Simple Yet Stable Bearing-Only Navigation

Tomáš Krajník, Jan Faigl, Vojtěch Vonásek, Karel Košnar, Miroslav Kulich, and Libor Přeučil
Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, 121 35 Praha 2, Karlovo náměstí 13, Czech Republic
e-mail: tkrajnik@labe.felk.cvut.cz, xfaigl@labe.felk.cvut.cz, vonasek@labe.felk.cvut.cz, kosnar@labe.felk.cvut.cz, kulich@labe.felk.cvut.cz, preucil@labe.felk.cvut.cz

Received 8 January 2010; accepted 7 June 2010

This article describes a simple monocular navigation system for a mobile robot based on the map-and-replay technique. 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. The method can navigate a robot while sensing only one landmark at a time, making it more robust than other monocular approaches. The aforementioned properties of the method allow even low-cost robots to effectively act in large outdoor and indoor environments with natural landmarks only. The basic idea is to utilize a monocular vision to correct only the robot's heading, leaving distance measurements to the odometry. The heading correction itself can suppress the odometric error and prevent the overall position error from diverging. The influence of a map-based heading estimation and odometric errors on the overall position uncertainty is examined. A claim is stated that for closed polygonal trajectories, the position error of this type of navigation does not diverge. The claim is defended mathematically and experimentally. The method has been experimentally tested in a set of indoor and outdoor experiments, during which the average position errors have been lower than 0.3 m for paths more than 1 km long. © 2010 Wiley Periodicals, Inc.

1. INTRODUCTION

The fundamental problem of mobile robotics is to autonomously navigate a mobile robot along a given path. To fulfill this task efficiently, a robot should maintain some knowledge about its surrounding environment, especially its position relative to the path or desired destination. Such knowledge may be represented in the form of a map, which can be used to estimate the robot position as well as for motion planning. The map is either known a priori and the robot performs localization or is created online and the mobile robot performs so-called simultaneous localization and mapping (SLAM).

The solid mathematical background of the Kalman filter (Kalman, 1960) allowed the research community to establish a sufficient theoretical framework for extended Kalman filter (EKF)-based SLAM. Proof of EKF convergence (Dissanayake, Newman, Clark, Durrant-Whyte, & Csorba, 2001) and lower bounds (Gibbens, Dissanayake, & Durrant-Whyte, 2000) on robot position uncertainty have been formulated. Upper bounds are discussed in the paper by Mourikis and Roumeliotis (2004), where the authors emphasize the importance of robot heading precision during the mapping process. To our knowledge, there is no other paper concerning upper bounds of EKF position estimation. Unfortunately, optimality of the Kalman filter is proven only for linear systems and therefore the main weakness of EKF methods lies in the linearization. The papers by Julier and Uhlmann (2001) and Martinelli, Tomatis, and Siegwart (2005) indicate that due to errors introduced in linearization, EKF methods might provide inconsistent results. Although the linearization process poses a significant threat to the consistency of the position estimation, it can be elegantly avoided using the inverse depth representation (Civera, Davison, & Montiel, 2008; Montiel, Civera, & Davison, 2006).

A multimedia file may be found in the online version of this article.

1.1. Vision-Based Navigation

The theoretical solutions of bearing-only SLAM have gained importance as the computational power of today's computers allows real-time image processing. The nature of visual information allows us to build sparse maps from well-distinguishable landmarks (Lowe, 1999, 2004), which are relatively easy to register. However, the range information is not provided directly by standard cameras. Some bearing-only methods use stereovision in order to obtain immediate range information (Kidono, Miura, & Shirai, 2000). Other methods substitute stereovision by motion and use a single monocular camera (Davison, Reid, Molton, & Stasse, 2007; Holmes, Klein, & Murray, 2008; Montiel et al., 2006).

Most monocular approaches are computationally complex and achieve low operational speeds when mapping large-scale environments. This problem can be solved by dividing a large global map into smaller maps with mutual position information (Bosse, Newman, Leonard, &

Teller, 2004; Clemente, Davison, Reid, Neira, & Tardós, 2007; Estrada, Neira, & Tardós, 2005; Williams, Cummins, Neira, Newman, Reid, et al., 2009).

A different approach is to build an environment map in advance and then use the map for localization (Blanc, Mezouar, & Martinet, 2005; Chen & Birchfield, 2009; Matsumoto, Inaba, & Inoue, 1996; Royer, Lhuillier, Dhome, & Lavest, 2007; Segvic, Remazeilles, Diosi, & Chaumette, 2007). In Royer et al. (2007), a monocular camera is carried through an environment and a video is recorded. The recorded video is then processed (in a matter of several minutes) and subsequently used to guide a mobile robot along the same trajectory. Chen and Birchfield (2006, 2009) present an even simpler form of navigation in a learned map. Their method utilizes a map consisting of salient image features remembered during a teleoperated drive. The map is divided into several conjoined segments, each associated with a set of visual features detected along it and a milestone image indicating the segment end. When a robot navigates a segment, its steering commands are calculated from positions of currently recognized and remembered features. The robot using this method moves forward with a constant speed and steers right or left with a constant velocity or does not steer at all. In Chen and Birchfield (2006), the segment end was detected by means of comparing the milestone image with the current view. The improved version (Chen & Birchfield, 2009) of the qualitative navigation uses a more sophisticated method to determine the segment end. The method takes into account the odometry, the current heading, and the similarity of the current and the milestone images. Still, the authors mention some problems with detection of the segment end. We claim that the segment end can be detected solely by the odometry and the comparison with the milestone image is not always necessary. Comparison with the milestone image increases robustness of the navigation method in cases of wheel slippage and other odometry errors.

The map-and-replay approach is closely related to visual servoing, in which the control of a robot is based on visual measurements (Chaumette & Hutchinson, 2006, 2007). Control inputs are either computed directly by a comparison of the current and reference images (Remazeilles & Chaumette, 2007; Segvic, Remazeilles, Diosi, & Chaumette, 2009) or by a computation of camera coordinates in the world reference frame (DeMenthon & Davis, 1992; Wilson, Hulls, & Bell, 1996). Normally, the visual servoing relies on a geometrical approach to calculate relations of map landmarks to the current image salient points (Segvic et al., 2009). These relations are used to calculate an interaction matrix (Chaumette & Hutchinson, 2006), which links observations to control inputs of the robot. The robot's control input can be computed from the Jacobian (Burschka & Hager, 2001), which relates world and image points, or from homography or fundamental matrices (Guerrero, Martinez-Cantin, & Sagüés, 2005; Remazeilles & Chaumette, 2007), which relate coordinates between actual and reference images. A strong reliance on the geometrical representation requires either camera calibration (Burschka & Hager, 2001; Remazeilles & Chaumette, 2007; Segvic et al., 2009) or structured environment (Guerrero et al., 2005; Remazeilles & Chaumette, 2007). A more detailed overview of visual servoing approaches related to the field of mobile robotics is presented in Chen and Birchfield (2009) and Segvic et al. (2009). Contrary to the visual servoing approach, our method does not require a calibrated camera and does not rely on the environment structure.

1.2. Motivation

The target of our efforts is to create a system that would be able to reliably navigate a mobile robot in an unstructured environment of any size. To achieve this challenging goal, we have decided that the navigation system should have the following properties:

- Scalability: Its computational complexity should be independent of the environment size.
- Simplicity: The method should be as simple as possible, because complex systems are more likely to contain errors.
- Swiftness: It has to satisfy real-time constraints.
- Standardness: It should use off-the-shelf equipment.
- Stability: The position uncertainty should not diverge with time.

The basic idea of the map-and-replay technique is similar to that of the industrial practice of programming stationary robots. One of the basic methods to program a stationary robot is by means of (tele)operation. A skilled operator guides the tip of the robot arm in order to perform a certain task (e.g., painting, welding). The robot records signals from its built-in receptors—typically incremental rotation sensors at its joints. During the robot operation, the recorded sequences serve as inputs for the robot's controllers. Though well established and efficient, this method is not applicable to mobile robots in unstructured environments due to the uncertainty in the robot–environment interaction. A typical example would be the use of odometry in a mobile robot localization—the uncertainty caused by wheel slippages tends to accumulate, which does not make odometry suitable for long-term localization and navigation. To effectively cope with the uncertainty in the robot's position, a mobile robot must use exteroreceptors to sense the surrounding environment. A mobile robot position and its heading can be estimated through measurements of the surrounding environment.

Several authors of SLAM algorithms acknowledge the fact that the uncertainty of robot heading is a crucial factor affecting the quality of the map and subsequently the quality of position estimation in the localization step. The influence of the heading estimation has been evaluated both

theoretically and practically (Frese, 2006). We extend the idea of heading estimation importance and claim that for long-term mobile robot localization it is sufficient to use exteroceptive sensors for heading estimation and the Cartesian coordinate estimation can be based just on proprioceptive sensors.

1.3. Paper Overview

A minimalistic approach to monocular localization and mapping is presented in this paper. We claim that for the navigation in a known environment, a robot needs a map just to estimate its heading and can measure its position by odometry. Formulating this particular instance of the navigation mathematically, we provide a formal proof of this claim. Furthermore, several large outdoor experiments confirm the expected system performance. We think that the most important contribution of our paper is not the presented method but the convergence proof presented in Section 3. The proof would apply to several other methods (Chen & Birchfield, 2009; Guerrero et al., 2005; Zhang & Kleeman, 2009) that use vision to correct heading and lateral position errors.

The rest of this paper is organized as follows. The proposed minimalistic navigation method is described in the next section. A mathematical model of this navigation method is outlined and its properties are examined in Section 3. A theorem that claims that this method prevents position uncertainty divergence is formulated and proven in the same section. The theoretical analysis is followed by a discussion on the practical issues of the navigation method. Experimental results verifying whether the system retains the expected properties are described in Section 5. A conclusion briefly discusses the properties of the proposed navigation method and outlines possible future improvements.

2. NAVIGATION SYSTEM DESCRIPTION

The proposed navigation procedure is based on the map-and-replay technique. The idea is simple: a robot is manually driven through an environment and creates a map of its surrounding environment. After that, the map is used for autonomous navigation. A similar technique for autonomous navigation based on computation of a robot steering from positions of remembered features has been described in Chen and Birchfield (2006), Royer et al. (2007), and Zhang and Kleeman (2009). To minimize the robot sensor equipment and to satisfy the "standardness" property mentioned in Section 1.2, we consider the most available sensors. The fundamental navigation property of a mobile vehicle is the traveled distance, which can be estimated by odometry. The odometric error is cumulative and therefore can be considered precise only in the short term. Another standard available sensor that does not require additional infrastructure is a compass. A fusion of data from the compass and odometry can provide position estimation but is still unsuitable for long-term navigation, because it lacks sufficient feedback from the robot's surrounding environment. To increase robot ability to sense the environment, one of the most advantageous sensors is a camera, which can provide lots of information.

Using these three main sensors, we have proposed the following simple navigation strategy:

- The robot is navigated along a sequence of straight line segments.
- At each segment start, the robot is turned to a direction according to the compass value.
- The steering control along the straight segment is computed from matched visual features providing a so-called visual compass.
- The end of each segment is recognized according to the traveled distance, which is measured by odometry.

The crucial component of the proposed navigation procedure is a map, which is created by guiding the robot along a path consisting of straight-line segments. Each segment has its own landmark map L_i, consisting of salient features detected in images captured by the robot's forward-looking camera, the initial robot orientation α and the segment length s. Once the map is created, the robot can travel autonomously within the mapped environment. During the navigation along a segment, the robot establishes correspondences of the currently seen and previously mapped landmarks and computes differences in the expected and recognized positions for each such correspondence. The robot steers in a direction that reduces those differences while moving straight at a constant speed until its odometry indicates that the current segment has been traversed. At the end of the segment, the robot switches to the next learned segment, turns to a direction of the initial orientation of the segment, and traverses the segment while keeping its direction according to matched features.

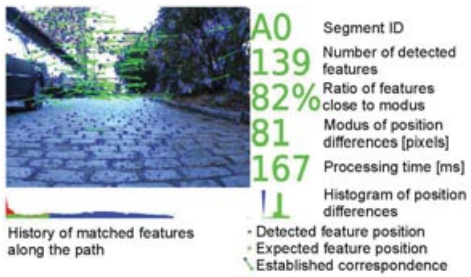
The next section describes the robot equipment and image processing. The algorithm for the map creation during the learning phase is described in Section 2.2, and the navigation algorithm is depicted in Section 2.3.

2.1. Robot Equipment

The proposed method has been verified on the P3AT robot with the Unibrain Fire-i601c camera, the TCM2 compass, and the HP 8710p laptop; see Figure 1(a). At first, the camera was equipped with a 7-mm objective with an electronically driven iris to prevent sunlight dazzle. The objective was replaced by a new one with a 4.5-mm focus length in 2008. At the same time, the electronically driven iris was substituted by a software exposure control. The laptop has Core2 Duo CPU running at 2.00 GHz and 1 GB of memory. Image processing is computationally demanding, and therefore the additional UPC70 battery had been used for longer experiments. To increase the robot action

(a) Robot configuration for experiments

(b) Image captured by robot camera and detected SURF positions



(c) Robot GUI with navigation phase data

Figure 1. Robot platform, detected features, and navigation graphical user interface (GUI).

radius, the three original batteries (connected in parallel) were replaced by one high-capacity battery and the robot's internal PC was disabled. The navigation system was implemented in C/C++ as a stand-alone Linux application.

The image processing algorithm is a critical component of the navigation system. The vision system must provide enough information to steer the robot in the right direction. Furthermore, it should be robust to real-world conditions, i.e., changing illumination, minor environment changes, and partial occlusions, and of course its performance should allow for a real-time response.

We have decided to use the speeded up robust features (SURF) (Bay, Tuytelaars, & Van Gool, 2006) method to identify landmarks in the image. The algorithm provides image coordinates of the salient features together with their descriptions. The SURF method is reported to perform better than most SIFT (Lowe, 1999) implementations in terms of speed and robustness to viewpoint and illumination changes. To achieve an additional speedup, the CPU implementation of the algorithm was adjusted to use both processor cores for parallel image processing. The captured image is horizontally divided, and the parts are processed in parallel. Later, we switched to the GPU (Cornelis & Van Gool, 2008) version of the algorithm. The GPU version has better real-time performance but is less distinctive than the CPU implementation (Svab, Krajnik, Faigl, & Preucil, 2009). Normally, the recognition of a 1,024 × 768 grayscale image provides descriptors of 150–300 features and takes 100–500 ms. The outdoor environment is usually richer in detected features, and image processing tends to be slower than indoors. A typical outdoor processed image with highlighted feature positions is shown in Figure 1(b).

2.2. Learning Phase

In the learning phase, the robot is manually guided through an environment in a turn-move manner and creates a map consisting of several straight segments. Each segment is described by its length s, its azimuth α, and a set of detected landmarks L. A landmark $l \in L$ is described by the tuple $(\mathbf{e}, k, \mathbf{u}, \mathbf{v}, f, g)$, where $\mathbf{e}$ is the SURF descriptor and k indicates the number of images in which the landmark was detected. Vectors $\mathbf{u}$ and $\mathbf{v}$ denote positions of the landmark in the captured image at the moment of its first and last detection, and f and g are the distances of the robot from the segment start in these moments.

The procedure that creates a map of one segment is shown in Algorithm 1. Before the robot starts to learn a segment, it reads compass data to establish the segment azimuth α and resets its odometric counters. After that, the robot starts to move forward, tracks detected features, and inserts them to the set L until the operator requests

Algorithm 1. Learn one segment

Input: α – an initial robot orientation (compass value)
Output: (α, s, L) – the data associated to the segment, where s is the traveled distance and L is a set of landmarks, a landmark is the tuple (k, e, u, v, f, g), where e is the SURF descriptor, k is a counter of feature detection, u and v are positions of the features in the image (at the moment of their first, resp. last occurrence), f and g denote distance from the segment start according to u, resp. v.

$L \leftarrow \emptyset$ // a set of learned landmarks
$T \leftarrow \emptyset$ // a set of tracked landmarks
$\alpha \leftarrow$ *compass value* // a robot orientation at the beginning of segment learning
repeat
  $d \leftarrow$ *current distance from the segment start*
  $S \leftarrow$ *extracted features with associated image position,* $(u, e) \in S$, *u position, e feature descriptor*
  foreach $\boldsymbol{t_i} = (e_i, k_i, u_i, v_i, f_i, g_i) \in T$ **do**
    $(u_a, e_a) \leftarrow \text{argmin}\{||e_i, e(s)|| \; | s \in S\}$ // select the best matching descriptor from S to e_i
    $(u_b, e_b) \leftarrow \text{argmin}\{||e_i, e(s)|| \; | s \in S \setminus \{(u_a, e_a)\}\}$ // select the next best matching descriptor
    if $||(e_i, e_a)|| \ll ||(e_i, e_b)||$ **then**
      $\boldsymbol{t_i} \leftarrow (e_i, k_i + 1, u_i, u_a, f_i, d)$ // update matched landmark
      $S \leftarrow S \setminus \{(u_a, e_a)\}$ // remove matched feature from the current set of detected features
    else
      $T \leftarrow T \setminus \{\boldsymbol{t_i}\}$ // remove $\boldsymbol{t_i}$ from the set of tracked landmarks
      $L \leftarrow L \cup \{\boldsymbol{t_i}\}$ // add $\boldsymbol{t_i}$ to the set of learned landmarks
  foreach $(u, e) \in S$ **do**
    $T \leftarrow T \cup \{(e, 1, u, u, d, d)\}$ // add new feature to the set of tracked landmarks
until *operator terminates learning mode*
$s \leftarrow d$ // the total traveled distance along the segment
$L \leftarrow L \cup T$ // add the current tracked landmarks to the set of learned landmarks

to stop. Images are continuously captured and processed during the movement. For each currently tracked landmark $\boldsymbol{t_i}$ (from the set T), two of the best matching features from the set of new features are found. If these two pairs are distinguishable enough (Bay et al., 2006), the best matching feature is associated to the tracked landmark, which is updated (values k, v, g). Each new feature is added to the set of tracked landmarks T, and its u and v are set to the value of the current distance from the segment start and the counter of the feature detection k is set to one. The segment description is saved at the end of the segment, and the operator can turn the robot to another direction and initiate mapping of a new segment. The format of the file, which stores the segment description, is shown in Table I.

2.3. Autonomous Navigation Mode

In the autonomous navigation mode, an operator enters a sequence of segments and indicates whether the robot should travel repeatedly. The robot is placed at the start of the first segment, loads the description of the segment, and turns itself to the segment azimuth and starts moving forward. The navigation procedure is shown in Algorithm 2. The relevant landmarks for the current robot position (i.e., according to the distance from the segment start) are selected from the set of the learned landmarks L. Correspondences between the mapped and the currently detected landmarks are established in the same way as in the learning phase. A difference in horizontal image coordinates of the features is computed for each such couple. A modus of those differences is estimated by the histogram voting method. The modus is converted to a correction value of the movement direction, which is reported to the robot's steering controller. After the robot travels a distance greater than or equal to the length of the given segment, the next segment description is loaded and the procedure is repeated. During the navigation, the robot displays the relevant states (mapped and recognized landmarks, recognition success ratio, etc.) on its graphical interface; see Figure 1(c).

An important aspect of this navigation algorithm is the fact that it does not need to explicitly localize the robot or to create a three-dimensional map of detected landmarks. It should also be noted that the proposed method is able to work in real time. Even though the camera readings are utilized only to correct the robot direction and the distance is measured by the imprecise odometry, the position uncertainty does not accumulate if the robot changes direction often enough. The stability of the proposed navigation method is discussed in the next section.

3. STABILITY OF BEARING-ONLY NAVIGATION

First, we describe in an informal way how the robot position uncertainty is changed as the robot travels a closed

Table I. A part of a segment map in a text file.

Record	Value	Meaning
Initial azimuth and length	2.13, 7.03	α, s
Landmark 0		
First position	760.74, 163.29	$\mathbf{u}_{\mathbf{l}_0}$
Last position	894.58, 54.44	$\mathbf{v}_{\mathbf{l}_0}$
Max visibility	128	k_{l_0}
First and last visible distance	0.00, 4.25	f_{l_0}, g_{l_0}
Descriptor	1, 0.116727, −0.000254, 0.000499, 0.000352, ...	$\mathbf{e}_{\mathbf{l}_0}$
Landmark 1		
First position	593.32, 381.17	$\mathbf{u}_{\mathbf{l}_1}$
Last position	689.89, 377.23	$\mathbf{v}_{\mathbf{l}_1}$
Max visibility	125	k_{l_1}
First and last visible distance	0.00, 6.73	f_{l_1}, g_{l_1}
Descriptor	−1, 0.070294, −0.006383, 0.012498, 0.006383, ...	$\mathbf{e}_{\mathbf{l}_1}$

Algorithm 2. Traverse one segment

Input: (α, s, L) – the data associated to the segment, where α is an initial angle of the robot orientation at the segment start, s is the traveled distance and L is a set of landmarks, a landmark is the tuple (e, k, u, v, f, g), where e is a SURF descriptor, k is a counter of feature detection, u and v are positions of feature in the image (at the moment of the first, resp. last, occurrence), f and g denote distances from the segment start according to u, resp. v.
Output: ω – a steering speed

```
turn(α)                                              // turn robot in the direction α
d ← current distance from the segment start
while d < s do
    T ← ∅                                            // a set of current tracked landmarks
    H ← ∅                  // a set of differences (horizontal position in the image) of matched features
    d ← current distance from the segment start
    S ← extracted features with associated image position, (u, e) ∈ S, u position, e feature descriptor
    foreach l_i = (e_i, k_i, u_i, v_i, f_i, g_i) ∈ L do
        if f_i ≥ d ≥ g_i then
            T ← T ∪ {l_i}                  // add landmark to the tracked landmarks according to the traveled distance
    while |T| > 0 do
        (e_i, k_i, u_i, v_i, f_i, g_i) ← argmax_{t∈T} k(t)        // get landmark with maximal number of occurrences k
        (u_a, e_a) ← argmin{||e_i, e(s)|| |s ∈ S}                 // select the best matching descriptor from S to e_i
        (u_b, e_b) ← argmin{||e_i, e(s)|| |s ∈ S \ {(u_a, e_a)}}        // select the next best matching descriptor
        if ||(e_i, e_a)|| ≪ ||(e_i, e_b)|| then
            p ← (v_i − u_i) (d − f_i)/(g_i − f_i) + u_i − u_a          // estimate angle to the matched landmark
            H ← H ∪ {p_x}                          // add horizontal difference to set of differences
        T ← T \ {(e_i, k_i, u_i, v_i, f_i, g_i)}                    // discard used landmark
    ω ← modus(H)                                   // determine new robot steering velocity
    report ω to steering controller
```

path. This should help to interpret the mathematical formalism describing the robot position uncertainty in geometrical terms and make the rest of this section more comprehensible. After that, we lay down a formal description of the proposed navigation method and analyze its stability. We outline a model of the robot movement and depict equations allowing the computation of the robot position uncertainty. Next, we use these equations to compute the robot position uncertainty for a closed path. Finally, we examine the properties of the proposed model and establish conditions ensuring that the robot position error does not diverge.

3.1. Geometrical Interpretation

Suppose that the learned path is a square and the robot has to travel it repeatedly. The robot is placed at a random [two-dimensional (2D) Gaussian distribution with zero mean] position near the first segment start; see Figure 2. The initial position uncertainty can therefore be displayed as a circle in which the robot is found with 90% probability. The navigation procedure is executed, and the robot starts to move along the first segment. Because it senses landmarks along the segment and corrects its heading, its lateral position deviation is decreased. However, owing to the odometric error, the longitudinal position error increases. At the end of the segment, the circle denoting position uncertainty becomes an ellipse, with a shorter axis perpendicular to the segment. Heading corrections are dependent on the value of the lateral deviation (see Section 3.2): the greater the deviation, the stronger the effect of heading corrections and therefore the lateral error decreases by a factor h for every traversed segment. The odometry error is independent of the current position deviation and is affected only by the length of the traversed segment, and therefore it can be modeled as an additive error o.

After the segment is traversed, the robot turns by 90 deg and starts to move along the next segment. The uncertainty changes again, but because of the direction change, the longer ellipse axis shrinks and the shorter is elongated due to the odometry error. This repeats for every traversed segment; the size of the uncertainty ellipse converges to a finite value. Because this particular trajectory is symmetric, axis lengths a, b of the "final" ellipse can be easily computed by the equations

$$\begin{aligned} a &= hb, \\ b &= a + o, \end{aligned} \tag{1}$$

where h is the coefficient of the lateral error reduction and o is the odometric error. The position error for $o = 1$ and $h = 0.25$ is shown in Figure 2. Though simple, this particular symmetric case gives us a basic insight into the problem. Now we will derive a broader mathematical model of the navigation, examine its properties, and show that the uncertainty does not diverge for nonsymmetrical trajectories as well.

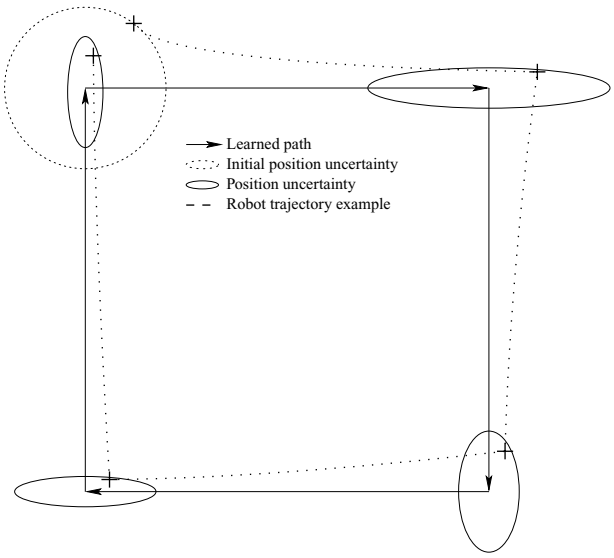


Figure 2. Position uncertainty evolution for a simple symmetric path.

3.2. Navigation

The proposed navigation method is based on the following assumptions:

- The robot moves in a plane.
- The map already exists in the form of a sequence of conjoined linear segments with landmark description.
- At least two segments of the mapped path are not collinear.
- The robot can recognize and associate a nonempty subset of mapped landmarks and determine their bearing.
- The robot can (imprecisely) measure the traveled distance by odometry.
- The camera is aimed forward, i.e., in the direction of the robot movement.

The path P consists of a sequence of linear segments p_i. The robot moves in a plane, i.e., its state vector is (x, y, φ). The robot we consider has a differential, nonholonomic drive, and therefore $\dot{x} = v\cos(\varphi)$ and $\dot{y} = v\sin(\varphi)$. For each segment p_i, there exists a nonempty subset of landmarks and a mapping between the robot position and the expected bearing of each landmark is established. At the start of each segment, the robot resets its odometry counter and turns approximately toward the segment end to sense at least one of the segment landmarks. The robot establishes correspondences of seen and mapped landmarks and computes differences in expected and recognized bearings. The robot steers in a direction that reduces these differences while moving forward until its odometry indicates that the current segment has been traversed.

Definition 1 (Closed-path stability property). *Assume that a robot navigates a closed path several times. Furthermore the robot is using an environment map only for heading corrections and measuring the distance by odometry. Then a path for which the robot position uncertainty does not diverge has the closed-path stability property.*

Theorem 1. *A path consisting of several conjoined segments retains the closed-path stability property if the assumptions in Section 3.2 are satisfied.*

3.3. Movement along One Segment

First, let us examine how a robot moves along one segment. We will focus on the position before and after traversing one segment and establish mapping from the robot position error at the segment start to the robot position error at the segment end.

To keep the model simple, we assume that the robot as well as the landmarks are positioned in a plane. We will consider having a map consisting of a single segment of length s with d landmarks, with positions represented as vectors $\mathbf{u_i}$. Because the robot is equipped with a forward-heading camera, learned landmark positions $\mathbf{u_i}$ are not assumed to be distributed uniformly along the path but rather shifted in the direction of the robot movement by a distance ρ. We can assume that $\rho \approx \frac{1}{d}\sum_{i=0}^{d-1} u_x i$, where d is the number of mapped landmarks. Let us place the segment start at the coordinate origin and the segment end at the position $[s, 0]^T$. We designate the robot position prior to the segment traversal as $\mathbf{a} = [a_x, a_y]^T$ and the final robot position as $\mathbf{b} = [b_x, b_y]^T$; see Figure 3. Let us assume that at every moment during the segment traversal, the robot recognizes a nonempty subset $\mathbf{W}$ of previously learned landmarks and the robot heads in a direction that minimizes the horizontal deviation of expected and recognized landmark positions. We denote the intersection of the robot heading with the learned segment axis as $\mathbf{w}$. At the beginning of the segment traversal, this position equals approximately $[\rho, 0]^T$ (i.e., $\mathbf{w} \approx [\rho, 0]^T$). As the robot traverses the segment, it loses sight of nearby landmarks and recognizes new ones. As new, more distant landmarks appear in the robot field of view and nearby landmarks disappear, the set $\mathbf{W}$ changes and the point $\mathbf{w}$ moves along the segment. It can be assumed that the point $\mathbf{w}$ moves approximately at the speed of the robot and therefore it is always ahead of the robot by the distance ρ.

Based on these premises, the robot position $[x, y]^T$ in terms of $y = f(x)$ can be established. The robot movement can be characterized by the following differential equation:

$$\frac{dx}{dy} = \frac{\rho}{-y}. \tag{2}$$

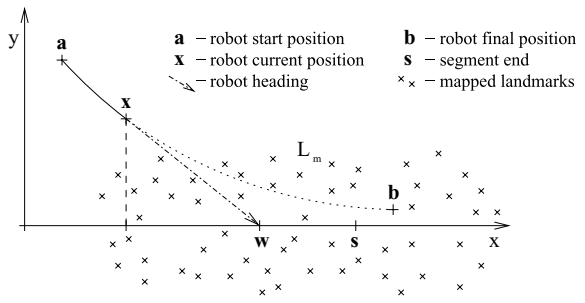


Figure 3. Robot movement model for a single path segment.

Solving Eq. (2) gives us a trajectory along which the robot moves:

$$y = ce^{-x/\rho}.$$

Considering a boundary condition $a_y = f(a_x)$, the constant c equals

$$c = \frac{a_y}{e^{-a_x/\rho}}.$$

Considering that the range of the robot's sensor is higher than the robot position uncertainty and that the segment length is higher than the robot lateral distance from the segment start (i.e., $\rho \gg a_x$, $s \gg |a_y|$), the constant c equals approximately a_y and the traveled distance is approximately equal to the segment length. Therefore, we can estimate the robot position after traveling a segment of length s by the following equations:

$$\begin{aligned} b_x &= a_x + s, \\ b_y &= a_y e^{-s/\rho}. \end{aligned} \tag{3}$$

We can transform Eqs. (3) to the matrix form

$$\begin{pmatrix} b_x \\ b_y \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & e^{-s/\rho} \end{pmatrix} \begin{pmatrix} a_x \\ a_y \end{pmatrix} + \begin{pmatrix} s \\ 0 \end{pmatrix}. \tag{4}$$

Equation (4) is valid for an error-free odometry. If the odometry error is modeled as a multiplicative uncertainty, the equation changes to

$$\begin{pmatrix} b_x \\ b_y \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & e^{-s/\rho} \end{pmatrix} \begin{pmatrix} a_x \\ a_y \end{pmatrix} + s \begin{pmatrix} 1+\upsilon \\ 0 \end{pmatrix}, \tag{5}$$

where υ is a random variable drawn from the Gaussian distribution with the zero mean and the variance ϵ. Accounting for the heading sensor noise, Eq. (5) changes to

$$\begin{pmatrix} b_x \\ b_y \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & e^{-s/\rho} \end{pmatrix} \begin{pmatrix} a_x \\ a_y \end{pmatrix} + \begin{pmatrix} s+s\upsilon \\ \xi \end{pmatrix}, \tag{6}$$

where ξ is a random variable of the Gaussian distribution with the zero mean and the variance τ. Consolidating Eq. (6), we can state

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

The aforementioned movement model holds for a segment aligned with the x axis. For a segment with an arbitrary orientation α, the movement model becomes

$$\mathbf{b} = \mathbf{R^T M R a} + \mathbf{R^T s}, \tag{7}$$

where

$$\mathbf{R} = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix}, \qquad \mathbf{M} = \begin{pmatrix} 1 & 0 \\ 0 & e^{-s/\rho} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & m \end{pmatrix}.$$

Equation (7) corresponds to aligning the segment with the x axis, applying $\mathbf{M}$, adding the odometric and the sensor

noise, and rotating the segment back to the direction α. In the following text, we use $\mathbf{N} = \mathbf{R}^\mathbf{T}\mathbf{M}\mathbf{R}$, which shortens Eq. (7) to

$$\mathbf{b} = \mathbf{N}\mathbf{a} + \mathbf{R}^\mathbf{T}\mathbf{s}. \tag{8}$$

All the aforementioned assumptions about the surrounding environment (landmark shift equal to ρ, $\rho \gg a_x$, etc.) can be relaxed as long as $m < 1$ for $s > 0$.

3.4. Position Uncertainty

Now, the dependence of the robot position uncertainty at the segment end to its uncertainty at the segment start can be examined. Consider that the robot position $\mathbf{a}$ before the segment traversal is a random variable drawn from a 2D normal distribution with the mean $\hat{\mathbf{a}}$ and the covariance matrix $\mathbf{A}$. To compute the robot position uncertainty after the segment traversal, we apply Eq. (8) to $\mathbf{a}$. Because the robot movement model in Eq. (8) has only linear and absolute terms, the robot position uncertainty after the segment traversal will constitute a normal distribution with the mean $\hat{\mathbf{b}}$ and the covariance matrix $\mathbf{B}$.

We denote $\mathbf{a} = \hat{\mathbf{a}} + \tilde{\mathbf{a}}$, where $\hat{\mathbf{a}}$ is the mean of $\mathbf{a}$ and $\tilde{\mathbf{a}}$ is a random variable of a normal distribution with the zero mean and the covariance $\mathbf{A}$. Similarly, we can denote $\mathbf{b} = \hat{\mathbf{b}} + \tilde{\mathbf{b}}$. Thus, we can rewrite Eq. (7) as follows:

$$\tilde{\mathbf{b}} = \mathbf{R}^\mathbf{T}\mathbf{M}\mathbf{R}\tilde{\mathbf{a}} + \mathbf{R}^\mathbf{T}\tilde{\mathbf{s}}.$$

We can claim that

$$\tilde{\mathbf{b}}\tilde{\mathbf{b}}^\mathbf{T} = (\mathbf{R}^\mathbf{T}\mathbf{M}\mathbf{R}\tilde{\mathbf{a}} + \mathbf{R}^\mathbf{T}\tilde{\mathbf{s}})(\mathbf{R}^\mathbf{T}\mathbf{M}\mathbf{R}\tilde{\mathbf{a}} + \mathbf{R}^\mathbf{T}\tilde{\mathbf{s}})^\mathbf{T}.$$

Because $\tilde{\mathbf{s}}$ and $\tilde{\mathbf{a}}$ are independent and do not correlate,

$$\tilde{\mathbf{b}}\tilde{\mathbf{b}}^\mathbf{T} = \mathbf{R}^\mathbf{T}\mathbf{M}\mathbf{R}\tilde{\mathbf{a}}\tilde{\mathbf{a}}^\mathbf{T}\mathbf{R}^\mathbf{T}\mathbf{M}^\mathbf{T}\mathbf{R} + \mathbf{R}^\mathbf{T}\tilde{\mathbf{s}}\tilde{\mathbf{s}}^\mathbf{T}\mathbf{R},$$

which rewritten in terms of covariance matrices is

$$\mathbf{B} = \mathbf{R}^\mathbf{T}\mathbf{M}\mathbf{R}\mathbf{A}\mathbf{R}^\mathbf{T}\mathbf{M}^\mathbf{T}\mathbf{R} + \mathbf{R}^\mathbf{T}\mathbf{S}\mathbf{R}, \tag{9}$$

where

$$\mathbf{S} = \begin{pmatrix} s^2\epsilon^2 & 0 \\ 0 & \tau^2 \end{pmatrix}.$$

Equation (9) allows us to compute the robot position uncertainty after traversing one segment.

3.5. Traversing Multiple Segments

Let us consider a path consisting of n chained segments denoted by $i \in \{0, \ldots, n-1\}$ with the end of the last segment equal to the start of the first segment, i.e., the considered path is closed. We denote length and orientation of the ith segment as s_i and α_i. The robot position before and after traversing the ith segment is noted as $\mathbf{a_i}$ and $\mathbf{b_i}$. Because the robot position at the end of the ith segment equals its start position at the segment $i+1$, we can state that $\mathbf{a_{i+1}} = \mathbf{b_i}$.

The movement model (9) for the ith traveled segment is

$$\mathbf{A_{i+1}} = \mathbf{B_i} = \mathbf{R_i^T}\mathbf{M_i}\mathbf{R_i}\mathbf{A_i}\mathbf{R_i^T}\mathbf{M_i^T}\mathbf{R_i} + \mathbf{R_i^T}\mathbf{S_i}\mathbf{R_i}. \tag{10}$$

Considering $\mathbf{N_i} = \mathbf{R_i^T}\mathbf{M_i}\mathbf{R_i}$ and defining $\mathbf{T_i} = \mathbf{R_i^T}\mathbf{S_i}\mathbf{R_i}$, we can rewrite Eq. (10) as

$$\mathbf{A_{i+1}} = \mathbf{N_i}\mathbf{A_i}\mathbf{N_i^T} + \mathbf{T_i}.$$

One can compute the robot position uncertainty in terms of the covariance matrix after traversing i path segments in the following terms:

$$\mathbf{A_i} = \left(\prod_{\mathbf{j=i-1}}^{\mathbf{0}} \mathbf{N_j}\right) \mathbf{A_0} \left(\prod_{\mathbf{j=0}}^{\mathbf{i-1}} \mathbf{N_j^T}\right) + \sum_{\mathbf{j=0}}^{\mathbf{i-1}} \left[\left(\prod_{\mathbf{k=i-1}}^{\mathbf{j}} \mathbf{N_k}\right) \mathbf{N_j^{-1}} \mathbf{T_j} \left(\mathbf{N_j^T}\right)^{-1} \left(\prod_{\mathbf{k=j}}^{\mathbf{i-1}} \mathbf{N_k^T}\right) \right]. \tag{11}$$

To examine how the robot position uncertainty changes after the robot travels the entire learned path i times, we define $\mathbf{C_i} = \mathbf{A_{in}}$ (e.g., $\mathbf{C_1} = \mathbf{A_n}$). Moreover, we denote

$$\check{\mathbf{N}} = \prod_{j=n-1}^{0} \mathbf{N_j}$$

and

$$\check{\mathbf{T}} = \sum_{\mathbf{j=0}}^{\mathbf{n-1}} \left[\left(\prod_{\mathbf{k=n-1}}^{\mathbf{j}} \mathbf{N_k}\right) \mathbf{N_j^{-1}} \mathbf{T_j} \left(\mathbf{N_j^T}\right)^{-1} \left(\prod_{\mathbf{k=j}}^{\mathbf{n-1}} \mathbf{N_k^T}\right) \right] \tag{12}$$

and rewrite Eq. (11) as

$$\mathbf{C_{i+1}} = \check{\mathbf{N}}\mathbf{C_i}\check{\mathbf{N}}^\mathbf{T} + \check{\mathbf{T}}. \tag{13}$$

By proving that $\mathbf{C_i}$ converges to a finite matrix as i grows to infinity, we prove Theorem 1.

3.6. Convergence Conditions

Expression (13) is the Lyapunov discrete equation (Lyapunov, 1992). If all eigenvalues of $\check{\mathbf{N}}$ lie within a unit circle and $\check{\mathbf{T}}$ is symmetric, then $\lim_{i\to\infty} \mathbf{C_i}$ is finite and equal to $\mathbf{C}_\infty$, which can be obtained by solution of the algebraic equation

$$\mathbf{C}_\infty = \check{\mathbf{N}}\mathbf{C}_\infty\check{\mathbf{N}}^\mathbf{T} + \check{\mathbf{T}}. \tag{14}$$

Because the matrix $\mathbf{S_i}$ is symmetric, $\mathbf{T_i} = \mathbf{R_i^T}\mathbf{S_i}\mathbf{R_i}$ is also symmetric. The product $\mathbf{X}\mathbf{T_i}\mathbf{X}^\mathbf{T}$ is symmetric for any $\mathbf{X}$ and therefore all addends in Eq. (12) are symmetric. Addition

preserves symmetry and therefore the matrix

$$\breve{\mathbf{T}} = \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{n-1}} \left[\left(\prod_{\mathbf{k}=\mathbf{n-1}}^{\mathbf{j}} \mathbf{N_k} \right) \mathbf{N_j^{-1}} \mathbf{T_j} \left(\mathbf{N_j^T} \right)^{-\mathbf{1}} \left(\prod_{\mathbf{k}=\mathbf{j}}^{\mathbf{n-1}} \mathbf{N_k^T} \right) \right]$$

is symmetric.

To prove that the eigenvalues of $\breve{\mathbf{N}}$ lie within a unit circle, we exploit the positiveness of the matrices $\mathbf{M_i}$ and $\mathbf{R_i}$. Because $\mathbf{M_i}$ is positive, $\mathbf{N_i} = \mathbf{R_i^T M_i R_i}$ is also positive. Moreover, as every $\mathbf{N_i}$ equals $\mathbf{R_i^T M_i R_i}$, its eigenvalues are the same as those of $\mathbf{M_i}$ and eigenvectors are columns of $\mathbf{R_i}$. The eigenvalues of $\mathbf{N_i}$ therefore correspond to one and e^{-s_i/ρ_i}. Because the product $\mathbf{XY}$ of a positive definite matrix $\mathbf{X}$ and a symmetric positive definite matrix $\mathbf{Y}$ is positive definite, the matrix $\breve{\mathbf{N}}$ is positive definite as well. Moreover, the dominant (maximal) eigenvalue of the product $\mathbf{XY}$ is lower than or equal to xy, where x and y are dominant eigenvalues of $\mathbf{X}$ and $\mathbf{Y}$. Because the dominant eigenvalue of every $\mathbf{N_i}$ is one, all eigenvalues of $\breve{\mathbf{N}}$ are smaller than or equal to one. The dominant eigenvalue of $\breve{\mathbf{N}}$ is equal to one if and only if the dominant eigenvalue of the product $\mathbf{N_{i+1}N_i}$ equals 1 for all i. Conditions satisfying that the eigenvalues of a product $\mathbf{N_{i+1}N_i}$ are lower than one ensure the existence of a finite solution of Eq. (14). Therefore, we have to find those conditions to support the closed-path stability property.

3.7. Convergence Proof

We will exploit the fact that a product of matrix eigenvalues equals the matrix determinant and the sum of eigenvalues equals matrix trace. Let us denote eigenvalues of the matrix product $\mathbf{N_{i+1}N_i}$ as $\lambda_{0,1}$ and the smaller eigenvalue of $\mathbf{N_i}$ as n_i ($n_i = e^{-s_i/\rho_i}$). For our convenience, we denote $j = i + 1$. Therefore

$$\det\left(\mathbf{N_j N_i}\right) = \det \mathbf{N_j} \det \mathbf{N_i} = \lambda_0 \lambda_1 = n_i n_j. \tag{15}$$

If $\lambda_{0,1} \in \langle 0, 1\rangle$, we can state that

$$(1 - \lambda_0)(1 - \lambda_1) \geq 0, \tag{16}$$

and therefore

$$\lambda_0 \lambda_1 - \lambda_0 - \lambda_1 + 1 \geq 0. \tag{17}$$

Combining Eq. (15) and inequality (17), we obtain

$$1 + n_i n_j \geq \lambda_0 + \lambda_1.$$

Considering that the sum of eigenvalues equals matrix trace, we get

$$\operatorname{trace}\left(\mathbf{R_j^T M_j R_j R_i^T M_i R_i}\right) \leq 1 + n_i n_j. \tag{18}$$

Because trace($\mathbf{AB}$) is equal to trace($\mathbf{BA}$), we can rewrite inequality (18) as

$$\operatorname{trace}\left(\mathbf{M_j}\left(\mathbf{R_i R_j^T}\right)^{\mathbf{T}} \mathbf{M_i R_i R_j^T}\right) \leq 1 + n_i n_j. \tag{19}$$

Both matrices $\mathbf{R_i}$ and $\mathbf{R_j}$ represent rotations. The matrix $\mathbf{R_i}$ denotes rotation by the angle α_i, and $\mathbf{R_j}$ denotes rotation by the angle α_j. Their product $\mathbf{R_i R_j^T}$ denotes rotation by $\alpha_i - \alpha_j$. If we denote $\beta = \alpha_i - \alpha_j$ and $\mathbf{R}_\beta = \mathbf{R_i R_j^T}$, inequality (19) is changed to

$$\operatorname{trace}\left(\mathbf{M_j R_\beta^T M_i R}_\beta\right) \leq 1 + n_i n_j. \tag{20}$$

By expanding matrices $\mathbf{M_j R_\beta^T}$, we obtain

$$\mathbf{M_j R_\beta^T} = \begin{pmatrix} \cos\beta & -n_j \sin\beta \\ \sin\beta & n_j \cos\beta \end{pmatrix}$$

and

$$\mathbf{M_i R}_\beta = \begin{pmatrix} \cos\beta & n_i \sin\beta \\ -\sin\beta & n_i \cos\beta \end{pmatrix}.$$

Inequality (20) can be rewritten to

$$(1 + n_i n_j)\cos^2\beta + (n_i + n_j)\sin^2\beta \leq 1 + n_i n_j$$

and further reduced to

$$1 + n_i n_j - (1 - n_i - n_j + n_i n_j)\sin^2\beta \leq 1 + n_i n_j.$$

Finally, we get

$$(1 - n_i)(1 - n_j)\sin^2\beta \geq 0. \tag{21}$$

Because $n_i = e^{-s_i/\rho_i}$, $n_i \in (0, 1)$ and $n_j \in (0, 1)$, and inequality (21) is strict for $\sin\beta \neq 0$. This fact implies that inequality (16) is strict as well, which means that both λ_0 and λ_1 are lower than one. Therefore both eigenvalues of the matrix product $(\mathbf{N_i N_j})$ are smaller than one if $\beta \neq n\pi | n \in \mathcal{N}$. The matrix $\breve{\mathbf{N}}$ has both eigenvalues smaller than one if and only if at least two conjoined segments of the path form an angle different from 0 or π.

Because all eigenvalues of $\breve{\mathbf{N}}$ lie within a unit circle and $\breve{\mathbf{T}}$ is symmetric, the covariance matrix $\mathbf{C}_\infty$ denoting the robot position uncertainty at the start of the first path segment is finite and obtainable by a solution of the algebraic equation

$$\mathbf{C}_\infty = \breve{\mathbf{N}} \mathbf{C}_\infty \breve{\mathbf{N}}^{\mathbf{T}} + \breve{\mathbf{T}}.$$

□

3.8. Convergence Proof Overview

We have established Eqs. (7) describing the movement of a robot using a navigation method, which is described in Section 2. Equation (10) allowed us to examine the robot position uncertainty evolution as the robot travels through a known environment. Modifying Eq. (10) to closed trajectories, we could rewrite it as Eq. (13). By examining the conditions under which Eq. (13) has a finite solution, we have proven Theorem 1 for closed paths that have at least two noncollinear segments. The existence of a finite solution of Eq. (13) means that if a mobile robot traverses repeatedly a

closed polygonal path using our method, its position error at every point of the traversed trajectory will stabilize at a certain value.

4. PRACTICAL ISSUES

The theoretical proof of convergence stands on several assumptions, which might not always be met. In this section, we will outline possible issues that might arise from incorrect assumptions and discuss their impact on navigation stability and accuracy. Moreover, we will discuss method requirements in terms of computational power, memory, and disk storage.

4.1. Convergence Proof from an Engineer's Point of View

Though elegant and useful, mathematical models and formal proofs are often based on a simplification of reality. A good mathematical model picks out the essence of the modeled problem and concentrates on an examination of the model properties. Some properties of the real system are not included in the model and therefore are not considered. An experienced engineer must be aware of these properties and realize the difference between math and reality. This applies to the proof presented in Section 3 as well.

In practice, extremely elongated (imagine a rectangle with sides 1,000 and 0.001 m long) paths will not retain the closed-path stability property because the movement along the shorter side will not compensate odometry errors accumulated over the long side. Moreover, the model does not cover the probability that the robot will not establish enough correct correspondences because its position error would grow too high. This might easily happen in case the robot has to avoid large obstacles as well as for paths with very long segments.

Moreover, the fact that the position error does not diverge might not be really useful in real robotic systems. In practice, the real precision is more important. The precision can be estimated using Eq. (13) if the learned path shape, landmark distribution ρ, camera noise τ, and odometry noise ϵ are known. Using $\rho = 20$ (i.e., most of the sensed landmarks are 20 m in front of the robot), the sensor noise $\tau = 0.1$, and the odometry noise $\epsilon = 0.0005$ for a square, 1-km-long path, the predicted navigation repeatability (i.e., computed from eigenvalues of $\mathbf{C}_\infty$) is 0.15 m. This value is in good accordance with the experimental results presented in Section 5, where the measured repeatability in outdoor scenarios was 0.14 m.

4.2. Reliance on Odometry

Odometry is regarded as unsuitable for long-term localization due to cumulative errors. Its error model is usually multiplicative with a precision around 1%. The error of the odometric pose estimation is caused mainly by the fact that the robot heading cannot be properly determined. On the other side, odometry can be very precise for traveled distance measurements. Moreover, an odometric error is usually systematic, which can be solved by precise calibration. Our experience with the P3AT robot shows that repeatability of its odometric measurements of the traveled distance on paved roads is better than 0.1%. This means that in the case of precise heading estimation, the robot would be able to travel 1 km with a position error lower than 1 m.

Our approach relies on the fact that the robot changes direction often enough. If the robot would travel in a straight direction for a long distance, its position error might grow beyond an acceptable level. This might be avoided either by forcing the robot to change directions during the learning phase or by complementing the distance measurement by methods without a long-term drift. An example of such a method might be global positioning system or a vision-based localization used in methods in Chen and Birchfield (2009), Royer et al. (2007), and Zhang and Kleeman (2009).

4.3. False Correspondences

The most troublesome issue is that correct correspondences might not be established. However, our algorithm works even in cases of a large number of outliers. Consider a situation in which the system is navigating and all of its established correspondences are false. The horizontal position deviation of detected and mapped features would be basically a random variable. Therefore a histogram H, which is built in order to establish the robot turning speed, will have its bins (approximately) equally filled. The robot turning speed will therefore be random. Now consider that there are a few correctly established correspondences. Each correctly established correspondence increases the value of the bin, which corresponds to the robot's true heading deviation. Therefore the probability that the correct bin has a maximal value increases with each correct correspondence.

This is different from the work presented in Chen and Birchfield (2009) and Segvic et al. (2007), where the authors choose a mean of horizontal differences instead of the modus. The modus is more invariant to the incorrect correspondences than the mean, which makes our method more precise and robust.

In reality, we get 80%–90% correctly established correspondences if the navigation phase follows mapping immediately. As the map gets older, the ratio of correct correspondences tends to drop. The rate of "map decay" depends on the environment and is caused mainly by two factors: short-term lighting changes caused by a change in the position of the sun and the current weather conditions and long-term environment changes caused by seasonal factors. Both illumination and long-term changes are not so significant in indoor environments, because lighting is typically artificial and seasonal changes do not happen. So it

is expected that illuminations and seasonal changes would play an important role in outdoor environments.

To evaluate the system robustness to lighting changes, we made an all-day experiment in which the robot traversed a 1-km-long path in an outdoor environment; see Section 5.5. To evaluate our system robustness to seasonal environment changes, we mapped a 50-m-long path in a park. The path was autonomously navigated and then relearned one month later. This was done in five consecutive months; see Section 5.4. The results of both experiments show that the system is robust to both long-term and short-term environment changes.

Dynamic objects and occlusions cause only a temporary and slight decrease in the ratio of correctly established correspondences. During the experiments, we did not notice any problems with moving objects in the robot field of view.

4.4. Obstacle Avoidance

The proposed navigation method itself does not include obstacle avoidance. However, it can be complemented by a collision avoidance module, which takes control of the robot whenever an obstacle in the robot's course is detected. Such a module guides the robot around the obstacle until the area between the robot and its path is clear again. After that, the visual-based navigation takes control and guides the robot by Algorithm 2.

Because obstacles have finite dimensions, the robot position error will grow by a finite value every time it passes an obstacle. From the theoretical point of view, random obstacles in the robot path can be modeled by the addition of a random vector with a zero mean to $\mathbf{s}$ in Eq. (8). Although the addition will increase the matrix $\mathbf{S}$ in Eq. (9), the symmetry of $\mathbf{S}$ will be preserved. Obstacles would therefore increase the matrix $\check{\mathbf{T}}$ in Eqs. (13) and (14), but because $\check{\mathbf{T}}$ remains symmetric, Eq. (14) will have a unique solution. However, the robot position uncertainty, represented by the matrix $\mathbf{C}_\infty$, will increase. Therefore, obstacle avoidance would decrease the precision of the robot navigation, but it should remain stable. This assumption is experimentally verified in Section 5.3.

It is clear that there exists a size of obstacles for which the algorithm will fail, because after circumnavigating the obstacle, the robot will not find previously mapped features.

4.5. Systematic Errors

Because the navigation algorithm relies on two sensors, there are two sources of systematic errors in our algorithm: the odometry and the camera.

The systematic error of the odometry means that if the robot traverses the distance d, it will report that the traveled distance is $d(1+\eta)$. Let us consider that the robot has 900% odometric error, i.e., $\eta = 9$. During mapping phases, the error will cause the segment lengths s and landmark data f and g to be 10 times higher in the map than in reality. However, in the navigation phases, the odometry error will give 10 times higher values of the robot distance from the segment start, and therefore errors in the map and robot position error will suppress each other.

The systematic error of the camera might be caused by the misalignment of the camera optical axis and the robot body. This causes the positions of landmarks $\mathbf{u}$, $\mathbf{v}$ in the map to be different from a case with an ideal camera. However, when the robot encounters the same location, detected landmark positions will be shifted the same way as in the learning phase. The systematic error will therefore cancel out as in the previous case.

A different case would be a change of the odometric error η or the camera angle θ between the learning and navigation phases. From a theoretical point of view, this would cause a change of the vector $\mathbf{s}$. Unlike in the previous case, the vector $\mathbf{s}$ will not be modified by a random vector but a fixed one. This means that $\tilde{\mathbf{s}}$ will remain the same and matrix $\check{\mathbf{T}}$ will preserve symmetricity. Systematic errors would cause the robot to traverse a trajectory slightly different from the learned one but should not affect navigation stability. However, the algorithm will fail to establish correct correspondences between the mapped and detected features if the systematic errors are too high.

The experimental evaluation of the influence of systematic errors on the navigation stability is described in Section 5.2. Even though the systematic errors were set to high values during the experimental evaluation, the navigation stability was preserved.

4.6. Necessity of a Compass

Relying on only a compass for heading estimation was shown to be a weak point during experiments performed in 2008 and 2009. During learning phases, the compass noise caused an incorrect azimuth estimation of some segments.

Therefore, we considered replacing the absolute azimuth measurements by relative ones. So, instead of recording α_i for the ith segment in the learning phase, the azimuth relative to the previous segment (i.e., $\Delta_i = \alpha_i - \alpha_{i-1}$) is recorded in the map. The relative azimuth Δ_i can be estimated either by odometry or by tracking of the features during transitions to the next segment. This approach is applicable in cases in which a robot is taught a path that is supposed to be traversed later. However, sometimes it is necessary to create a more complex, graph-like map; see Section 5.7.

In more complex cases, the robot creates a map of the large environment in several mapping runs and traverses the given sequence of segments in an order different from the mapped sequence. In this case, sole knowledge of the relative segment azimuths is not sufficient, because angles between nonconsecutive segments are not known to us. Of course an angle between the ith and $(i+j)$th segments can

be estimated by summing all relative angles between segments i and $i + j$, but because every Δ_i contains a small error, the error of the sum is too large for high j.

To deal with these more complex cases, we have implemented a simple Kalman filter, which fuses data from odometry and compass. The filter suppresses the compass noise and causes the absolute heading measurements to be more reliable. However, the filter was implemented at the end of 2009, so in the previous experiments the compass noise caused trouble.

4.7. Computational and Storage Requirements

To estimate computational and storage requirements, we have used data from the experiment described in Section 5.5.

We evaluated the computational requirements in terms of required computational time spent in various stages of the algorithm. The most computationally intensive stage of the algorithm is the feature extraction, which takes 260 ms on average. About 30 ms is taken by establishing proper correspondences. The histogram voting time is less than 1 ms. During experiments, the camera image and additional parameters of the algorithm were displayed for debugging purposes. Drawing these data on a computer screen takes about 60 ms. Thus, the entire control loop takes about 350 ms.

The average landmark density is about 140 landmarks per meter of the path. The map is stored on the hard drive in a text format (see Table I), and one landmark occupies about 800 bytes. Therefore, the disk storage needed for 1 km of path is about 112 MB. Once loaded to the computer memory, a landmark is represented in binary and occupies less than 300 bytes. Thus, a segment 1 km long would take 42 MB of computer memory.

5. EXPERIMENTS

The assumptions formed in Sections 3 and 4 were verified in several real-world experiments. The experimental evaluation was performed in seven different scenarios examining the following:

1. Convergence for two types of paths–with and without the closed-path stability property
2. The impact of systematic errors to the navigation precision
3. Feasibility of complementing the method by the collision avoidance module
4. Robustness to environment changes and variable lighting conditions
5. Performance in environments with landmark deficiency
6. Navigation efficiency for long paths in an outdoor environment with diverse terrain
7. Real deployment of the navigation procedure in RoboTour 2008 and RoboTour 2009 contests (Iša & Dlouhý, 2010)

During these scenarios, the robot autonomously traversed more than 3 km of indoor and more than 25 km of outdoor paths. The P3AT platform with the configuration described in Section 2 was used in all testing scenarios.

The robot learned different closed paths and was requested to navigate these paths several times in the first six scenarios. The relative position $\mathbf{c_i}$ of the robot to the path start was measured every time the robot completed the ith path loop. To evaluate the quality of the navigation algorithm, accuracy and repeatability values as in Chen and Birchfield (2009) were used. The accuracy ε_{acc} and the repeatability ε_{rep} are computed as the rms of the Euclidean distance or the standard deviation of the robot's final positions from the path start:

$$\varepsilon_{\text{acc}} = \sqrt{\frac{1}{n-j}\sum_{i=j}^{n}\|\mathbf{c_i}\|^2}, \quad \varepsilon_{\text{rep}} = \sqrt{\frac{1}{n-j}\sum_{i=j}^{n}\|\mathbf{c_i}-\mu\|^2}, \tag{22}$$

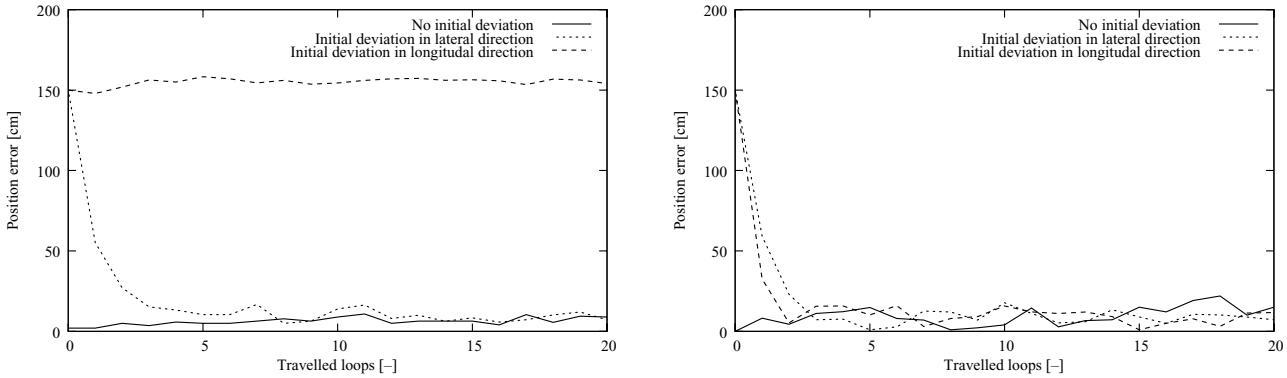
where $\mathbf{c_i}$ is the robot position relative to the path start after completing the ith loop and $\mu = \sum_{i=j}^{n}\mathbf{c_i}/(n-j)$. In most scenarios, the initial robot position was intentionally changed to be 1.5 m apart from the learned path start. In these cases, we do not set j to 1 but wait five loops until the initial position error diminishes. Thus, the repeatability and the accuracy are computed for $j = 5, n = 20$ in the first four scenarios.

5.1. Stability of Robot Position for Different Types of Paths

The scenario examines Theorem 1 for paths with and without the closed-path stability property. The conclusions made in Section 3 indicate that paths with all collinear segments do not retain the closed-path stability property, whereas other paths do. The following paths have been considered: a path with only two collinear segments (i.e., a "back and forth" line path) and a square path. At first, the robot was taught these closed paths in an indoor hall. After that, the robot was placed either directly at the path start or 1.5 m away and requested to navigate along the learned path 20 times. The robot position $\mathbf{c_i}$ was measured after each completed loop.

The first (degenerate) path was formed of two collinear, 5-m-long, segments. The square path was composed of four segments, each 5 m long, with the end of the last segment identical to the segment at the start of the path. The distance of the robot from the path start after each loop (i.e., $\|\mathbf{c_i}\|$) is shown in Figure 4.

The values indicate that for square trajectories, the robot was able to correct the position error that was introduced at the beginning of navigation along the learned path. Only was the error in the y coordinate (i.e., the coordinate axis normal to path segments) partially corrected for collinear trajectories, while the x coordinate remained



(a) The position error for line (degenerate) path

(b) The position error for square (normal) path

Figure 4. The position error for paths without and with the stability property.

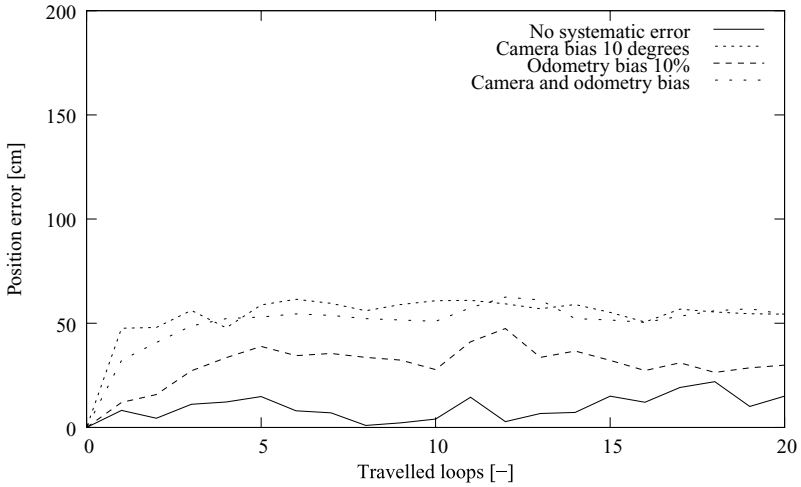


Figure 5. Systematic errors effect: the position error for different types of systematic errors.

uncorrected. These experimental results confirm the theoretical assumptions described in Section 3.8, stating that the navigation is unstable for paths with only collinear segments.

The robot traversed more than 1.8 km in this scenario. Both the accuracy and the repeatability for the 20-m-long square paths were 0.10 m.

5.2. Effect of Systematic Errors

The effect of the systematic errors on navigation precision was evaluated in this scenario. Two sources of systematic errors were considered: the camera and the odometry. An error of the camera can be caused by its optical axis deviation, and an odometric error can be caused by a tire pressure change. To show the effect of the parameter change, it is necessary to modify these parameters between the learning and the navigation phases; otherwise a path is learned with the systematic errors, and therefore the errors do not have an effect.

The following experiments were performed to verify that small-scale systematic errors do not affect navigation stability. At first, the robot camera was panned by 10 deg, and the robot was requested to traverse the square path learned during scenario 5.1 20 times. Then, a 10% systematic odometry error was introduced.[1] Finally, the robot was requested to traverse the path 20 times with both 10% odometry and 10-deg camera bias. As in the previous cases, the robot position $\mathbf{c_i}$ was measured each time the robot reached the learned path start. The measured distances from the path start are shown in Figure 5.

[1]This was done in software—a distance of the robot from the segment start measured by odometry was multiplied by a factor of 1.1 before it was passed to the navigation algorithm.

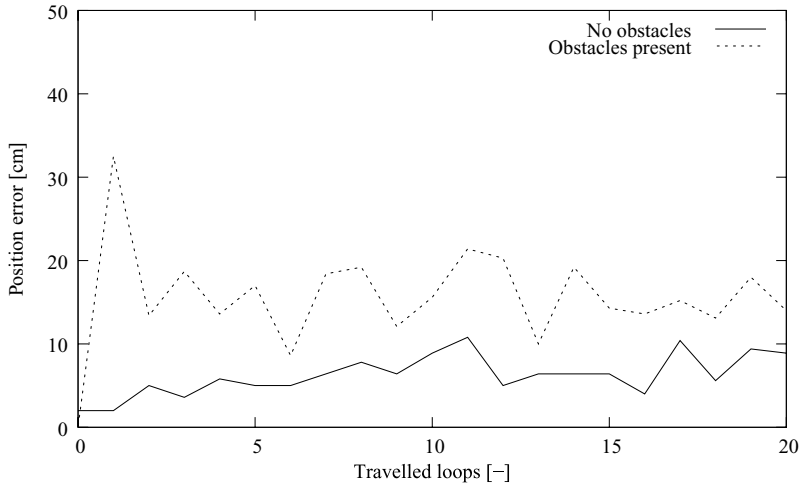


Figure 6. Obstacle avoidance experiment: the position errors with and without obstacles.

The results show that the odometric and the camera biases cause errors in robot positioning but the error does not diverge. After a few loops, the position error does not grow anymore and the system reaches a steady state. The overall accuracy is lower than without the systematic errors, but repeatability remains similar to the previous scenarios.

The robot traversed more than 1.2 km in this scenario. The average accuracy with the camera bias was 0.58 m, and the odometry bias caused the accuracy to change to 0.34 m. When both the odometry and the camera were biased, the accuracy was 0.55 m. The average repeatability was lower than in the previous scenario, i.e., 0.06 m.

5.3. Obstacle Avoidance

A simple collision avoidance module (based on the robot sonars) was activated in this scenario. The collision avoidance is based on the Tangent Bug algorithm with a finite range sensor (Choset, Lynch, Hutchinson, Kantor, Burgard, et al., 2005). When the robot detects an obstacle on its course, the visual-based navigation is suppressed and the robot starts to circumnavigate the detected obstacle. During the circumnavigation, odometry is used to determine the robot position and the sonar data are used to estimate the obstacle position. The visual navigation algorithm is resumed when the path between the robot and the end of the current segment is clear.

The robot was taught a square path similar to the one used in scenario 5.1. After that, one obstacle[2] was placed on each path segment, and the robot navigated the path 20 times. The robot autonomously navigated approximately 0.4 km with an accuracy of 0.16 m and a repeatability of 0.08 m. It is clear that the position precision was affected but did not diverge. See Figure 6.

[2]Obstacle dimensions were approximately half of the robot size.

5.4. Environment and Lighting Changes

The effects of variable lighting conditions and long-term environment changes were examined in this scenario. At first, the robot was taught a closed, 50-m-long path consisting of five segments in the Stromovka park located in the city of Prague. One month later, the robot was placed 1.5 m away from the path start and was requested to navigate the path 20 times. This procedure was repeated five times, i.e., the test was done every month from November 2009 until April 2010. In each experiment, the robot used a map created in a previous month. The measured distances are shown in Figure 7.

Not only did the lighting conditions differ every time but also the environment went through seasonal changes. To document these changes, a picture from the onboard camera was stored every time the mapping was initiated; see Figure 8. There were considerably fewer correct correspondences between recognized and learned features. With a map created just before the navigation, the robot usually correctly recognizes 70%–90% of learned landmarks. Using a 1-month-old map, the ratio of the correctly recognized landmarks drops to 10%–40%. Nevertheless, the robot was able to correct its initial position error and was able to traverse the path faultlessly in all cases.

The robot autonomously navigated more than 6 km with an average accuracy of 0.24 m in this scenario. Unlike in previous scenarios, we did not measure the robot position $\mathbf{c_i}$ after completion of each loop; we recorded the robot distance only from the path start, i.e., $\|\mathbf{c_i}\|$. Therefore, the repeatability cannot be calculated by Eqs. (22). Except for winter months, pedestrians regularly crossed the robot path and moved into the robot's field of view.

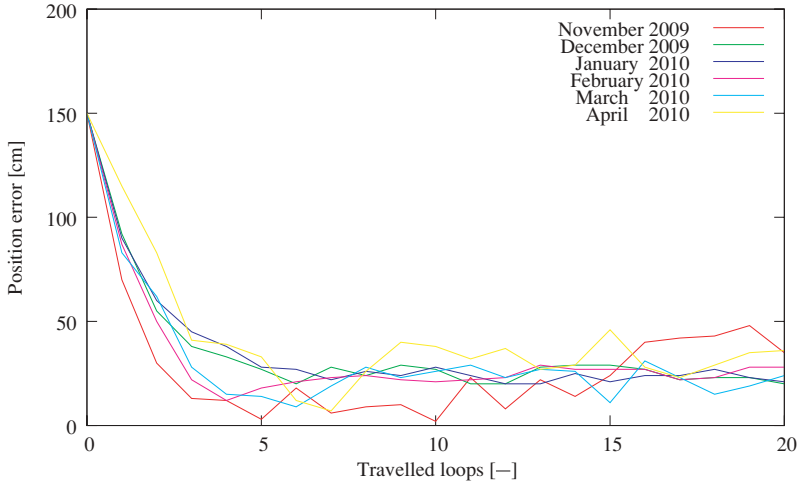


Figure 7. The position errors in different months of the long-term experiment.

(a) October 2009 (c) November 2009 (e) December 2009

(b) January 2010 (d) February 2010 (f) March 2010

Figure 8. Long-term experiment: the view from the robot's camera at the path start in different months.

5.5. One-Day Outdoor Experiment

The system performance for long paths was evaluated in a realistic outdoor environment. The experiment was performed around the Proboštov pond[3] in Proboštov, Czech Republic, at the end of March 2010. The robot was taught a 1-km-long path around the pond in the morning. The path went through a variable nonflat terrain with asphalt paths, dirt roads, footpaths, and grass terrain in an approximately equal ratio (see Figure 9). After the path was taught, the robot was placed 1.5 m away from the path start and requested to traverse it repeatedly. Every time it reached the path start, its position was measured and its batteries replaced (the robot was not moved during the battery exchange). It took approximately 1 h for the robot to traverse the learned path, and the battery replacement took 15 min. The weather changed from cloudy/light rain to partly cloudy/sunny during the experiment. In the afternoon, a lot of pedestrians showed up and either entered the robot's field of view or crossed its path.

Nevertheless, the robot was able to complete the learned path six times before nightfall. The robot traversed

[3] 50°39′58.716″N, 13°50′18.35″E.

Figure 9. One-day experiment: the path around Proboštov pond and the dirt road terrain example.

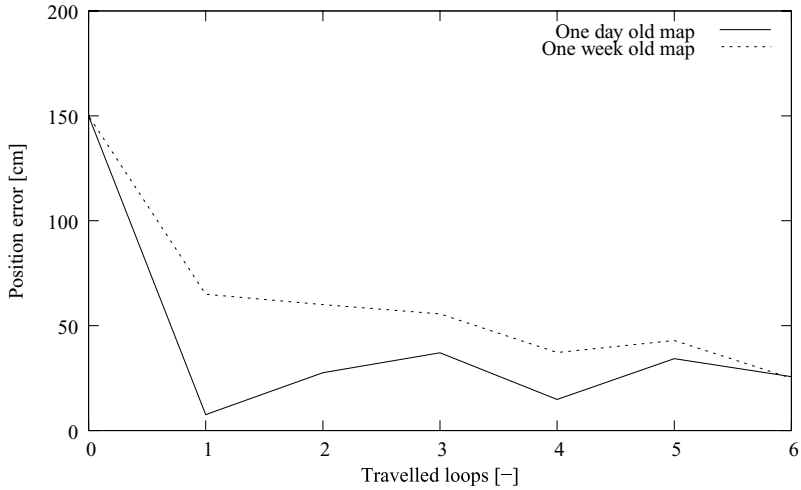


Figure 10. The position errors of the one-day experiment.

6 km with an accuracy[4] of 0.26 m and a repeatability of 0.14 m. The experiment was repeated (without the learning phase) 1 week later, and the robot traversed the path six times with an accuracy of 0.31 m and a repeatability of 0.20 m. The measured distances are shown in Figure 10.

5.6. Landmark Deficiency Experiment

We have claimed that the system is able to operate in an environment that contains a low number of landmarks. To verify this assumption, we taught the robot an outdoor path during night and let it navigate using only streetlamp lights. The robot was taught a 0.3-km-long path on paved roads in a residential area. The onboard camera iris was fully opened, and the camera exposure time was set to 0.37 s. The path was taught at midnight, so more than 90% of the mapped landmarks were streetlamps and illuminated windows.

After the path was learned, the robot was placed 1.5 m from the path start and requested to traverse it 10 times. As opposed to in the daytime experiments, in which the robot detected typically 150–300 landmarks, during the nighttime, the typical number of landmarks was 3. The robot traversed 3 km with an accuracy[5] of 0.32 m and a repeatability of 0.16 m. See Figure 11.

5.7. The RoboTour Outdoor Delivery Challenge

The RoboTour contest (Dlouhy & Winkler, 2009; Iša & Dlouhý, 2010) is an international autonomous robot delivery challenge organized by robotika.cz. The participating

[4]In this case, ε_{acc} and ε_{rep} were computed with $j = 3$ and $n = 6$ in Eqs. (22).

[5]In this case, ε_{acc} and ε_{rep} were computed with $j = 3$ and $n = 10$ in Eqs. (22).

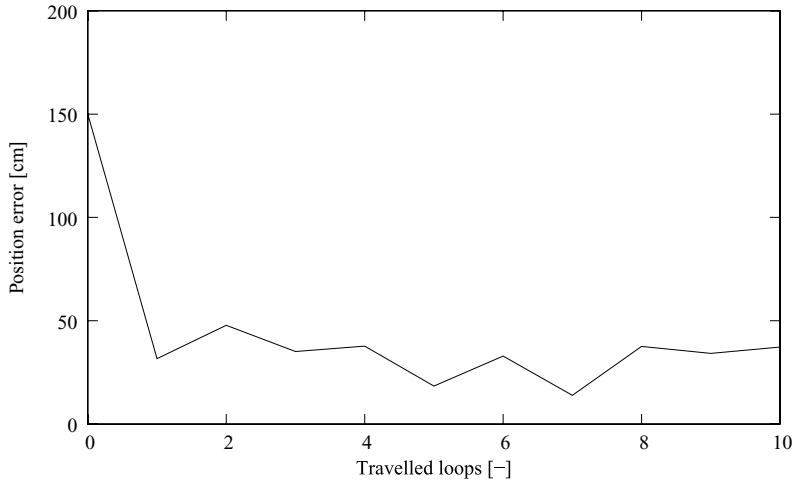
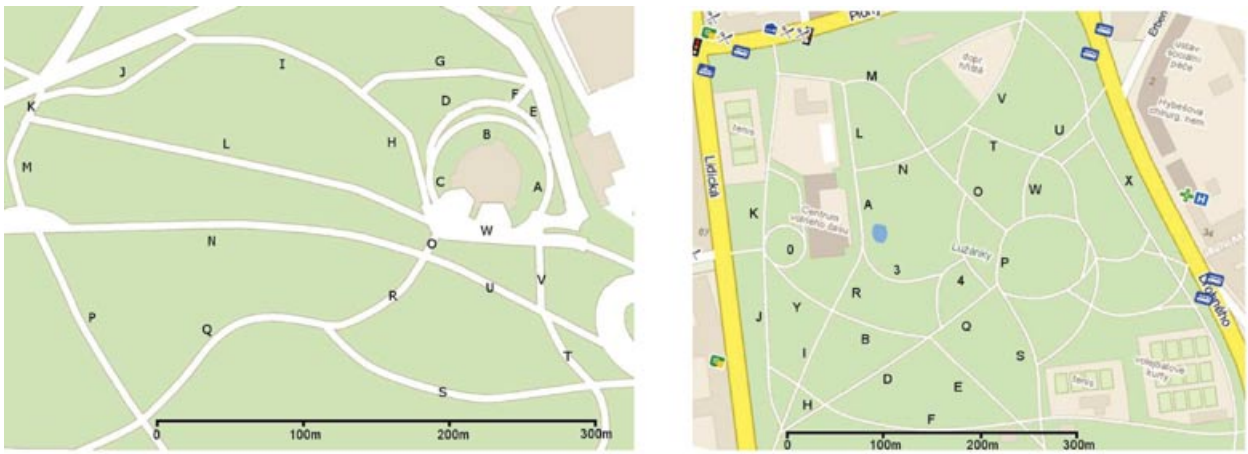


Figure 11. The position errors of the landmark deficiency (night) experiment.



(a) RoboTour 2008 (b) RoboTour 2009

Figure 12. RoboTour 2008/2009 pathway maps.

teams are mostly from Czech and Slovak universities. The competition is a perfect event for an independent verification of system functions and comparison with other navigation methods. However, not only are the navigation methods evaluated, but also the complete systems including all hardware parts are tested.

Fully autonomous robots have to travel a random path in a park, stay on the pavements, and detect randomly placed obstacles in this challenge. A map of the park with its pathways (designated by letters) is given to the teams in advance. The competition consists of several rounds, each with a different path. Thirty minutes before each round, referees choose a random closed path and announce it as a sequence of letters. Competing teams place their robots at the starting positions and execute their autonomous navigation algorithms. Robots must travel without leaving the pathway and without colliding with any random obstacles. The robot score is determined according to its traveled distance. In 2008, the competition was held in Stromovka park[6] in Prague, Czech Republic. One year later, the contest moved to park Lužánky[7] in Brno, Czech Republic.

The Stromovka park pathways were mapped 2 days prior to the competition. The competition had five rounds with different pathways; see Table II and Figure 12(a). The robot completed the required path four times of these five attempts. During two of these attempts, the robot did not leave the pathway at all, and during two others, the robot had partially left (i.e., with two side wheels) the pathway. In

[6] 50°6′18.778″N, 14°25′33.395″E.
[7] 49°12′25.516″N, 16°36′29.81″E.

Table II. RoboTour 2008 and 2009 paths.

Pathway sequence length (m)			
RoboTour 2008		RoboTour 2009	
ABCW	250	ALK0JIR	800
ORQPMKJIH	850	BESQDGHJ0Y	1,050
ABCHGFDCW	500	34PWTMLA	750
HGFEAWOUVW	600	0YR34SFHJ	600
UTSROCDEBCW	700		
Total	2,900	Total	2,200

cases when the robot left the pathway partially, it was left to continue moving (without additional scores) and reached the goal area. One failed attempt was caused by a battery failure.

The competition in Lužánky park was performed in a larger part of the environment; hence the mapping took 3 days. The total length of the mapped pathways was 8 km; the map consisted of approximately one million landmarks, which took 834 MB of disk space. The competition had four rounds; see Table II and Figure 12(b). The robot was able to complete the required paths two times. One attempt failed due to a wrong compass reading during the path learning, but after a manual correction of the robot heading, the robot caught up and reached the goal area. The other failed attempt was caused by a human factor.

Although the performance was not perfect, the robot was able to travel the required trajectory, and our team reached the first rank for both events in 2008 and 2009.

5.8. Experiment Summary

The results of the aforementioned experiments not only confirm theoretical assumptions stated in Section 3, but they also show that our method is feasible for use in real-world conditions. The proposed method is able to cope with diverse terrain, dynamic objects, obstacles, systematic errors, variable lighting conditions, and seasonal environment changes. The summary of experiments in Table III indicates that the localization precision of our method is slightly worse than in closely related methods presented in Royer et al. (2007) and Zhang and Kleeman (2009). Lower precision is probably caused by heavy reliance on odometry and suboptimal use of visual information.

Compared to the similar method presented in Chen and Birchfield (2009), our method accuracy and repeatability is better in outdoor environments. A probable reason for this is that we used a modus to determine robot heading. Chen and Birchfield (2009) use a mean of horizontal deviations, which is less robust to data association errors.

6. CONCLUSION

A simple navigation method based on bearing-only sensors and odometry was presented. In this method, a robot navigating a known environment uses a map of the environment and a camera input to establish its heading, while measuring the traveled distance by odometry. We claim that this kind of navigation is sufficient to keep the robot position error limited. This claim is formulated as a closed-path stability property and proved for polygonal paths with at least two noncollinear segments. The property allows us to estimate the robot position uncertainty based on the landmark density, robot odometry precision, and path shape.

The proposed method was experimentally verified by a mobile robot with a monocular camera. The robot builds a SURF-based (Bay et al., 2006; Cornelis & Van Gool, 2008) landmark map in a guided tour. After that, it uses the aforementioned method to autonomously navigate in the mapped environment.

We conducted experiments indicating that theoretical results and assumed conditions are sound.

The proposed navigation method has surprising properties different from the properties of other navigation and localization methods, mainly the following:

- The robot can perform 2D localization by heading estimation, which is a one-degree-of-freedom method.
- If the robot travels between two points, it is better to use a "zigzag" trajectory rather than a straight one.
- Traveling a closed trajectory might reduce the robot position uncertainty.

Table III. Proposed method accuracy and repeatability in various scenarios.

	Indoor		Outdoor		
	Clear	Obstacles	Long term	1 day	Night
Accuracy (m)	0.10	0.16	0.24	0.25	0.32
Repeatability (m)	0.10	0.08	N/A	0.14	0.16
Loop length (m)	20	20	50	1,040	330

We believe that the convergence proof does not apply only to our system but is valid for many other algorithms. The proof suggests that any algorithm that decreases the lateral position error of a robot is stable for closed polygonal trajectories. This might be the case for even simpler and faster methods, such as the one presented in Zhang and Kleeman (2009). However, this is merely a hypothesis, which needs to be thoroughly examined.

The fundamental limitation of our method is its reliance on odometric measurements. Other visual-based navigation methods use odometry only as an auxiliary measurement or do not require odometry at all. Therefore, these methods would perform better in scenarios in which wheel slippages have to be taken into account. Although our method is limited in application and its precision is lower compared to methods presented in Royer et al. (2007) and Zhang and Kleeman (2009), we believe that it is interesting from an academic point of view.

In the future, we would like to test our algorithm with robots that have only imprecise odometry or inaccurate dead reckoning. The preliminary tests conducted with the AR-Drone quadrotor helicopter, which estimates the traveled distance by accelerometers, seem to be promising.

7. APPENDIX A: INDEX TO MULTIMEDIA EXTENSIONS

The video is available as Supporting Information in the online version of this article.

Extension	Media type	Description
1	Video	Algorithm 1 implemented on a UAV

8. APPENDIX B

This Appendix presents values measured during experiments.

8.1. Stability of Robot Position for Different Types of Paths

Table B.I contains data measured during the first experimental scenario presented in Section 5.1. It shows the measured positions for the paths that do and do not retain the stability property. Note that line (degenerate) paths do not correct the longitudal position deviation, which is in accordance with the convergence proof presented in Section 3.

Table B.I. Convergence for degenerate and normal paths: indoors.

	Position relative to start (m)					
	Line (degenerate) path			Square (normal) path		
Loop	$d_{0,0}$	$d_{0,1.5}$	$d_{1.5,0}$	$d_{0,0}$	$d_{0,1.5}$	$d_{1.5,0}$
00	0.00, 0.00	0.00, 1.50	1.50, 0.00	0.00, 0.00	0.00, 1.50	1.50, 0.00
01	−0.02, 0.00	0.05, 0.55	1.46, −0.24	0.08, −0.02	0.12, 0.58	0.32, −0.06
02	−0.03, −0.04	−0.03, 0.27	1.49, −0.30	0.02, 0.04	0.02, 0.23	0.05, −0.02
03	−0.03, −0.02	−0.06, 0.14	1.53, −0.32	−0.02, 0.11	0.04, 0.06	0.10, 0.12
04	−0.05, 0.03	−0.03, 0.13	1.53, −0.25	0.10, 0.07	−0.07, −0.03	0.05, 0.15
05	−0.05, 0.00	−0.03, 0.10	1.56, −0.27	0.10, 0.11	−0.01, 0.00	0.05, 0.09
06	−0.04, 0.03	−0.03, 0.10	1.55, −0.25	0.08, 0.00	−0.02, 0.02	0.00, −0.16
07	−0.05, 0.04	−0.05, 0.16	1.53, −0.22	0.01, 0.07	0.04, −0.12	−0.03, 0.00
08	−0.05, 0.06	−0.05, 0.00	1.54, −0.25	0.00, −0.01	0.05, 0.11	0.07, 0.04
09	−0.04, 0.05	−0.06, 0.02	1.53, −0.15	−0.01, 0.02	0.05, −0.05	0.07, 0.06
10	−0.04, 0.08	−0.05, 0.13	1.53, −0.21	0.00, −0.04	−0.08, −0.16	0.09, 0.13
11	−0.06, −0.09	−0.04, 0.16	1.54, −0.25	−0.04, −0.14	0.02, −0.11	0.12, 0.02
12	−0.05, 0.01	−0.04, −0.07	1.54, −0.31	0.02, −0.02	0.05, 0.02	0.02, −0.11
13	−0.05, 0.04	−0.08, 0.06	1.55, −0.27	0.03, −0.06	0.00, 0.06	−0.01, −0.12
14	−0.04, 0.05	−0.06, 0.02	1.53, −0.31	0.06, 0.04	−0.09, −0.10	0.02, −0.09
15	−0.05, −0.04	−0.06, 0.06	1.55, −0.21	−0.01, −0.15	−0.01, 0.09	0.01, 0.00
16	−0.04, 0.00	−0.05, −0.03	1.54, −0.24	−0.02, −0.12	−0.05, 0.01	0.01, 0.05
17	−0.03, 0.10	−0.07, 0.02	1.52, −0.21	−0.03, −0.19	0.07, 0.08	0.05, −0.06
18	−0.04, 0.04	−0.09, −0.05	1.55, −0.24	0.02, −0.22	0.09, 0.05	0.01, 0.03
19	−0.03, 0.09	−0.11, 0.05	1.56, −0.10	−0.02, −0.10	0.08, 0.04	0.07, 0.09
20	−0.04, 0.08	−0.07, 0.02	1.54, −0.07	0.01, −0.15	0.02, 0.07	0.04, −0.11

Table B.II. Effect of systematic errors on navigation convergence.

	Position relative to start (m) Systematic error (bias)			
Loop	None	Camera	Odom.	Both
00	0.00, 0.00	0.00, 0.00	0.00, 0.00	0.00, 0.00
01	0.08, −0.02	−0.30, 0.37	0.08, 0.09	−0.16, 0.28
02	0.02, 0.04	−0.28, 0.39	0.09, 0.13	−0.15, 0.38
03	−0.02, 0.11	−0.35, 0.44	0.13, 0.24	−0.16, 0.46
04	0.10, 0.07	−0.29, 0.38	0.15, 0.30	−0.15, 0.50
05	0.10, 0.11	−0.31, 0.50	0.17, 0.35	−0.03, 0.53
06	0.08, 0.00	−0.33, 0.52	0.13, 0.32	−0.24, 0.49
07	0.01, 0.07	−0.34, 0.49	0.19, 0.30	−0.14, 0.52
08	0.00, −0.01	−0.32, 0.46	0.13, 0.31	−0.18, 0.49
09	−0.01, 0.02	−0.33, 0.49	0.12, 0.30	−0.13, 0.50
10	0.00, −0.04	−0.36, 0.49	0.10, 0.26	−0.14, 0.49
11	−0.04, −0.14	−0.35, 0.50	0.13, 0.39	−0.33, 0.47
12	0.02, −0.02	−0.32, 0.50	0.18, 0.44	−0.39, 0.49
13	0.03, −0.06	−0.35, 0.45	0.13, 0.31	−0.38, 0.48
14	0.06, 0.04	−0.33, 0.49	0.14, 0.34	−0.18, 0.49
15	−0.01, −0.15	−0.32, 0.45	0.14, 0.29	−0.13, 0.50
16	−0.02, −0.12	−0.31, 0.40	0.11, 0.25	−0.12, 0.49
17	−0.03, −0.19	−0.36, 0.44	0.11, 0.29	−0.16, 0.51
18	0.02, −0.22	−0.31, 0.46	0.11, 0.24	−0.15, 0.54
19	−0.02, −0.10	−0.35, 0.42	0.12, 0.26	−0.21, 0.53
20	0.01, −0.15	−0.32, 0.44	0.13, 0.27	−0.12, 0.53

8.2. Effect of Systematic Errors

Table B.II contains data from the experimental scenario in Section 5.2, in which effects of the camera and the odometry bias were measured.

8.3. Obstacle Avoidance

Table B.III contains data from the experiment scenario described in Section 5.3, in which we verified the methods ability to deal with obstacles.

8.4. Environment and Lighting Changes

Table B.IV contains data from the experiment scenario described in Section 5.4, in which the algorithm was tested in an outdoor environment with long-term environment changes.

8.5. One-Day Outdoor Experiment

Table B.V contains data from experiment scenario 5.5, in which the algorithm was tested in an outdoor environment with variable terrain.

Table B.III. Positioning errors with and without obstacles.

	Position relative to start (m)	
Loop	Clear path	Obstacles
00	0.00, 0.00	0.00, 0.00
01	0.08, −0.02	0.24, 0.22
02	0.02, 0.04	0.12, 0.06
03	−0.02, 0.11	0.17, 0.08
04	0.10, 0.07	0.11, −0.08
05	0.10, 0.11	0.15, −0.08
06	0.08, 0.00	−0.05, −0.07
07	0.01, 0.07	0.14, −0.12
08	0.00, −0.01	0.15, −0.12
09	−0.01, 0.02	0.02, −0.12
10	0.00, −0.04	0.14, −0.07
11	−0.04, −0.14	0.17, −0.13
12	0.02, −0.02	0.20, −0.04
13	0.03, −0.06	0.08, −0.06
14	0.06, 0.04	0.19, −0.03
15	−0.01, −0.15	0.14, −0.03
16	−0.02, −0.12	0.13, 0.04
17	−0.03, −0.19	0.15, −0.03
18	0.02, −0.22	0.02, −0.13
19	−0.02, −0.10	0.17, −0.06
20	0.01, −0.15	0.14, −0.01

Table B.IV. Long-term algorithm reliability.

	Distance to path start (m)					
Loop	Nov	Dec	Jan	Feb	Mar	Apr
00	1.50	1.50	1.50	1.50	1.50	1.50
01	0.70	0.92	0.90	0.87	0.83	1.15
02	0.30	0.55	0.60	0.50	0.62	0.83
03	0.13	0.38	0.45	0.22	0.28	0.41
04	0.12	0.33	0.38	0.12	0.15	0.39
05	0.03	0.27	0.28	0.18	0.14	0.33
06	0.18	0.20	0.27	0.21	0.09	0.12
07	0.06	0.28	0.22	0.23	0.19	0.07
08	0.09	0.24	0.26	0.24	0.28	0.26
09	0.10	0.29	0.24	0.22	0.23	0.40
10	0.02	0.27	0.28	0.21	0.26	0.38
11	0.23	0.20	0.24	0.22	0.29	0.32
12	0.08	0.20	0.20	0.23	0.23	0.37
13	0.22	0.28	0.20	0.29	0.27	0.27
14	0.14	0.29	0.25	0.27	0.26	0.29
15	0.24	0.29	0.21	0.27	0.11	0.46
16	0.40	0.27	0.24	0.27	0.31	0.28
17	0.42	0.22	0.24	0.22	0.23	0.23
18	0.43	0.23	0.27	0.23	0.15	0.29
19	0.48	0.23	0.23	0.28	0.19	0.35
20	0.45	0.20	0.21	0.28	0.24	0.36

Table B.V. One-day outdoor experiments.

	Position relative to start (m) by map age	
Loop	Up to date	1 week
00	0.00, 1.50	0.00, 1.50
01	−0.07, 0.03	0.63, 0.15
02	−0.21, 0.18	0.59, 0.11
03	−0.34, 0.15	0.53, 0.17
04	−0.05, −0.14	0.32, 0.19
05	0.34, 0.05	−0.09, 0.22
06	−0.25, 0.06	0.15, 0.20

Table B.VI. Landmark deficiency experiment.

Loop	Position relative to start (m)
00	0.00, 1.50
01	−0.13, −0.29
02	0.21, −0.43
03	−0.09, −0.34
04	−0.32, −0.20
05	0.12, −0.14
06	−0.08, −0.32
07	−0.05, −0.13
08	−0.29, −0.24
09	−0.09, −0.33
10	−0.10, −0.36

8.6. Landmark Deficiency Experiment

Table B.VI contains data from the experiment scenario described in Section 5.6, in which the algorithm was tested during night in low-visibility conditions.

ACKNOWLEDGMENTS

We would like to thank our colleagues for valuable remarks and friends for help with the outdoor tests. The work presented in this paper has been supported by the Ministry of Education of the Czech Republic under projects MSM 6840770038 and 2C06005, by CTU research grant SGS10/185/OHK3/2T/13, and by the FP7-ICT project "REPLICATOR" No. 216240.

REFERENCES

Bay, H., Tuytelaars, T., & Van Gool, L. (2006, May). SURF: Speeded up robust features. In Proceedings of the Ninth European Conference on Computer Vision, Graz, Austria.

Blanc, G., Mezouar, Y., & Martinet, P. (2005, April). Indoor navigation of a wheeled mobile robot along visual routes. In Proceedings of International Conference on Robotics and Automation, Barcelona, Spain.

Bosse, M., Newman, P., Leonard, J. J., & Teller, S. (2004). SLAM in large-scale cyclic environments using the Atlas framework. International Journal of Robotics Research, 23(12), 1113–1139.

Burschka, D., & Hager, G. (2001, May). Vision-based control of mobile robots. In Proceedings International Conference on Robotics and Automation, Seoul, Korea.

Chaumette, F., & Hutchinson, S. (2006). Visual servo control, part I: Basic approaches. IEEE Robotics and Automation Magazine, 13(4), 82–90.

Chaumette, F., & Hutchinson, S. (2007). Visual servo control, part II: Advanced approaches. IEEE Robotics and Automation Magazine, 14(1), 109–118.

Chen, Z., & Birchfield, S. T. (2006, May). Qualitative vision-based mobile robot navigation. In 2006 (ICRA), IEEE International Conference on Robotics and Automation (ICRA), Orlando, Florida (pp. 2686–2692).

Chen, Z., & Birchfield, S. T. (2009). Qualitative vision-based path following. IEEE Transactions on Robotics and Automation, 25(3), 749–754.

Choset, H., Lynch, K. M., Hutchinson, S., Kantor, G. A., Burgard, W., Kavraki, L. E., & Thrun, S. (2005). Principles of robot motion: Theory, algorithms, and implementations. Cambridge, MA: MIT Press.

Civera, J., Davison, A. J., & Montiel, J. (2008). Inverse depth parametrization for monocular SLAM. IEEE Transactions on Robotics, 24(5), 932–945.

Clemente, L. A., Davison, A. J., Reid, I. D., Neira, J., & Tardós, J. D. (2007). Mapping large loops with a single hand-held camera. In W. Burgard, O. Brock, and C. Stachniss (Eds.), Robotics: Science and systems. Cambridge, MA: MIT Press.

Cornelis, N., & Van Gool, L. (2008, June). Fast scale invariant feature detection and matching on programmable graphics hardware. In CVPR 2008 Workshop (June 27th), Anchorage AK.

Davison, A. J., Reid, I. D., Molton, N. D., & Stasse, O. (2007). MonoSLAM: Real-time single camera SLAM. IEEE Transactions on Pattern Analysis and Machine Intelligence, 29(6), 1052–1067.

DeMenthon, D., & Davis, L. S. (1992, May). Model-based object pose in 25 lines of code. In ECCV '92: Proceedings of the Second European Conference on Computer Vision, Santa Margherita Ligure, Italy.

Dissanayake, M., Newman, P., Clark, S., Durrant-Whyte, H., & Csorba, M. (2001). A solution to the simultaneous localization and map building (SLAM) problem. IEEE Transactions on Robotics and Automation, 17(3), 229–241.

Dlouhy, M., & Winkler, Z. (2009). Robotour outdoor delivery challenge. http://robotika.cz/competitions/en. Accessed June 26, 2010.

Estrada, C., Neira, J., & Tardós, J. D. (2005). Hierarchical SLAM: Real-time accurate mapping of large environments. IEEE Transactions on Robotics, 21(4), 588–596.

Frese, U. (2006). A discussion of simultaneous localization and mapping. Autonomous Robots, 20(1), 25–42.

Gibbens, P., Dissanayake, G., & Durrant-Whyte, H. (2000, December). A closed form solution to the single degree

of freedom simultaneous localisation and map building (SLAM) problem. In Proceedings of the 39th IEEE Conference on Decision and Control, Sydney, Australia (pp. 191–196).

Guerrero, J. J., Martinez-Cantin, R., & Sagüés, C. (2005). Visual map-less navigation based on homographies. Journal of Robotic Systems, 22(10), 569–581.

Holmes, S., Klein, G., & Murray, D. W. (2008, May). A square root unscented Kalman filter for visual monoSLAM. In 2008 IEEE International Conference on Robotics and Automation, Pasadena, CA (pp. 3710–3716).

Iša, J., & Dlouhý, M. (2010, September). Robotour—Robotika.cz outdoor delivery challenge. In Proceedings of the 1st Slovak–Austrian International Conference on Robotics in Education, Bratislava, Slovakia (to appear).

Julier, S., & Uhlmann, J. (2001, May). A counter example to the theory of simultaneous localization and map building. In 2001 IEEE International Conference on Robotics and Automation, Seoul, Korea (pp. 4238–4243).

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. Transactions of the ASME—Journal of Basic Engineering, 82(Series D), 35–45.

Kidono, K., Miura, J., & Shirai, Y. (2000, July). Autonomous visual navigation of a mobile robot using a human-guided experience. In Proceedings of 6th International Conference on Intelligent Autonomous Systems, Venice, Italy (pp. 620–627).

Lowe, D. G. (1999, September). Object recognition from local scale-invariant features. In ICCV '99: Proceedings of the International Conference on Computer Vision, Kerkyra, Corfu, Greece (p. 1150).

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60(2), 91–110.

Lyapunov, A. (1992). The general problem of stability in motion. International Journal of Control, 55(3), 531–773.

Martinelli, A., Tomatis, N., & Siegwart, R. (2005, August). Some results on SLAM and the closing the loop problem. In International Conference on Intelligent Robots and Systems, Edmonton, Canada (pp. 334–339).

Matsumoto, Y., Inaba, M., & Inoue, H. (1996, April). Visual navigation using view-sequenced route representation. In Proceedings of the International Conference on Robotics and Automation, Minneapolis, MN.

Montiel, J., Civera, J., & Davison, A. (2006, August). Unified inverse depth parametrization for monocular SLAM. In Proceedings of Robotics: Science and Systems, Philadelphia, PA.

Mourikis, A., & Roumeliotis, S. I. (2004, September). Analysis of positioning uncertainty in simultaneous localization and mapping (SLAM). In Proceedings of the IEEE/RSJ International Conference on Robotics and Intelligent Systems (IROS), Sendai, Japan.

Remazeilles, A., & Chaumette, F. (2007). Image-based robot navigation from an image memory. Robotics and Autonomous Systems, 55(4), 345–356.

Royer, E., Lhuillier, M., Dhome, M., & Lavest, J.-M. (2007). Monocular vision for mobile robot localization and autonomous navigation. International Journal of Computer Vision, 74(3), 237–260.

Segvic, S., Remazeilles, A., Diosi, A., & Chaumette, F. (2007, June). Large scale vision based navigation without an accurate global reconstruction. In IEEE International Conference on Computer Vision and Pattern Recognition, Minneapolis, MN.

Segvic, S., Remazeilles, A., Diosi, A., & Chaumette, F. (2009). A mapping and localization framework for scalable appearance-based navigation. Computer Vision and Image Understanding, 113(2), 172–187.

Svab, J., Krajnik, T., Faigl, J., & Preucil, L. (2009, November). FPGA-based speeded up robust features. In 2009 IEEE International Conference on Technologies for Practical Robot Applications, Boston, MA.

Williams, B., Cummins, M., Neira, J., Newman, P., Reid, I., & Tardós, J. (2009). A comparison of loop closing techniques in monocular SLAM. Robotics and Autonomous Systems, 57(12), 1188–1197.

Wilson, W., Hulls, C., & Bell, G. (1996). Relative end-effector control using Cartesian position based visual servoing. Robotics and Automation, 12(5), 684–696.

Zhang, A. M., & Kleeman, L. (2009). Robust appearance based visual route following for navigation in large-scale outdoor environments. International Journal of Robotics Research, 28(3), 331–356.