Pattern Recognition Letters 53 (2015) 118–128

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Hybrid vision-based navigation for mobile robots in mixed indoor/outdoor environments[☆]

Pablo De Cristóforis[a,*], Matias Nitsche[a], Tomáš Krajník[b,c], Taihú Pire[a], Marta Mejail[a]

[a] *Laboratory of Robotics and Embedded Systems, Faculty of Exact and Natural Sciences, University of Buenos Aires, Argentina*
[b] *Lincoln Centre for Autonomous Systems, School of Computer Science, University of Lincoln, United Kingdom*
[c] *Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Republic*

ARTICLE INFO

Article history:
Received 14 March 2014
Available online 29 October 2014

Keywords:
Vision-based navigation
Mixed indoor/outdoor environments
Mobile robotics

ABSTRACT

In this paper we present a vision-based navigation system for mobile robots equipped with a single, off-the-shelf camera in mixed indoor/outdoor environments. A hybrid approach is proposed, based on the teach-and-replay technique, which combines a path-following and a feature-based navigation algorithm. We describe the navigation algorithms and show that both of them correct the robot's lateral displacement from the intended path. After that, we claim that even though neither of the methods explicitly estimates the robot position, the heading corrections themselves keep the robot position error bound. We show that combination of the methods outperforms the pure feature-based approach in terms of localization precision and that this combination reduces map size and simplifies the learning phase. Experiments in mixed indoor/outdoor environments were carried out with a wheeled and a tracked mobile robots in order to demonstrate the validity and the benefits of the hybrid approach.

1. Introduction

Autonomous navigation can be roughly described as the process of moving safely along a path between a starting and a final point. In mobile robotics, different sensors have been used for this purpose, which has led to a varied spectrum of solutions. Active sensors such as sonars [1], laser range finders [2] and radars [3] have been used in autonomous navigation methods. These sensors are inherently suited for the task of obstacle detection and can be used easily because they directly measure the distances from obstacles to the robot.

Other sensors that are becoming a standard part in mobile robotic systems, particularly in field robotics, are visual sensors. Quality cameras have become increasingly affordable, they are small and can provide high resolution data in real time. They are passive and therefore do not interfere with other sensors. Unlike range sensors, they can not only be used to detect obstacles, but also to identify forbidden areas and navigate mobile robots using human-defined rules (i.e. keep off the grass). Such forbidden areas are not obstacles, since they are in the same plane as the path, but should be considered as non-traversable. Moreover, and probably more important, nowadays computational power required by image processing techniques is readily available in consumer hardware. For these reasons, in recent years vision-based navigation for mobile robots has been a widely studied topic.

Many visual navigation methods generally rely on the extraction of salient visual features, using algorithms such as well-known SIFT (scale invariant feature transform) or SURF (speeded up robust features), among others. These are then used as external references from which information about the structure of the surrounding environment and/or the ego-motion of the robot is estimated.

A classical approach to visual navigation is known as teach-and-replay. This technique is closely related to visual servoing [4,5], where the task is to reach a desired pose using vision for control feedback. Several teach-and-replay works employ visual features as landmarks to guide the autonomous navigation. A particularly successful work has been presented by Furgale and Barfoot [6]. This method uses a stereo-camera to build a manifold map of overlapping submaps as the robot is piloted along a route. Each submap represent a metric reconstruction of a portion of the environment. The use of local submaps allows the rover to faithfully repeat long routes without the need for an accurate global environment reconstruction.

In contrast to metric reconstruction of the environment, the authors of Krajník et al. [7], and Chen and Birchfield [8] present a simplistic monocular navigation algorithm that performs autonomous navigation without building metric maps, so-called appearance-based or qualitative navigation. In their approach, the robot steering is calculated directly from horizontal displacement of the currently perceived and previously mapped image features. The article of Krajník et al. [7]

☆ This paper has been recommended for acceptance by G. Sanniti di Baja.
* Corresponding author. Tel.: +54 1145763300.
E-mail address: pdecris@dc.uba.ar (P. De Cristóforis).

http://dx.doi.org/10.1016/j.patrec.2014.10.010
0167-8655/© 2014 Elsevier B.V. All rights reserved.

contains a mathematical proof of the stability of their feature-based navigation.

In recent years, teach-and-replay methods have been studied for both terrestrial and aerial robots. On the one hand, many works use metric reconstruction techniques, such as stereo-based triangulation proposed by Ostafew et al. [9] and Pfrunder et al. [10]. On the other hand, several works still employ solely appearance-based navigation [11–14].

The main drawback of the aforementioned methods lies in the fact that the robot workspace is limited only to the regions mapped during the training step. The user has to guide the robot all around the entire path before performing autonomous navigation, which may represent a very tedious process, especially in large outdoor environments. We argue that the mathematical proof presented in Krajník et al. [7] is not limited only to feature-based map-and-replay navigation methods. This fact should allow to design hybrid navigation systems that use both map-and-replay and map-less reactive navigation methods.

An example of such a method is a real-time monocular path following algorithm that allows a mobile robot navigate through semi-structured or unstructured paths [15]. This method uses a probabilistic approach similar to the proposed by Dahlkamp et al. [16], learning a visual model of the nearby road using an area in front of the vehicle as a reference. The traversable path appearance is modeled as a mixture of Gaussians and the classification between navigable or not navigable area is done of segments rather than pixel level. From the previously determined navigable area, a path center trajectory is estimated and using a motion control law the robot is guided to the center of the detected path and maintain it away from path edges during navigation.

Though different solutions have been proposed to address the indoor or outdoor vision-based navigation, there are not many works that solve the problem when it comes to mixed indoor and outdoor environments. The main contribution of this work is to propose a hybrid navigation method that can work successfully both in indoor and outdoor environments while overcoming the aforementioned drawbacks of existing methods, which are generally oriented towards a specific scenario. The proposed method combines a teach-and-replay feature-based method [7] and a segmentation-based approach for following paths [15]. Both methods have the same principle behind: to correct the heading of the robot to guide the autonomous navigation.

When transitioning to environment contains semi-structured paths, the visual path-following method is used. When there is no path to follow, a map is built using feature-based approach. By recording odometric information for establishing the length of each segment in the map, a large mixed indoor/outdoor environment can be traversed autonomously. We claim that even though neither of the two methods explicitly estimates the robot position in the environment, the heading corrections themselves keep the robot position error bound. In other words, by combining two navigation methods based on the same principle, i.e. heading correction, a hybrid navigation approach which takes the best of each can be obtained, without loosing accuracy in autonomous navigation. In this paper this is proved in both theoretical and practical point of views.

The remainder of this paper is organized as follows: the hybrid navigation approach is presented in Section 2, description of the feature-based method is introduced in Section 3, description of the segmentation-based method for following path is introduced in Section 4, the convergence analysis is performed in Section 5, evaluation criteria and experimental results are discussed in Section 6 and the paper is concluded in Section 7.

2. Hybrid visual navigation

The main idea behind the proposed hybrid navigation method is to combine two separate approaches, feature-based and segmentation-based, by alternating between one and the other depending on the part of the environment the robot is at.

Both of the aforementioned methods correct the robot's heading so it is steered towards the intended path. However, neither of the methods can compute the localization of the robot within the environment. Odometry is only used to determine the relative position of the robot within a segment, and therefore, when the robot has to change between the two navigation methods. One might argue that the cumulative nature of the odometric error would eventually cause the navigation method to fail. However, Krajník et al. [7] indicate that the heading correction itself can compensate for odometric errors and keep the robot position error bound. In this article, we will extend this idea of heading compensation to a broader set of monocular navigation systems based on robot's heading correction. In particular, we will show that stability of bearing-only navigation is not limited to landmark-based localization only, but can be applied to reactive path-following methods as well.

The advantages of the proposed approach compared to the one presented in Krajník et al. [7] can be summarized as follows:

1. It is not necessary to teach the robot the entire trajectory in areas where a detectable path is present. This significantly simplifies and shortens the learning phase.
2. Since the path-following method is adaptive, it is more robust to environment appearance variations compared to the feature-based method. This increases the overall reliability of the navigation.
3. The path following method reduces robot positioning error faster than the feature-based approach. Thus, combination of both methods allows for a more precise navigation.
4. The path following method does not need to store information about the entire edge that is to be traversed. Rather than that, it just need information about the path texture, which dramatically reduces spatial requirements for storage of the topological map especially when traversing long outdoor paths.

2.1. Method overview

The topological map that the method uses is a graph where nodes represent places in the environment and edges represent the navigation methods that can be used to move between these places. While the nodes are associated with azimuths indicating directions of the outgoing edges, the edges contain information that allows the robot to traverse between the individual places. In other words, an edge of the hybrid topological map represents an action that the robot should take during its length in different parts of the route: i.e. either the path-following or the feature-based navigation.

The environment map is created during a training phase where the user builds a topological map by driving a semi-autonomously moving robot through the environment. During the training, the robot either moves forwards while creating an image-feature based map or it follows a pathway of the environment. Whenever the user terminates the current behavior, the robot records the length of the edge and creates a topological node. Then, the user can manually set the robot orientation and start path following or feature mapping again. It is important to note that if the length of the path-following edge is known (e.g. from an overhead map), there is no need to traverse the edge during the learning phase (see Fig. 1).

In the autonomous navigation or "replay" phase, the robot traverses a sequence of edges by either one of the two movement primitives. Note that neither or the methods performs localization of the robot. Both methods simply keep the robot close to the trained paths while using odometric reading to decide whether the particular edge has been traversed or not. In Section 5 we will show that despite of simplicity of the approach, the robot position error does not diverge over time.

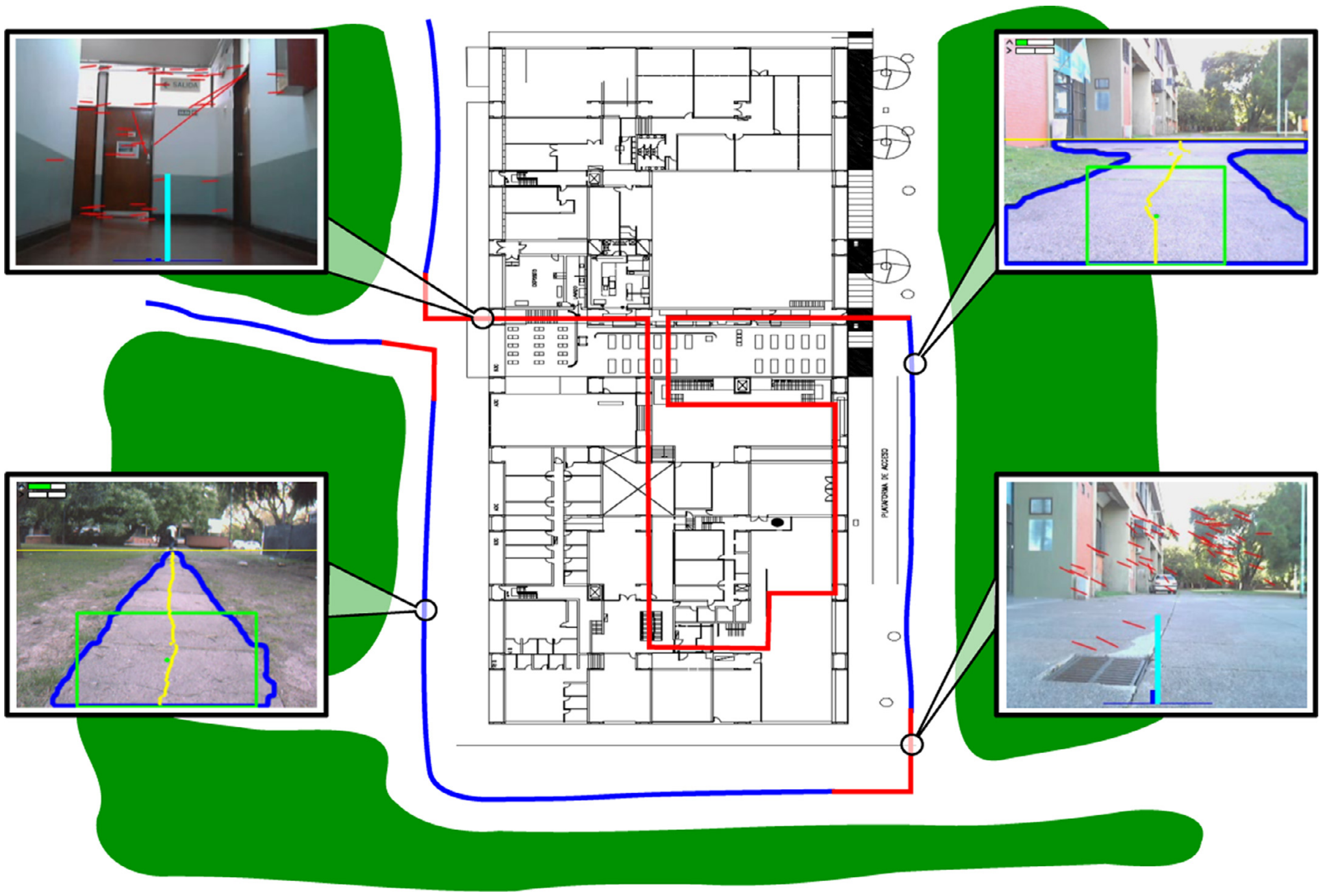


Fig. 1. Schematic of an example map of the hybrid method for a given indoor/outdoor scenario. Lines marked in blue consist of the portions of the map which are traversed using the path-following method, while lines marked in red, portions which are traversed using the feature-based approach. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

In the next sections we will describe each navigation movement primitive: feature-based for open areas (indoor or outdoor) where there is not bounded path to go through, or cross intersection of two or more paths, and image segmentation based method conceived to follow structured or semi-structured paths.

3. Feature-based visual navigation

The feature-based navigation method used as a motion primitive of the proposed approach is based on Krajník et al. [7]. This method uses the map-and-replay technique which is described in this section. This method allows a robot to follow a given path by performing an initial learning phase where features extracted from the camera images are stored. In a second replay phase, the robot autonomously navigates through the learned path while correcting its heading by comparing currently seen and previously stored features.

3.1. Image features

Since image features are considered as visual landmarks, the feature detector and descriptor algorithms are a critical component of the navigation method. Image features must provide enough information to steer the robot in a correct direction. Furthermore, they should be robust to real world conditions, i.e. changing illumination, scale, viewpoint and partial occlusions and of course its performance must allow real-time operation. In the original version of the method, Krajník et al. used SURF algorithm proposed by Bay et al. [17] to identify visual landmarks in the image. The SURF method is reported to perform better than most SIFT [18] implementations in terms of speed and robustness to viewpoint and illumination change.

For this work, further evaluation of a variety of extractor and descriptor algorithms, in the context of visual navigation, was performed. The goal of these off-line experiments was to determine which algorithm combination is the best choice for visual navigation in terms of performance, robustness and repeatability [19]. As a result of this evaluation, we conclude that the combination of STAR based on CenSurE (Center Surround Extremas) algorithm [20] to detect image features and BRIEF (binary robust independent elementary features) algorithm [21] to build the descriptor of the feature outperforms the other image feature extraction algorithms. The STAR extractor gives better results to detect landmarks than SURF, because keypoints extracted with STAR are more salient and thus, also more stable than keypoints extracted with SURF. The BRIEF descriptor uses a simpler coding scheme to describe a feature, this reduces storage requirements and increases matching speed. The SURF use 64 (or 128 in the extended version) double-precision floating-point values to describe a keypoint, this results in 512 bytes (or 1024) of data for each descriptor. In contrast, BRIEF uses a string of bits that can be packed in 32 bytes. Taking into account the high number of features present in the map, this can amount to hundreds of megabytes which are saved in large-scale maps. Furthermore, in terms of speed, instead of using the Euclidean distance to compare descriptors as it is done with SURF, BRIEF descriptors can be compared using Hamming distance, which can be implemented extremely fast using SSE instructions.

3.2. Learning phase

In the initial learning phase, the robot is manually guided through the environment while the map is built. The operator can either let the robot go forward or stop it and turn it in a desired direction. A map can therefore be built, which will have the shape of a series of segments of certain length as computed by the robots odometry. During forward movement, the robot tracks features extracted from the robot's camera images. For each feature, its position in the image is saved along its descriptor. Also, initial and final distances (as measured by the robot's odometry) along the segment where the feature was first and last seen are also saved. Tracking is performed by extracting features for the present frame and comparing these to others that were continuously seen in previous frames. While features remain in the set of tracked landmarks, their information can be updated. When features are not seen anymore, they are added to the map and removed from this set. Matching is performed by comparing descriptors.

The procedure which creates a map of one segment is as follows in the listing 1. Each landmark has an associated descriptor l_{desc}, pixel position l_{pos_0}, l_{pos_1} and robot relative distance l_{d_0}, l_{d_1}, for when the landmark was first and last seen, respectively.

Algorithm 1: Learning Phase

```
Input: $\varphi$: initial robot orientation
Output: $L$: landmarks learned for the current segment, $s$:
        segment length, $\varphi_0$: segment orientation
$\varphi_0 \leftarrow \varphi$ read robot orientation from gyro or compass
$L \leftarrow \emptyset$            /* clear the learned landmark set */

$T \leftarrow \emptyset$            /* clear the tracked landmark set */

repeat
    $F$ ←features extracted from the current camera image
    foreach $l \in T$ do
        $f \leftarrow$ find_match($l$,$F$)
        if no match then
            $T \leftarrow T - \{l\}$                /* stop tracking */
            $L \leftarrow L \cup \{l\}$             /* add to segment */
        else
            $F \leftarrow F - \{f\}$
            $l \leftarrow (l_{desc}, l_{pos_0}, f_{pos}, l_{d_0}, d)$ /* update image
               coordinates & robot position          */
    foreach $f \in F$ do
        $T \leftarrow T \cup \{(f_{desc}, f_{pos_0}, f_{pos_0}, d, d)\}$        /* start
        tracking new landmark */
until operator terminates learning mode
$L \leftarrow L \cup T$                     /* save tracked landmarks */
$s \leftarrow d$                  /* remember the segment's length */
```

The map obtained by this approach is thus composed of a series of straight segments, each described by its length s, azimuth ϕ_0 and a set of detected landmarks L. A landmark $l \in L$ is described by a tuple $(\mathbf{e}, k, \mathbf{u}, \mathbf{v}, f, g)$, where $\mathbf{e}$ is its BRIEF descriptor and k indicates the number of images, in which the feature was detected. Vectors $\mathbf{u}$ and $\mathbf{v}$ denote positions of the feature in the captured image at the moment of its first and last detection and f and g are distances of the robot from the segment start at these moments.

3.3. Autonomous navigation phase

In order to navigate the environment using this method, the robot is initially placed near the starting position of a known segment. Then, the robot moves forward at constant speed while correcting its heading. This is performed by retrieving the set of relevant landmarks from the map at each moment and to estimate their position in the current camera image. These landmarks are paired with the currently detected ones and the differences between the estimated and real positions are calculated. As these differences in image coordinates are related to a displacement of the robot in the world, it is possible to actively minimize these by moving the robot accordingly, and thus the previously learnt path is replayed.

Algorithm 2: Navigation Phase

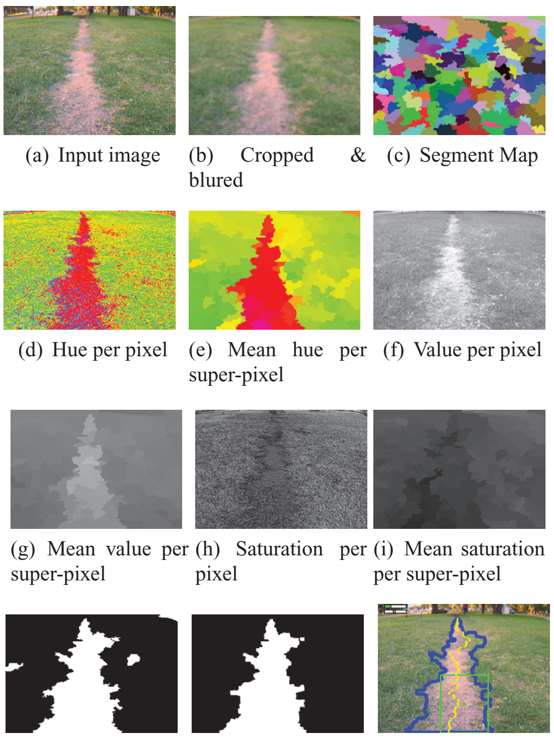
```
Input: $L$: landmarks of the current segment $s$: current
       segment length
Output: $\omega$: angular robot speed $v$: forward robot speed
$v \leftarrow v_0$                    /* start moving forwards */
while $d < s$ do
    $H \leftarrow \emptyset$            /* pixel-position differences */
    $T \leftarrow \emptyset$                    /* tracked landmarks */
    foreach $l \in L$ do
        if $l_{d_0} < d < l_{d_1}$ then
            $T \leftarrow T \cup \{l\}$                /* get expected
            landmarks according to $d$ */
    $F$ ←features extracted from the current camera image
    while $T$ not empty do
        $f \leftarrow$ find_match($l$,$F$)
        if matched then
            /* compare feature position to
               estimated current landmark
               position by interpolation                */
            $h \leftarrow f_{pos} - \left((l_{pos_1} - l_{pos_0})\frac{d - l_{d_0}}{l_{d_1} - l_{d_0}} + l_{pos_0}\right)$
            $H \leftarrow H \cup \{h\}$
        $T \leftarrow T - \{l\}$
    $\omega \leftarrow \gamma$ mode($H$)
$v, \omega \leftarrow 0$           /* segment completed, stop moving */
```

More formally, the replay phase is as follows. Each time a new image is captured, the robot updates the estimated traveled distance d from the start of the segment and builds the set T by retrieving landmarks in L that were visible at distance d during learning. Then, features are extracted from the current camera image and put in the set F. A pairing between the sets F and T is established as in the learning phase. After establishing these correspondences, for each pair of matching features their image position is compared. The expected image position of landmarks in T is obtained by linearly interpolating according to d, since only initial and last image positions of landmarks are stored during mapping to reduce map size. By taking the horizontal difference in image positions a heading correction for the robot can be established. To obtain a single correction value, the most likely horizontal position needs to be obtained since the matching process may produce outliers. Thus, this single deviation value is obtained by sampling all horizontal differences in a histogram and taking its mode.

While the robot moves forward at constant speed, this heading correction is performed for each frame until d equals the segment length stored in the map. The listing 2 presents the algorithm used to traverse or 'replay' one segment.

4. Segmentation-based path-following

The path-following method used as motion primitive of the proposed approach is based on a previous work by De Cristóforis et al. [15]. Here, a mobile robot is autonomously steered in order to remain inside a semi-structured path by means of image processing alone.


(a) Input image (b) Cropped & blured (c) Segment Map

(d) Hue per pixel (e) Mean hue per super-pixel (f) Value per pixel

(g) Mean value per super-pixel (h) Saturation per pixel (i) Mean saturation per super-pixel

(j) Super-pixel classification as binary mask (k) Filtered binary mask (l) Path contour

Fig. 2. Pipeline description: (a) input image as acquired from the camera, (b) cropped image below automatically detected horizon, (c) segment map. For each segment, mean and covariance are computed: (e), (i) and (g) show segment hue, saturation and value means respectively. (d), (h) and (f) show pixel hue, saturation and value for reference. (j) binary mask obtained from classification using ground models from ROI. (k) Morphological opening filter is applied in order to 'close' small gaps in the binary mask. (l) Path contour extracted from processed binary mask and middle points computed, on top-left linear and angular speeds values obtained by control law.

The method works by first dividing the image into regions or segments of similar appearance and then classifying these as belonging to traversable or non-traversable areas. It does not require a learning phase since it uses the area immediately in front of the robot in order to obtain a sample of traversable terrain. A Gaussian-mixture model is used to describe this sample (using the Hue, Saturation and Value components) and compare to the rest of the regions in the image. After classification, the most likely group of interconnected image segments is used to obtain a contour of traversable path. With this contour a simple control law which infers the path middle line and steers the robot to follow it is used.

Since the method only requires the area of the image below the horizon level (i.e. the ground) to steer the robot, an initial horizon detection method can be used to infer this area and restrict processing to the meaningful portion of the image. Fig. 2 shows the image processing pipeline.

4.1. Image segmentation

In outdoor semi-structured or unstructured environments, the traversable area is often cluttered by small objects with a color of the forbidden area, for example grass, tree leaves, water or even snow in the middle of the road. In this case, most classification methods working at a pixel level, like the one proposed by Dahlkamp et al. [16], would perform worse than methods which first segment the image to several areas with similar texture or color. Thus, the initial step in the path-detection process is to segment the image into a set of super-pixels. After the image is cropped to contain the region below the horizon (either automatically or manually), the result is initially filtered to reduce noise (using a median-blur, which preserves edge contrast) and then segmented. The segmentation algorithm used is the graph-based approach proposed by Felzenszwalb and Huttenlocher [22]. The algorithm first constructs a fully connected graph where each node corresponds to a pixel in the image. Pixel intensities between neighbors are analyzed and edges are broken whenever a threshold is exceeded. The resulting unconnected sub-graphs define the segments (or super-pixels).

4.2. Classification

To achieve a robust classification of the segments, a probabilistic approach is used, based on Dahlkamp et al. [16]. This step determines if a population sample (a segment), modeled by a Gaussian probability distribution $\mathcal{N}(\mu, \Sigma)$, with mean vector μ and covariance matrix Σ, represents an instance of a more general model of the 'navigable path' class or not. This navigable path class is represented in turn by a Mixture-of-Gaussians (MoG) model. The steps involved in the classification task are as follows:

1. Segment model computation: each segment is modeled by its mean μ and covariance Σ of HSV color values.
2. Navigable class computation: a rectangular ROI (region of interest) in the lower part of the image is used as a reference of navigable path (see Fig. 2). All models contained in this ROI are merged by similarity (using Mahalanobis distance). The resulting models which cover the ROI area by more than a defined percentage are used as references for navigable regions in the image.
3. Classification of all segments: All segments in the image are compared to the reference models for navigable path using the Mahalanobis distance. A binary mask for all pixels is created indicating the membership to this class.
4. Contour extraction: the binary mask is post-processed by a morphological opening operation to reduce artifacts and from all regions classified as navigable path, the area intersecting the ROI region is chosen. The contour of this region is computed.

4.3. Motion control

The goal of the motion control is to correct the heading of the robot to keep it in the middle of the road. From the previously determined contour, a path center trajectory is estimated in order to guide the robot to the center of the detected path and maintain it away from path edges. First, by going row-by-row in the image, the middle point for the current row is obtained from the leftmost and rightmost pixel's horizontal positions of the path contour. The list of middle points is then used to compute angular and linear speeds with a simple yet effective control-law as follows. From the list of n horizontal values u_i of the ith middle point of the detected path region, angular speed ω and forward speed v are computed as follows:

$$\omega = \alpha \sum_{i}^{n} \left(x_i - \frac{w}{2} \right), \qquad v = \beta n - |\omega| \tag{1}$$

where w is the width of the image and α and β are constants specific to the robot's speed controllers.

The effects of this control law are such that the robot will turn in the direction where there is the highest deviation, in average, from middle points with respect to a vertical line passing through the image center. This line can be assumed to be the position of the camera, which is mounted centered on the robot. Therefore, whenever a turn in the path or an obstacle is present, the robot will turn to remain inside the path and avoid the obstacle. The linear speed of the robot is reduced accordingly to the angular speed determined by the previous computation. This has the effect of slowing down the robot whenever it has to take a sharp turn.

5. Robot position error analysis

In this section we show that despite the fact that the robot localization is based on odometry, the robot position error does not diverge over time due to the heading corrections caused by the two movement primitives.

Our analysis assumes that a mobile robot moves through a sequence of n waypoints by using reactive navigation algorithms (or motion primitives) that steer the robot towards the waypoint's connecting lines. We describe robot position at each waypoint in the waypoint's coordinate frame and show that the robot's relative position to a particular waypoint can be calculated by a recursive formula $\mathbf{x_{k+1}} = M_k x_k$, where the matrix $\mathbf{M_k}$ eigenvalues are 1 and m_k. First, we show that m_k is lower than 1 because of the heading corrections introduced by the motion primitive used to navigate between kth and $(k+1)$th waypoint. Finally, we show that if all the path waypoints do not lie on a straight line, $\mathbf{M} = \prod k = 0^{n-1} M_k$ dominant eigenvalue is lower than one, meaning that the robot position error is reduced as it travels the entire path.

5.1. Robot movement model

Let us assume that the robot moves on a plane and its position and heading is x, y, φ respectively. Assume that the edge to be traversed starts at coordinate origin and leads in the direction of the x-axis.

The path following method presented in Section 4.1 calculates the robot's steering speed ω directly from image coordinates of the segmented path center. These coordinates are affected by both robot heading ϕ and lateral displacement y. Larger robot displacement and larger heading deviation cause larger values of the steering speed, i.e.

$$\omega \sim -c_0 y - c_1 \varphi. \tag{2}$$

Given that the robot's speed v is much lower compared to the maximal rotation speed ω, the robot can change its heading much faster than its horizontal displacement y. Thus, a robot steered by the method presented in Section 4 will quickly stabilize its heading and reach a state when its rotation speed ω will be close to zero. This allows to rewrite Eq. (2) to

$$0 \approx -c_0 y - c_1 \varphi. \tag{3}$$

Thus, heading φ of a robot following the path can be expressed as

$$\varphi \approx -k_0 y, \tag{4}$$

where k_0 is a positive constant.

The case of the feature-based method is similar. If the robot is displaced in a lateral direction from the path, i.e. in the direction of the y-axis, the perceived features will not be visible at the expected positions, but will be shifted to the right for $y > 0$ and to the left $y < 0$.

Since the feature-based method steers the robot in a direction that minimizes the difference (in horizontal coordinates) of the expected and detected features, the robot heading will be stabilized at a value that is proportional to its lateral displacement. In other words, the robot heading φ satisfies a similar constrain as in the previous case:

$$\varphi \approx -k_1 y, \tag{5}$$

where k_1 is a positive constant.

Thus, both of the motion primitives of the hybrid method adjust the robot heading φ proportionally to its lateral displacement from the path, i.e. $\varphi \approx -ky$, where $k > 0$. Both methods move the robot along the edge until the robots odometric counter reports that the distance travelled exceeded the length of the edge stored in the map. Assuming that the robot heading φ is small, we can state that

$$\frac{dy}{dx} = \varphi = -ky. \tag{6}$$

Let the initial position of the robot be (a_x, a_y), where the values of (a_x, a_y) are significantly smaller than the segment length. Solving this differential equation with boundary conditions $a_y = f(a_x)$ allows us to compute the robot position:

$$y = a_y e^{-kx}. \tag{7}$$

Considering a segment of length s, we can calculate the robot position (b_x, b_y) after it traverses the entire segment as:

$$\begin{aligned} b_x &= a_x + s \\ b_y &= a_y e^{-ks}. \end{aligned} \tag{8}$$

Eq. (8) would hold for an error-less odometry and noiseless camera. Considering the camera and odometry noise, Eq. (8) will be rewritten to

$$\begin{pmatrix} b_x \\ b_y \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & e^{-sk} \end{pmatrix} \begin{pmatrix} a_x \\ a_y \end{pmatrix} + \begin{pmatrix} s\upsilon \\ \xi \end{pmatrix}, \tag{9}$$

where υ and ξ are normally distributed random variables, $\mu(\upsilon) = 1$, $\sigma(\upsilon) = \epsilon$, $\mu(\xi) = 0$ and $\sigma(\xi) = \tau$. A compact form of (9) is

$$\mathbf{b} = \mathbf{Ma} + \mathbf{s}. \tag{10}$$

For a segment with an arbitrary azimuth, one can rotate the coordinate system by the rotation matrix $\mathbf{R}$, apply (10) and rotate the result back by $\mathbf{R^T}$. Thus, Eq. (10) would become

$$\mathbf{b} = \mathbf{R^T}\left(\mathbf{MRa} + \mathbf{s}\right) = \mathbf{R^T MRa} + \mathbf{R^T s}. \tag{11}$$

Using (11), the robot position $\mathbf{b}$ at the end of the segment can be computed from its starting position $\mathbf{a}$. Eq. (11) allows to calculate robot position as it traverses along the intended path.

5.2. Position error

Let the robot initial position $\mathbf{a}$ be a random variable drawn from a two-dimensional normal distribution with the mean $\hat{\mathbf{a}}$. Since Eq. (11) has only linear and absolute terms, the final robot position $\mathbf{b}$ will have a normal distribution as well. Let $\mathbf{a} = \hat{\mathbf{a}}+\tilde{\mathbf{a}}$, where $\tilde{\mathbf{a}}$ is a random normal variable with zero mean. Assuming the same notation for $\mathbf{b}$ and $\mathbf{s}$, Eq. (11) can be rewritten as

$$\tilde{\mathbf{b}} = \mathbf{R^T MR}(\hat{\mathbf{a}} + \tilde{\mathbf{a}}) + \mathbf{R^T}(\hat{\mathbf{s}} + \tilde{\mathbf{s}}) - \hat{\mathbf{b}}. \tag{12}$$

Substituting $\mathbf{R^T MR}\hat{\mathbf{a}} + \mathbf{R^T}\hat{\mathbf{s}}$ for $\hat{\mathbf{b}}$, Eq. (12) becomes

$$\tilde{\mathbf{b}} = \mathbf{R^T MR}\tilde{\mathbf{a}} + \mathbf{R^T}\tilde{\mathbf{s}}, \tag{13}$$

where $\tilde{\mathbf{a}}, \tilde{\mathbf{b}}, \tilde{\mathbf{s}}$ are Gaussian random variables with zero mean. The $\tilde{\mathbf{a}}$ and $\tilde{\mathbf{b}}$ represent the robot position error at the start and end of the traversed segment.

5.3. Traversing multiple segments

Let the robot path is a closed chain of n linear segments denoted by numbers from 0 to $n-1$. Let a segment i be oriented in the direction α_i and its length be s_i. Let the robot positions at the start and end of the ith segment are $\mathbf{a_i}$ and $\mathbf{b_i}$ respectively. Since the segments are joined, $\mathbf{b_i} = \mathbf{a_{i+1}}$ and Eq. (13) for the ith traveled segment is

$$\tilde{\mathbf{a}}_{\mathbf{i+1}} = \mathbf{N_i}\tilde{\mathbf{a}}_{\mathbf{i}} + \mathbf{R_i^T}\tilde{\mathbf{s}}_{\mathbf{i}}, \tag{14}$$

where $N_i = R_i^T M_i R_i$. Thus, the robot position after traversing the entire path consisting of the n segments will be

$$\tilde{\mathbf{a}}_{\mathbf{n}} = \check{\mathbf{N}}\tilde{\mathbf{a}}_{\mathbf{0}} + \check{\mathbf{t}}, \tag{15}$$

where

$$\mathbf{\breve{N}} = \prod_{j=n-1}^{0} \mathbf{N_j} \text{ and } \mathbf{\breve{t}} = \sum_{\mathbf{i=0}}^{\mathbf{n-1}} \left(\prod_{\mathbf{j=i+1}}^{\mathbf{n-1}} \mathbf{N_j} \right) \mathbf{R_i^T \tilde{s}_i}. \tag{16}$$

If the robot traverses the entire path k-times, its position can be calculated in a recursive way by

$$\mathbf{\tilde{a}_{(k+1)n}} = \mathbf{\breve{N}\tilde{a}_{kn}} + \mathbf{\breve{t}}. \tag{17}$$

Since the mean of every $\tilde{s}_i$ is equal to zero and $\tilde{s}_i$ have normal distribution, $\breve{t}$, that is a linear combination of $\tilde{s}_i$, has a normal distribution with zero mean as well. Therefore, Eq. (17) describes a linear discrete stochastic system, where the vector $\breve{t}$ represents a disturbance vector. If the system characterized by (17) is stable, then the position deviation of the robot from the intended path $\tilde{a}_i$ does not diverge for $k \to +\infty$. The system is stable if all eigenvalues of $\mathbf{\breve{N}}$ lie within a unit circle.

As every $\mathbf{N_i}$ equals to $\mathbf{R_i^T M_i R_i}$, its eigenvalues lie on the diagonal of $\mathbf{M_i}$ and its eigenvectors constitute columns of $\mathbf{R_i}$. Therefore, each matrix $\mathbf{N_i}$ is positive-definite and symmetric. Since the dominant eigenvalue of every $\mathbf{N_i}$ equals to one, eigenvalues of $\mathbf{\breve{N}}$ are smaller or equal to one. Since the dominant eigenvalue of $\mathbf{\breve{N}}$ is equal to one if and only if the dominant eigenvalues of products $\mathbf{N_{i+1}N_i}$ are equal 1 for all i. However, dominant eigenvalue of a product $\mathbf{N_{i+1}N_i}$ equals 1 only if the dominant eigenvectors of both $\mathbf{N_i}$ and $\mathbf{N_{i+1}}$ are linearly dependent, which corresponds to collinearity of ith and $(i+1)$th segment. Thus, a dominant eigenvalue $\breve{n}$ of the matrix $\mathbf{\breve{N}}$ equals 1 if and only if all path segments are collinear, i.e. the entire path is just a straight line. For any other case, the spectral radius of $\mathbf{\breve{N}}$ is lower than one, meaning that the system described by (17) is stable. This means that if the robot travels the trajectory repeatably, its position error $\mathbf{\tilde{a}_{kn}}$ does not diverge.

6. Experimental results

The proposed hybrid vision-based navigation system was implemented in C/C++ within the ROS (Robot Operating System) framework as a set of separate ROS modules[1]. The performance of the navigation system has been evaluated in an indoor/outdoor environment. The experiments were performed outside and inside of the Pabellón 1 building, Ciudad Universitaria, Buenos Aires, Argentina. Two sets of experiments were performed with two different robots. Two robotic platforms were used for experiments: a small differential caterpillar-tracked robot called ExaBot [23] and a four-wheeled P3-AT from MobileRobots. Both platforms were equipped with a single off-the-shelf camera and a Core i5 laptop as the sensing and processing elements, respectively.

Additionally, the motion model correctness was also demonstrated in practice, by teaching and repeating a single straight segment in an outdoor scenario, using both methods. Finally, the computational efficiency of the method was measured.

6.1. Motion model correctness

The first scenario was aimed at verification of the assumptions given in Section 5.1. In particular, we wanted to calculate if the robot motion model established by Eq. (7) conforms to real robot motion along one straight segment. To verify the motion model, we have taught the robot to traverse a 5.6 m long straight path. Then, we executed the autonomous navigation system in order to repeat the path 5 times, with an initial lateral displacement of about 0.5 m from the original starting point. The robot motion was tracked using an external localization system [24]. We used both segmentation- and feature-based methods for this test. The recovered trajectories, compared with an expected exponential shape given by (7), are presented in Fig. 3, where the stability of the system is appreciated. The trajectories described by both methods correspond to the exponential model presented in Eq. (7).

As can be seen, both navigation methods converge. However, it can be easily noted that in this outdoor scenario, the segmentation-based path following method converges much faster and with less error. Since the scene is comprised mostly of distant elements, the feature-based method aligns its view and converges much slower. On the other hand, on indoor scenarios, the convergence of the feature-based method is much faster. Thus, it makes much more sense to use a faster converging method such as path-following when the robot has to navigate outdoors through a structured or semi-structured path.

6.2. Algorithm computational efficiency

During the aforementioned experiment, we have also measured the real-time performance of both navigation algorithms. Typically, one cycle of the feature-based navigation algorithm takes approximately 30 ms, out of which one half is spent with the image features extraction and the other with the frame-to-map matching. Segmentation of one image into superpixels typically takes 150 ms of a total of 215 ms that takes the whole algorithm. Thus, the feature- and segmentation-based algorithms issue about 30 and 4 steering commands per second respectively, which allows to quickly stabilize the robot at the heading desired.

6.3. Large-scale experiments

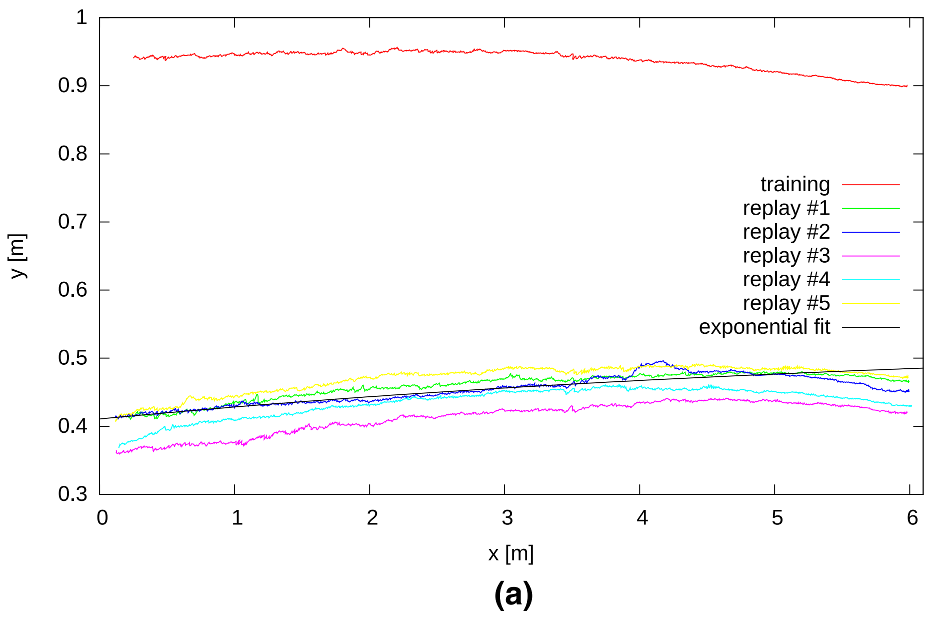
The purpose of the large-scale experiments was to test the robot in realistic outdoor conditions. However, it was not technically possible to cover the entire operation area by the localization system. Therefore, the algorithm precision has been assessed in the same way as Chen and Birchfield [8] and Krajník et al. [7], i.e. the relative robot position has been measured each time the robot traversed an entire path.

In both cases, a closed path was first taught during a tele-operated run, traversing an open area around the entrance of the building and a square path inside the building hall. On the outdoor portion, when a path was available, the path-detection method was used. Otherwise, the feature-based navigation method was selected. During this training phase, a hybrid map was built, which was later used during autonomous navigation. For the P3-AT, the outdoor portion was considerably longer, of a total of 150 m (see Fig. 4), while for the ExaBot, the path length was of 68 m. Two sets of experiments were performed with the P3-AT on different days using different mappings of the environment.

In order to test the robustness of the approach, a series of autonomous replays of the previously taught path were performed. While this type of navigation has been theoretically proved to reduce odometric errors by means of processing visual information, during the experiments the aim was to test this aspect in practice. To this end, the position of the robot at the end of each replay was measured and compared to the ending position. Furthermore, in order to better appreciate the previous error reduction effect, during the experiments performed with the ExaBot, the robot was laterally displaced by 0.6 m before the first repetition while still remaining inside the bounds of the path. This was possible due to the reduced size of the ExaBot. On the other hand, due to its larger size, the P3-AT was only displaced about 0.3 m. However, given such a long path, the expected errors of odometry alone (generally around of 10%) would be higher than this amount and what is of importance is that after several repetitions the robot always reaches the expected position without significant error.

For the first set of experiments with the ExaBot, the positions differences were measured by marking the final robot positions on the

[1] We will plan to release an open version of the software if the paper is accepted.



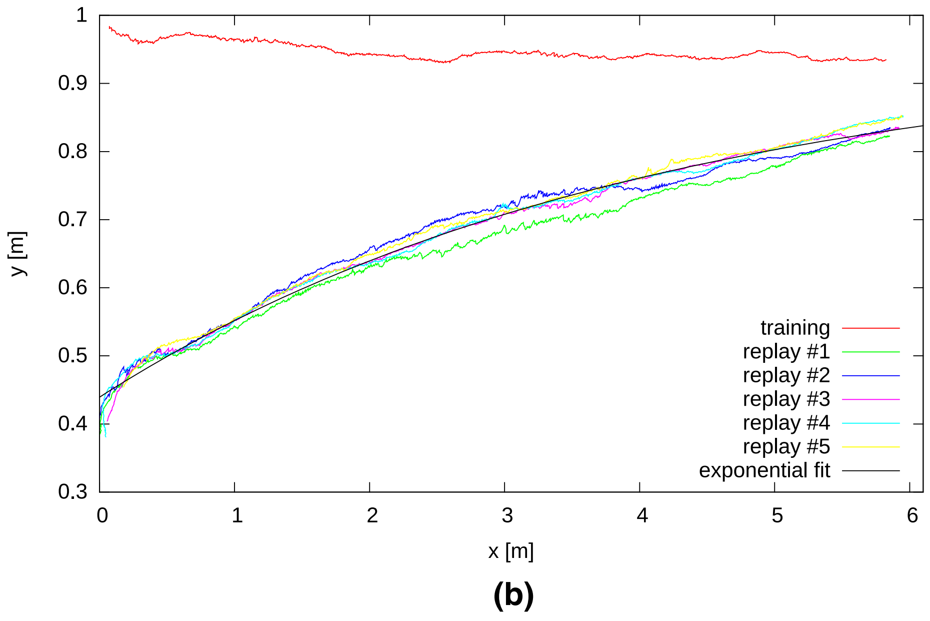


Fig. 3. Robot trajectory along a straight segment path. (a) shows the results using the feature-based method and (b) using the path following method. Five replays were performed for each case. The data points of the recorded trajectories are fitted using the equation $\mathbf{y} = \mathbf{a_y} e^{-\mathbf{kx}}$ to verify the motion model used in the navigation method.

floor and measuring by hand. In contrast, for the second set of experiments performed with the P3-AT, an external visual localization system, WhyCon [24], was used as ground-truth data of the robot position. The localization system allows obtaining the pose of the robot within its 2D plane of motion, with respect to a user set coordinate system, by detecting a circular pattern attached to the robot by means of an external fixed camera. This localization system has been shown to have around 1% relative error (in relation to the measurement area).

The results for both sets can be seen in Table 1, corresponding to five repeats for the case of the P3-AT and three for the ExaBot (due to limited autonomy). As can be seen, the second set of repeats with the P3-AT (second column) gave better results than the first one (first column). This may be because in the first case the learning phase was made in one day and the autonomous navigation phase in another, while the weather changed from cloudy/light rain to partly cloudy/sunny, affecting the learned map of the environment. In the second set of repeats with the P3-AT the whole experiment (both learning and replay phase) was done in the same day. In total, using the presented hybrid method, the P3-AT robot was able to autonomously navigate 1.5 km. Example images corresponding to the robot views during navigation can be seen in Fig. 5.

Table 1
Position errors of the indoor/outdoor experiment with the hybrid method, for each robot. Errors were measured after each repetition with respect to the ending positions reached after training.

	P3-AT				ExaBot	
Replay	**Abs. err. (m)**		**Rel. err.** (%)		**Abs. err. (m)**	**Rel. err. (%)**
1	0.40	0.09	0.27	0.06	0.12	0.12
2	0.58	0.19	0.39	0.12	0.07	0.10
3	0.90	0.20	0.60	0.13	0.03	0.05
4	0.86	0.21	0.57	0.13	–	–
5	0.80	0.25	0.53	0.16	–	–
Avg.	0.71	0.19	0.47	0.12	0.07	0.09

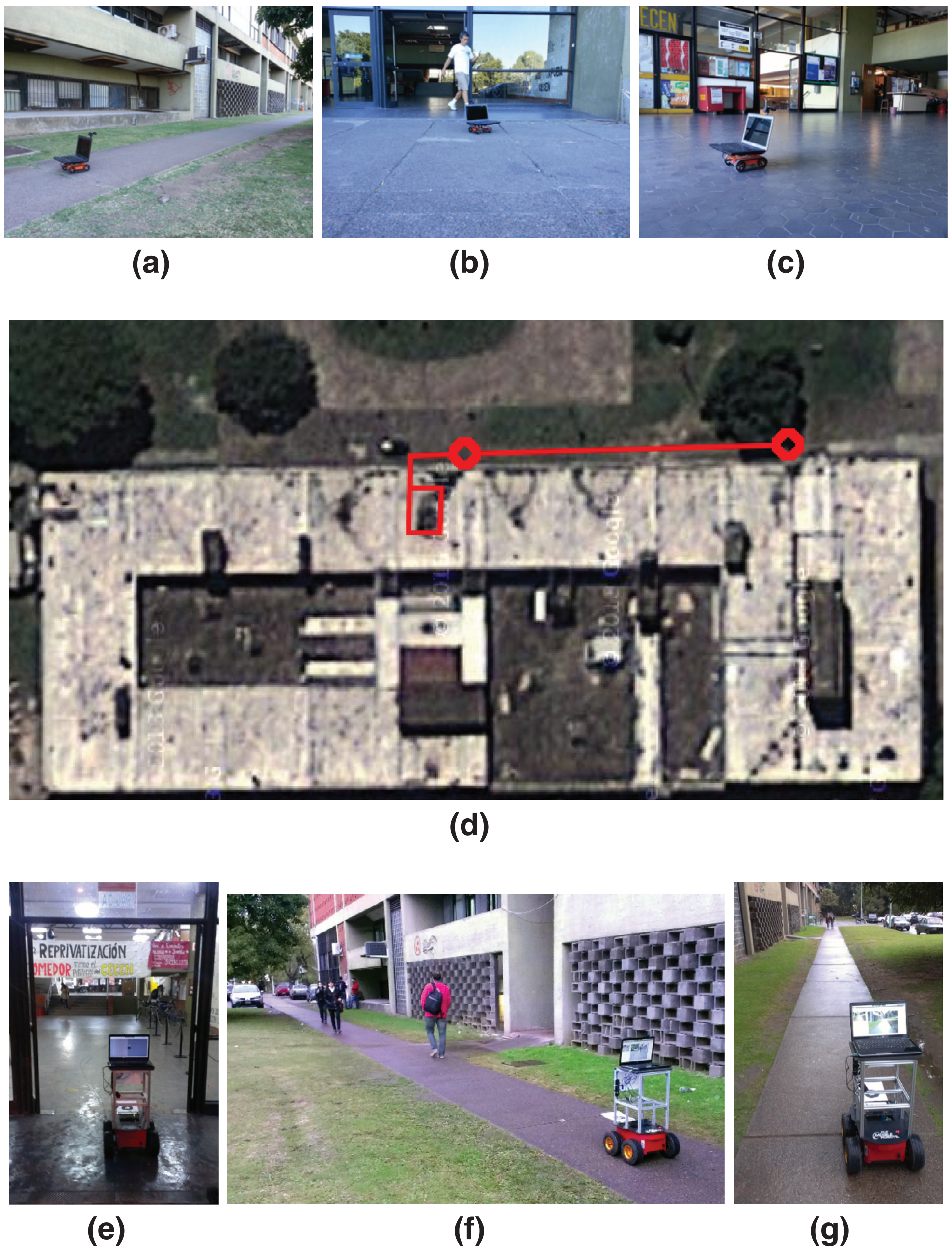


Fig. 4. Images of the experiments and of the path traversed by the robot around and inside the Pabellón 1 building, Ciudad Universitaria, Buenos Aires, Argentina. (a), (b) and (c) show the autonomous navigation phase with the ExaBot, (e), (f) and (g) with the Pioneer 3P-AT, and (d) shows the environment where the experiments took place and the traversed path, taken from Google Maps.

6.4. *Performance comparison*

To compare the algorithm precision with other teach-and-repeat systems, we have decided to use accuracy ε_{acc} and the repeatability ε_{rep} measures as introduced in Chen and Birchfield [8]. The ε_{acc} and εrep are computed as the RMS of Euclidean distance or a standard deviation of the robot's final positions from the path start by equations

$$\varepsilon_{acc} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \|\mathbf{x_i} - \mathbf{x_f}\|^2} \qquad \varepsilon_{rep} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \|\mathbf{x_i} - \mu\|^2}, \tag{18}$$

where $\mathbf{x_i}$ is the robot position after completing the ith loop, $\mathbf{x_f}$ the final position after training and $\mu = \sum_{i=1}^{n} \mathbf{c_i}/n$.

Table 2 compares repeatability and accuracy of the our results to the ones presented in Krajník et al. [7] and Chen and Birchfield [8]. For this analysis the second experiment with the P3-AT was taken.

Table 2
Comparison of the method's accuracy and repeatability to other works.

		Our method	Other methods	
		P3-AT	Krajnik	Chen
Accuracy	(m)	0.19	0.26	0.66
Repeatability	(m)	0.08	0.14	1.47
Path length	(m)	150	1040	40

This summary indicates that the navigation accuracy of our method is comparable with the closely related methods presented in articles [7,8], showing better results for both accuracy and repeatability analysis.

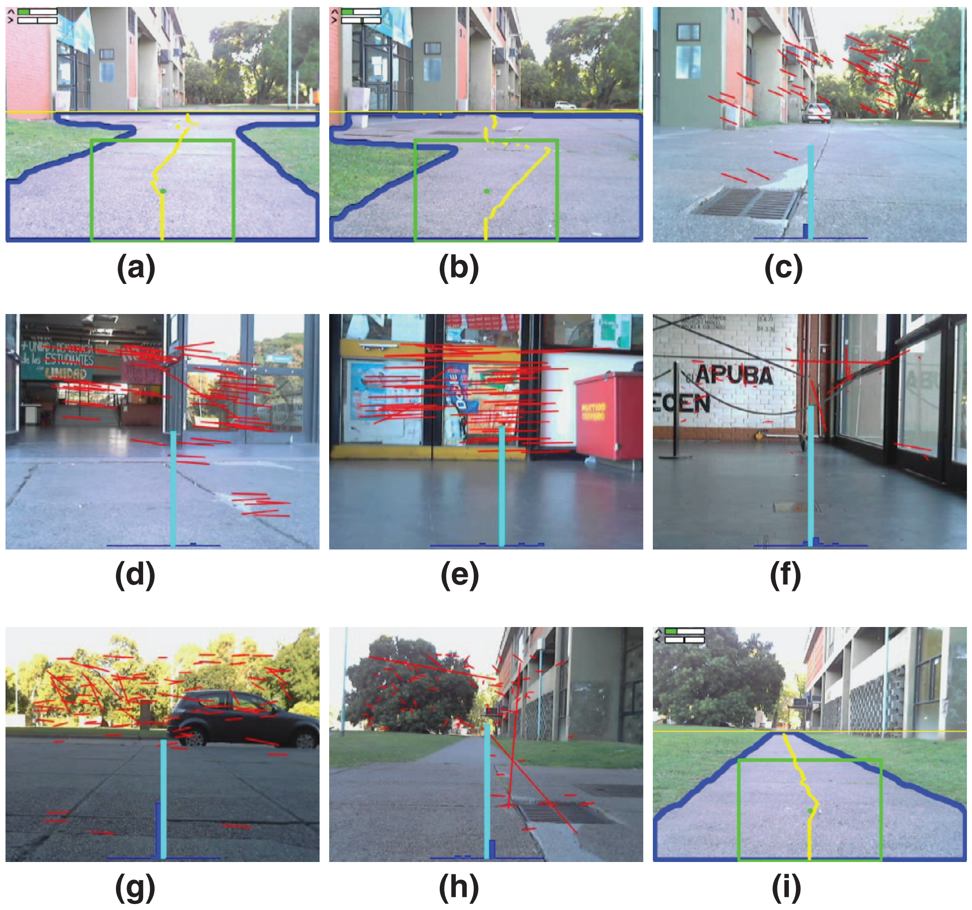


Fig. 5. Screenshots extracted from the ExaBot robot: (a), (b) and (i) during the segmentation-based navigation and (c), (d), (e), (f), (g) and (h) during the landmark-based navigation.

7. Conclusion

In this paper we propose a hybrid visual navigation system for mobile robots in indoor/outdoor environments. The method enables to use both segmentation-based and feature-based navigation as elementary movement primitives for the entire system. A topological map of the environment is defined that can be thought as a graph, where the edges are navigable paths and the nodes are open areas. As we already saw segmentation-based navigation fits very well to path following (edges) and landmark-based navigation is suitable for open areas (nodes).

The presented method is robust and easy to implement and does not require sensor calibration or structured environment, and its computational complexity is independent of the environment size. We also present a convergence analysis to prove both theoretically and empirically that despite the robot localization within a segment is based on pure odometry, if the robot traverses a closed path repeatedly, its position error does not diverge over time due to the heading corrections caused by the properties of the two movement primitives. The aforementioned properties of the method allow even low-cost robots equipped only with a single, off-the-shelf camera to effectively act in large in mixed indoor/outdoor environments.

Acknowledgments

The research was supported by EU-ICT project 600623 STRANDS, CZ-ARG joint project 7AMB14ARo15 and UBACYT project 20020130300035BA.

References

[1] J. Borenstein, Y. Koren, Obstacle avoidance with ultrasonic sensors, IEEE J. Robot. Autom. 4 (2) (1988) 213–218.
[2] H. Surmann, K. Lingemann, A. Nüchter, J. Hertzberg, A 3d laser range finder for autonomous mobile robots, in: Proceedings of the 32nd ISR (International Symposium on Robotics), vol. 19, 2001, pp. 153–158.
[3] K. Kaliyaperumal, S. Lakshmanan, K. Kluge, An algorithm for detecting roads and obstacles in radar images, IEEE Trans. Veh. Technol. 50 (1) (2001) 170–182.
[4] F. Chaumette, S. Hutchinson, Visual servo control. I. Basic approaches, IEEE Robot. Autom. Mag. 13 (4) (2006) 82–90.
[5] F. Chaumette S. Hutchinson, Visual servo control. II. Advanced approaches [tutorial], IEEE Robot. Autom. Mag. 14 (1) (2007) 109–118.
[6] P. Furgale, T.D. Barfoot, Visual teach and repeat for long-range rover autonomy, J. Field Robot. 27 (5) (2010) 534–560.
[7] T. Krajník, J. Faigl, V. Vonásek, K. Košnar, M. Kulich, L. Přeučil, Simple yet stable bearing-only navigation, J. Field Robot. 27 (5) (2010) 511–533.
[8] Z. Chen, S. Birchfield, Qualitative vision-based path following, IEEE Trans. Robot. 25 (3) (2009) 749–754.
[9] C.J. Ostafew, A.P. Schoellig, T.D. Barfoot, Visual teach and repeat, repeat, repeat: Iterative Learning Control to improve mobile robot path tracking in challenging outdoor environments, 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2013, pp. 176–181. 10.1109/IROS.2013.6696350
[10] A. Pfrunder, A.P. Schoellig, T.D. Barfoot, A proof-of-concept demonstration of visual teach and repeat on a quadrocopter using an altitude sensor and a monocular camera, in: 2014 Canadian Conference on Computer and Robot Vision (CRV), IEEE, 2014, pp. 238–245.
[11] K. Ok, D. Ta, F. Dellaert, Vistas and wall-floor intersection features-enabling autonomous flight in man-made environments, in: Workshop on Visual Control of Mobile Robots (ViCoMoR): IEEE/RSJ International Conference on Intelligent Robots and Systems, 2012.
[12] M. Augustine, F. Ortmeier, E. Mair, D. Burschka, A. Stelzer, M. Suppa, Landmark-Tree map: a biologically inspired topological map for long-distance robot navigation, in: International Conference on Robotics and Biomimetics (ROBIO), 2012, pp. 128–135.
[13] T. Krajnik, M. Nitsche, S. Pedre, L. Preucil, M.E. Mejail, A simple visual navigation system for an uav, in: 2012 9th International Multi-Conference on Systems, Signals and Devices (SSD), IEEE, 2012, pp. 1–6. 10.1109/SSD.2012.6198031

[14] M. Nitsche, T. Pire, T. Krajník, M. Kulich, M. Mejail, Monte carlo localization for teach-and-repeat feature-based navigation, in: M. Mistry, A. Leonardis, M. Witkowski, C. Melhuish (Eds.), Advances in Autonomous Robotics Systems, volume 8717 of *Lecture Notes in Computer Science*, Springer International Publishing, 2014, pp. 13–24. 10.1007/978-3-319-10401-0_2

[15] P. De Cristóforis, M.A. Nitsche, T. Krajník, M. Mejail, Real-time monocular image-based path detection, J. Real-Time Image Process. (2013) 1–14.

[16] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, G. Bradski, Self-supervised monocular road detection in desert terrain, in: Proceedings of Robotics: Science and Systems (RSS), 2006.

[17] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, Speeded-up robust features (SURF), Comput. Vis. Image Und. 110 (3) (2008) 346–359.

[18] D. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vis. 60 (2) (2004) 91–110.

[19] T. Krajník, P. De Cristóforis, J. Faigl, H. Szücsová, M. Nitsche, L. Preucil, M. Mejail, Image features for long-term mobile robot autonomy, in: ICRA 2013, Workshop on Long-Term Autonomy, 2013.

[20] M. Agrawal, K. Konolige, M. Blas, Censure: center surround extremas for realtime feature detection and matching, Computer Vision–ECCV 2008, 2008, pp. 102–115.

[21] M. Calonder, V. Lepetit, C. Strecha, P. Fua, Brief: binary robust independent elementary features, Computer Vision–ECCV 2010, 2010, pp. 778–792.

[22] P. Felzenszwalb, D. Huttenlocher, Efficient graph-based image segmentation, Int. J. Comput. Vis. 59 (2) (2004) 167–181.

[23] S. Pedre, M. Nitsche, F. Pessacg, J. Caccavelli, P. De Cristóforis, Design of a multi-purpose low-cost mobile robot for research and education, in: Towards Autonomous Robotic Systems, volume 8717 of *Lecture Notes in Computer Science*, Springer, 2014.

[24] T. Krajník, M. Nitsche, J. Faigl, P. Vaněk, M. Saska, L. Přeučil, T. Duckett, M. Mejail, A practical multirobot localization system, J. Intell. Robot. Syst. (2014) 1–24.