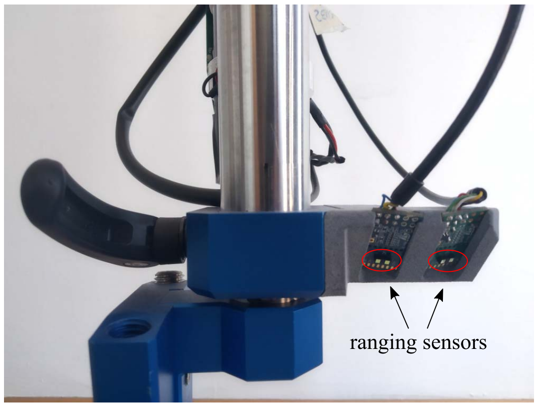

Fig. 2. The two ranging sensors mounted on the exoskeleton calf.

frame. Two ranging sensors are located on the calf of each leg (see Fig. 2). Such sensors measure the distance between the calf and the staircase surfaces. The surface estimation system proposed in this article is based on the knowledge of positions $\mathbf{p}_l$ and $\mathbf{p}_r$, corresponding to the points in which the laser rays intercept the staircase surfaces (see also Fig. 1), measured w.r.t. an inertial frame located on the supporting foot. Both positions can be calculated from the knowledge of

- the kinematic parameters of the exoskeleton,
- the 4 joint variables $(\theta_{l_1}, \theta_{l_2}, \theta_{r_1}, \theta_{r_2})$,
- the torso orientation,
- the distances acquired by the ranging sensors.

When one of the two legs starts climbing the staircase, a point cloud is progressively built and analyzed in order to identify the step surfaces. During the motion, each sensor draws an approximately linear segment on each surface (see Fig. 1). Since an infinite number of planes passes through a single linear segment, a second one is required in order to uniquely define the surface: this motivates the use of 2 ranging sensors for each leg.

The exoskeleton is governed by a Arm Cortex-M4 32bit processor running at 168MHz, with a sampling rate of the control loop equal to 1 kHz. The limited computational capabilities of such processor imposed developing an identification approach based on a lightweight strategy. The identification method proposed in this work exploits the incremental acquisition of the point cloud. Practically, surfaces are identified by means of an iterative procedure which progressively updates the surfaces parameters. Such approach, which drastically reduces the number of operations that must be executed at each sampling instant, will be described in next Section III.

III. The Proposed Recursive Total Least Squares Approach

Surface identification problems are typically solved by means of least squares approaches or their orthogonal counterpart, i.e., Total Least Squares (TLS) strategies. TLS approaches are applied in different contexts, from the robot localization problem [25], to the estimation of parameters [26], [27]. Furthermore, they are used for signal processing applications and FIR filtering [28], [29], but also for the estimation of the batteries capacity [30], [31]. Mentioned works propose several recursive TLS techniques, but the emphasis is always posed on a recursive formulation of the Singular Value Decomposition (SVD) problem which must be internally solved. Conversely, for the problem at hand, the SVD problem, due to the dimension of the involved matrices, has a negligible computational impact, but the sequential acquisition of the point cloud requires to continuously update the internal matrices of the algorithm: an iterative procedure is used in order to reduce the computational burden of the update process.

In this work, the staircase surfaces are identified by means of a RTLS approach similar to the one proposed in [23], [24]. Similarly to the standard TLS method [32], [33], each plane is identified from an over-determined system of equations. Such equations are obtained by assuming that all the points associated to the same surface are actually contained by the same point cloud but, since acquisitions are affected by noise, the plane coefficients need to be estimated through a best fitting approach.

Unfortunately, due to the acquisition procedure adopted, it may happen that points are assigned to the wrong surface. The RTLS algorithm has been consequently modified, in order to update surfaces by adding new points or by removing wrongly assigned ones.

This section will briefly recall the basic concepts of the TLS strategy and, then, it will show how it can be reformulated into a recursive form.

Any planar surface can be analytically represented as follows

$$v_x(x - \bar{x}) + v_y(y - \bar{y}) + v_z(z - \bar{z}) = 0, \quad (1)$$

where $\mathbf{v} := [v_x \; v_y \; v_z]^T$ is a generic vector which is orthogonal to the plane and $\bar{\mathbf{p}} := [\bar{x} \; \bar{y} \; \bar{z}]^T$ is a generic point lying on the plane. A point $\mathbf{p} := [x\ y\ z]^T$ belongs to the plane if it satisfies (1). Thus, a plane is defined through the knowledge of vectors $\mathbf{v}$ and $\bar{\mathbf{p}}$.

Given a cloud of points $\mathbf{p}_i = [x_i\ y_i\ z_i]^T, i = 1, 2, \dots, n$, the corresponding best fitting plane can be found by means of several methods, among which the TLS is the most commonly used one. The plane coefficients are obtained by minimizing the squared distances between the given points and the plane itself, i.e., by minimizing the following cost index

$$J = \sum_{i=1}^{n} \frac{\left[v_x(x_i - \bar{x}) + v_y(y_i - \bar{y}) + v_z(z_i - \bar{z})\right]^2}{v_x^2 + v_y^2 + v_z^2}. \tag{2}$$

Any cloud of points admits a centroid defined as follows

$$\mathbf{e} := \frac{1}{n}\sum_{i=1}^{n} \mathbf{p}_i, \tag{3}$$

or, alternatively,

$$\mathbf{e} := \frac{\mathbf{P}\mathbf{o}^T}{n}, \tag{4}$$

where $\mathbf{P} := [\mathbf{p}_0\ \mathbf{p}_1 \cdots \mathbf{p}_n] \in \mathbb{R}^{3\times n}$ is a matrix which contains the points and $\mathbf{o} := [1\ 1 \cdots 1]$ is a row vector containing n ones.

As shown in [34], the centroid associated to the point cloud belongs to the best fitting plane, so that $\bar{\mathbf{p}}$ can be assumed as follows

$$\bar{\mathbf{p}} = \mathbf{e}. \tag{5}$$

Bearing in mind (5), (2) can be rewritten into the following form

$$J = \left\|\mathbf{M}_p\hat{\mathbf{n}}\right\|^2 = \hat{\mathbf{n}}^T\mathbf{M}_p^T\mathbf{M}_p\hat{\mathbf{n}} \tag{6}$$

where $\mathbf{M}_p \in \mathbb{R}^{n\times 3}$ and $\hat{\mathbf{n}} \in \mathbb{R}^3$ are defined as follows

$$\mathbf{M}_p := (\mathbf{P} - \bar{\mathbf{p}}\mathbf{o})^T = \begin{bmatrix} x_1 - e_x & y_1 - e_y & z_1 - e_z \\ \vdots & \vdots & \vdots \\ x_n - e_x & y_n - e_y & z_n - e_z \end{bmatrix}, \tag{7}$$

$$\hat{\mathbf{n}} = \frac{1}{\sqrt{v_x^2 + v_y^2 + v_z^2}} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}. \tag{8}$$

Evidently, $\hat{\mathbf{n}}$ is an unit vector which is orthogonal to the estimated plane and can be obtained by minimizing J, i.e., by solving the following optimization problem

$$\hat{\mathbf{n}}^* = \arg\min_{\hat{\mathbf{n}}}\left\{\hat{\mathbf{n}}^T\mathbf{M}_p^T\mathbf{M}_p\hat{\mathbf{n}}\right\} \tag{9}$$

Since matrix $\mathbf{M}_p^T\mathbf{M}_p \in \mathbb{R}^{3\times 3}$ is real, symmetric and positive semi-definite, the solution of (9) is well known and can be obtained through an SVD decomposition of the same matrix. In particular, $\mathbf{M}_p^T\mathbf{M}_p$ can always be written as follows

$$\mathbf{M}_p^T\mathbf{M}_p = \left[\hat{\mathbf{u}}_1\ \hat{\mathbf{u}}_2\ \hat{\mathbf{u}}_3\right] \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{u}}_1^T \\ \hat{\mathbf{u}}_2^T \\ \hat{\mathbf{u}}_3^T \end{bmatrix}, \tag{10}$$

where $\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2$ and $\hat{\mathbf{u}}_3$ are the vectors of an orthonormal basis of $\mathbb{R}^3$ composed by the eigenvectors of $\mathbf{M}_p$ and $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$ are the corresponding eigenvalues. According to [34], the solution of (9) is given by $\hat{\mathbf{n}}^* = \hat{\mathbf{u}}_3$, where $\hat{\mathbf{u}}_3$ is the eigenvector associated to λ_3, i.e., to the smallest eigenvalue.

The SVD decomposition of matrix $\mathbf{M}_p^T\mathbf{M}_p$, owing to its reduced dimensions, is not time consuming and can be efficiently obtained in real time even through processors with reduced computational capabilities. However, as anticipated in Section II, the point cloud is continuously updated, so that the dimension of $\mathbf{M}_p$ continuously grows and the re-evaluation of $\mathbf{M}_p^T\mathbf{M}_p$ can become time consuming.

The mentioned problem has been overcome by implementing an iterative procedure for the estimation of the plane coefficients.

Bearing in mind (5) and (7), as well as the definition of vector $\mathbf{o}$, product $\mathbf{M}_p^T\mathbf{M}_p$ can be expanded as follows

$$\begin{aligned} \mathbf{M}_p^T\mathbf{M}_p &= (\mathbf{P} - \bar{\mathbf{p}}\mathbf{o})(\mathbf{P} - \bar{\mathbf{p}}\mathbf{o})^T, \\ &= \mathbf{P}\mathbf{P}^T - \mathbf{e}\mathbf{o}\mathbf{P}^T - \mathbf{P}\mathbf{o}^T\mathbf{e}^T + \mathbf{e}\mathbf{o}\mathbf{o}^T\mathbf{e}^T, \\ &= \mathbf{P}\mathbf{P}^T - \mathbf{e}\mathbf{o}\mathbf{P}^T - \mathbf{P}\mathbf{o}^T\mathbf{e}^T + n\mathbf{e}\mathbf{e}^T. \end{aligned}$$

By further recalling (4), the following equation is obtained

$$\mathbf{M}_p^T\mathbf{M}_p = \mathbf{P}\mathbf{P}^T - n\mathbf{e}\mathbf{e}^T - n\mathbf{e}\mathbf{e}^T + n\mathbf{e}\mathbf{e}^T = \mathbf{P}\mathbf{P}^T - n\mathbf{e}\mathbf{e}^T. \tag{11}$$

Equation (11) can be used for the recursive update of $\mathbf{M}_p^T\mathbf{M}_p$. Let us suppose that at step n, product $\mathbf{M}_{p_n}^T\mathbf{M}_{p_n}$ has been obtained by means of the following equation

$$\mathbf{M}_{p_n}^T\mathbf{M}_{p_n} = \mathbf{P}_n\mathbf{P}_n^T - n\mathbf{e}_n\mathbf{e}_n^T, \tag{12}$$

and that terms $\mathbf{V}_n := \mathbf{P}_n\mathbf{P}_n^T \in \mathbb{R}^{3\times 3}$ and $\mathbf{e}_n \in \mathbb{R}^3$ have been stored into 2 variables. At step $n+1$, product $\mathbf{M}_{p_{n+1}}^T\mathbf{M}_{p_{n+1}}$ necessarily assumes the following form

$$\mathbf{M}_{p_{n+1}}^T\mathbf{M}_{p_{n+1}} = \mathbf{P}_{n+1}\mathbf{P}_{n+1}^T - (n+1)\mathbf{e}_{n+1}\mathbf{e}_{n+1}^T, \tag{13}$$

where

$$\mathbf{P}_{n+1} := \left[\mathbf{P}_n|\mathbf{p}_{n+1}\right]. \tag{14}$$

Evidently, term $\mathbf{P}_{n+1}\mathbf{P}_{n+1}^T$ in (13) can be obtained, with a limited computational burden, by means of the following expression

$$\mathbf{V}_{n+1} = \mathbf{P}_{n+1}\mathbf{P}_{n+1}^T = \mathbf{P}_n\mathbf{P}_n^T + \mathbf{p}_{n+1}\mathbf{p}_{n+1}^T = \mathbf{V}_n + \mathbf{p}_{n+1}\mathbf{p}_{n+1}^T. \tag{15}$$

Similarly, $\mathbf{e}_{n+1}$ can be updated by means of the following mean recursion formula

$$\mathbf{e}_{n+1} := \frac{\mathbf{P}_{n+1}\mathbf{o}_{n+1}^T}{n+1} = \frac{\mathbf{P}_n\mathbf{o}_n^T + \mathbf{p}_{n+1}}{n+1} = \frac{n\mathbf{e}_n + \mathbf{p}_{n+1}}{n+1}. \tag{16}$$

As shown in next Section IV, the acquired points may also be assigned to the wrong surface due to the incremental acquisition of the point cloud. As a consequence, it is important to have the possibility to remove wrongly assigned points from a surface, in order to discard or to assign them to a new plane. The same reasoning followed for the synthesis of (15) and (16) leads to the following recursive equations, which can be used to remove a point from a surface

$$\mathbf{P}_{n-1}\mathbf{P}_{n-1}^T = \mathbf{V}_n - \mathbf{p}_A\mathbf{p}_A^T, \tag{17}$$

$$\mathbf{e}_{n-1} := \frac{n\mathbf{e}_n - \mathbf{p}_A}{n-1}. \tag{18}$$

IV. Plane Fitting Algorithm

As explained in previous sections, the staircase steps are identified by means of economic ranging sensors, so that the point cloud grows, point by point, during the motion. As a consequence, when a new point $\mathbf{p}_r$ or $\mathbf{p}_l$ is acquired, its membership to a given surface is not known in advance. For such reason, the first problem to be solved is the identification of the surface on which acquired points are located.

A staircase is made of a sequence of planar surfaces. A structure $S_{id} = \{\mathbf{n}, \mathbf{B}, \mathbf{V}, \mathbf{e}, \hat{\mathbf{n}}^*, \sigma, ok\}$ is defined for each of them. It contains all the elements required for the analytical representation of the surface. Each structure is defined as follows:

- id, surface IDentification number. 0 indicates the ground plane;
- $\mathbf{n} = [n_r, n_l]$, number of points assigned to S_{id}, deriving from the right and the left ranging sensors, respectively;
- $\mathbf{B} \in \mathbb{R}^{b\times 3}$, First Input First Output (FIFO) register which stores the last b points associated to S_{id};
- $\mathbf{e} := [e_x\, e_y\, e_z]^T \in \mathbb{R}^3$, centroid of S_{id};
- $\mathbf{V} \in \mathbb{R}^{3\times 3}$, matrix $\mathbf{PP}^T$ used by the RTLS (see Section III);
- $\hat{\mathbf{n}}^* \in \mathbb{R}^3$, normal vector of S_{id};
- σ_n, standard deviation of the distances between the acquired points and S_{id};
- ok, Boolean variable which indicates if the plane estimate is reliable or not.

Structures are collected into a vector of five elements $\mathbf{S} := [S_0, S_1, \ldots, S_4]^T$, since during a standard climbing step the acquired points belong to a maximum of 5 surfaces. Even ids correspond to horizontal surfaces, while odd ids are relative to vertical ones.

It is natural to assume that, while climbing, the first acquired points are located on S_0, immediately followed by other points falling on the first riser, S_1, then by others lying on the first tread, S_2, and so on.

The functions proposed in Algorithm 1 exploit such idea for the identification of the planes. The first one, i.e., *Assign*, associates the acquired point to the correct surface, while the second one, i.e., *SurfUpdate*, updates the corresponding surface equations through the recursive technique proposed in Section III. Details of both functions are given in Sections IV-A and IV-B respectively, while the workflow of the algorithm is proposed in Algorithm 1. The calling main program is omitted for conciseness. At each sampling time, the main program computes the coordinates of a new point through a forward kinematic function, and then passes them to *Assign*. Furthermore, the calling program keeps track of an index, id, which identifies the surface currently under investigation. During the motion, the first points will certainly belong to S_0, so that, initially, $id = 0$.

The general purpose version of the algorithm does not make assumptions concerning the surfaces orientations. However, in case of regular staircases, even and odd planes can be assumed horizontal and vertical, respectively. Such assumptions simplify the RTLS procedure: for horizontal planes the normal vector is perpendicular to the ground, while for the vertical ones, matrix $\mathbf{V}$ becomes 2-dimensional, so that the SVD procedure is faster.

Algorithm 1: The Algorithm Which Assigns the Point to the Correct Surface and Updates Its Information

```
Data: $\mathbf{p}_i$, acquired point, $\mathbf{Q}_i$, FIFO queue which stores $q$ points from a single ranging sensor
Result: $S_{id}$, $id \in [0, s]$, structure for the $s$ surfaces identified
1  Function Assign($\mathbf{p}$, $\mathbf{Q}$, $\mathbf{S}$, $f$, $id$)
2      if $\mathbf{Q}$ is full then
3          Compute $\bar{z}$ and $\sigma_z$;
4          $\mathbf{q} \leftarrow$ Extract($\mathbf{Q}$);
5          if IsEven($id$) & ($\sigma_z < th_z$) & ($|q_z - \bar{z}| < \sigma_z$) then
6              SurfUpdate($\mathbf{q}$, $S_{id}$, $f$);
7          else if IsEven($id$) & ($\sigma_z \geq th_z$) & ($q_z > S_{id}.e_z + 1.5\, th_z$) then
8              $id \leftarrow id + 1$;
9              SurfUpdate($\mathbf{q}$, $S_{id}$, $f$);
10         else if IsOdd($id$) & ($\sigma_z \geq th_z$) then
11             SurfUpdate($\mathbf{q}$, $S_{id}$, $f$);
12         else if IsOdd($id$) & ($\sigma_z < th_z$) then
13             if ($q_z > S_{id-1}.e_z + th_{pl}$) then
14                 $id \leftarrow id + 1$;
15             SurfUpdate($\mathbf{q}$, $S_{id}$, $f$);
16     $\mathbf{Q} \leftarrow$ Insert($\mathbf{p}$);
17 Function SurfUpdate($\mathbf{q}$, $S_{id}$, $f$)
18     if $\mathbf{B}$ is full then
19         $\mathbf{r} \leftarrow$ Extract($S_{id}.\mathbf{B}$);
20         $d$ = Dist($\mathbf{r}$, $S_{id}$);
21         if not($|d| \leq S_{id}.\sigma_n$) then
22             RemoveFromSurf($\mathbf{r}$, $S_{id}$, $f$);
23     $S_{id}.\mathbf{B} \leftarrow$ Insert($\mathbf{q}$);
24     AddToSurf($\mathbf{q}$, $S_{id}$, $f$);
25     if $S_{id}.n_r \geq N$ & $S_{id}.n_l \geq N$ then
26         $S_{id}.ok \leftarrow 1$;
27     if $id > 0$ & $S_{id}.ok = 1$ then
28         Shift($S_{id-1}$, $S_{id}$);
```

A. Phase I: Points Are Assigned to the Proper Surface

Each time a new point $\mathbf{p} := [p_x\, p_y\, p_z]^T$ is acquired, function *Assign* is called. As previously asserted, points are preliminary analyzed in order to allocate them on the correct surface. To this purpose, they are not immediately assigned to a surface but, conversely, they are initially stored into a (First Input First Output) FIFO queue, named $\mathbf{Q} := [\mathbf{p}_0\, \mathbf{p}_1 \cdots \mathbf{p}_{q-1}] \in \mathbb{R}^{3\times q}$, which contains the last acquired positions. The queue dimension, i.e., q, is kept small, so as to guarantee that $\mathbf{Q}$ contains points belonging to a maximum of two adjacent planes. Two different FIFOs are used, one for the left laser sensor, $\mathbf{Q}_l$, and the other for the right one, $\mathbf{Q}_r$. The proper FIFO is passed to *Assign* by the calling program, together with a flag f which is equal to "r" for the right sensor and to "l" for the left one. Since the two FIFOs are managed with the same strategy, from now on, $\mathbf{Q}$ will be used to indifferently indicate one of them.

The algorithm starts assigning points to a surface only when $\mathbf{Q}$ is full. Conversely, new points are simply stored into the FIFO (line 16). When $\mathbf{Q}$ is full (line 2), i.e., when it contains q elements, stored points are analyzed in order to assign the oldest one to its corresponding plane. The basic idea is that points belonging to horizontal surfaces have similar p_z components. As a consequence, if all the points in $\mathbf{Q}$ belong to the same plane, then the standard deviation of their p_z component, i.e.,

$$\sigma_z = \sqrt{\frac{1}{q}\sum_{k=0}^{q-1}\left(p_{z_k} - \bar{z}\right)^2},$$

with

$$\bar{z} = \frac{1}{q}\sum_{k=0}^{q-1} p_{z_k},$$

must be smaller than a given threshold, $th_z > 0$. Such threshold depends on the acquisition noise.

Bearing in mind such idea, the oldest point in $\mathbf{Q}$ is extracted and placed in $\mathbf{q} := [q_x\, q_y\, q_z]^T$ (line 4). If the current *id* is even, and $\sigma_z < th_z$, i.e., the points in $\mathbf{Q}$ belong to a horizontal surface, and $q_z \in [\bar{z} - \sigma_z, \bar{z} + \sigma_z]$, then point $\mathbf{q}$ is assigned to the plane identified by the current *id* (lines 5, 6).

Still considering an even *id* – which means that the current plane is horizontal – if $\sigma_z > th_z$, then $\mathbf{Q}$ also contains points belonging to a vertical surface: $\mathbf{q}$ may belong to the horizontal plane or to the subsequent vertical one. If $S_{id}.\mathbf{e} := S_{id}.[e_x\, e_y\, e_z]^T$ is the centroid associated to the current horizontal plane and the analyzed point has a q_z component which is larger than $S_{id}.e_z + 1.5\, th_z$, then *id* is incremented and $\mathbf{q}$ is consequently assigned to the subsequent vertical plane (lines 7–9).

If function *Assign* is called with an odd *id* (the current plane is vertical), until condition $\sigma_z \geq th_z$ applies, points are assigned to the vertical surface (lines 10, 11). Conversely, if $\sigma_z < th_z$ a further test is made in order to check if q_z is higher than a minimum value given by the elevation of the past horizontal plane ($S_{id-1}.e_z$) plus a threshold th_{pl}. Such threshold corresponds to the minimum height which is expected for a riser and has been introduced in order to avoid false switches to the subsequent horizontal plane. If the test is passed, *id* is increased and $\mathbf{q}$ is correctly assigned (lines 12–15).

Any other situation is considered unacceptable and $\mathbf{q}$ is consequently discarded, since it is probably affected by too much noise.

B. Phase II: The Surfaces Equations Are Updated

When a point is finally assigned to a plane by the *Assign* function, the corresponding surface structure S_{id} must be updated. Hence *Assign* calls *SurfUpdate*($\mathbf{q}, S_{id}, f$), which acts on the plane data. It should be noticed that points $\mathbf{q}$ are assigned to a given surface on the basis of considerations deriving from the analysis of a short FIFO ($\mathbf{Q}$), so that it may happen that points are assigned to the wrong surface. Therefore, function *SurfUpdate*($\mathbf{q}, S_{id}, f$) re-elaborates them by exploiting the entire point cloud assigned to each surface: some points will be discarded, others will be removed from a surface to be assigned to another.

Let us deeper analyze function *SurfUpdate*($\mathbf{q}, S_{id}, f$). Any new point $\mathbf{q}$, associated to a surface through its *id*, is stored into FIFO $S_{id}.\mathbf{B}$ (line 23) for future elaborations and, simultaneously, it is used to update the plane equations (line 24). In particular, *AddToSurf*($\mathbf{q}, S_{id}, f$) updates matrix $S_{id}.\mathbf{V}$ and vector $S_{id}.\mathbf{e}$ through (15) and (16), respectively. Furthermore, it calculates $S_{id}.\hat{\mathbf{n}}^*$ from the eigenvectors of $\mathbf{M}_p^T\mathbf{M}_p$.

A surface is declared "reliable" when its point cloud contains at least N samples for each one of the two ranging sensors (lines 25 and 26). For such reason, *AddToSurf*($\mathbf{q}, S_{id}, f$) keeps track of the number of samples obtained from the left and the right ranging sensors by properly updating field $S_{id}.\mathbf{n}$.

The remaining lines of the code are conceived to improve the surface identification. In the literature, surface outliers are usually managed through techniques based on Random Sample Consensus (RANSAC) algorithm [15], [35] or on Least Trimmed Squares and Principal Component Analysis [36]. Such methods cannot be adopted in this context due to their computational burden, so that an alternative strategy has been adopted. In particular, if FIFO $\mathbf{B}$ is full, lines 18–22 extract from it the oldest point, i.e., $\mathbf{r}$, and verify its distance from the plane: if such distance is higher than the standard deviation of all the points associated to the plane, i.e., $S_{id}.\sigma_n$, $\mathbf{r}$ is removed from the surface by means of *RemoveFromSurf*($\mathbf{q}, S_{id}, f$), which uses equations (17) and (18). $S_{id}.\mathbf{n}$ is updated accordingly. It should be noticed that, as mentioned before, if the assumption of perpendicular planes can be made, functions *AddToSurf* and *RemoveFromSurf* simplifies since the problem involves a single dimension for the horizontal planes and two for the vertical ones.

A final refinement is eventually performed through function *Shift*(S_{id-1}, S_{id}). If S_{id} is "reliable", such function checks if some of the points of S_{id-1} are actually closer to the current surface. In that case, such points are removed from S_{id-1} and added to S_{id}.

V. Results

The proposed algorithm has been tested by means of the experimental exoskeleton shown in Fig. 3. The first experiments involved a single staircase which was approached from different angles. Generic surfaces were assumed in Section V-A, while regular surfaces – with perfectly horizontal treads and vertical risers – were considered in Section V-B. An extensive set of experiments, involving three different staircases, was then executed to verify the reliability and the efficiency of the whole algorithm. The corresponding results are compared in Section V-C with the analogous ones achieved through three alternative methods proposed in the literature.

A. Identification of Staircases With Generic Surfaces

The first tests considered the following three cases. Case A is relative to a nominal operating condition: the exoskeleton is posed in front of a staircase whose risers are 0.162 m high and whose treads are 0.28 m depth. The rising foot is posed

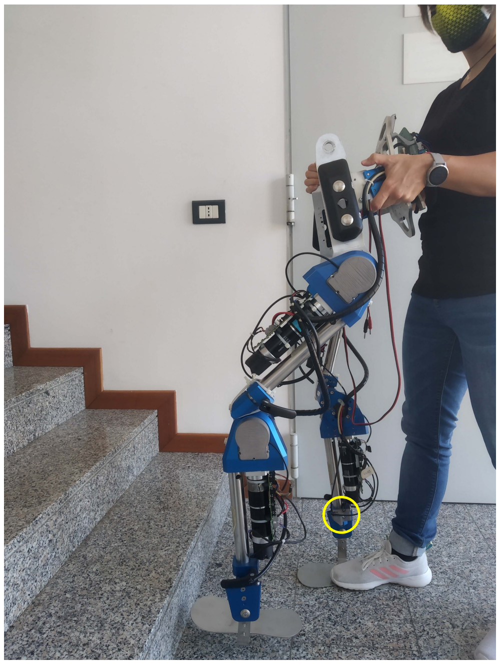

Fig. 3. The experimental exoskeleton used for the tests. A yellow circle points out the ranging sensors positions.

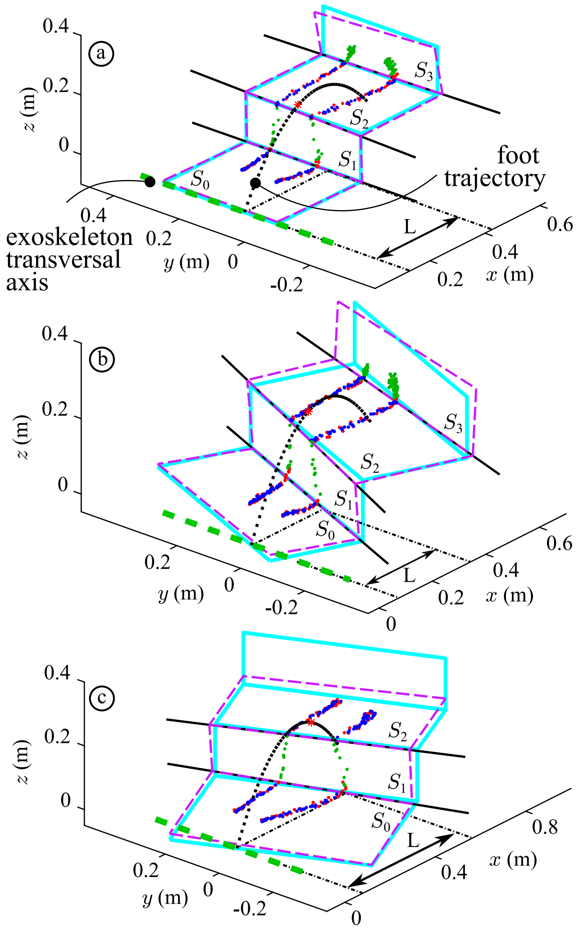

Fig. 4. Staircase reconstruction obtained through Algorithm 1 when (a) the exoskeleton is located frontally w.r.t. the staircase, (b) it is turned anticlockwise or (c) clockwise around its vertical axis. Light blue solid segments are used for the actual profile of the staircase, while the estimated surfaces are represented through magenta dashed lines. The thick green dashed segment points out the exoskeleton orientation w.r.t. the staircase, while the black dotted curve is the foot trajectory. A red asterisk along such trajectory indicates the position in which the estimate of S_2 is considered reliable. Colors of the acquired points change depending on the surface to which they are assigned: blue points have been used for treads, while green ones have been used for risers. Discarded points are highlighted in red.

at an initial distance $L = 0.305$ m from the first riser, so as to execute a regular climbing step. Conversely, Case B and Case C consider two unusual working conditions. As shown in Fig. 4b, in Case B the exoskeleton is counterclockwise turned around its vertical axis by 23 deg, so as to see the staircase on its right side. Furthermore, the climbing foot is initially located closer to the staircase, more precisely at a distance $L = 0.27$ m from the first riser. Conversely, in Case C (see Fig. 4c) the exoskeleton is clockwise turned by 23 deg around its vertical axis and its foot is posed at a distance $L = 0.49$ m from the staircase. Cases B and C are unusual in real operating conditions since, for evident reasons which also include safety, therapists who assist the patient activate the climbing procedure only when the exoskeleton is placed exactly in front of the staircase. Both cases were considered so as to verify the robustness of the approach. Indeed, they admit different distances between the two sensors and the first step of the staircase and different incidence angles between the laser beams and the steps surfaces. In facts, both variables influence the sensors precision and, consequently, may affect the estimate accuracy. Furthermore, in both configurations, points acquired at the same sampling instants by the 2 sensors may belong to different surfaces, thus complicating their assignment to the right plane. In all the three cases, a trajectory was planned for the right leg, so as to make it climb the first step. The acquired points were processed in real time by means of a Arm Cortex-M4 32bit processor running at 168 MHz. The following thresholds were assumed for Algorithm 1: $th_z = 8 \cdot 10^{-3}$ m and $th_{pl} = 13 \cdot 10^{-2}$ m.

Figure 4 shows the outcomes of the three experiments. Since this work focuses on the staircase estimation, a simple parabolic trajectory was used, and the same was not updated after the steps were identified. A red asterisk along the trajectory points out the position in which S_2 is first identified. As can be noticed, the surface is known when the foot is still far from its final destination, so that the trajectory can be promptly modified in order to obtain the best foot placement.

Table I makes it possible to evaluate quantitatively the outcomes of the three test cases. Columns 2, 4, and 6 show, for each *id*, angle α between the normal vectors of the actual and the estimated surfaces (see also Fig. 5). Similarly, columns 3, 5, and 7 list distances d between the centroids of the estimated surfaces and the actual planes. In all the test cases, a good matching between identified and actual surfaces was obtained. The sole exceptions are represented by surfaces S_3 for Cases A and B and S_2 for Case C. The reason of such performance deterioration is given by the vibrations in the exoskeleton structure, which appear at the end of each transient. Due to such vibrations, the acquisition precision reduces and points are spread on the corresponding surface (see Fig. 4).

The most important row in Table I is the one relative to S_2, i.e., the touch down surface. For Cases A and B, α is always smaller than 3 deg and the centroids associated to the

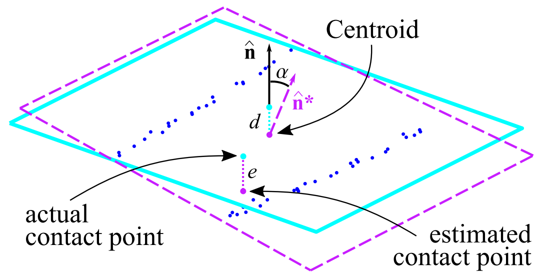

Fig. 5. The blue surface with its normal vector $\hat{\mathbf{n}}$ represents a plane of the actual staircase, while the red surface is identified through the strategy proposed in Section IV. α is the angle between the normal vectors, d is the distance between the point cloud centroid and the staircase surface, e is the contact point elevation error.

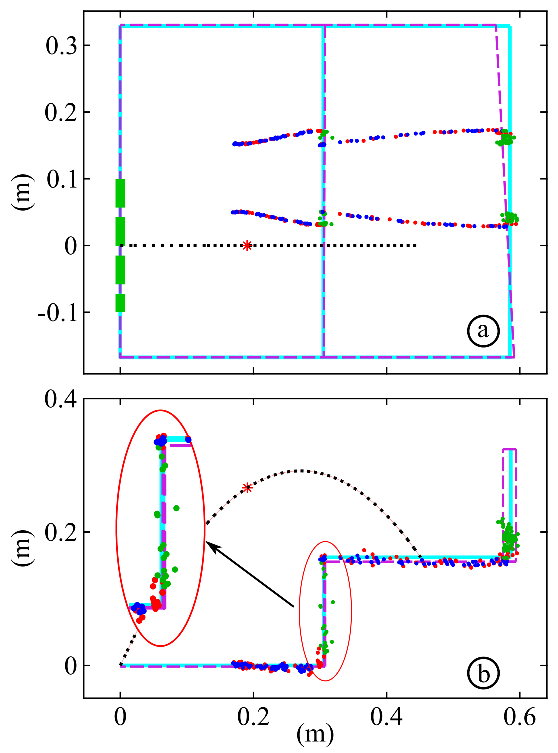

Fig. 6. Case A: staircase identification under the hypothesis of horizontal treads and vertical risers. (a) top view (b) side view. Colors are the same used for Figure 4.

surfaces are closer than 3 millimeters to the actual staircase. Such tolerances make it possible to reasonably select the final touch down position for the foot. In such position, the height of the estimated tread is equal to 0.1579 m for Case A, i.e., the estimated tread was roughly $4 \cdot 10^{-3}$ m lower than the actual one (see error e in Fig. 5), while for Case B it is equal to 0.1603 m, i.e., the step was $2 \cdot 10^{-3}$ m lower than the real one. As anticipated, Case C returned the worst tread estimation. However, in the touch down position the height of the estimated tread is equal to 0.1551 m, i.e., the error is similar to the one obtained for Case A. It is important to point out that such precisions have been achieved with ranging sensors whose acquisition noise, measured under static conditions, is equal to $\pm 1.5 \cdot 10^{-3}$ m.

TABLE I

ANGLE α BETWEEN NORMAL VECTOR $\hat{\mathbf{n}}$ OF THE ACTUAL SURFACE AND $\hat{\mathbf{n}}^*$ OF THE ESTIMATED ONE, AND DISTANCE d BETWEEN POINT CLOUD CENTROIDS $\bar{\mathbf{p}} = \mathbf{e}$ AND THE STAIRCASE SURFACES

ID	Case A, Fig. 4a		Case B, Fig. 4b		Case C, Fig. 4c	
	α (deg)	d (m)	α (deg)	d (m)	α (deg)	d (m)
0	2.104	$-9.30 \cdot 10^{-5}$	4.340	$-6.82 \cdot 10^{-4}$	2.295	$4.90 \cdot 10^{-3}$
1	1.999	$1.10 \cdot 10^{-3}$	5.287	$-1.30 \cdot 10^{-5}$	3.916	$2.10 \cdot 10^{-3}$
2	2.607	$-3.64 \cdot 10^{-4}$	2.350	$2.50 \cdot 10^{-3}$	6.744	$1.32 \cdot 10^{-2}$
3	10.749	$-5.90 \cdot 10^{-3}$	8.147	$-1.62 \cdot 10^{-2}$	-	-

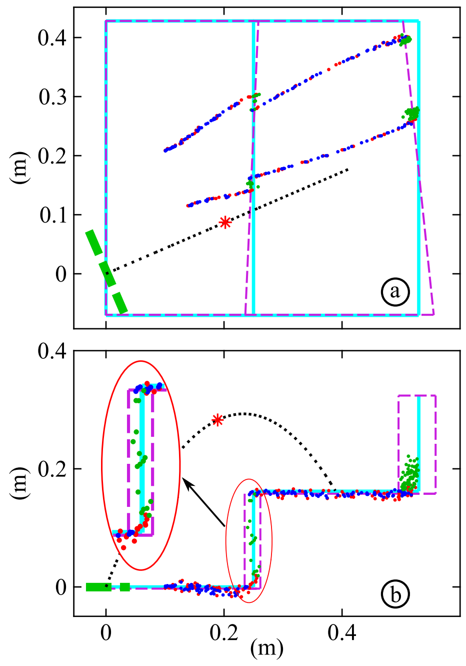

Fig. 7. Case B: staircase identification under the hypothesis of horizontal treads and vertical risers. (a) top view (b) side view. Colors are the same used for Figure 4.

TABLE II

RESULTS OBTAINED BY ADMITTING HORIZONTAL TREADS AND VERTICAL RISERS. DATA ARE REPORTED IN THE SAME WAY OF TABLE I. ANGLE α BETWEEN NORMAL VECTOR $\hat{\mathbf{n}}$ OF THE ACTUAL SURFACE AND $\hat{\mathbf{n}}^*$ OF THE ESTIMATED ONE, AND DISTANCE $d = e$ BETWEEN POINT CLOUD CENTROIDS $\bar{\mathbf{p}} = \mathbf{e}$ AND THE STAIRCASE SURFACES

ID	Case A, Fig. 4a		Case B, Fig. 4b		Case C, Fig. 4c	
	α (deg)	d (m)	α (deg)	d (m)	α (deg)	d (m)
0	-	$-1.64 \cdot 10^{-3}$	-	$-2.91 \cdot 10^{-3}$	-	$-1.76 \cdot 10^{-3}$
1	0.214	$1.47 \cdot 10^{-3}$	2.541	$-0.72 \cdot 10^{-3}$	1.096	$4.12 \cdot 10^{-3}$
2	-	$-6.36 \cdot 10^{-3}$	-	$-4.22 \cdot 10^{-3}$	-	$1.65 \cdot 10^{-3}$
3	3.139	$-3.50 \cdot 10^{-3}$	5.979	$-1.50 \cdot 10^{-2}$	-	-

B. Identification of Staircases With Horizontal Treads and Vertical Raisers

So far, the proposed algorithm hypothesized the identification of staircases whose surfaces may be generically oriented in the space. However, it is almost always possible to assume that treads are horizontal and risers are vertical. The identification algorithm considerably simplifies: the treads estimation process becomes one-dimensional, while the risers one becomes two-dimensional. Algorithm 1 can also manage such

TABLE III
A COMPARATIVE ANALYSIS BETWEEN THE EXPERIMENTAL RESULTS OBTAINED IN THIS WORK WITH THE HTVR METHOD, AND THE ONES ACHIEVED WITH THE ALTERNATIVE TECHNIQUES PROPOSED IN [18], [19], AND [20]. THE STATISTICS OF THE FIRST 3 ROWS REFER TO 3 DIFFERENT STAIRCASES

	Laser scanner	Camera	Ranging Sensors	Accuracy								Acquisition and computation time	Sensors cost
				Risers				Treads					
				actual (m)	mean (m)	std (m)	error (%)	actual (m)	mean (m)	std (m)	error (%)	(s)	($)
HTVR			✓	0.153	0.157	$2.8 \cdot 10^{-3}$	2.61 ± 1.83	0.294	0.290	$3.0 \cdot 10^{-3}$	1.36 ± 1.02		
HTVR			✓	0.162	0.169	$3.4 \cdot 10^{-3}$	4.32 ± 2.10	0.280	0.272	$3.2 \cdot 10^{-3}$	2.86 ± 1.14	$(4.3 \pm 1.3) \cdot 10^{-4}$	8
HTVR			✓	0.175	0.179	$3.2 \cdot 10^{-3}$	2.29 ± 1.83	0.270	0.264	$3.5 \cdot 10^{-3}$	2.22 ± 1.30		
[18]	✓	✓		0.070	0.074	$3.1 \cdot 10^{-3}$	6.00 ± 4.43	0.180	0.192	$6.2 \cdot 10^{-3}$	6.50 ± 3.44	$(2.5 \pm 0.1) \cdot 10^{-2}$	≥ 800
[19]	✓			0.050	0.053*	–	6.00*	0.180	0.186*	–	3.33*	16	≥ 2000
[20]		✓		0.115	0.115	$1.2 \cdot 10^{-3}$	0.0087 ± 1.04	0.280	0.278	$8.24 \cdot 10^{-4}$	0.893 ± 0.29	–	≥ 300

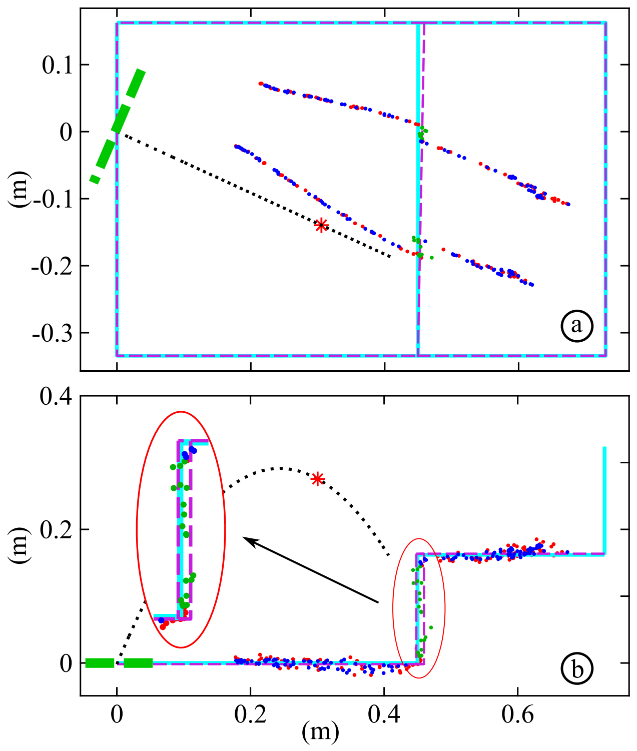

Fig. 8. Case C: staircase identification under the hypothesis of horizontal treads and vertical risers. (a) top view (b) side view. Colors are the same used for Figure 4.

simplified problems. In particular, the complexity of functions *AddToSurf* and *RemoveFromSurf* reduces.

The same points acquired for Cases A, B, and C have been re-elaborated by admitting such hypothesis. The obtained results are visually shown in Figs. 6, 7, and 8, while numerical errors are listed in Table II. Due to the imposed parallelism, errors d and e coincides, and α is always 0 for all the treads. A comparison with the homologous results listed in Table I shows that, due to the reduced degrees of freedom, slightly higher errors are obtained. However, the maximum errors for S_2, i.e., the foot contact surface, is equal to $-6.36 \cdot 10^{-3}$ m.

In terms of computational time, the proposed algorithm meets the expectations. With the adopted processor, i.e., a Cortex-M4 32bit running at 168 MHz, the average computational time for the general purpose algorithm was equal to $0.685 \pm 0.437 \cdot 10^{-3}$ s. For the horizontal treads version, computational times reduce to $0.427 \pm 0.132 \cdot 10^{-3}$ s. Practically, the identification algorithm is sufficiently fast to be executed between two subsequent acquisitions of the laser sensors, whose sampling time is equal to $50 \cdot 10^{-3}$ s.

C. Reliability Tests on Different Staircases

The approach reliability has been tested through a set of additional experiments involving 3 different staircases. For each of them, 20 different positions and orientations of the exoskeleton were considered. The experimental results are summarized in the first 3 rows of Table III and have been obtained by assuming Horizontal Treads and Vertical Risers (HTVR). In the same table, they are compared with the outcomes of the following alternative systems:

- Reference [18] a NAO robot, additionally equipped with a laser range finder (Hokuyo URG-04LX);
- Reference [19] a hexapod robot, equipped with a stepper motor and a 2D laser range finder (Sick LMS111) in order to acquire 3D point cloud of a staircase;
- Reference [20] an exoskeleton, equipped with a depth camera (Percipio FM830).

In terms of accuracy, the performance of the novel method is comparable with the ones achieved in [18] and [19]. Better results are obtained in [20], which however use a dense point cloud, which is elaborated offline by means of MATLAB. As shown by the last two columns of Table III, different conclusions can be drawn in terms of computational times and implementation costs: despite the low power processor adopted, the novel method is evidently the fastest and its implementation cost is very small.

Summarizing, for all the experiments, the elevation error associated to the touch down position of the foot was lower than 10^{-2} m. Such tolerance is acceptable for the application at hand, since minor errors are compensated by the patient who wears the exoskeleton. Furthermore, if required, additional ranging sensors could be added to the system, in order to improve the final approach to the step.

VI. CONCLUSION

The algorithm proposed in this work was conceived to identify in real time the 3D shape of a staircase, so as to allow an exoskeleton to consequently adapt its gait. Differently from other strategies in the literature, the target is reached by means of a low cost system based on an IMU, two ranging sensors, and an Arm Cortex processor. Despite the architectural simplicity, the 3D shape of the staircase is reconstructed with a precision which is sufficient for the correct planning of a climbing step. Furthermore, the computational time of the algorithm is fully compatible with the cycle time of the system.

A possible drawback of the method proposed in this work is related to the incremental acquisition of the point cloud. In fact, each new sample point is obtained at a different instant and its Cartesian position is computed by solving a forward kinematics problem. As a consequence, an accurate knowledge of the kinematic parameters of the system is required in order to properly build the point cloud: the mentioned accuracy was achieved through a careful identification of the system parameters. Such accuracy can also be affected by unmeasured deformations of the structure. As previously reported, good estimates have been obtained despite the system vibrations. This result was possible because the exoskeleton structure is sufficiently rigid and thanks to the angular sensors located on the joint axes, which compensate for possible backslashes along the actuation chains.

The limited computational burden of the proposed method will allow, in the future, to improve the identification precision by processing the signals of additional sensors. Currently, the on-line gait planning problem is under investigation: once the staircase has been identified, the exoskeleton step must be adapted, so as to smoothly and correctly reach the final touch down position.

REFERENCES

[1] D. Pinto-Fernandez *et al.*, "Performance evaluation of lower limb exoskeletons: A systematic review," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 28, no. 7, pp. 1573–1583, Jul. 2020.

[2] J. Jang, K. Kim, J. Lee, B. Lim, J.-K. Cho, and Y. Shim, "Preliminary study of online gait recognizer for lower limb exoskeletons," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots and Syst. (IROS)*, 2017, pp. 5818–5824.

[3] F. Gao, G. Liu, F. Liang, and W.-H. Liao, "IMU-based locomotion mode identification for transtibial prostheses, orthoses, and exoskeletons," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 28, no. 6, pp. 1334–1343, Jun. 2020.

[4] I. Kang, P. Kunapuli, and A. J. Young, "Real-time neural network-based gait phase estimation using a robotic hip exoskeleton," *IEEE Trans. Med. Robot. Bionics*, vol. 2, no. 1, pp. 28–37, Feb. 2020.

[5] H.-Y. Huang, J.-S. Chen, and C.-E. Huang, "Toward the gait analysis and control of a powered lower limb orthosis in ascending and descending stairs," *Procedia Eng.*, vol. 79, pp. 417–426, Jan. 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877705814009424

[6] R. Auberger, M. F. Russold, R. Riener, and H. Dietl, "Patient motion using a computerized leg brace in everyday locomotion tasks," *IEEE Trans. Med. Robot. Bionics*, vol. 1, no. 2, pp. 106–114, May 2019.

[7] Z. Li *et al.*, "Hybrid brain/muscle signals powered wearable walking exoskeleton enhancing motor ability in climbing stairs activity," *IEEE Trans. Med. Robot. Bionics*, vol. 1, no. 4, pp. 218–227, Nov. 2019.

[8] D. Novak and R. Riener, "A survey of sensor fusion methods in wearable robotics," *Robot. Auton. Syst.*, vol. 73, pp. 155–170, Nov. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889014001705

[9] Y. Kanazawa and K. Kanatani, "Reliability of plane fitting by range sensing," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, vol. 2, May 1995, pp. 2037–2042.

[10] C. Wang, H. Tanahashi, H. Hirayu, Y. Niwa, and K. Yamamoto, "Comparison of local plane fitting methods for range data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1, Dec. 2001, pp. 663–669.

[11] O. Fernández, "Obtaining a best fitting plane through 3D georeferenced data," *J. Struct. Geol.*, vol. 27, no. 5, pp. 855–858, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0191814105000143

[12] J. W. Weingarten, G. Gruener, and R. Siegwart, "Probabilistic plane fitting in 3D and an application to robotic mapping," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA04)*, vol. 1, Apr. 2004, pp. 927–932.

[13] S.-W. Kwon, F. Bosche, C. Kim, C. T. Haas, and K. A. Liapi, "Fitting range data to primitives for rapid local 3D modeling using sparse range point clouds," *J. Autom. Construct.*, vol. 13, no. 1, pp. 67–81, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0926580503000773

[14] Z. Tian, F. Gao, Z. Jin, and X. Zhao, "Dimension measurement of hot large forgings with a novel time-of-flight system," *Int. J. Adv. Manuf. Technol.*, vol. 44, no. 1, pp. 125–132, Sep. 2009. [Online]. Available: https://doi.org/10.1007/s00170-008-1807-8

[15] K. Zhang *et al.*, "Environmental features recognition for lower limb prostheses toward predictive walking," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 27, no. 3, pp. 465–476, Mar. 2019.

[16] K. Klasing, D. Althoff, D. Wollherr, and M. Buss, "Comparison of surface normal estimation methods for range sensing applications," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2009, pp. 3206–3211.

[17] P. Michel, J. Chestnutt, S. Kagami, K. Nishiwaki, J. Kuffner, and T. Kanade, "GPU-accelerated real-time 3D tracking for humanoid locomotion and stair climbing," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2007, pp. 463–469.

[18] S. Oßwald, J. Gutmann, A. Hornung, and M. Bennewitz, "From 3D point clouds to climbing stairs: A comparison of plane segmentation approaches for humanoids," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots (Humanoids)*, 2011, pp. 93–98.

[19] Z. Zheng, G. Zhong, and H. Deng, "A method to detect stairs with three-dimensional scanning for hexapod robot stair climbing," in *Proc. IEEE Int. Conf. Mechatronics Autom. (ICMA)*, 2016, pp. 2541–2546.

[20] Y. Feng, L. Xia, Y. He, C. Wang, Z. Yan, and X. Wu, "Stairs reconstruction with 3D point cloud for gait generation of lower limb exoskeleton robot," in *Proc. IEEE Int. Conf. Robot. Biomim. (ROBIO)*, 2019, pp. 2019–2024.

[21] S. Kumar, C. Savur, and F. Sahin, "Dynamic awareness of an industrial robotic arm using time-of-flight laser-ranging sensors," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2018, pp. 2850–2857.

[22] S. Kumar, S. Arora, and F. Sahin, "Speed and separation monitoring using on-robot time-of-flight laser-ranging sensor arrays," in *Proc. IEEE Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2019, pp. 1684–1691.

[23] J. Poppinga, N. Vaskevicius, A. Birk, and K. Pathak, "Fast plane detection and polygonalization in noisy 3D range images," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots and Syst. (IROS)*, Sep. 2008, pp. 3378–3383.

[24] R. Dubé *et al.*, "Incremental-segment-based localization in 3-D point clouds," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1832–1839, Jul. 2018.

[25] D. L. Boley, E. S. Steinmetz, and K. T. Sutherland, "Robot localization from landmarks using recursive total least squares," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, vol. 2, Apr. 1996, pp. 1381–1386.

[26] D. Kubus, T. Kroger, and F. M. Wahl, "On-line estimation of inertial parameters using a recursive total least-squares approach," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2008, pp. 3845–3852.

[27] S. Rhode and F. Gauterin, "Online estimation of vehicle driving resistance parameters with recursive least squares and recursive total least squares," in *Proc. IEEE Intell. Veh. Symp. (IV)*, Jun. 2013, pp. 269–276.

[28] C. E. Davila, "An efficient recursive total least squares algorithm for FIR adaptive filtering," *IEEE Trans. Signal Process.*, vol. 42, no. 2, pp. 268–280, Feb. 1994.

[29] D.-Z. Feng, X.-D. Zhang, D.-X. Chang, and W. X. Zheng, "A fast recursive total least squares algorithm for adaptive FIR filtering," *IEEE Trans. Signal Process.*, vol. 52, no. 10, pp. 2729–2737, Oct. 2004.

[30] G. Plett, "Recursive approximate weighted total least squares estimation of battery cell total capacity," *J. Power Sources*, vol. 196, no. 4, pp. 2319–2331, Feb. 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S037877531001654X

[31] T. Kim, Y. Wang, Z. Sahinoglu, T. Wada, S. Hara, and W. Qiao, "A rayleigh quotient-based recursive total-least-squares online maximum capacity estimation for lithium-ion batteries," *IEEE Trans. Energy Convers.*, vol. 30, no. 3, pp. 842–851, Sep. 2015.

[32] G. Golub and C. Van Loan, "An analysis of the total least squares problem," *SIAM J. Numer. Anal.*, vol. 17, no. 6, pp. 883–893, 1980.

[33] S. Van Huffel and J. Vandewalle, "Analysis and solution of the non-generic total least squares problem," *SIAM J. Matrix Anal. Appl.*, vol. 9, no. 3, pp. 360–372, 1988.

[34] Y. Nievergelt, "Total least squares: State-of-the-art regression in numerical analysis," *SIAM Rev.*, vol. 36, no. 2, pp. 258–264, 1994.

[35] X. Qian and C. Ye, "NCC-RANSAC: A fast plane extraction method for 3-D range data segmentation," *IEEE Trans. Cybern.*, vol. 44, no. 12, pp. 2771–2783, Dec. 2014.

[36] S. Miyagawa, S. Yoshizawa, and H. Yokota, "Trimmed median PCA for robust plane fitting," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2018, pp. 753–757.