This is a pre-peer reviewed version of the following article:
Krajník, T., Faigl, J., Vonásek, V., Košnar, K., Kulich, M. and Přeučil, L., Simple yet stable bearing-only navigation. Journal of Field Robotics, n/a. doi: 10.1002/rob.20354,
which has been published in final form at
http://onlinelibrary.wiley.com/doi/10.1002/rob.20354/abstract.
The published paper is considerably extended over this draft.

# Simple, yet stable bearing-only navigation

**Tomáš Krajník***
Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague
121 35 Praha 2, Karlovo náměstí 13, Czech Republic
tkrajnik@labe.felk.cvut.cz

**Jan Faigl**
Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague
xfaigl@labe.felk.cvut.cz

**Miroslav Kulich**
Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague
kulich@labe.felk.cvut.cz

**Libor Preucil**
Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague
preucil@labe.felk.cvut.cz

## Abstract

This article describes a simple monocular navigation system for a mobile robot based on map and replay technique. The presented method is robust, easy to implement, 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 effectivelly act in large outdoor and indoor environments with natural landmarks only.

The basic idea is to utilize monocular vision to correct robot heading only and leaving distance measurements to odometry. The heading correction itself can suppress odometric error keeping the position error bound.

The article examines influence of map-based heading estimation and odometric errors on the overall position uncertainty. A claim that for a certain set of trajectories the localization error of this type of navigation remains bound is stated. This claim is defended mathematically and by simulated and real-world experiments. This method was demonstrated in a

*Use footnote for providing further information about author (webpage, alternative address). Acknowledgments to funding agencies should go in the **Acknowledgments** section at the end of the paper.

Robotour autonomous robot competitions, during which it has successfully completed paths over 1 km length with localization errors lower than 0.5 m.

# 1 Introduction

One of the fundamental problems of mobile robotics is to autonomously navigate a given path. To fulfill this task effectivelly, the robot should maintain some knowledge about its surrounding, especially its position relative to the destination. This knowledge might be represented in several ways, (J. Svab and Preucil, 2009a) is mostly represented in form of a map, which is used by the robot to estimate its position up to some error. The map is either known a priory and the robot performs localization, or is created on-line and the mobile robot performs simultaneous localization and mapping (SLAM).

The solid mathematical background of Kalman filter (Kalman, 2005) allowed the research community to build a sufficient theoretical base for EKF-based SLAM. Proofs of EKF convergence (Dissanayake et al., 2001) and lower bounds (Gibbens et al., 2000) on robot position uncertainty have been formulated. Upper bounds are discussed in paper (Mourikis and Roumeliotis, 2004), where authors emphasize the importance of robot heading precision during the mapping process. Up to our knowledge, there is no other paper concerned with upper bounds of EKF position estimation. Unfortunately, Kalman filter optimality is proven only for linear systems and therefore the weakness of EKF methods is the linearization. Papers (Julier and Uhlmann, 2001) (Martinelli et al., 2005) indicate, that due to errors introduced in linearization, EKF methods might provide inconsistent results. The linearization process poses a significant threat to the consistency of robot position estimation especially if the sensors do not provide bearing and range simultaneously. Range-only methods (Leonard et al., 2002) using EKF were studied in order to model a sonar-based SLAM. However, methods based on bearing-only sensors seem to be even more nonlinear in nature and therefore, they are usually assumed to provide range measurements by triangulation (Bailey, 2003).

## 1.1 Vision-based navigation

The theoretical solutions of bearing-only SLAM have gained importance as the computational power of nowadays computers allows real time image processing. The nature of visual information allows to build sparse maps from well distinguishable landmarks (Se et al., 2001), which are easy to register. However, the range information is still not provided by most cameras. Some bearing only methods use stereo in order to obtain immediate range information (Kidono et al., 2000). Other methods substitute stereo by motion and use a single monocular camera (Davison et al., 2007), (Holmes et al., 2008).

However, due to the aforementioned linearization problems, monocular approaches remain computationally complex and cannot usually map more than thousands of landmarks, while maintaining a real-time operation speed. Methods, which do not perform SLAM, but rather build an environment map in advance and then use the map for localization (Blanc et al., 2005),(Matsumoto et al., 1996) are therefore still being developed (Royer et al., 2007), (Chen and Birchfield, 2006). In article (Royer et al., 2007), monocular camera is carried through the environment while it is recording a video, which is then processed (in matter of hours) and subsequently used to guide the robot along the same trajectory. Another article (Chen and Birchfield, 2006) presents even simpler form of navigation in a learned map. This method utilizes a map consisting of salient image features rememberred during a teleoperated drive. Robot steering commands are calculated from positions of recognized and rememberred features.

## 1.2 Motivation

The target of our efforts is to create a system, which would be able to navigate reliably in unstructured environment of any size. To achieve this, we have decided the navigation algorithm 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, as complex systems are more likely to contain errors,
- swiftness - it has to satisfy real-time constraints,
- standardness - it should use off-the shelf equipment and sensors,
- stability - the position uncertainty should be bound.

The basic idea of map and replay method is similar to 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 labourer guides the tip of the robot arm in order to perform a certain task (e.g. painting, welding). The robot records signals from its intrareceptors - typicaly incremental rotation sensors at its joints. During robot operation, the recorded sequences serve as inputs for robots' controllers. Though well established and efficient, this method is not applicable to mobile robots in unstructured environments due to the uncertainty in robot-environment interaction. A typical example of the case would be the use of odometry in mobile robot localization - the uncertainty, caused by wheel slipping, tends to accumulate and the robot finally loses track of its position, which denies to use odometry in long-term localization. To effectivelly cope with uncertainty in position, a mobile robot must use extrareceptors to sense the surrounding environment. Through measurements of the surrounding environment, a mobile robot estimates its position and heading.

Several authors of SLAM algorithms acknowledge the fact, that uncertainty of robot heading is a crutial factor influencing the quality of the generated map and subseqently the quality of position estimation in the localization step. The influence of the heading estimation has been evaluated both theoretically and in real applications (Frese, 2006).

In our article, we extend this idea and claim, that using extrareceptive sensors for heading estimation is sufficient for long term robot localization and that cartesian coordinate estimation can be based solely on intrareceptive sensors.

### 1.3 Paper overview

This paper presents a minimalistic approach to monocular localization and mapping. We claim, that for navigation in a known environment, a robot needs a map to estimate its heading only and can measure its position by odometry. Formulating this particular instance of navigation mathematically, we provide a formal proof of our this claim. Furthermore, several large outdoor experiments confirm system performance.

The paper is organized as follows: First, an overview of current state of vision-based map building and localization methods is presented. In the next section, a minimalistic navigation method based on map-based heading assessment and odometry measurements is proposed. A mathematical model of this navigation method is outlined and its properties are examined. Theorem, which claims that this method can keep robot position uncertainty bound is formulated and proven. Experiments verifying, whether the system has properties of the outlined mathematical model are described in the next section. Conclusion resumes the paper and shortly discusses properties of proposed navigational method.

# 2 Navigation system description

Proposed navigation procedure is based on record and re-play technique. The idea is simple, a robot is manually driven through an environment and creates a map of its surrounding. After that, the map is used for autonomous navigation. Similar technique for autonomous navigation based on computation of robot steering from positions of remembered features has been described by (Chen and Birchfield, 2006; Royer et al., 2007; Zhang and Kleeman, 2009). To minimize robot sensor equipment and to satisfy the "standard" condition 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. However odometric error is cumulative, and therefore is precise only in short term. Another standard available sensor which does not require additional infrastructure is a compass. A fusion of inputs from compass and odometry can improve localization, but it still unsuitable for long-term navigation, because it lacks sufficient feedback from surrounding environment. To increase robot ability to sense the environment, one of the most advantageous sensors is camera, which can provide lots of information.

Using these three main sensors, we have proposed a simple navigation strategy.

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

The key component of the proposed navigation procedure is a map, which is created by guiding the robot along a path consisting of straight segments. Each segment has its own landmark map $L_i$, consisting of salient features detected in the image captured by robots forward looking camera, initial robot orientation $\alpha$ and segment length $s$. After the map is created, the robot can travel autonomously within the mapped environment. During navigation along a segment, the robot establishes correspondences of currently seen and previously mapped landmarks and computes differences in expected and recognized positions for each such pair. The robot steers in a direction which 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 next learned segment, turns into remembered direction and traverses the segment while keeps direction according to matched features.

The next section describes robot equipment and image processing. Algorithm for creation of map during learning phases is described in section 2.2 and the navigation algorithm is depicted in subsection 2.3.

## 2.1 Robot equipment

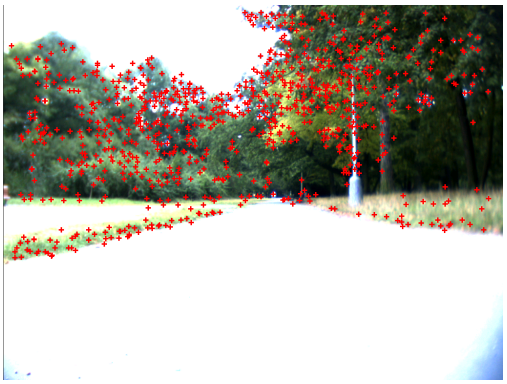
The proposed method was implemented on a P3AT robot with a Unibrain Fire-i601c camera, TCM2 compass and HP 8710p laptop, see Figure 1a. The camera was equipped with a 7 mm objective with electronically-driven iris to prevent sunlight dazzle. The laptop had Core2 Duo CPU running at 2.00GHz and 1 GB of memory. Image processing is computationally demanding and therefore additional UPC70 battery had to be used for longer experiments. Three robot batteries were replaced by one large, robot internal PC has been disabled in order to increase its action radius. Navigation system has been implemented in C/C++ as a standalone Linux application.

**Image processing** algorithm is a critical component of the navigation system. The vision system must provide enough information to steer the robot in a right direction. Furthermore, it should be robust to real

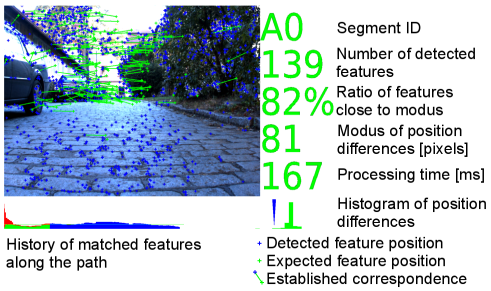
world conditions, i.e. changing illumination, minor environment changes and partial occlusions and of course its performance should allow real-time response.

We have decided to use to use Speeded Up Robust Features (SURF) (Bay et al., 2006) to identify landmarks in the image. The SURF method is reported to perform better than most SIFT (Se et al., 2001) implementations in terms of speed and robustness to viewpoint and illumination change. To achieve additional speedup the CPU implementation of the algorithm has been adjusted to use two cores for parallel image processing. The captured image is horizontally divided and both parts are processed in parallel. Later on, we switched to the GPU (Cornelis and Gool, 2008) version of the algorithm. The GPU version has better real-time performace, but is less distinctive than the CPU implementation (J. Svab and Preucil, 2009b). Normally, recognition of a 1024x768 grayscale image provides positions and descriptors of 150-300 features and takes 100-500 ms. Outdoor environment is usually richer in detected features and image processing tends to be slower than indoors. This algorithm provides image coordinates of salient features together with their descriptions. Typical outdoor processed image with highlighted feature positions is in Figure 1b.

(a) Robot configuration.



(b) Captured image and detected features.




(c) Robot GUI during navigation.

Figure 1: Robot platform, detected features and navigation GUI

### 2.2 Learning phase

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

The procedure which creates a map of one segment is shown in algorithm 1. Before the robot starts to

learn a segment, it resets its odometric counters and reads compass data to establish segment azimuth $\alpha$. After that the robot starts to move forwards, tracks detected features and puts them to set the $L$ until the operator requests to stop. During the movement, images are continuously captured and processed. For each currently tracked landmark $\boldsymbol{t_i}$ (from the set $T$) two the best matching features from set of new features are determined. 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 $T$ its $u$ and $v$ are set to the value of current traveled distance from the segment start and the counter of feature detection $k$ is set to one. At the end of the segment, segment description is saved and the operator can turn the robot to another direction and initiate mapping of another segment. A part of the file with segment description is shown in table 1.

**Algorithm 1**: Learn one segment

**Input**: $\alpha$ – initial robot orientation (compass value)

**Output**: $(\alpha, s, L)$ – associated data to segment, where $s$ is traveled distance and $L$ is set of landmarks, a landmark is senary $(k, e, u, v, f, g)$, where $e$ is a SURF descriptor, $k$ is counter of feature detection, $u$ and $v$ is position of feature in the image (at the moment of first, resp. last occurrence), $f$ and $g$ denotes distance from segment start according to $u$, resp. $v$.

$L \leftarrow \emptyset$ // set of learned landmarks
$T \leftarrow \emptyset$ // set of tracked landmarks
$\alpha \leftarrow$ *compass value* // robot orientation at the beginning of segment learning
**repeat**
  $d \leftarrow$ *current traveled 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 next the 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 current set of detected features
    **else**
      $T \leftarrow T \setminus \{\boldsymbol{t_i}\}$ // remove $\boldsymbol{t_i}$ from set of tracked landmarks
      $L \leftarrow L \cup \{\boldsymbol{t_i}\}$ // add $\boldsymbol{t_i}$ to set of learned landmarks
  **foreach** $(u, e) \in S$ **do**
    $T \leftarrow T \cup \{(e, 1, u, u, d, d)\}$ // add new feature to set of tracked landmark
**until** *robot is in the learning mode*
$s \leftarrow d$ // total traveled distance along segment
$L \leftarrow L \cup T$ // add current tracked landmarks to the set of learned landmarks

### 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 or not. The robot is placed at the start of the first segment, loads description of the segment and turns itself to the segment azimuth and starts moving forwards. Navigation procedure is shown in algorithm 2. Relevant landmarks for current robot position (distance from the segment start) are selected from the set of learned landmarks $L$. Similarly to the learning phase two the best matching features, which are distinguishable enough, for relevant landmarks are used as correspondence criterion. Corresponding features (learned landmark and new detected feature) are coupled. A difference in horizontal image coordinate of the features is computed for each such couple. A modus of those differences is estimated by histogram voting method. The modus is converted to a correction value of movement direction, which is reported to robot steering controller. After the robot travels distance greater or equal to the length of given

Table 1: Segment map a text file

| Record | value | meaning |
|---|---|---|
| Initial azimuth and length: | 2.13, 7.03 | $\alpha, s$ |
| Landmark 0: | | |
| First position: | 760.74, 163.29 | $\mathbf{u_{l_0}}$ |
| Last position: | 894.58, 54.44 | $\mathbf{v_{l_0}}$ |
| 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,... | $\mathbf{e_{l_0}}$ |
| Landmark 1: | | |
| First position: | 593.32, 381.17 | $\mathbf{u_{l_1}}$ |
| Last position: | 689.89, 377.23 | $\mathbf{v_{l_1}}$ |
| 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,... | $\mathbf{e_{l_1}}$ |

segment, the next segment description is loaded and the procedure is repeated. During navigation, the robot displays relevant states (mapped and recognized landmarks, recognition success ratio etc.) on its graphical interface, see Figure 1c.

**Algorithm 2**: Traverse one segment

**Input**: $(\alpha, s, L)$ – associated data to segment, where $\alpha$ is initial angle of robot orientation at segment start, $s$ is traveled distance and $L$ is set of landmarks, a landmark is senary $(e, k, u, v, f, g)$, where $e$ is a SURF descriptor, $k$ is counter of feature detection, $u$ and $v$ is position of feature in the image (at the moment of first, resp. last occurrence), $f$ and $g$ denotes distance from segment start according to $u$, resp. $v$.

**Output**: $\omega$ – steering controller input

*turn*$(\alpha)$ // turn robot in direction $\alpha$

$d \leftarrow$ *current traveled distance from the segment start*

**while** $d < s$ **do**

  $T \leftarrow \emptyset$ // set of current tracked landmarks

  $H \leftarrow \emptyset$ // set of differences (horizontal position in the image) of matched features

  $d \leftarrow$ *current traveled distance from the segment start*

  $S \leftarrow$ *extracted features with associated image position,* $(u, e) \in S$*, u position, e feature descriptor*

  **foreach** $\boldsymbol{l_i} = (e_i, k_i, u_i, v_i, f_i, g_i) \in L$ **do**

    **if** $f_i \geq d \geq g_i$ **then**

      $T \leftarrow T \cup \{\boldsymbol{l_i}\}$ // add landmark to tracked landmarks according to traveled distance

  **while** $|T| > 0$ **do**

    $(e_i, k_i, u_i, v_i, f_i, g_i) \leftarrow \text{argmax}_{\boldsymbol{t} \in T}\, k(\boldsymbol{t})$ // get landmark with maximal number of occurrences $k$

    $(u_a, e_a) \leftarrow \text{argmin}\{||e_i, e(s)|| \; | s \in S\}$ // 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 next the best matching descriptor

    **if** $||(e_i, e_a)|| \ll ||(e_i, e_b)||$ **then**

      $p \leftarrow \frac{|v_i - u_i|}{|g_i - f_i|}(d - f_i) + u_i - u_a$ // estimate angle to the matched landmark

      $H \leftarrow H \cup \{p\}$ // add horizontal difference to set of differences

    $T \leftarrow T \setminus \{(e_i, k_i, u_i, v_i, f_i, g_i)\}$ // discard used landmark

  $\omega \leftarrow \text{modus}(H)$ // determine new robot steering velocity

  *report* $\omega$ to steering controller

An important aspect of this navigation algorithm is that fact it does not need to localize the robot or to create a three-dimensional map of detected objects. 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 direction and the distance is measured by imprecise odometry, it is shown, that if the robot changes direction often enough, it will keep close to learned path. The stability of proposed navigation method is discussed in next section.

# 3 Stability of bearing-only navigation

First, we describe in an informal way how robot position uncertainty changes as the robot travels a closed path. That should help to interpret mathematical formalism describing robot position uncertainty in geometrical terms and make the following chapter more comprehensible. After that, we lay down a formal description of proposed navigation method and analyze its stability. We outline a model of robot movement, and depict equations allowing computation of robot position uncertainty. After that, we show how the robot position error changes after traversing a closed path. Finally, we examine properties of this model and establish conditions ensuring that 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 (2D Gaussian distribution with zero mean) position near the first segment start see Figure 2. The initial position uncertainty can be therefore displayed as a circle, in which the robot is found with (e.g.) 90% probability. The navigation procedure is switched on 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, due to the odometric error, the longitudal position error increases. At the end of the segment, the circle denoting position uncertainty therefore becomes an ellipse with the shorter axis perpendicular to the segment. Heading corrections are dependent on the lateral deviation (see chapter 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 influenced only by the length of the traversed segment and therefore is modeled as an additive error $o$.

After the segment is traversed, the robot turns by 90° and starts to move along the second segment. The uncertainty changes again, but because of the direction change, the longer ellipse axis shrinks and the shorter is elonged due to odometry error. As this repeats for every traversed segment, the size of the uncertainty ellipse might converge to a finite value. Since this particular trajectory is symmetric, axes lengths $a$,$b$ of the "final" ellipse can be easily computed by the equations (1)

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

where $h$ is a coefficient of lateral error reduction and $o$ is odometric error ($o = 1$, $h = 0.25$) on Figure 2. Though simple, this particular symmetric case gives us a basic insight into the problem. In the next chapter, we will derive a broader mathematical model of the navigation, examine its properties and show, that the uncertainty converges for other, nonsymmetrical trajectories as well.

## 3.2 Navigation

The proposed navigation method is based on the following assumptions:

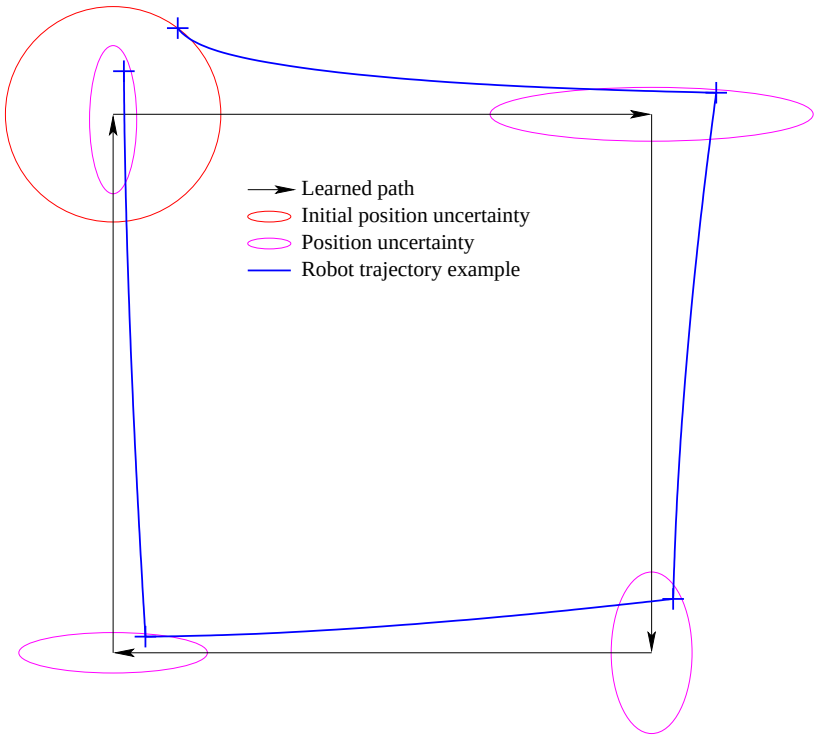



Figure 2: Position uncertainty evolution

- the robot moves in a plane,
- the map already exists in a form of a sequence of conjoined linear segments with landmark description,
- 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 an odometry,
- the camera is aimed in the direction of robot movement.

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

**Definition 1 (Closed path navigation property)** *Assume a robot using an environment map only for heading corrections, while measuring the distance by odometry navigates a closed path several times. Then path, for which the robot position uncertainty at any point is bound, has a closed path navigation property.*

**Theorem 1** *A path consisting of several conjoined noncolliear segments retains the closed path navigation property if conditions 3.2 are satisfied.*

### 3.3 Movement along one segment

At 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 robot position error at segment start to robot position error at segment end.

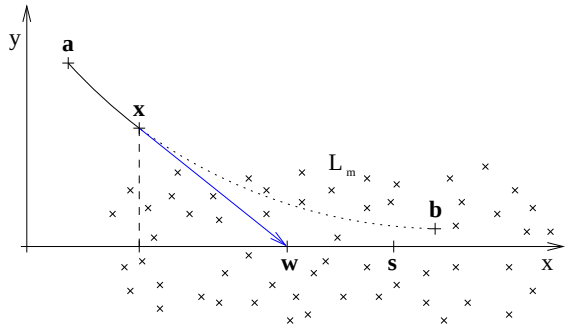



Figure 3: Navigation model for one segment

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 single segment of length $s$ with $m$ landmarks, with positions represented as vectors $\mathbf{u_i}$. Since 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 robot movement by a distance $\rho$. We can assume, that $\rho \approx \frac{1}{v}\sum_{i=0}^{v-1} u_x i$ where $v$ is the number of mapped landmarks. Let us place segment start at coordinate origin and segment end at position $[s, 0]^T$. We designate robot position prior to segment traversal as $\mathbf{a} = [a_x, a_y]^T$ and final robot position as $\mathbf{b} = [b_x, b_y]^T$, see Figure 3. Let us assume that at every moment during segment traversal the robot recognizes a non-empty subset $\mathbf{W}$ of previously learned landmarks and heads in a direction, which minimizes the horizontal deviation of expected and recognized positions. We denote intersection of robot heading with learned segment axis as $\mathbf{w}$. The position of $\mathbf{w} \approx [\rho, 0]^T$ at the beginning of segment traversal. As the robot traverses the segment, it loses sight of nearby landmarks and recognizes new ones. As new, more distant landmarks appear in robot view and nearby disappear, the set $\mathbf{W}$ changes and $\mathbf{w}$ moves along the segment. It can be assumed, that $\mathbf{w}$ moves approximately at the speed of the robot and therefore it is always ahead of the robot by $\rho$.

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

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

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

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

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

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

Considering, that the range of robots sensor is higher than robot position uncertainty and therefore $\rho \gg a_x$,

the constant $c \approx a_y$ and we can estimate robot position after traveling a segment of length $s$ by following equations:

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

We can transform (3) to matrix form

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

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

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

where $\upsilon$ is a random variable drawn from Gaussian distribution with zero mean and variance $\epsilon$.Consolidating previous equation (5), we can state

$$\mathbf{b} = \mathbf{M}\mathbf{a} + \mathbf{s}.$$

This movement model holds for a segment aligned with $x$-axis. For a segment with an arbitrary orientation $\alpha$, the movement model becomes more complex:

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

where

$$\mathbf{R} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \mathbf{M} = \begin{pmatrix} 1 & 0 \\ 0 & e^{-\frac{s}{\rho}} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & m \end{pmatrix}. \tag{6}$$

This corresponds to aligning the segment with x-axis, applying $\mathbf{M}$, adding odometric noise and rotating the segment back to direction $\alpha$. In the following text, we define $\mathbf{N} = \mathbf{R^T}\mathbf{M}\mathbf{R}$, which shortens equation (6) to

$$\mathbf{b} = \mathbf{N}\mathbf{a} + \mathbf{R^T}\mathbf{s}. \tag{7}$$

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

### 3.4 Position uncertainty

Now, the dependence of robot position uncertainty at segment end to its uncertainty at segment start can be examined. Consider, that robot position $\mathbf{a}$ before segment traversal is a random variable drawn from a two-dimensional normal distribution with mean $\hat{\mathbf{a}}$ and covariance matrix $\mathbf{A}$. To compute robot position uncertainty after segment traversal, we apply (7) to $\mathbf{A}$. Since robot movement model (7) has only linear and absolute terms, robot position uncertainty after segment traversal will constitute a normal distribution with mean $\hat{\mathbf{b}}$ covariance matrix $\mathbf{B}$.

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

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

We can claim, that:

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

Since $\tilde{\mathbf{s}}$ 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

$$\mathbf{B} = \mathbf{R}^{\mathbf{T}}\mathbf{M}\mathbf{R}\mathbf{A}\mathbf{R}^{\mathbf{T}}\mathbf{M}^{\mathbf{T}}\mathbf{R} + \mathbf{R}^{\mathbf{T}}\mathbf{S}\mathbf{R}, \tag{8}$$

where

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

Equation (8) allows us to compute 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 $i$th segment as $s_i$ and $\alpha_i$ respectively. Robot position before and after traversing $i$th segment is noted as $\mathbf{a_i}$ and $\mathbf{b_i}$ respectively. Since the robot position at the end of $i^{th}$ segment equals to its start position at segment $i+1$, we can state, that $a_{i+1} = b_i$.

The movement model (8) for $i^{th}$ traveled segment is

$$\mathbf{A_{i+1}} = \mathbf{B_i} = \mathbf{R_i^T}\mathbf{M_i}\mathbf{R_i}\mathbf{A_i}\mathbf{R_i^T}\mathbf{M_i^T}\mathbf{R_i} + \mathbf{R_i^T}\mathbf{S_i}\mathbf{R_i} \tag{9}$$

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(9) as follows:

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

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

$$\mathbf{A_i} = \left(\prod_{\mathbf{j=i-1}}^{\mathbf{0}} \mathbf{N_j}\right) \mathbf{A_0} \left(\prod_{\mathbf{j=0}}^{\mathbf{i-1}} \mathbf{N_j^T}\right) + \sum_{\mathbf{j=0}}^{\mathbf{i-1}} \left(\left(\prod_{\mathbf{k=i-1}}^{\mathbf{j}} \mathbf{N_k}\right) \mathbf{N_j^{-1}}\mathbf{T_j} \left(\mathbf{N_j^T}\right)^{-1} \left(\prod_{\mathbf{k=j}}^{\mathbf{i-1}} \mathbf{N_k^T}\right)\right). \tag{10}$$

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

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

and

$$\breve{\mathbf{T}} = \sum_{\mathbf{j=0}}^{\mathbf{n-1}} \left(\left(\prod_{\mathbf{k=n-1}}^{\mathbf{j}} \mathbf{N_k}\right) \mathbf{N_j^{-1}}\mathbf{T_j} \left(\mathbf{N_j^T}\right)^{-1} \left(\prod_{\mathbf{k=j}}^{\mathbf{n-1}} \mathbf{N_k^T}\right)\right) \tag{11}$$

and rewrite equation (10) as

$$\mathbf{C_{i+1}} = \breve{\mathbf{N}}\mathbf{C_i}\breve{\mathbf{N}}^{\mathbf{T}} + \breve{\mathbf{T}}. \tag{12}$$

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

### 3.6 Convergence conditions

Expression (12) is the Lyapunov discrete equation (Lyapunov, 1992). If all eigenvalues of $\breve{\mathbf{N}}$ lie within a unit circle and $\breve{\mathbf{T}}$ is symmetric, then $\lim_{i\to\infty} C_i$ is finite and equal to $C_\infty$, which can be obtained by solving

$$\mathbf{C}_\infty = \breve{\mathbf{N}}\mathbf{C}_\infty\breve{\mathbf{N}}^{\mathbf{T}} + \breve{\mathbf{T}}. \tag{13}$$

Since matrix $\mathbf{S_i}$ is constructed as diagonal, $\mathbf{T_i} = \mathbf{R_i^T S_i R_i}$ is symmetric. The product $\mathbf{X T_i X^T}$ is symmetric for any $\mathbf{X}$ and therefore all addends in (11) are symmetric. Addition preserves symmetry and therefore the matrix

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

is symmetric.

To prove that the eigenvalues of $\breve{\mathbf{N}}$ lie within a unit circle, we exploit positiveness of the matrices $\mathbf{M_i}$ and $\mathbf{R_i}$. Since $\mathbf{M_i}$ 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 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 $e^{-\frac{s_i}{\rho}}$. Since 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. Moreover, the maximum eigenvalue of the product $\mathbf{XY}$ is lower or equal than $xy$, where $x$ and $y$ are dominant eigenvalues of $\mathbf{X}$ and $\mathbf{Y}$. Since the dominant eigenvalue of every $\mathbf{N_i}$ is one, all eigenvalues of $\breve{\mathbf{N}}$ are smaller 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 existence of a finite solution of equation (13). Therefore, we have to find those conditions to support the 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 matrix product $\mathbf{N_{i+1}N_i}$ as $\lambda_{0,1}$ and the smaller eigenvalue of $\mathbf{N_i}$ as $n_i$ ($n_i = e^{-\frac{s_i}{\rho}}$). For our convenience, we denote $j = i + 1$. Therefore

$$\det\left(\mathbf{N_j N_i}\right) = \det \mathbf{N_j} \det \mathbf{N_i} = \lambda_0\lambda_1 = n_i n_j \tag{14}$$

If $\lambda_{0,1} \in < 0, 1 >$, we can state, that

$$(1 - \lambda_0)(1 - \lambda_1) \geq 0, \tag{15}$$

and therefore

$$\lambda_0\lambda_1 - \lambda_0 - \lambda_1 + 1 \geq 0. \tag{16}$$

Combining (14) and (16), we obtain

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

considering that the sum of eigenvalues equals matrix trace, we get:

$$trace\left(\mathbf{R_j^T M_j R_j R_i^T M_i R_i}\right) \leq 1 + n_i n_j. \tag{17}$$

Since $trace(\mathbf{AB}) = trace(\mathbf{BA})$, we rewrite (17) as

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

Matrices $\mathbf{R_i}$ and $\mathbf{R_j}$ are rotations, $\mathbf{R_i}$ denotes rotation by angle $\alpha_i$ and $\mathbf{R_j}$ denotes rotation by angle $\alpha_j$. Their product $\mathbf{R_i}\mathbf{R_j^T}$ denotes rotation by $\alpha_i - \alpha_j$. If we denote $\beta = \alpha_i - \alpha_j$, equation (18) is changed to

$$trace\left(\mathbf{M_j}\mathbf{R}_\beta^{\mathbf{T}}\mathbf{M_i}\mathbf{R}_\beta\right) \leq 1 + n_i n_j. \tag{19}$$

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

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

and

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

Inequality (19) can be rewritten to

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

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

$$(1 - n_i)(1 - n_j)\sin^2\beta \geq 0. \tag{20}$$

Since $n_i = e^{-\frac{s_i}{\rho}}$ and therefore $n_i \in (0,1)$ and $n_j \in (0,1)$, inequality (20) is sharp for $\sin\beta \neq 0$. This implies that inequality (15) is strict as well, which means that both $\lambda_0$ and $\lambda_1$ are lower than one. Both eigenvalues of the matrix product $(\mathbf{N_i}\mathbf{N_j})$ are therefore smaller than one if $\beta \neq n\pi | n \in \mathcal{N}$. Therefore, the matrix $\breve{\mathbf{N}}$ has both eigenvalues smaller than one if and only if any two conjoined segments of the path form an angle different from 0 or $\pi$.

Since all eigenvalues of $\breve{\mathbf{N}}$ lie within a unit circle and $\breve{\mathbf{T}}$ is symmetric, covariance matrix $\mathbf{C}_\infty$ denoting robot position uncertainty at the start of the first path segment is finite and obtainable by solution of 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 equations (6) describing movement of a robot equipped using a navigation method described before.This allowed us to examine robot position uncertainty evolution (9) as it travels through a known environment. Modifying equation (9) to closed trajectories, we could rewrite it as (12). By examining conditions, under which (12) has a finite solution, we have proven Theorem 1 for closed paths, which have at least two non-collinear segments.

# 4 Experiments

To verify theoretical assumptions formed in the chapter 3, real-world experiments were performed in three different scenarios. The first scenario examines stability of robot positions for simple line path and triangular path. Two next scenarios evaluate performance of the robot localization in a large outdoor environment. In particular, the second scenario is performed for fifty-meters long closed path that is traversed repeatedly and after each loop an error of the robot position is measured. The last testing scenario represents real deployment of proposed navigation procedure, experiments were performed during the RoboTour 2008 and RoboTour 2009 competitions. Experimental results for these three scenarios are consecutively presented in following sections. In all testing scenarios the P3AT platform with configuration described in chapter2 was used.

## 4.1 Stability of Robot Position for Line and Triangular Paths

The conclusions made in chapter 3 indicate, that the navigation is not suitable for paths with collinear segments only. To verify this assumption, we tested the navigation for two paths - first path consisted of two collinear segments (i.e. line path) and the second one was triangular, i.e. consisted of three noncolinear segments. In this scenario the robot was first learned both closed paths. After that it was placed close to the path start and requested to navigate along learned path for four times. The robot position was measured after each complete round relatively to start of the learned path. As mentioned, two types of paths were examined: line and triangular. The line path is formed of two collinear segments, one for each direction. At the end of each segment, a robot is turned about 180°. The triangular path is composed from three line segments, the end of the last segment is identical to start of the path. Both types of paths were tested in indoor and outdoor environment. Measured deviations of the robot position after each round are shown in tables 2 and 3.

Table 2: Indoor test results

**Line trajectory**

| **Loop no.** | **Position difference to start point [m]** | | |
|---|---|---|---|
| | **Round 1** | **Round 2** | **Round 3** |
| 0 | 0.00, 0.00 | -1.00, 0.00 | 0.00, 1.00 |
| 1 | -0.05, 0.07 | -0.95, 0.30 | -0.03, 0.03 |
| 2 | -0.07, 0.09 | -0.93, 0.38 | -0.05, 0.05 |
| 3 | -0.10, 0.10 | -0.93, 0.47 | -0.07, 0.07 |
| 4 | -0.13, 0.12 | -0.92, 0.47 | -0.14, 0.14 |

**Triangular trajectory**

| **Loop no.** | **Position difference to start point [m]** | | |
|---|---|---|---|
| | **Round 1** | **Round 2** | **Round 3** |
| 0 | 0.00, 0.00 | 1.00, 0.00 | 0.00, -1.00 |
| 1 | 0.08, -0.08 | 0.47, 0.14 | 0.02, -0.47 |
| 2 | 0.09, -0.10 | 0.26, 0.07 | 0.18, -0.19 |
| 3 | -0.05, 0.05 | 0.18, -0.08 | 0.03, -0.10 |
| 4 | -0.05, 0.05 | 0.08, -0.02 | 0.01, -0.07 |

Measured error indicates, that for triangular trajectories, the robot was able to correct position error 1 m that was introduced at the beginning of navigation along learned path. For line trajectories, only the error in $y$-coordinate (i.e. coordinate axis normal to all path segments) was partially corrected, while the $x$-coordinate

Table 3: Outdoor test results

**Line trajectory**

| Loop no. | Position difference to start point [m] | | |
|---|---|---|---|
| | **Round 1** | **Round 2** | **Round 3** |
| 0 | 0.00, 0.00 | -1.00, 0.00 | 0.00, -1.00 |
| 1 | 0.09, 0.08 | -0.98, 0.00 | -0.02, -0.76 |
| 2 | -0.23, 0.20 | -1.16, -0.16 | 0.03, -0.62 |
| 3 | -0.18, 0.26 | -1.25, -0.32 | 0.07, -0.35 |
| 4 | -0.11, 0.29 | -1.31, -0.32 | 0.15, -0.59 |

**Triangular trajectory**

| Loop no. | Position difference to start point [m] | | |
|---|---|---|---|
| | **Round 1** | **Round 2** | **Round 3** |
| 0 | 0.00, 0.00 | 1.00, 0.00 | 0.00, 1.00 |
| 1 | 0.12, -0.15 | 0.92, 0.33 | 0.32, 0.62 |
| 2 | 0.23, -0.16 | 0.59, 0.35 | 0.14, 0.22 |
| 3 | -0.15, -0.03 | 0.35, -0.12 | 0.05, 0.21 |
| 4 | -0.10, 0.03 | 0.23, -0.03 | -0.12, 0.10 |

slowly diverged. These experimental results confirm theoretical assumptions described in section 3.8, stating that navigation is unstable for paths with only collinear segments.

### 4.2 Closed Loop Navigation in Outdoor Environment

A method described in previous section was used to examine robot localization error in a large outdoor environment. The robot was learned a closed, square-like path approximately 50 m long. The path was learned in the Stromovka park located in the city of Prague. After a month, the robot was placed 150 cm away from the beginning of the path and requested to navigate the path fifty times. Every time the robot indicated path completion, its distance from the path origin was measured. Recorded values are plotted in Figure 4.

It is shown that the robot quickly reduced initial position error and then traversed the path faultlessly, the average position error is 21 cm, which is competitive with similar navigation configurations.

### 4.3 The RoboTour Outdoor Delivery Challenge

The RoboTour contest (Dlouhy and Winkler, 2009) is an autonomous robot delivery challenge organized by `robotika.cz` with participation of major universities of Czech Republic. The competition is a perfect event for an independent verification of the system functions and comparison with other navigation methods, however not only navigation methods is tested, but also complete system including all hardware parts.

Fully autonomous robots have to travel a random path in a park, stay on the pavements and detect randomly placed obstacles. A map of the park with pathways denoted 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 circular path and announce it as a sequence of letters. Competing teams place their robots on the beginning of the path and start their autonomous navigation algorithm. Robot score is determined according to its traveled distance. Robots must travel without leaving the pathway and without

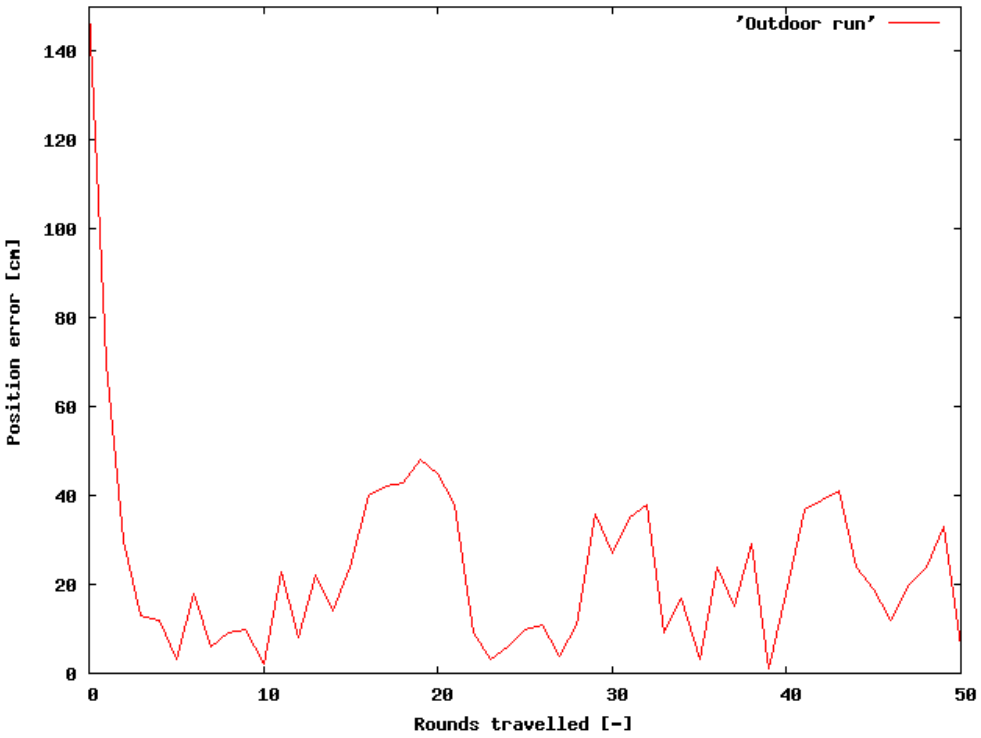



Figure 4: Position error

colliding with a random obstacle. In the year 2008, the competition was held in Stromovka park[1] in Prague, Czech republic. One year later, the contest moved to park Lužánky[2] in Brno, Czech republic.

The Stromovka park pathways have been mapped two days prior to the competition. The competition had five rounds with different pathways, see table 4. Out of these five attempts, the robot completed the required path four times. 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. The robot was left to continue moving (without additional scores) and reached the goal area. One failed attempt was caused by a battery failure.

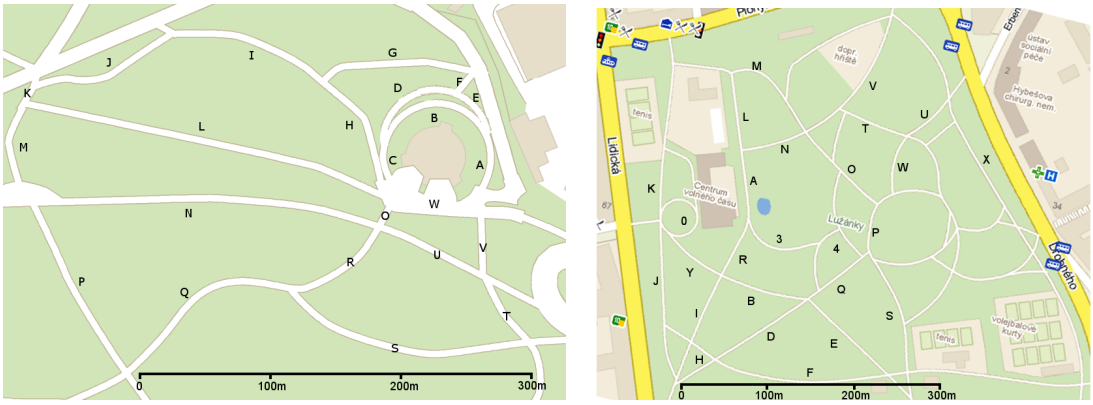



Figure 5: RoboTour 2008/2009 pathway maps

[1]Kralovska obora aka Stromovka50°6'18.778"N, 14°25'33.395"E 50°4'53.579"N
[2]Lužánky park 49°12'25.516"N, 16°36'29.81"E

Table 4: RoboTour 2008/2009 paths

| Pathway sequence | length [m] | Pathway sequence | length [m] |
|---|---|---|---|
| ABCW | 250 | ALK0JIR | 800 |
| ORQPMKJIH | 850 | BESQDGHJ0Y | 1050 |
| ABCHGFDCW | 500 | 34PWTMLA | 750 |
| HGFEAWOUVW | 600 | 0YR34SFHJ | 600 |
| UTSROCDEBCW | 700 | | |
| Total | 2 900 | Total | 2 200 |

The competition in the Lužánky park was performed in larger part of the environment, hence and mapping took three days. The total length of mapped pathways was 8 km. The competition had four rounds, see table 4. Our robot was able to complete the required paths two times. One attempt failed due to wrong compass reading, but after manually correction of the robot heading, the robot caught up and reached the goal area. The other failed attempt was caused by human factor.

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

# 5 Conclusion

A simple navigation method based on bearing-only sensors and odometry was presented. A robot navigating known environment by this method uses the map and camera input to establish its heading, while measuring the traveled distance by an odometry. We claim, that this kind of navigation is sufficient to keep robot position error limited for a certain set of trajectories. By modeling robot position uncertainty, this claim is formulated as convergence property and proved for a specified class of paths. The property allows to estimate robot position uncertainty upper bound based on landmark density, robot odometry precision and path shape.

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

We have 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, e.g

- the robot can perform 2D localization by heading estimation, which is 1-DOF method,
- if the robot travels between two points, its better to use a "zig-zag" trajectory rather than straight one,
- traveling a closed trajectory can reduce robot position uncertainty.

# Acknowledgements

We would like to thank our colleagues for valuable remarks and friends for help with outdoor tests. This work was supported by the Research program xxxxxxxxxxxxxxxxx.

**References**


Bailey, T. (2003). Constrained initialisation for bearing-only SLAM. In *2003 IEEE International Conference on Robotics and Automation, vols 1-3, Proceedings*, IEEE International Conference ON Robotics AND Automation, pages 1966–1971. 20th IEEE International Conference on Robotics and Automation (ICRA), Taipei, Taiwan, Sep 14-19, 2003.

Bay, H., Tuytelaars, T., and Gool, L. (2006). Surf: Speeded up robust features. In *Proceedings of the ninth European Conference on Computer Vision*.

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

Chen, Z. and Birchfield, S. T. (2006). Qualitative vision-based mobile robot navigation. In *2006 IEEE International Conference on Robotics and Automation (ICRA), vols 1-10*, IEEE International Conference on Robotics and Automation, pages 2686–2692, 345 E 47TH ST, NEW YORK, NY 10017 USA. IEEE, IEEE. IEEE International Conference on Robotics and Automation (ICRA), Orlando, FL, May 15-19, 2006.

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

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

Dissanayake, M., Newman, P., Clark, S., Durrant-Whyte, H., and 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. and Winkler, Z. (2009). Robotour outdoor delivery challenge.

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

Gibbens, P., Dissanayake, G., and Durrant-Whyte, H. (2000). 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, vols 1-5*, IEEE Conference on Decision and Control - Proceedings, pages 191–196. 39th IEEE Conference on Decision and Control, Sydney, Australia, Dec 12-15, 2000.

Holmes, S., Klein, G., and Murray, D. W. (2008). A Square Root Unscented Kalman Filter for visual monoSLAM. In *2008 IEEE International Conference ON Robotics AND Automation, VOLS 1-9*, IEEE International Conference ON Robotics AND Automation, pages 3710–3716. IEEE International Conference on Robotics and Automation, Pasadena, CA, MAY 19-23, 2008.

J. Svab, T. K. and Preucil, L. (2009a). Fpga-based speeded up robust features. In *IEEE International Conference on Technologies for Practical Robot Applications 2009*.

J. Svab, T. K. and Preucil, L. (2009b). Fpga-based speeded up robust features. In *2009 IEEE International Conference on Technologies for Practical Robot Applications 2009*.

Julier, S. and Uhlmann, J. (2001). A counter example to the theory of simultaneous localization and map building. In *2001 IEEE International Conference on Robotics and Automation, vols I-IV, Proceedings*, IEEE International Conference on Robotics and Automation, pages 4238–4243. IEEE International Conference on Robotics and Automation, Seoul, South Korea, May 21-26, 2001.

Kalman, R. E. (2005). A new approach to linear filtering and prediction problems.

Kidono, K., Miura, J., and Shirai, Y. (2000). Autonomous visual navigation of a mobile robot using a human-guided experience. In *Proceedings of 6th Int. Conf. on Intelligent Autonomous Systems*, pages 620–627.

Leonard, J. J., Rikoski, R. J., Newman, P. M., and Bosse, M. (2002). Mapping partially observable features from multiple uncertain vantage points. *International Journal of Robotics Research*, 21:943–975.

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

Martinelli, A., Tomatis, N., and Siegwart, R. (2005). Some results on SLAM and the closing the loop problem. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vols 1-4*, pages 334–339. IEEE/RSJ International Conference on Intelligent Robots and Systems, Edmonton, Canada, Aug 02-06, 2005.

Matsumoto, Y., Inaba, M., and Inoue, H. (1996). Visual navigation using view-sequenced route representation. In *Proceedings of the International Conference on Robotics and Automation*, pages 83–88, Minneapolis, USA.

Mourikis, A. I. and Roumeliotis, S. I. (2004). Analysis of positioning uncertainty in simultaneous localization and mapping (SLAM). In *Proceedings of the IEEE/RSJ International Conference on Robotics and Intelligent Systems (IROS)*, pages 13–20, Sendai, Japan.

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

Se, S., Lowe, D., and Little, J. (2001). Vision-based mobile robot localization and mapping using scale-invariant features. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2051–2058, Seoul, Korea.

Zhang, A. M. and Kleeman, L. (2009). Robust appearance based visual route following for navigation in large-scale outdoor environments. *Int. J. Rob. Res.*, 28(3):331–356.