J Intell Robot Syst (2014) 76:119–136
DOI 10.1007/s10846-013-9984-6

A Real-time Door Detection System for Domestic Robotic Navigation

C. Fernández-Caramés · V. Moreno · B. Curto · J. F. Rodríguez-Aragón · F. J. Serrano

Received: 20 May 2013 / Accepted: 27 September 2013 / Published online: 30 November 2013

Abstract Human beings use doors to access rooms and corridors, to know where they are, to know where they have to go, etc. Similarly, it would be quite useful for robots to be able to detect doors in order to accomplish more complex and flexible navigation tasks. Such a goal is even more desirable when domestic environments are taken into account. Moreover, if the human-robot interaction is considered, the use of this semantic information can be broadly used. In this paper we present a solid and complete door detection system which fuses data from an end-user camera and a laser rangefinder. By using both Haar-like features and the Integral Image, the computation time is significantly reduced when compared to other methods found in the literature. Extensive tests in real-world environments have been performed in order to prove the efficiency, robustness and real-time ability of our system.

Keywords Door detection · Robotics perception · Real-time applications

C. Fernández-Caramés · V. Moreno (✉) · B. Curto · J. F. Rodríguez-Aragón · F. J. Serrano
Robotics and Society Research Group, Faculty of Science, Salamanca, Spain
e-mail: vmoreno@usal.es
URL: http://gro.usal.es

1 Introduction

In urban environments, both indoors (buildings) and outdoors (streets), doors are significant landmarks that are unconsciously and constantly used by human beings in their daily lives. They represent access and connection points between the outside and the inside, and once inside, between rooms and corridors. Furthermore, we use them to know where to go when we want to find someone or something. Once we recognize a door we are able to know where we are by means of a comparison with an environment representation.

Within domestic environments, service robots can take advantage of the use of high-level information. However, robots still lack proper spatial cognition or, in other words, the ability to understand and to interpret their surrounding environment as well as the objects encountered in it. This is an important shortcoming and arises from the fact that replicating the immense knowledge that people possess about environments is an utterly difficult task for robots [31].

Visual door recognition has been widely used in robot localization, which is an extensive matter. Many different and successful strategies have hitherto been developed [8, 12, 13, 15]. Vertical edges have long been used by the robotics community as a first step for door detection. Using this type of visual landmarks, Sugihara [23] presented one of the first approaches to solve the problem of monocular vision localization. Assuming the camera parameters were known and the observations were error-free, Sugihara obtained a set of vertical edges from each image and estimated the robot position and heading by means of triangulation. To do so, he computed the angle between pairs of consecutive observations and match them to existing landmarks in an a priori 2D world model. At least three observations are required, though more can also be used. Extending this approach, Krotkov [10] proposed a similar model where, using a worst-case analysis, the effects of uncertainty were also taken into account. Under the same assumptions, Atiya and Hager [30] represented errors using tolerances, which led them to propose a set-based algorithm for solving the matching problem. They also analyzed the effect of observing false positives, unlike the aforementioned methods, which is a common problem in mobile robot applications. Madsen and Andersen [7] showed that using landmark triplets for triangulation, the observation error can be so severe that navigation may prove impossible, but it can be solved predicting the uncertainty of the triplets and selecting the most precise ones. All these methods deal with the problem of absolute localization.

More sophisticated error models, such as the one used in extended Kalman Filters (EKF), have also been applied to localization based on vertical edges. EKF-based methods usually provide incremental localization, and a popular technique consists of projecting the location of map features onto the camera image using the estimation of the robot pose given by odometry. This allows to narrow the search space when it comes to detect landmarks in the images. Following this approach, Crowley and Chenavier [13] presented an EKF in order to fuse odometry with vision using the Hough transform for finding vertical lines. The tests they performed were quite uncomplicated, involving a map consisting of five landmarks, preplanned robot paths, and the camera to be manually positioned toward the expected features. A similar but much more thorough and comprehensive work was the complex FINALE system by Kosaka and Kak [2], which also relied on the Hough transform for vertical line detection but assumed that geometric 3D models of the environment were available. Arras and Tomatis [20] and Arras et al. [21] also use the EKF for fusing vertical features detected in images with walls detected with a 2D laser rangefinder.

Other probabilistic-based methods rely on the Symmetries and Perturbations model (SPmodel) [19] for representing uncertainty. Some examples using vertical edges and the SPmodel include combining sensory information from vision and laser 2D using an EKF [18], an Extended Information Filter [16] or range and intensity laser 3D data [17]. One of the main limitation comes from the fact that the detected features such as vertical lines, do not have a significant semantic value to be used with other kind of information like maps.

Our proposal is concerned with high-level features like doors or corridors which are interesting because they are considered as key elements in urban buildings to achieve a localization with a high semantic or symbolic processing capabilities. Many other high-level tasks can be performed having this kind of features successfully detected and located in a symbolic map. Our proposal gives as result the position of the surrounding doors by fusing the information from a monocular webcam and a 2D laser rangefinder. By considering a real-world environment, we will demonstrate that our proposal may perform the door detection task very reliably with a computational cost that allows the procedure to be used with a light on-board computer and an end-user camera. Even using non specialized end-user components, our procedure has a remarkable performance. Therefore it is shown as a very valuable tool for a robotic platform working within urban buildings.

This paper is organized as follows. A depth review of the works that have considered the door detection problem will be presented in Section 2, showing their strengths and weaknesses. A global description of the procedure will be shown in Section 3. Several and heterogeneous environments have been considered in order to show the validity of our proposal. Finally, the suitability of our procedure to run under real-time conditions will be proved and computation times are shown in Section 4.

2 Related Works Concerning Door Detection

Perhaps the most naive approach to door detection is to classify open doors attending to their width, using laser 2D data [24, 27]. Similarly, Cariñena et al. [26] employed ultrasonic sensors and fuzzy reasoning, but only one door at a time can be detected and it has to be within one meter from the robot. Anguelov et al. [9] use a 2D laser scanner combined with stripes of color from an omnidirectional camera. They mainly detect doors from range data, assuming that doors are dynamic objects that change their state (open or closed) over time. Additionally, they detect a few static doors presuming an uniform color model.

The main shortcoming of methods based on range data is that they will not work unless a door is either open or the door plane is clearly distinguishable from the wall plane. When doors are flushed with the wall, which is very common in some environments, a vision sensor becomes essential. Using a trinocular stereo vision system, Kim and Nevatia [8], extracted vertical and horizontal segments, verifying door candidates by measuring real distance data obtained from the stereo setting.

Despite the undeniable usefulness of real distance data, numerous researchers rely just on monocular vision as the only sensor, which force them to devise pixel-based metrics in order to classify objects as doors. An important drawback affecting these methods is that the success of the classifiers is effectively restricted to those doors that are at a particular distance and orientation from the robot camera. Therefore, only doors nearby and parallel to the image plane are usually recognized, as door proportions are severely affected by perspective. Under this rationale, Monasterio et al. [11] measured distances between vertical edges detected with a Sobel filter. Muñoz-Salinas et al. [29] developed a fuzzy system based on horizontal and vertical segments extracted with the Hough transform. Yang and Tian [32] combined edge and corner detection with a number of pixel-based thresholds. Murillo et al. [3] present a probabilistic approach built upon a model described by the geometry and appearance of doors. The appearance model is learned from hand-labeled data and hence depends on the trained door colors. The geometry model is defined in terms of pixels, and just as the previous methods, the system fails when doors are perpendicular to the image plane, since doors are much narrower. Furthermore, the authors admit to have a relatively high rate of false positives. Shi and Samarabandu [34] manage to detect doors farther away from the camera using a more informed approach that tests whether a door jamb candidate fall into the floor area or not. Surprisingly enough, their system cannot detect doors too close, because they reject doors without the top bar or door widths greater than one third of the image.

A very interesting alternative to the aforementioned methods is the machine learning approach by Chen and Birchfield [35], where they combine several weak classifiers using the Adaboost algorithm to produce a strong classifier. Hensler et al. [14] augment Chen's framework by adding 2D laser range data to estimate the concavity and width of the doors. Some of the weak classifiers used in these two pieces of work include tests for vertical lines, concavity, gap between the door and the floor, color, texture, kick plates, door width and door knobs. A strong point of these methods is that they achieve an extremely low false positive rate. However, they still focus just on nearby doors, and cannot achieve real-time performance even using low resolution images (320×240, 5 fps), as a result of the number of tests needed to be run per image. In [36], Chen et al. improve their previous work by using a data-driven Markov chain Monte Carlo process.

Finally, it is noteworthy that one of the most important elements that makes doors different from other objects is door handles. Nevertheless, these elements are difficult to be detected due to their extremely small size and varying shape, and therefore, usually more reliably detected from a very short distance range. Exploiting the fact that door handles protrude from the door surface, Rusu et al. [28] used 3D range data to successfully detect door handles with a PR2 robot and subsequently open them with its robotic arm. The capability of opening doors is very desirable because allows a robot to enlarge its navigation autonomy.

3 Proposed Door Detection Procedure

In this section, our door detection procedure will be explained. Figure 1 shows a block diagram of the whole procedure. In the following subsections, every functionality needed for the correct performance of our proposal will be detailed.

Two sensors are considered: a single end-user camera and a laser range finder. In this way, our procedure receives each time step as parameters an image (*Image*), which is captured from the on-board camera, and the data obtained from the laser rangefinder ($\mathcal{S}$). At each time step t our SICK laser rangefinder obtains a set of $n = 361$ range measurements, spanning a semicircle:

$$\mathcal{S} = \{s_1, \ldots, s_n \mid s_i = (\rho_i, \theta_i)\} \quad (1)$$

The measurements are acquired in a counterclockwise ordered sequence, and each tuple (ρ_i, θ_i) contains the polar coordinates of the i-th scan point. The angular resolution of the device is fixed and equal to $\Delta\theta = \theta_{i+1} - \theta_i = 0.5°$.

These information sources are fused in order to get more information. The treatment of the laser data for the corridor detection and the image processing methods done in order to discover the visible doors are explained in Sections 3.1 and 3.3, respectively. The final decision method is explained in Section 3.4.

Our robot, which is shown in Fig. 2, is equipped with a laser scanner and a monocular camera, and using the calibration procedure of Guan et al. [6] the laser measurements can be projected onto the image and converted to pixels.

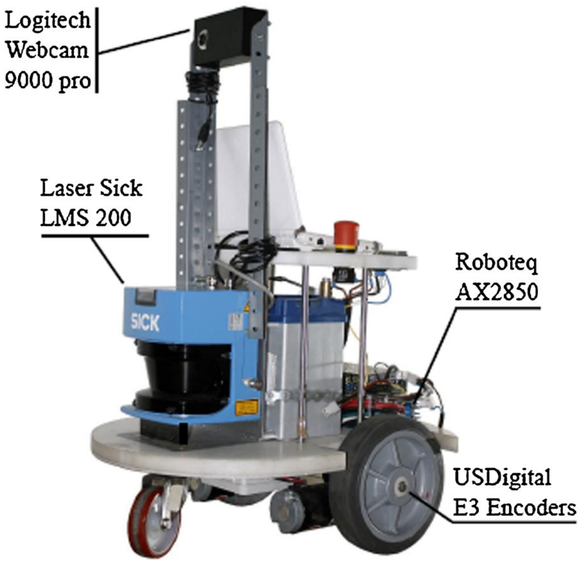
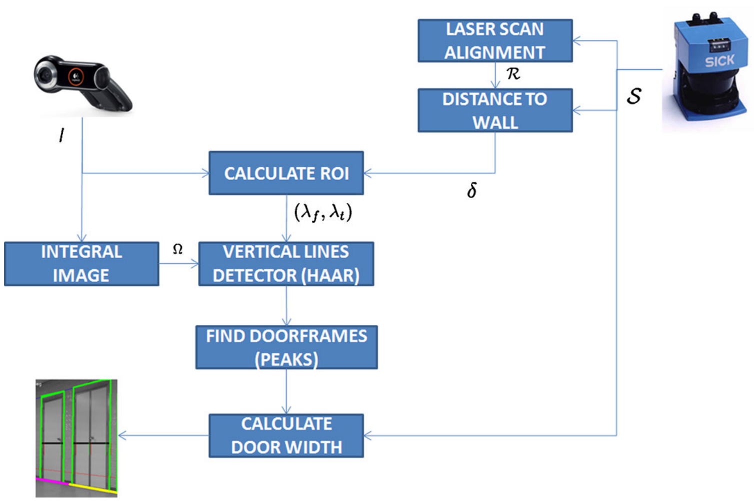


Fig. 2 MoRLaCo: mobile robot laser controlled

3.1 Laser Scan Alignment

In order to find out the angle ψ that the corridor forms with the heading direction of the robot, we use a modified angle histogram technique shown in Algorithm 1, based on the work from Weiß et al. [33]. If the scan points are interpreted as a set of vectors $\vec{\mathcal{V}}$ (2), a new set $\vec{\mathcal{W}}$ (3) can be then defined having each $\vec{w}_i$ as the difference between

Fig. 1 Block diagram of the door detection system



$\vec{v}_i$ and its j-th successor. The angle histogram is just the angle distribution of the elements in $\vec{\mathcal{W}}$, and the histogram maximum corresponds to the corridor angle ψ.

$$\vec{\mathcal{V}} = \{\vec{v}_1, \ldots, \vec{v}_n \mid \vec{v}_i = (\rho_i \cdot \cos(\theta_i), \rho_i \cdot \sin(\theta_i))\} \tag{2}$$

$$\vec{\mathcal{W}} = \{\vec{w}_1, \ldots, \vec{w}_{n-j} \mid \vec{w}_i = (\vec{v}_{i+j} - \vec{v}_i)\} \tag{3}$$

When $\|\vec{w}_i\|$ is on the order of the laser measurement noise (σ_l), which is $\sigma_l \approx 5$ mm for our laser, the vector angle happens to be very noisy. This is a recurring problem that Weiß et al. suggest to solve by choosing j appropriately. However, this solution still results in noisy maxima in the angle histogram, because the j value is fixed and $\|\vec{w}_i\|$ may still be on the order of σ_l.

To achieve less noisy histograms, we propose to compute the set $\vec{\mathcal{W}}$ as shown in Eq. 4. Thus, rather than using a fixed j value, j is adaptively chosen such that $\vec{v}_{i+j}$ is the vector that lets the inequality $\|\vec{w}_i\| \geq d_h$ hold true. By choosing $d_h \gg \sigma_l$, the angle of $\vec{v}_{i+j}$ is not significantly affected by sensor noise. Additionally, so as to make the histogram more reliable, we compute another set $\vec{\mathcal{W}}'$ defined just as $\vec{\mathcal{W}}$, but using the constraint $\|\vec{w}_i\| \geq 2 \cdot d_h$

$$\vec{\mathcal{W}} = \{\vec{w}_i, \ldots, \vec{w}_t \mid \vec{w}_i = (\vec{v}_{i+j} - \vec{v}_i),\ \|\vec{w}_i\| \geq d_h\} \tag{4}$$

Thus, the number of points used to calculate the corridor orientation is now doubled. Since the resulting histogram is better defined, errors in the histogram interpretation were reduced.

Once the corridor orientation has been determined the $\mathcal{S}$ set must be rotated in order to find the walls more easily, as it will be explained in the next subsection. Equation 5 shows the expressions that are applied to rotate the laser scan point set.

Algorithm 1 Relative corridor orientation

```
1:  procedure CorridorOrientation($\mathcal{S}$)
2:      $\vec{\mathcal{V}} \leftarrow polar\_to\_cartesian(\mathcal{S})$
3:      $x_{min} \leftarrow \min(\mathcal{S})$
4:      $x_{max} \leftarrow \max(\mathcal{S})$
5:      $h \leftarrow histogram(x_{min}, x_{max})$
6:      for $i \leftarrow 1, \cdots, n-1$ do
7:          $\vec{w}_i \leftarrow \vec{v}_{i+j} - \vec{v}_i$ such that $\|\vec{w}_i\| \geq d_h$
8:          $increment(h, ang(\vec{w}_i))$
9:          $\vec{w}'_i \leftarrow \vec{v}_{i+k} - \vec{v}_i$ such that $\|\vec{w}_i\| \geq 2 \cdot d_h$
10:         $increment(h, ang(\vec{w}'_i))$
11:     end for
12:     $\psi \leftarrow \max(h)$
13:     return $\alpha$
14: end procedure
```

$$\mathcal{R} = \{r_1, \ldots, r_n \mid r_i = (x_i, y_i)\}$$
$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix} \cdot \begin{bmatrix} \rho_i \cos(\theta_i) \\ \rho_i \sin(\theta_i) \end{bmatrix} \tag{5}$$

The result of the scan rotation can be observed in Fig. 3.

3.2 Wall Detection and ROI Calculation

Once the scan points have been aligned with the Y axis, the corridor walls can be efficiently detected using an X-histogram.

The X-histogram is a technique robust to outliers, such as the Hough transform, but requires less computer resources. It represents the x coordinates distribution of the rotated scan point set $\mathcal{R}$. When both corridor walls are visible, the histogram has two maxima indicating the distance to the left and right corridor walls (δ_l, δ_r). The

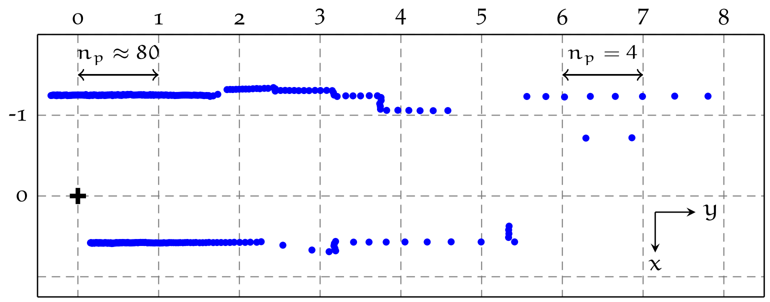


Fig. 3 Laser scan aligned with the Y axis

maxima are clearly identifiable, because the left wall has always negative x values, whereas the right wall has positive x values.

A shortcoming is that there are much more points close to the sensor than far from it as shown in Fig. 3. For example, close to the sensor there are 80 points per meter and there are 4 points per meter far from it. Consequently, the surfaces close to the sensor will determine the result of the x-histogram, making it potentially unreliable. How to find the distance to the left wall is detailed in Algorithm 2. The distance to the right wall must be computed in a similar way. A set B is computed where each value b_i is true if the point n_i is a breakpoint with respect to the point n_{i-1}, or it is false otherwise. This set is computed using the Distance-based Convolution Clustering (DCC), as explained in [4].

Once we have estimated the corridor orientation (ψ) and the distance to the left (δ_l) and right (δ_r) walls. In the robot coordinate system, $\{X_R, Y_R, Z_R\}$, these parameters suffice to define a vertical plane corresponding to the wall as:

$$x_r \cos(\psi) + y_r \sin(\psi) = \delta \tag{6}$$

where δ is either δ_l or δ_r.

If the left and right wall planes are intersected with the floor plane $z_r = 0$ and with a horizontal plane at the top of the door height $z_r = 2.0$, four lines are obtained that define the area in which doors are expected to be found. If this four lines are projected onto the image, then the image lines $\lambda_f, \lambda_t, \xi_f, \xi_t$ are obtained, which define *regions of interest* (ROI) in the image where doors can be found. This is illustrated in Fig. 4.

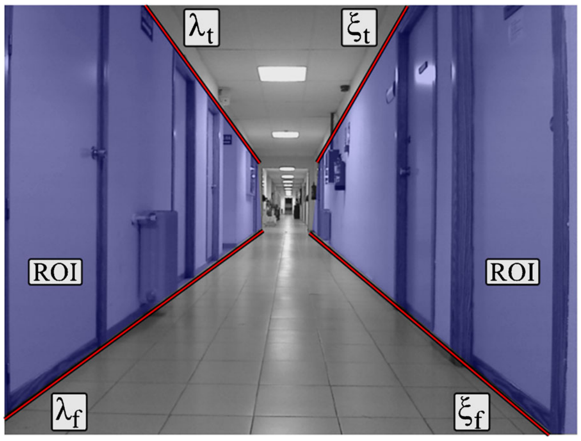


Fig. 4 Wall lines and regions of interest in the image

3.3 Vertical Line Detection

When a door is considered, at least two vertical lines define the contour of a doorframe. On the other hand vertical lines are useful because they are not affected by the a perspective deformation if the camera optical axis is set horizontal.

Algorithm 2 Find the distance to the left wall

```
1: procedure DISTANCE_TO_LEFT_WALL($\mathcal{R}$)
2:     $\mathcal{N} \leftarrow$ points from $\mathcal{R}$ with $r_i^x < 0$
3:     $x_{min} \leftarrow \min(\mathcal{N}^x)$
4:     $x_{max} \leftarrow \max(\mathcal{N}^x)$
5:     $m \leftarrow num\_points(\mathcal{N})$
6:     $h \leftarrow x\_histogram(x_{min}, x_{max})$
7:     $\mathcal{B}(b_1, \cdots, b_m) \leftarrow dcc\_breakpoints(\mathcal{N})$        ▷ $b_i \leftarrow true|false$
8:     for $i \leftarrow 1, \cdots, m-1$ do
9:         if $(n_{i+1}^y - n_i^y) > (n_{i+1}^x - n_i^x)$ then
10:            if $b_{i+1}$ is false then
11:                $increment(h, euclidean\_dist(n_i, n_{i+1}))$
12:            end if
13:        end if
14:    end for
15:    return $\max(h)$        ▷ Distance to left wall
16: end procedure
```

Line detection usually involves the following steps: (1) image smoothing, (2) edge detection, (3) constructing lines from individual edges by means of connected components analysis, and (4) line filtering [12]. The first two steps can be performed by means of a convolution operation, using a gaussian filter for smoothing and a Prewitt or Sobel filter for edge detection.

If a kernel of size $P \times Q$ is convolved with an image having N_{px} pixels, the standard convolution needs $\tau_n = N_{\mathrm{px}} \cdot P \cdot Q$ operations, whereas a separable convolution (suitable for gaussian, Prewitt or Sobel) is faster and takes τ_s operations:

$$\tau_s = N_{\mathrm{px}} \cdot P + N_{\mathrm{px}} \cdot Q = N_{\mathrm{px}} * (P + Q) \tag{7}$$

Instead of this approach we propose the use of the Haar-like features that will be calculated using integral image operation. In the following the procedure will be exposed.

Having an image I of dimensions $w \times h$, the integral image is a matrix S with the same dimensions, where the element of coordinates (x, y) has the sum of every pixel found above and on the left of the point (x, y), including such point. Thus, the integral image value for the point (x, y) can be expressed by the sum:

$$S(x, y) = \sum_{i=0}^{x} \sum_{j=0}^{y} I(i, j) \tag{8}$$

The computational cost to calculate the integral image varies greatly depending on its formulation. According to Kisačanin [22], the Eq. 8 takes $\frac{1}{4}w^2h^2$ operations. Using a recursive approach, the computational cost of the integral image calculation can be significantly improved. Taking advantage of the integral image properties, each of its elements can be recursively calculated in a single pass, as:

$$\begin{aligned} S(x, y) = {} & S(x - 1, y) + S(x, y - 1) \\ & - S(x - 1, y - 1) + I(x, y) \end{aligned} \tag{9}$$

Having this formulation, $3 * w * h$ operations are only needed. However, at [25] Viola–Jones a most efficient way $(2 * w * h)$ is presented where a pair of recurrences appear

$$R(x, y) = R(x, y - 1) + I(x, y) \tag{10}$$

$$S(x, y) = S(x - 1, y) + (x, y) \tag{11}$$

where $r(x, y)$ is the cumulative sum of the pixels of a row.

Instead of searching for all possible lines in the image (using a generic line detector) we have opted for a more informed search that is possible because a data fusion has been performed. As depicted in Fig. 5, the lines λ_f and λ_t not only define the ROI where doors can be found in the left wall. They also specify the expected height of a doorframe taking into account the perspective deformation. Using this valuable information together with adaptive vertical Haar-like features, vertical lines corresponding to doorframes can be found with a minimum number of calculations.

The complete algorithm is specified in Listing 3. From now on we use the term *haar window* to refer to a vertical haar-like feature. In Fig. 5, haar windows are illustrated as rectangular pixel areas vertically split in two: a black half (subtraction) and a white half (addition). After adding and

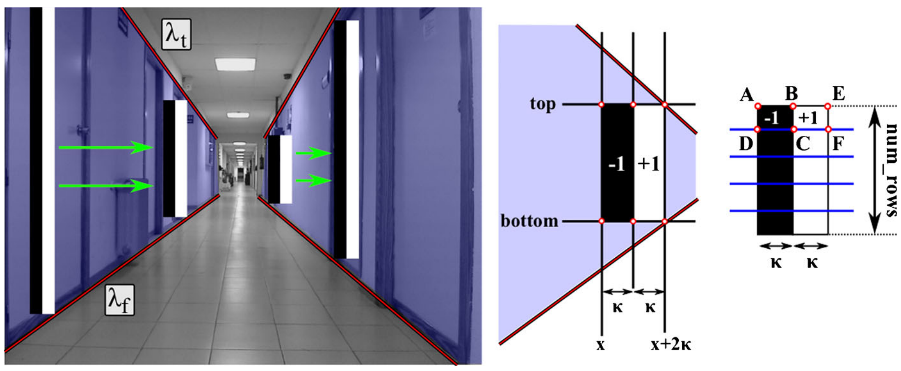


Fig. 5 Combining haar windows and regions of interest for detecting vertical lines corresponding to door frames

subtracting the pixels in both halves, the result is nonzero if there is an abrupt vertical change of intensity in the image. The parameter κ defines half of the window width, and we have set it to $\kappa = 3$ pixels. If the parameter κ is smaller it might not be possible to detect an abrupt change of intensity. Moreover, if κ is increased, the window may occupy an entire doorframe, especially when doors are far and smaller in the image. At this point, a key element of our procedure is found, due to the use of the integral image operation. This one provides a strong reduction of the computation because, for any ROI size, only four operations will be needed instead of the consideration of the whole set of the ROI points.

Our approach consists of moving the haar window from the left to the right of the image, obtaining an array of values which are nonzero where vertical lines with the height of a door are located. Thus, the height of each haar window is adapted so that it touches both λ_f and λ_t. For each pixel x, the vertical line $x + 2\kappa$ is intersected with λ_f and λ_t, obtaining the horizontal lines *top* and *bottom*, needed to adaptively adjust the size of the window.

In order to make the procedure more robust, each haar window is subdivided in n_r rows. Then, for each row, a haar subwindow is computed by taking the difference between the areas $BEFC$ and $ABCD$. This difference can be computed with just six references to the integral image. The resulting difference is normalized dividing by the area of half a window. If the normalized value is bigger than a threshold value T_i, it means that a vertical edge in that subwindow exists, and the number of edges n_e is increased. Finally, if the percentage of edges (n_e/n_r) is bigger than a threshold percentage T_p, the normalized sum of all the haar

Algorithm 3 Vertical lines detection

```
1: procedure ADAPTIVE_HAAR($x, \lambda_f, \lambda_t, \Omega$)
2:     $(top, bottom) \leftarrow intersect\_lines(x, \lambda_f, \lambda_t)$
3:     $sum \leftarrow 0, n_e \leftarrow 0$
4:     $\Delta r = (top - bottom)/n_r$
5:     for $i \leftarrow 0$ to $n_r - 1$ do
6:         $A \leftarrow (x, top + i * \Delta r)$                ▷ $A = (A_x, A_y)$
7:         $D \leftarrow (x, A_y + \Delta r)$
8:         $temp \leftarrow haar\_window(A, D, \Omega)$
9:         $sum \leftarrow sum + temp$
10:        if $temp > T_i$ then
11:            $n_e \leftarrow n_e + 1$
12:        end if
13:    end for
14:    $sum \leftarrow sum/n_r$                ▷ Sum normalization
15:    if $(n_e/n_r) < T_p$ then
16:        $sum \leftarrow 0$
17:    end if
18:    return $sum$
19: end procedure
20: procedure HAAR_WINDOW($A, D, \Omega$)
21:    $B \leftarrow (A_x + \kappa, A_y)$
22:    $C \leftarrow (D_x + \kappa, D_y)$
23:    $E \leftarrow (B_x + \kappa, B_y)$
24:    $F \leftarrow (C_x + \kappa, C_y)$
25:    $sum \leftarrow 2\Omega(B) + \Omega(D) + \Omega(F) - [\Omega(A) + \Omega(E) + \Omega(2C)]$
26:    return $sum/area(ABCD)$
27: end procedure
```

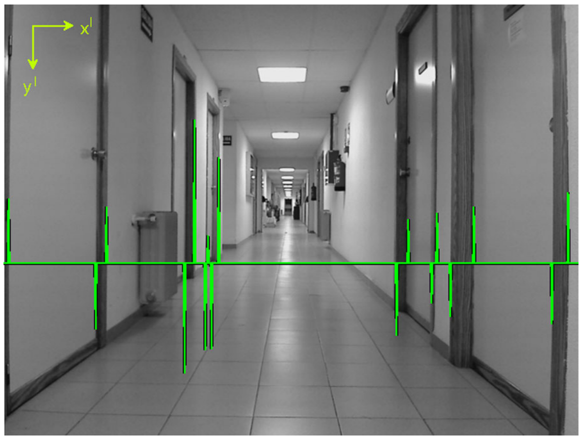


Fig. 6 Intensity variation function

subwindows is the final intensity value. Otherwise, zero is returned. The purpose of this technique is that choosing n_r and T_p appropriately, partially occluded doors can be detected. In our case, $n_r = 6$. A higher value would imply a small size of each row for distant doors and this would cause errors. The percentage value used for the acceptance of an edge is $T_p = 0.8$. Using this value, partially occluded doors are detected. Using a smaller value, there is a risk of accepting as edges other elements of the image which are not doorframes. The result of the algorithm is shown graphically in Fig. 6. As it can be seen the results are very sharp, because the haar windows average the pixel values.

3.4 Final Door Detection

The next step is to look for peaks in the intensity variation function. In sum, a peak is both a maximum and a minimum of that function. A doorframe is a variation of the intensity values in two directions: while entering from the wall to the doorframe there will be a first variation and an inverse variation will take place while exiting the door frame to the wall. Therefore, our aim is to find two consecutive and inverse intensity breaks. That is what we will consider as a door frame. While looking for peaks, if a couple of peaks with the same sign are found, the one with the largest absolute value will prevail.

The pixels that are on both ends of every door frame are saved. Using the laser projection previously explained, the actual points of both pixels are estimated interpolating the two laser points whose projection on the image are closer to the pixels that define the door frame. This let us calculate actual distances between door frames. Taking as a possible door width a distance between 0.7 m and 1.1 m, a door will be registered if the distance between two detected door frames is found between those values. The distance between the doorframes is chosen according to the normalization rule UNE 56801:2008 of the Spanish Normalization Agency [1], which gives the measurements of the door sets in buildings.

Therefore, our proposal results in a set of candidate doors that meet the following characteristics:

- 2 doorframes are detected sequentially (vertical lines)
- these doorframes have at least a certain height
- the distance between both doorframes is within the ranges established by the normalization rule
- both doorframes (vertical lines) are located on the line of the wall

4 Experimental Results

In order to check the success of our classifier, we have generated several test datasets from different locations and environments within the Faculty of Science of the University of Salamanca. For this purpose, we have used our robot *MoRLaCo* (*Mobile Robot Laser Controlled*), which is a customized robot from the Robotics and Society Group of the University of Salamanca, equipped with a SICK LMS 220 laser sensor and an ordinary Logitech QuickCam Pro 9000 camera, both connected to an onboard laptop with a Intel®Core™i3 CPU U380 at 1.33 GHz CPU and 2 GB RAM, running Fedora 16. In our experiment, we use a scanning angle of 180, a maximum scan range of 7.0 m, an angular resolution of 0.5° and a sampling rate of 30 Hz. For the camera, a

Table 1 Resulting parameters of the laser calibration

	Laser parameters
α	0.96°
H	0.252 m

Table 2 Resulting parameters of the camera calibration

Parameter		Value
Intrinsic	f	3.736 mm
	K_1	$3.621 \cdot 10^{-3} \ \frac{1}{\mathrm{mm}^2}$
	Cx	772 px
	Cy	676 px
Extrinsic	T_x	−34.538 mm
	T_y	−743.826 mm
	T_z	29.232 mm
	R_x	94.07°
	R_y	0.31°
	R_z	0.45°

640 × 480 resolution has been selected. The calibration parameters of the laser scanner are shown in Table 1, where α is the tilt angle of the laser and H is the distance between the laser sensor and the camera. The intrinsic and extrinsic parameters of the camera are shown in Table 2, where f is the focal length of the camera, K_1 is the radial lens distortion coefficient, C_x, C_y are the coordinates of center of radial lens distortion, T_x, T_y, T_z are the translation components for the transformation between the world and camera coordinates, and R_x, R_y, R_z are the rotation angles for the transformation between the world and camera coordinates.

4.1 Testing the Door Detection Strategy

The test data were captured while navigating through the corridors of the building. They include different environments, with different objects, occlusions, door positions and different door types, colours, etc.

Figure 7a shows a corridor of a department, with doors in both sides and objects such as bookshelves, radiators, etc. Figure 7b shows the Dean's Office corridor, where the doors are different from the previous ones and there are plants and benches that do not let a complete door visibility. Figure 7c is taken in the main corridor of the Faculty which is a wider corridor where different kind of doors are found, having shiny tiles on the walls. Figure 7d shows the wall of another department with opened doors on one side and columns and

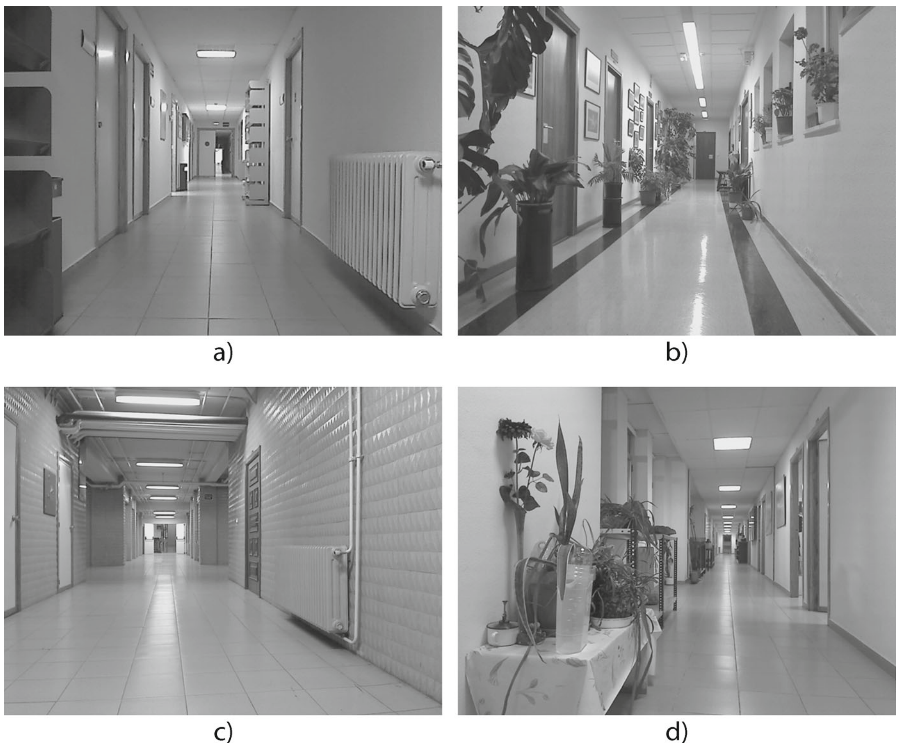


Fig. 7 Different environments from the test datasets. **a** Corridor of the Computer Science and Automation Department. **b** Corridor of the Dean's Office. **c** Main corridor of the Faculty. **d** Corridor of the Geological Engineering Department

Table 3 Results of our classifier

Distance	Number of total doors	% of hits	% false positive
$d < 3$	278	91	2
$3 < d < 5$	348	84	5
$5 < d < 7$	296	57	6

objects on the other wall. In sum, there is a set of heterogeneous environments in order to test the generality of our classifier. The test dataset consists of 500 images with their respective laser scan data. We have established three categories based on the distance between the robot and the door to observe the performance of our classifier, while the distance to the possible doors increases: less than 3 m, between 3 and 5 m, and between 5 and 7 m. The result of the performance of our classifier over the dataset is shown in Table 3.

Figure 8 shows the classifier performance for different cases. The corridor is identified with the top and bottom lines. The bottom line is divided into three sections with three different colors used to identify the distance: blue (< 3 m), yellow (3 to 5 m) and magenta (> 5 m). The red and green dots are the points of the laser sensor projected on the image. Figure 8a and b shows the door detection in two different environments with doors of different characteristics. Figure 8c shows the detection of a door in an adverse situation, having occlusions caused by plants, bins, etc. A failure detecting the closest door can also be observed. Figure 8d shows the detection of both opened doors, proving that our classifier works correctly in more or less adverse situations.

Obviously, the performance of the classifier decreases as the distance to the doors increases. This occurs due to the lower definition in the image of the doorframes, as well as to the lower resolution of the laser sensor depending on the distance.

Although the tests were performed while navigating through different corridors of a building, it is not necessary that the vision system captures two walls in order to get the doors detected properly. Using the laser sensor data, with an aperture range of 180°, the regions of the walls on which potential candidate doors are located will be identified. Afterwards, using the image cap-

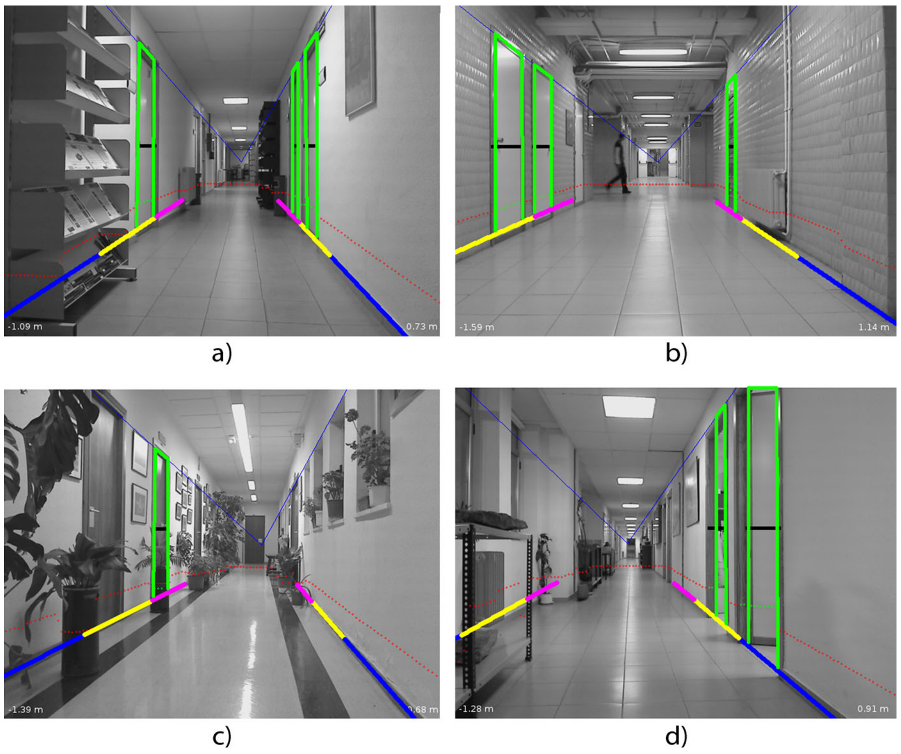


Fig. 8 Performance of the classifier in different environments. *Blue line*: $d < 3$ m; *Yellow line*: $3 \text{ m} < d < 5$ m; *Magenta line*: $5 \text{ m} < d < 7$ m

tured by the camera, the doors on those regions of interests will be detected. That makes that our system works not only in corridors but in diverse environments where both walls are not identified.

Figure 9a and c shows the detected doors found on a single wall. Figure 9b shows that our system detects successfully an incomplete door which is captured frontally in a room. Finally, Fig. 9d shows a door with a different morphology to the previous ones. It is located in a hall and that is the reason why the camera only captures a unique wall. Figure 9d also shows the strength of our system taking into account that there is no false positive in objects which may have similar characteristics to the doors, as the bulletin boards and the gap for the bin.

A study of the false positives found is necessary. The finding of a false positive during a navigation may generate a multitude of errors in location issues. The number of detected false positives in the images also depends on the distance from the robot. Table 3 shows the false positives found in the test dataset. In general, we are talking about a 4 % of false positives on average for all distances.

Figure 10 shows several examples of false positives. Figure 10a shows a detected door at the site occupied by a bookshelf. Two windows are detected as doors by our classifier in Fig. 10c. However, these false positives are quickly fixed in the following executions of the classifier. Our classifier does not detect doors erroneously in previous or subsequent frames, as it can be observed in Fig. 10b, which corresponds to the next frame to Fig. 10a. Figure 10d is the frame previous to Fig. 10c, and both windows are not identified as doors this time. Therefore, the effect of the false positives is minimized during the navigation of the robot in the environment caused by this quickly correction of our classifier.

With all these considerations, we may consider our door detector very reliable for distances less than 5 m and it provides good information for distances greater than 5 m. Contextualizing the problem of door detection within the field of the robot location during the navigation inside a building, our classifier has been proved as very useful for detecting nearby doors with confidence. We can also state that, even losing performance, it

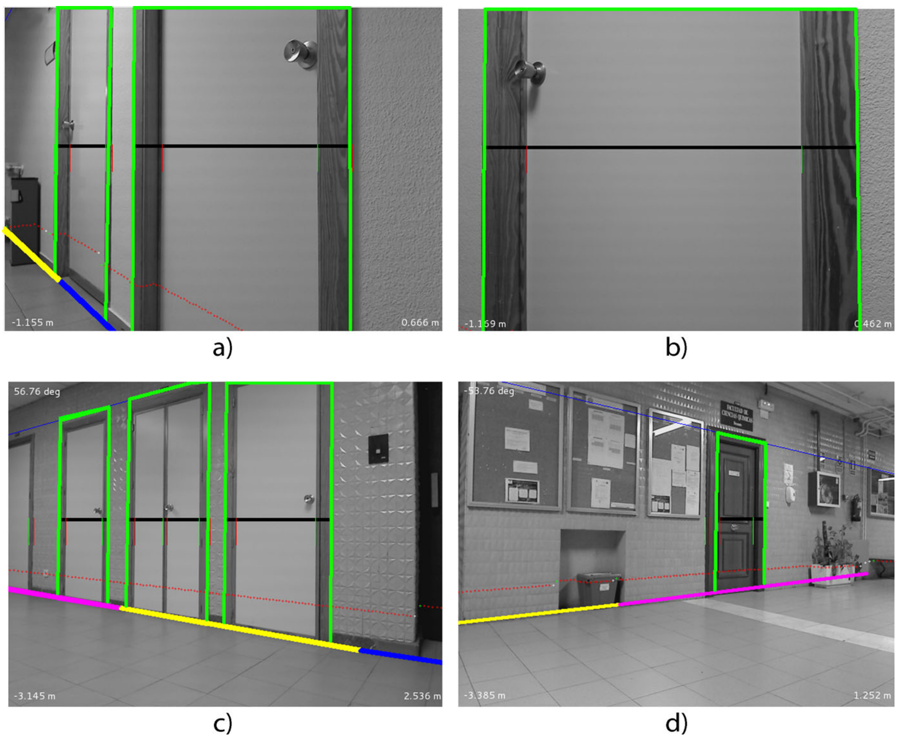


Fig. 9 **a** Two doors in the same wall of a corridor. **b** Door in a room detected frontally. **c** Doors in a wall (single and double doors). **d** Door in a hall and with different morphology. *Blue line*: $d < 3$ m; *Yellow line*: $3\text{ m} < d < 5$ m; *Magenta line*: $5\text{ m} < d < 7$ m

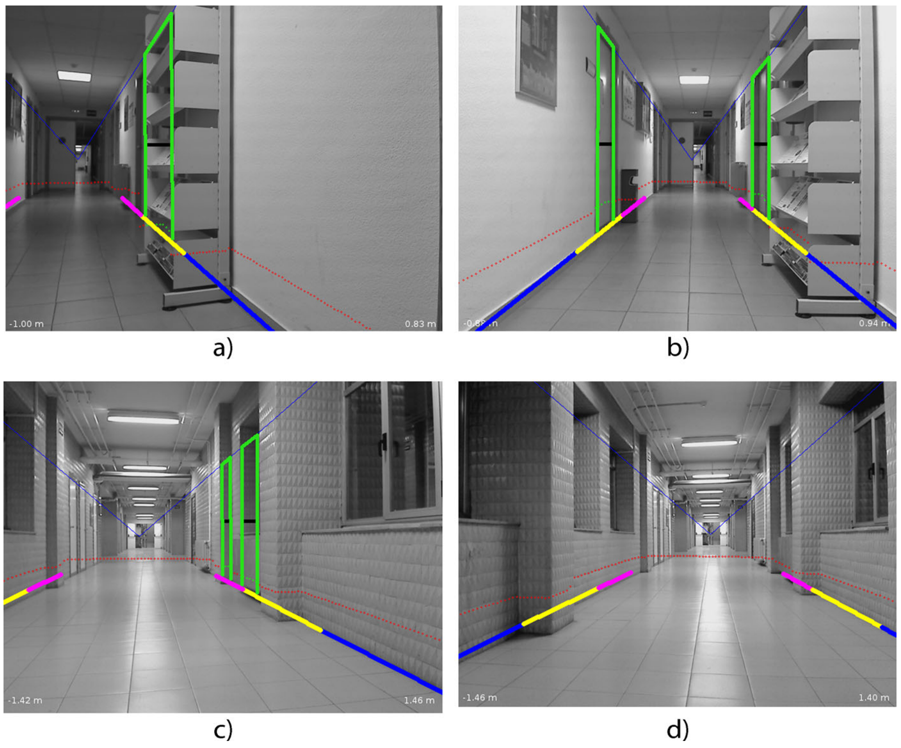


Fig. 10 False positives. **a** False positive found as a bookshelf. **b** False positive corrected in the next frame. **c** False positives found as two windows. **d** Any false positive found in the previous frame. *Blue line*: $d < 3$ m; *Yellow line*: 3 m $< d < 5$ m; *Magenta line*: 5 m $< d < 7$ m

continues working in an acceptable way to detect distant doors. This allows us to build a dynamic door probability map while moving through a corridor, giving greater weight to the detected doors depending on the proximity to the robot.

4.2 Suitability to Real-time Execution

It is essential to have small execution times, which allows the classifier to perform the necessary calculations to be able to use the door detection in localization processes during navigation. In our case, each execution of our classifier takes 1.35 milliseconds. With this timing results, there is absolutely no problem in considering our classifier running in real-time localization processes with a capability of more than 740 fps. Our camera has a capability of 30 fps which can be acquired with no restriction coming from the classifier execution.

As an example, considering our robot moving at a human walking speed, which is 1.5 m/s, it would take almost 4 s to advance 5 m. This is the distance over which our classifier works very reliably. As each cycle of image capture plus door detection takes 1.35 ms, it could run almost 3.000 door detections until reaching the place where the classifier loses efficiency. Actually, the limits are given by the capture periods of the camera, more than by our classifier. Thus, there will be 120 effective executions of our classifier before accomplishing these 5 m. Therefore, this performance loss is completely corrected and a very reliable probabilities map could be made using all the information provided by our classifier during the time spent to advance these 5 m.

A study on how the enlargement of the captured image affects the classifier efficiency was performed taking advantage of the good execution times. The same images were captured, but in this time with an image resolution of 1600×1200 pixels. The results are similar to the ones obtained while using smaller images. The conclusion reached is that the laser resolution prevents the improvement of these results. To avoid such an excessive dependency with the laser resolution, the points obtained from that sensor were interpolated simulating a straight line, as it is assumed in a wall. In this way, every point on the line of

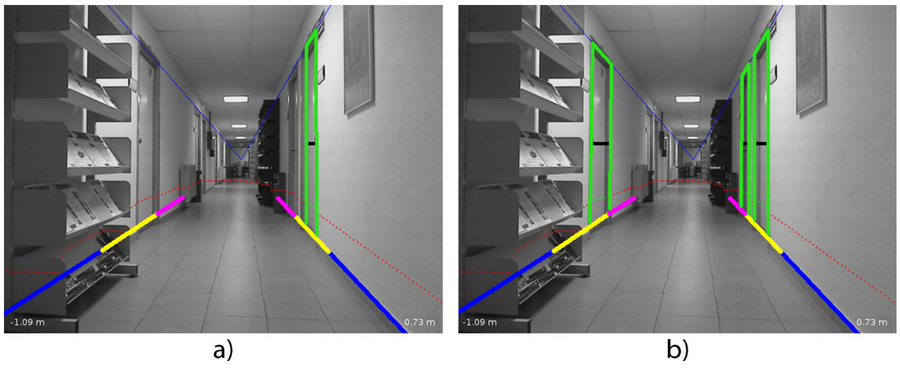


Fig. 11 1600 × 1200 images using laser point interpolation. **a** Without using laser point interpolation. **b** Using laser point interpolation. *Blue line*: $d < 3$ m; *Yellow line*: 3 m $< d <$ 5 m; *Magenta line*: 5 m $< d <$ 7 m

the wall can be projected on the image and the distances can be estimated. The actual laser points are measured exactly, but every point between two known laser points can be estimated.

Using this laser point interpolation process, we could detect 90 % of the possible doors at a distance shorter than 3 m, 80 % of the doors between 3 and 5 m, and 55 % of the doors that are beyond 5 m. An example of this improvement is observed in Fig. 11.

Figure 11a shows the result of our classifier without the implementation of the laser point interpolation. There are two doors undetected. Both of them are beyond 5 m. Figure 11b shows the same situation but in this time including the implementation of the laser point interpolation. Both undetected doors are now correctly detected by our classifier.

The execution time of our classifier using images with a resolution of 1600 × 1200 pixels is 7.45 milliseconds. This means that improving the laser sensor resolution, our classifier will also be able to work with larger images in real-time applications. These timing results will give us the capability of working more than 130 fps in a resolution of 1600 × 1200 pixels. This will allows us to better detect distant doors. The main advantage of using larger images is that distant doors are now bigger in the image and the edge detection is now more accurate. On the other hand, a better laser resolution is required to measure distances in distant doors. Laser interpolation estimates the width of distant doors, but is still not enough for a correct detection. The width of distant doors would be measured more accurately with a higher laser resolution and therefore, errors would be reduced.

Moreover, an improvement and an optimization of the different operations that take place in our algorithm have been achieved. The real-time suitability was imperative for the viability and the implementation of the system. Thus, a study of the execution in different runtime environments was done in order to compare the execution times on different machines with different specifications. In this way, the real-time suitability of our approach is proved not depending on the used runtime environment.

The first runtime environment was an Asus Eee PC 901 with an Intel Atom CPU N270 at 1.60 GHz and 1 GiB RAM running Ubuntu 10.04. The second experimental environment was a desktop computer with an Intel Pentium D CPU at 2.80 GHz with 3 GiB RAM running Ubuntu 10.04.

Table 4 Integral image calculation times

	$3 * w * h$ (ms)	$2 * w * h$ (ms)
Env 1 (640 × 480)	8.38	6.09
Env 1 (1600 × 1200)	52.38	38.03
Env 1 (3200 × 2400)	211.54	154.13
Env 2 (640 × 480)	4.94	0.94
Env 2 (1600 × 1200)	32.04	5.70
Env 2 (3200 × 2400)	124.62	22.21
Env 3 (640 × 480)	1.96	0.51
Env 3 (1600 × 1200)	14.69	5.63
Env 3 (3200 × 2400)	57.50	20.49
Env 4 (640 × 480)	4.09	0.95
Env 4 (1600 × 1200)	27.85	6.20
Env 4 (3200 × 2400)	106.19	24.89

Table 5 Execution times of several related operations

	Prewitt (ms)	Conv (ms)	OpenCV Prewitt (ms)	OpenCV Conv (ms)
Env 1 (640×480)	10.08	24.33	14.73	19.71
Env 1 (1600×1200)	62.33	151.84	89.95	123.01
Env 1 (3200×2400)	250.35	609.99	354.95	486.00
Env 2 (640×480)	3.83	9.8	6.8	6.65
Env 2 (1600×1200)	22.80	50.70	35.94	40.03
Env 2 (3200×2400)	92.45	203.29	132.69	162.95
Env 3 (640×480)	3.46	3.73	3.94	4.31
Env 3 (1600×1200)	23.22	23.62	23.08	26.26
Env 3 (3200×2400)	104.01	95.71	89.36	104.96
Env 4 (640×480)	42.98	8.52	10.52	11.77
Env 4 (1600×1200)	18.32	53.30	59.56	73.35
Env 4 (3200×2400)	71.52	213.25	229.77	293.17

The third environment was another desktop computer with an Intel Core2 Duo CPU E8400 at 3.00 GHz and 3.25 GB RAM running Fedora 16. Finally, the fourth environment was the onboard laptop of our robotic platform itself, which is a Lenovo Thinkpad Edge with an Intel Core i3 CPU U380 1.33 GHz CPU and 2 GB RAM, running Fedora 16.

We calculated separately the execution times of each operation of the algorithm. The highest computational cost occurs in the integral image calculation. Therefore, an implementation of Eq. 9 and an implementation of Eqs. 10 and 11 are compared. The first one takes $3 * w * h$ operations and the second one takes $2 * w * h$ operations. Table 4 shows the results in milliseconds of the execution times of both methods in the four different runtime environments. Besides, the executions were done using three different image sizes: 640×480, $1600 \times 1200 (\times 6.25)$ and $3200 \times 2400 (\times 25)$ pixels to remark the time variation depending on the image size.

Moreover, the execution times of several operations were also compared. These operations were a separable convolution implementation of a 3×3 Prewitt kernel, an implementation of a 3×3 non separable convolution, and the implementations of these two operations using the OpenCV library. Table 5 shows the execution times for the three image sizes previously quoted.

Finally, the execution times of our procedure are shown in Table 6. The first column is the total time and the second column represents the time obtained by subtracting the integral image calculation time from the total time. Thus, the second column is the time of the other needed operations in the procedure. This actually proves that the integral image calculation is the operation with the highest computational cost. Besides, the execution time of the rest of the operations are quite similar despite the increase of the image size.

Many of the proposals found in the literature require a prior training because they are probabilistic solutions, they need to create maps, etc. Examples of such proposals are Anguelov et al. [9], Chen and Birchfield [35], Murillo et al. [3], Hensler et al. [14], Shi et al. [24] and Chen et al. [36]. The execution times of our proposal are clearly better than those shown in previous works as Barber et al. [27] (3 fps), Chen and Birchfield [35] Chen et al. [36] (5 fps), Hensler et al. [14] (12 fps) and Salinas et al. [29] (6 fps).

Table 6 Procedure time execution

	Procedure (ms)	Subtracting integral image time (ms)
Env 1 (640×480)	7.6	1.51
Env 1 (1600×1200)	40.11	2.08
Env 1 (3200×2400)	158.63	4.50
Env 2 (640×480)	1.39	0.45
Env 2 (1600×1200)	6.46	0.76
Env 2 (3200×2400)	23.62	1.41
Env 3 (640×480)	0.72	0.21
Env 3 (1600×1200)	6.40	0.77
Env 3 (3200×2400)	21.58	1.09
Env 4 (640×480)	1.35	0.40
Env 4 (1600×1200)	7.45	1.25
Env 4 (3200×2400)	26.59	1.70

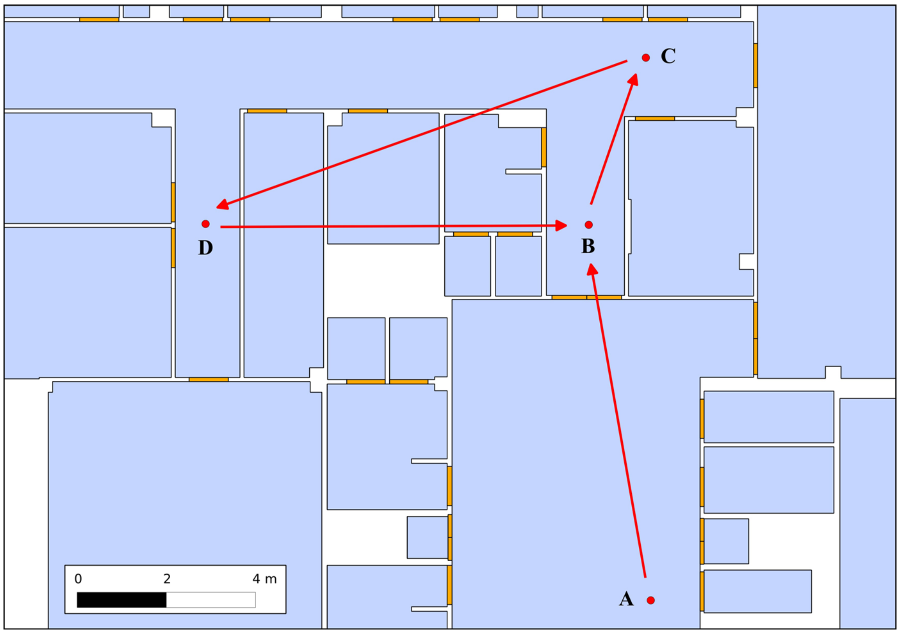


Fig. 12 Path performed by the robot during the navigation mission: $A - [B - C - D]^4$

4.3 Application of Door Detection in a Real-time Robot Navigation

The proposed work has been used as a door detection tool in several navigation missions of MoRLaCo robot in an office environment [5]. A map, where door locations are known, is available by the system during the navigation mission.

Figure 12 shows the path ordered to the robot during the mission. From an initial location A, which is known, the cycle $B - C - D$ is performed four times. Since there is not a direct path from the

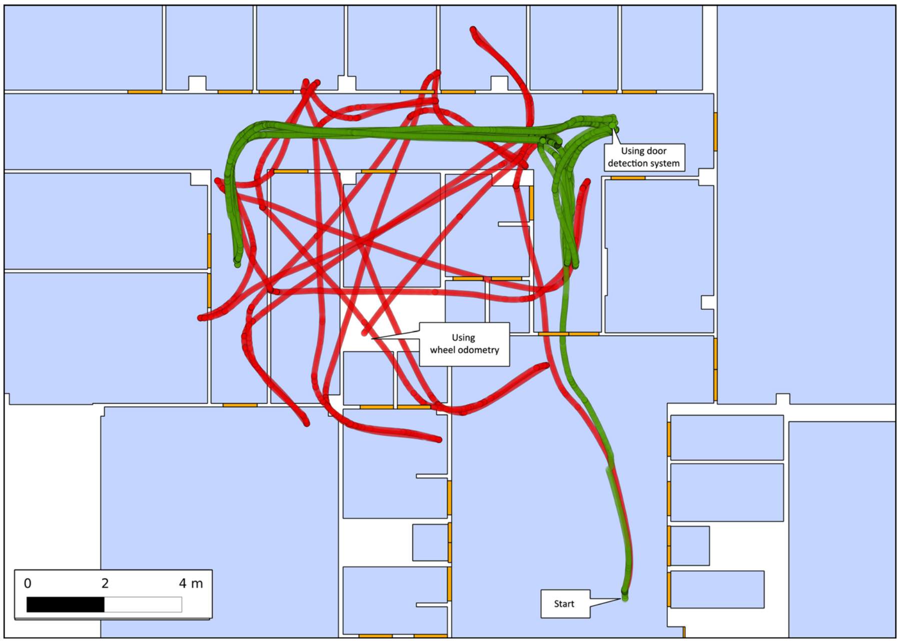


Fig. 13 Result of the navigation mission. *Red*: robot location using only wheel odometry. *Green*: robot location using wheel odometry and the door detection system)

point D to the point B, the robot must retrace its steps through the same corridor where it reached the point D.

Figure 13 shows the result of the navigation. The robot location at every instant of time using only the information from the wheel odometry is shown in red color. As it can be seen, a systematic error has been deliberately added. On the other hand, the robot location at every instant of time using the wheel odometry data and corrected by the door detection system is shown in green color.

It is clearly demonstrated that the robot location calculation using only the wheel odometry data is not accurate. That inaccuracy would provoke that the robot task result impossible to be performed. However, the robot location is accurate at every instant of time when using the door detection system and the map to correct the robot position.

This demonstrates the benefits of using our door detection system in a navigation mission of a domestic robot in an urban building using doors as significant objects if a map is known.

5 Conclusions

A robust procedure to detect doors in buildings is here proposed. The problem has been obviously dealt by many research groups, so a deep review of existing results had to be done. Our proposal consists of a solid combination of tasks organized in an algorithm. Each of them is optimized in order to find an optimal global solution: the region of interest in the image is chosen using laser information, prior calculation of integral image, edge detector proposed. By reducing the execution time of the whole process, our door detection system works at maximum rates of 740 fps using images with a resolution of 640×480 pixels. That will allow the robot to navigate at higher speed without reducing the success rate.

To prove its robustness, several building environments have been considered for a complete testing work where quite different environmental conditions were found: doors with different colors and morphology, different corridors and rooms, different points of view and robot orientations, etc. Therefore, our proposal has a greater generalization compared to existing works where a previous training process was necessary for a particular environment. As a result of these tests, it has been shown that when low and medium distances are considered and a fast human-like walking speed is applied, the classifier provides good quality results with a very high level of confidence. Finally, taking into account the extremely low computing times required, it can be concluded that just off-the-shelf perception and minimal computation resources are needed. Therefore, the real-time capabilities of the procedure are guaranteed.

References

1. AENOR, Spanish Normalization Agency: UNE 56801:2008, Unidad de hueco de puerta de madera. Terminología, definiciones y clasificación. COMITE AEN/CTN 56 - MADERA Y CORCHO (2008)
2. Kosaka, A., Kak, A.C.: Fast vision-guided mobile robot navigation using model-based reasoning and prediction of uncertainties. CVGIP: Image Underst. **56**(3), 271–329 (1992)
3. Murillo, A.C., Kosecká, J., Guerrero, J.J., Sagüés, C.: Visual door detection integrating appearance and shape cues. Robot. Auton. Syst. **56**(6), 512–521 (2008)
4. Fernández-Caramés, C., Moreno, V., Curto, B., Vicente, J.A.: Clustering and line detection in laser range measurements. Robot. Auton. Syst. **58**, 720–726 (2010)
5. Fernández-Caramés, C.: PhD Thesis: Técnicas de navegación para un robot móvil utilizando sistemas de razonamiento espacial. Directed by: V. Moreno, B. Curto, Universidad de Salamanca, Salamanca (2012)
6. Guan, C.-H., Gong, J.-W., Chen, Y.-D., Chen, H.-Y.: An application of data fusion combining laser scanner and vision in real-time driving environment recognition system. In: International Conference on Machine Learning and Cybernetics, vol. 6, pp. 3116–3121 (2009)
7. Madsen, C.B., Andersen, C.S.: Optimal landmark selection for triangulation of robot position. Robot. Auton. Syst. **23**(4), 277–292 (1998)
8. Kim, D., Nevatia, R.: Recognition and localization of generic objects for indoor navigation using functionality. Image Vis. Comput. **16**(11), 729–743 (1998)
9. Anguelov, D., Koller, D., Parker, E., Thrun, S.: Detecting and modeling doors with mobile robots. In: Proceedings of the IEEE International Conference on Robotics and Automation, vol. 4, pp. 3777–3784 (2004)
10. Krotkov, E.: Mobile robot localization using a single image. In: Proceedings of the 1989 IEEE International Conference on Robotics and Automation, vol. 2, pp. 978–983 (1989)
11. Monasterio, I., Lazkano, E., Rañó, I., Sierra, B.: Learning to traverse doors using visual information. Math. Comput. Simul. **60**(3), 347–356 (2002)

12. Brian Burns, J., Hanson, A.R., Riseman, E.M.: Extracting straight lines. IEEE Trans. Pattern Anal. Mach. Intell. **8**(4), 425–455 (1986)
13. Crowley, J.L., Chenavier, F.: Position estimation for a mobile robot using vision and odometry. In: Proceedings of the 1992 IEEE International Conference on Robotics and Automation, vol. 3, pp. 2588–2593 (1992)
14. Hensler, J., Blaich, M., Bittel, O.: Real-time door detection based on adaboost learning algorithm. In: Research and Education in Robotics - EUROBOT 2009. Communications in Computer and Information Science, vol. 82, pp. 61–73 (2010)
15. Gutmann, J.-S., Burgard, W., Fox, D., Konolige, K.: An experimental comparison of localization methods. In: Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 2, pp. 736–743 (1998)
16. Castellanos, J.A., Neira, J., Strauss, O., J.D. Tardós: Detecting high level features for mobile robot localization. In: IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems, 1996, pp. 611–618 (1996)
17. Neira, J., Tardós, J.D., Horn, J., Schmidt, G.: Fusing range and intensity images for mobile robot localization. IEEE Trans. Robot. Autom. **15**(1), 76–84 (1999)
18. Neira, J., Ribeiro, M.I., Tardós, J.D.: Mobile robot localization and map building using monocular vision. In: Proceedings of the 5th International Symposium on Intelligent Robotic Systems, pp. 275–284, Stockholm, Sweden (1997)
19. Tardos, J.D.: Representing partial and uncertain sensorial information using the theory of symmetries. In: Proceedings of the 1992 IEEE International Conference on Robotics and Automation, vol. 2, pp. 1799–1804 (1992)
20. Arras, K.O., Tomatis, N.: Improving robustness and precision in mobile robot localization by using laser range finding and monocular vision. In: Proceedings of the 1999 Third European Workshop on Advanced Mobile Robots, pp. 177–185 (1999)
21. Arras, K.O., Tomatis, N., Jensen, B., Siegwart, R.: Multisensor on-the-fly localization: precision and reliability for applications. Robot. Auton. Syst. **34**(2–3), 131–143 (2001)
22. Kisačanin, B.: Integral image optimizations for embedded vision applications. In: IEEE Southwest Symposium on Image Analysis and Interpretation 2008, pp. 181–184 (2008)
23. Sugihara, K.: Some location problems for robot navigation using a single camera. Comput. Vis. Graph. Image Process. **42**(1), 112–129 (1988)
24. Shi, L., Kodagoda, S., Dissanayake, G.: Laser range data based semantic labeling of places. In: Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5941–5946 (2010)
25. Viola, P., Jones, M.J.: Robust real-time face detection. Int. J. Comput. Vis. **57**(2), 137–154 (2004)
26. Cariñena, P., Regueiro, C.V., Otero, A., Bugarín, A.J., Barro, S.: Landmark detection in mobile robotics using fuzzy temporal rules. IEEE Trans. Fuzzy Syst. **12**(4), 423–435 (2004)
27. Barber, R., Mata, M., Boada, M.J.L., Armingol, J.M., Salichs, M.A.: A perception system based on laser information for mobile robot topologic navigation. In: Proceedings of the IEEE 28th Annual Conference of the Industrial Electronics Society, vol. 4, pp. 2779–2784 (2002)
28. Rusu, R.B., Meeussen, W., Chitta, S., Beetz, M.: Laser-based perception for door and handle identification. In: Proceedings of the International Conference on Advanced Robotics, pp. 1–8 (2009)
29. Muñoz-Salinas, R., Aguirre, E., García-Silvente, M., González, A.: Door-detection using computer vision and fuzzy logic. WSEAS Trans. Syst. **10**(3), 3047–3052 (2004)
30. Atiya, S., Hager, G.D.: Real-time vision-based robot localization. IEEE Trans. Robot. Autom. **9**(6), 785–800 (1993)
31. Thrun, S.: Robotic mapping: a survey. In: Lakemeyer, G., Nebel, B. (eds.) Exploring Artificial Intelligence in the New Millennium, pp. 1–36. Morgan Kaufmann, San Mateo, CA (2002)
32. Yang, X., Tian, Y.: Robust door detection in unfamiliar environments by combining edge and corner features. In: Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, pp. 57–64 (2010)
33. Weiß, G., Wetzler, C., von Putkammer, E.: Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans. In: Proceedings of the International Conference on Intelligent Robots and Systems (1994)
34. Shi, W., Samarabandu, J.: Investigating the performance of corridor and door detection algorithms in different environments. In: International Conference on Information and Automation, pp. 206–211 (2006)
35. Chen, Z., Birchfield, S.T.: Visual detection of lintel-occluded doors from a single image. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, pp. 1–8 (2008)
36. Chen, Z., Li, Y., Birchfield, S.T.: Visual detection of lintel-occluded doors by integrating multiple cues using a data-driven Markov chain Monte Carlo process. Robot. Auton. Syst. **59**(11), 966–976 (2011). ISSN 0921-8890