

Towards Efficient and Scalable Visual Homing

Journal Title
XX(X):2–43
©The Author(s) 2016
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/ToBeAssigned
www.sagepub.com/


Annett Stelzer¹, Mallikarjuna Vayugundla¹, Elmar Mair¹, Michael Suppa², Wolfram Burgard³

Abstract

Visual homing describes the ability of a robot to autonomously return to its starting position along a previously traversed path using visual information. In this paper, we propose a method for visual homing that is solely based on bearing angles to landmarks. During the first traversal of a path, the robot creates a sequence of viewframes, which are rotationally aligned landmark angle configurations at certain locations. During homing, the robot calculates homing vectors which subsequently lead it in the direction to align the currently perceived set of landmark observations with the reference viewframe until the home location is reached. This paper discusses methods for homing vector calculation and proposes new methods which are more robust and yield straighter homing paths in case of non-isotropic landmark distributions and false landmark matches. Furthermore, we present the Trail-Map, which is a novel data structure for storing a sequence of viewframes in a non-redundant and scalable way. The Trail-Map exploits the fact that the bearing angles to distant landmarks and landmarks in the direction of movement hardly change when the robot moves, whereas close landmarks change their bearing angles quickly. Thus, the Trail-Map allows easy downscaling by deleting observations that correspond to nearby, quickly-changing landmarks and, thus, retaining the stable, translation invariant landmark information. We show the memory efficiency and scalability of the data structure in simulations and in real-world indoor and outdoor experiments. This makes the proposed method for visual homing suitable for mobile robots with limited computational and memory resources.

Keywords

visual homing, range-free navigation, omnidirectional vision

1 Introduction

In recent years, mobile robots have become part of our daily lives. They vacuum our floors, mow our lawns and clean our windows. While these robots work in well-defined environments and perform well-defined and simple tasks fully autonomously, mobile robots are also demanded to work in unstructured and unpredictable areas. For example, the exploration of foreign planets, mines, or disaster sites poses high risks to humans, which drives the demand for mobile robots as robust tools to support scientists or rescue workers. In such scenarios, full robot autonomy is neither required nor desired, because the decision of what places are interesting and what information to record is better to be made by a human, who can rapidly respond to unforeseeable events (Murphy (2014)). Rather, the robot should be a tool which supports the human operators by providing an overview of the situation, which is necessary for further decisions and task planning. For this, the robot should provide basic autonomy, such as obstacle avoidance, autonomous navigation to a given goal location, and autonomous homing, i.e. returning to the start location once the operator has finished the task or in case objects or samples have to be returned to a base. Robots which are applicable to this task usually have very limited computational resources and limited memory, because they have to be small and agile for rough terrain locomotion, or they have to use space-qualified hardware for planetary exploration. For this reason, the robot's autonomous skills have to be implemented in a very efficient way.

This manuscript aims at the development of an efficient method for visual homing. The term *homing* is borrowed from biology, where it describes the ability of insects to return to their nests after foraging. This special navigation task comprises learning and retracing a path. The robot memorizes a path while either being remotely controlled by an operator, or while autonomously navigating to intermediate waypoints given by the operator. For this, it uses an omnidirectional camera for observing the surrounding visual panorama. Once the robot is commanded to return to its base, it uses the memorized information for retracing the path. To allow for long-range homing, the stored path should be scalable. For this, the robot should store all information as long as possible, but then in case of memory shortage forget non-crucial path information to extend its motion range. This should only affect the homing accuracy, but should still enable the robot to reach its home position. Furthermore, considering the limited computational resources of the robot, it is important that the computational requirements of the homing method stay constant with respect to the length of the travelled path.

In literature, techniques for retracing learned paths can be divided into appearance-based and feature-based navigation approaches. In *appearance-based* navigation

¹German Aerospace Center (DLR), Institute of Robotics and Mechatronics, Oberpfaffenhofen, Germany

²Roboception GmbH, Munich, Germany

³University of Freiburg, Department of Computer Science, Freiburg, Germany

Corresponding author:

Annett Stelzer, Institute of Robotics and Mechatronics, DLR German Aerospace Center, Münchener Str. 20, 82234 Oberpfaffenhofen, Germany.

Email: annett.stelzer@dlr.de

methods, the robot memorizes full images or special image properties in a topological map (Tapus et al. (2006)) during a training phase and then navigates by matching the stored information with the current view (Matsumoto et al. (1996); Kosecka et al. (2003); Vardy (2006); Zhang and Kleeman (2009)). *Feature-based* approaches only store the configurations of landmarks in the environment at certain places. Cartwright and Collett (1983, 1987) developed the *snapshot* model based on experiments with honey bees. This model stores the perceived landmark configuration at a certain location in a so-called snapshot, which contains the bearing angles and the sizes of the landmarks projected on the insect's retina. For homing, the robot computes the direction that matches the current landmark configuration with the stored snapshot. Dai and Lawton (1993) introduced the term *viewframe*, which consists of a set of landmarks and their corresponding bearing angles as they are observed from a certain location. Kawamura et al. (2002) transferred the viewframe concept into 3-dimensional space by describing distinct places by the projection of landmarks on the surrounding egosphere. Other works on retracing paths also make use of features, but are not explicitly inspired by insect navigation models. Argyros et al. (2005) proposed robot homing by using only angular information of visual features in panoramic images. The robot tracks image corners to build up a visual memory containing the life cycle of all features. For homing, the robot selects intermediate milestone positions that allow tracking of at least three image features in-between. Then, it employs a local control strategy to subsequently move to the milestone positions until the home position is reached. Goedemé et al. (2005) presented a method for visual path following of an automatic wheelchair based on sparsely captured omnidirectional images of the environment. Local 2D maps of image features are created by triangulation and corrected using a SLAM approach, while the robot performs a homing motion to the location of the goal. Šegvić et al. (2007) introduced a hierarchical environment representation, which contains a graph of key images with extracted 2D features at the top level, and local 3D reconstructions at the bottom level. The information in the top level enables robust navigation by visual servoing, while the bottom level is used for predicting feature locations to support tracking. Using this approach, the robot can cover large distances without a consistent reconstruction of the environment. Furgale and Barfoot (2010b) proposed *Visual Teach and Repeat*, which enables a robot to follow a taught path over several kilometers. In this work, overlapping feature submaps are created during the teach pass which are used for localization in the repeat pass. Global consistency is not enforced, since local consistency is sufficient for the task. This approach requires about 348MB of data per kilometer on average. In the method introduced by Krajník et al. (2010) the robot learns straight line sequences, where each segment is associated with a landmark map, the initial orientation of the robot and the segment length. For homing, the robot uses the landmark map only for correcting its current heading, but then moves straight until it has traversed the segment according to its odometry measurements. Krajník et al. (2010) reported that this method required 848MB for a run of 8km length, which means an average of 106MB per kilometer. Cherubini and Chaumette (2013) proposed a method in which the robot stores a sequence of key images along its path, such that subsequent images contain common static features. For repeating the path, the robot extracts and matches common features in the current

and the goal image and moves to align the x-coordinates of the centroids of the feature point clouds.

All works mentioned above fail to give information about how to scale the resulting maps and how to efficiently organize the information to save memory and computation time. To our knowledge, the first work addressing the problem of scalability and memory efficiency for feature-based homing is the *Landmark Tree-Map (LT-Map)* developed by Augustine et al. (2012). The LT-Map organizes landmark views in a tree so that slowly changing, translation invariant landmarks are located near the root of the tree while translation variant landmarks are located in the leaves. The LT-Map can be scaled by pruning the tree and, thus, discarding the information about quickly changing landmarks.

In this paper, we develop a constant-time and memory efficient visual homing method based on the idea of the LT-Map. This method is independent of any distance information, but solely relies on bearing angle measurements to landmarks, which are extracted from an omnidirectional sensor. The robot memorizes the landmark bearing configurations, so-called *viewframes*, of certain locations and stores them in the Trail-Map (*Translation Invariance Level Map*), which is a novel, non-redundant data structure that can easily be pruned in case of memory shortage and outperforms the LT-Map. For homing, the robot retrieves the viewframes and computes homing vectors which subsequently lead the robot to the previous viewframe until the original home position is reached. Using this approach the robot can reliably retrace long-range paths without the need to maintain a metrically correct Cartesian map. Thus, the method runs in constant time independent of the length of the path and is suitable for robots with limited computational resources.

The paper is organized as follows: In the next section, we will explain the concept of viewframe-based homing and introduce viewframe dissimilarity measures and homing vector calculation methods. Section 3 introduces the Trail-Map, a novel, scalable data structure for viewframe-based homing. Section 4 gives simulation results of homing using the Trail-Map and compares it with the LT-Map data structure. In Section 5 we explain what is necessary to apply the Trail-Map-based homing method to a real robot and Section 6 gives experimental results in indoor and outdoor real-world environments. Section 7 concludes this paper.

2 Viewframe-Based Homing

A viewframe (VF) is defined as the configuration of landmark views which corresponds to a certain location in two- or three-dimensional Euclidean space (ref. Fig. 1). Each landmark view (LV) contains the landmark's ID, its descriptor and its bearing angle containing the azimuth $\phi_{i,a}$ (and elevation $\phi_{i,e}$ in the 3D case) under which the landmark L_i is observed. The landmark views are extracted from omnidirectional images. All viewframes are assumed to be rotationally aligned with each other, either using compass information or an estimated orientation. The unit vector pointing in the

direction of landmark L_i is l_i and can be computed from $(\phi_{i,a}, \phi_{i,e})$ as

$$l_i = \begin{bmatrix} \cos \phi_{i,e} \cos \phi_{i,a} \\ \cos \phi_{i,e} \sin \phi_{i,a} \\ \sin \phi_{i,e} \end{bmatrix}. \quad (1)$$

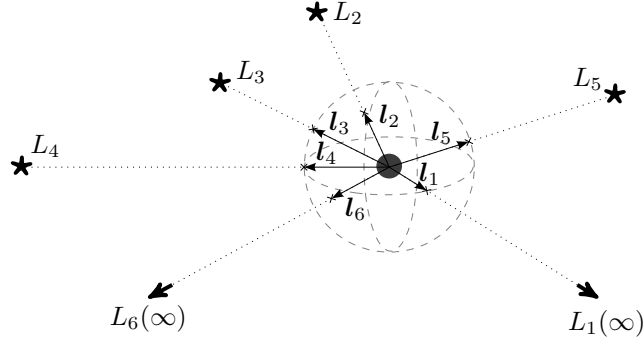


Figure 1. Illustration of a viewframe (adapted from Mair et al. (2014)). L_i are the landmarks and l_i the unit vectors pointing to them.

In a nutshell: The robot records viewframes during a mapping phase while it explores unknown regions either autonomously or remotely controlled by an operator. A *dissimilarity measure* is required to decide when a new viewframe is recorded. At some point, the robot is commanded to return to its starting position. For this, it has to compute *homing vectors* that successively give the direction to the previously recorded viewframe until the viewframe corresponding to the starting position is reached. The robot decides whether a viewframe is reached based on another dissimilarity measure, which can be, but not necessarily has to be the same measure as for viewframe recording. More details are provided in the following.

2.1 Viewframe Dissimilarity Measures

In the mapping phase the robot has to detect when the current view becomes significantly different from the viewframe that it previously recorded. Additionally, during homing it is important for the robot to recognize a known place or to know when the desired goal viewframe is reached. Thus, a measure of viewframe dissimilarity has to be computed.

There are different ways for computing a dissimilarity measure δ_{diss} . In this paper, we use the average angle between the N corresponding landmark unit vectors l'_i of the current view and l_i of the goal viewframe:

$$\delta_{\text{diss}}^{\text{ang}} = \frac{1}{N} \sum_{i=1}^N \text{acos}(l_i^T l'_i). \quad (2)$$

When using this measure, large angle changes of nearby landmarks can be compensated by a high number of distant landmarks. To prevent this, the k th maximum

value of all difference angles between corresponding unit vectors can be used:

$$\delta_{\text{diss}}^{\text{max}} = k\text{th-max} \left\{ \text{acos}(\mathbf{l}'_i \mathbf{l}_i) \right\}, \quad (3)$$

where the value for k can be set according to the number of corresponding landmarks, for example as a ratio, which would result in a percentile rank. As a disadvantage, this measure is strongly affected by noise in the angle measurements and by false landmark matches. Thus, we will use the k th maximum dissimilarity measure only in simulations with perfect landmark observations, and switch to the average angle measure for real world experiments.

A new viewframe is acquired when the dissimilarity measure of the current view compared to the previously recorded viewframe exceeds the threshold $\xi_{\delta_{\text{diss}}}$. Thus, the density of the viewframes depends on the local landmark configuration: In areas with only few close landmarks, the viewframes will be further apart than in areas with many nearby landmarks. That leads to an implicit adaption of the map resolution to the local conditions. The smaller the threshold $\xi_{\delta_{\text{diss}}}$, the higher the resolution and, hence, the accuracy and the memory requirements of the resulting map. The threshold should be chosen according to the measurement accuracy of the bearing sensor and to the required path accuracy. Since usually no high accuracy is required for the traversal between different workspaces, higher thresholds should be preferred for the benefit of less memory.

2.2 Homing Vector Calculation Methods

To move from the current position to a goal viewframe, the moving direction has to be computed from the landmark angle information that the robot currently perceives compared to the stored configuration in the viewframe. The resulting direction is usually represented by a vector, called the *homing vector*.

In literature, different methods for calculating homing vectors from the current view to a goal view have been proposed. Apart from image-based methods, where whole images are compared for homing vector calculation (Franz et al. (1998); Zeil et al. (2003)), we will focus on the landmark-based methods for homing vector calculation.

The homing vector calculation method of the *snapshot model* (Cartwright and Collett (1983)) is based on dark and bright sectors on the insect's retina caused by projections of the surrounding landmarks. Several modifications of this method were proposed by Lambrinos et al. (2000), e.g. the proportional vector model, the average landmark vector (ALV) model and the difference vector model. However, all these methods assume that the projection of a landmark on the retina results in a bright or dark sector. Furthermore, the landmark is assumed to have a perceivable size that only changes with the distance from the landmark but not with the bearing the landmark is perceived at. That is only true for cylindrical objects which can clearly be distinguished from the background. This is not always the case in real world scenarios. Landmarks that are detected using feature detection algorithms appear as characteristic points in images. They can have a scale but they usually do not have a size which is clearly perceivable.

Considering landmarks as point image features without a perceivable size, the difference vector model proposed by Lambrinos et al. (2000) can be adjusted to

calculate the homing vector \mathbf{h} as

$$\mathbf{h} = \frac{1}{N} \sum_{i=1}^N (\mathbf{l}'_i - \mathbf{l}_i) , \quad (4)$$

where \mathbf{l}'_i are the unit landmark vectors in the current view and \mathbf{l}_i are the corresponding unit landmark vectors of the goal view. Here, the sum of the difference vectors is normalized by the number of corresponding landmarks N .

In this model, the difference vectors are always secants of the unit circle around the current viewframe. For the method to work well, it is assumed that the landmarks are distributed isotropically around each viewframe and that a 360° panorama of the scene is taken. Then, errors in the orthogonal direction to the homing vector cancel each other out. When landmarks are located only in one direction of the viewframe, errors cannot cancel out. Hence, the homing vector is biased, especially when the landmarks appear only in the homing direction. This behavior is shown in Fig. 2a, where the homing vectors near the connecting line between the landmark cluster and the home position are approximately perpendicular to the desired homing direction. This leads to zigzag-like viewframe approaching behaviors (ref. Fig. 4a). The case of having a landmark cluster in the direction of motion must be considered when downscaling the Trail-Map data structure, which will be introduced in the next section.

Creating radial homing vector components using the apparent size of the landmark would solve this problem. However, when landmarks are assumed to be points, they do not have an apparent size. In that case, the apparent width of an imaginary landmark between two landmark observations can be used to achieve more direct homing vectors. We propose the improved difference vector model, which uses the angles between two landmarks and the robot as apex to generate radial homing vector components. When moving towards two landmarks, the angle between them increases. Hence, a component x_i in the positive direction of the bisecting line of the two landmarks is added when the angle between those landmarks in the goal viewframe is greater than in the current viewframe. Otherwise, the component is subtracted:

$$\mathbf{h} = \frac{1}{N} \sum_{i=1}^N (\mathbf{l}'_i - \mathbf{l}_i) + \frac{1}{N-1} \sum_{i=1}^{N-1} \frac{(\mathbf{l}'_i + \mathbf{l}'_{i+1})}{2} x_i \quad (5)$$

$$x_i = \left| \text{acos}(\mathbf{l}'_i{}^T \mathbf{l}'_{i+1}) \right| - \left| \text{acos}(\mathbf{l}_i{}^T \mathbf{l}_{i+1}) \right| = \beta_{i,i+1} - \beta'_{i,i+1}.$$

The construction of a homing vector with and without the angle differences is illustrated in Fig. 3 for an environment with only two landmarks.

The resulting streamline and angle deviation plot is shown in Fig. 2b. The homing vectors have now improved when only landmarks in the direction of movement are available as shown in Fig. 4b. However, the width of an imaginary landmark not only decreases when the robot travels away from it, but also when the imaginary landmark is perceived at a flat angle. Hence, in these areas the resulting homing vectors are still erroneous.

We propose another variant of the difference vector model, called the normalized difference vector model. In contrast to the original difference vector model, the homing

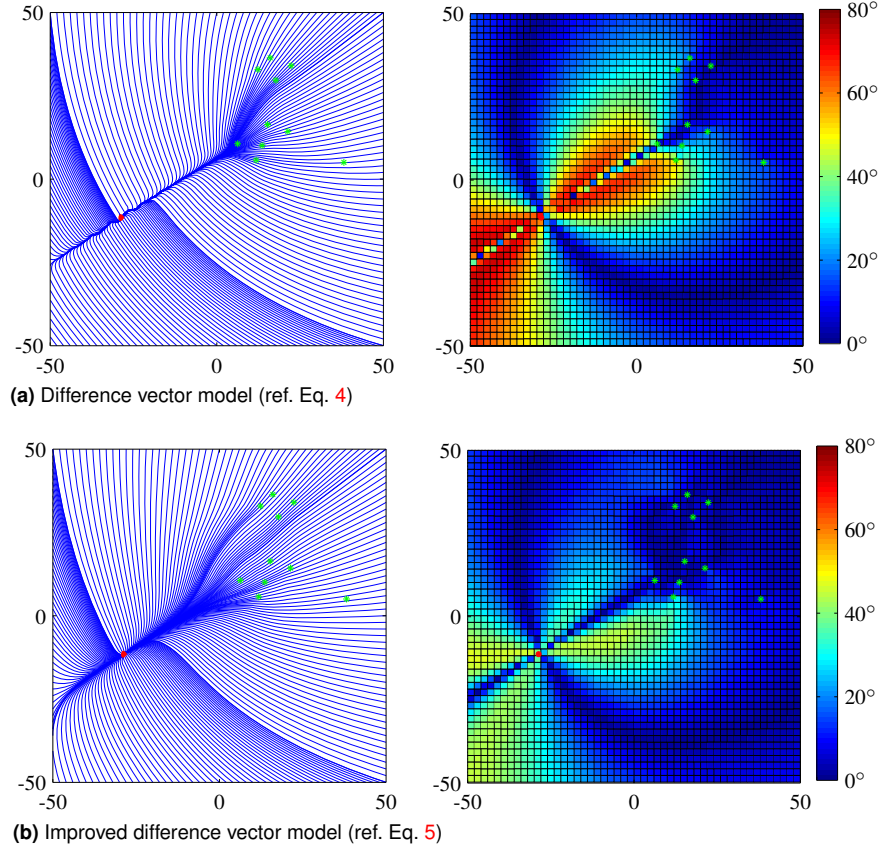


Figure 2. Homing vector streamlines (left) and angle deviations from the direct path (right) for the difference vector model and improved difference vector model with a landmark cluster (green asterisks). The red circle denotes the goal location. The x and y coordinates are given in units.

vector is calculated by summing the normalized difference vectors, as

$$\mathbf{h} = \frac{1}{N} \sum_{i=1}^N \frac{(\mathbf{l}'_i - \mathbf{l}_i)}{|\mathbf{l}'_i - \mathbf{l}_i|}. \quad (6)$$

The same kind of normalization can be applied to the improved difference vector model:

$$\mathbf{h} = \frac{1}{N} \sum_{i=1}^N \frac{(\mathbf{l}'_i - \mathbf{l}_i)}{|\mathbf{l}'_i - \mathbf{l}_i|} + \frac{1}{N-1} \sum_{i=1}^{N-1} \frac{(\mathbf{l}'_i + \mathbf{l}'_{i+1})}{|\mathbf{l}'_i + \mathbf{l}'_{i+1}|} \text{sign}(x_i) \quad (7)$$

$$x_i = \left| \text{acos}(\mathbf{l}'_i{}^T \mathbf{l}_{i+1}) \right| - \left| \text{acos}(\mathbf{l}'_i{}^T \mathbf{l}'_{i+1}) \right| = \beta_{i,i+1} - \beta'_{i,i+1}.$$

The advantage of the normalization becomes apparent when false landmark matches occur. Since false landmark matches are likely to yield unit vectors pointing in a

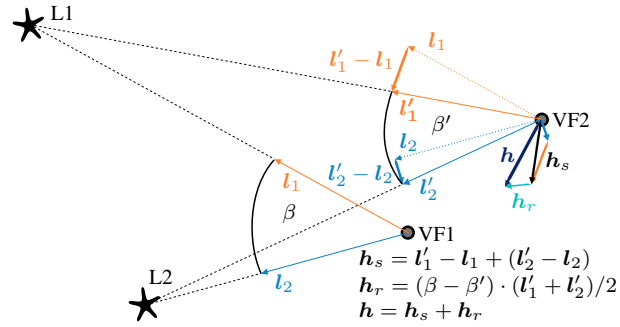


Figure 3. Homing vector construction (adapted from Stelzer et al. (2014)): h_s : homing vector using difference vector model with only secant components. h_r : radial component from angle differences. h : homing vector using improved difference vector model

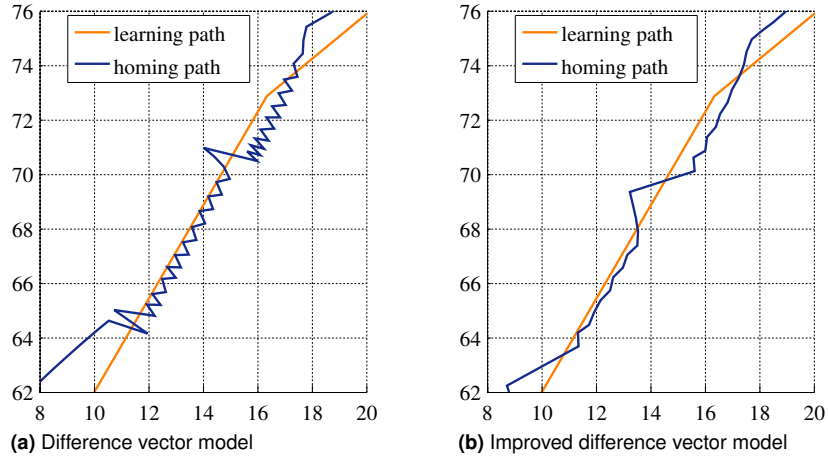


Figure 4. Resulting homing paths using the difference vector model and the improved difference vector model (adapted from Stelzer et al. (2014)). The x and y coordinates are given in units.

very different direction than the unit vector corresponding to the true match, the resulting difference vectors are often large. Thus, these false matches have a very strong influence on the homing vector direction. This influence is decreased when all difference vectors are used in the normalized form. Fig. 5a shows the streamlines and deviations of the homing vectors computed by the improved difference vector model when 10% landmark outliers are present. In this case, homing would most probably fail when the robot starts at the lower left corner of the plot. In contrast, when the normalized version is used, the homing vectors are more stable in the presence of outliers (ref. Fig. 5b).

Other authors also proposed methods for homing vector calculation. Weber et al. (1999) presented a method for homing vector calculation based on tangential correction

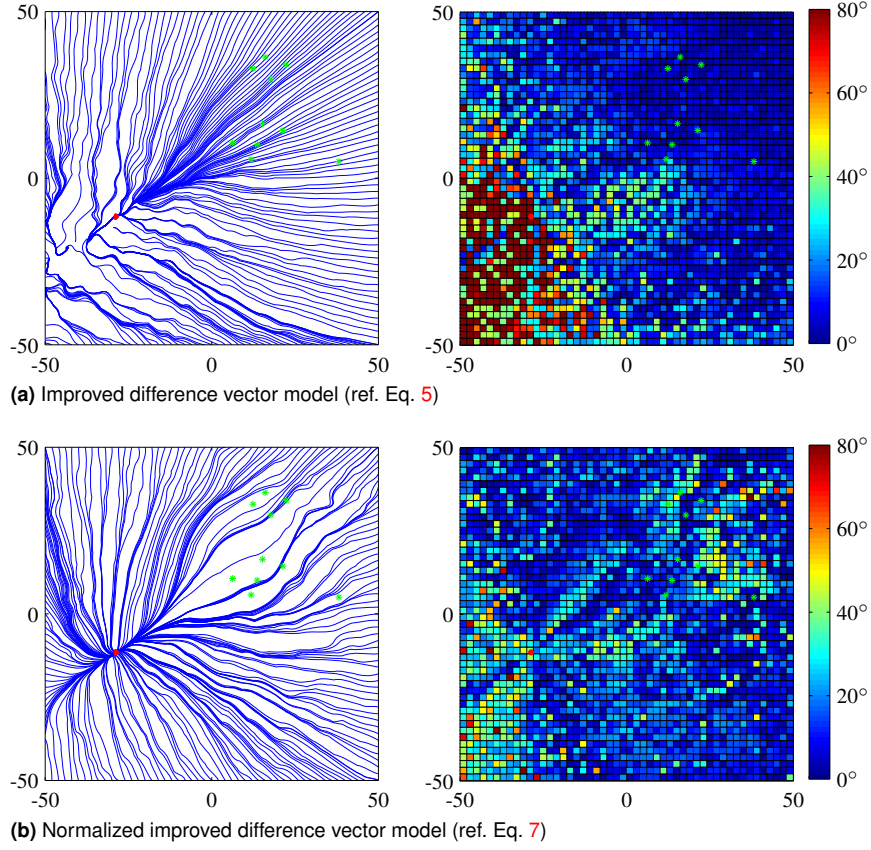


Figure 5. Homing vector streamlines (left) and angle deviations from the direct path (right) for the improved difference vector model and the normalized improved difference vector model with a landmark cluster (green asterisks) and 10% outliers. The red circle denotes the goal location. The x and y coordinates are given in units.

vectors that are perpendicular to the current landmark bearing ϕ'_i and proportional to the difference between the current and the goal bearing ϕ_i as

$$\mathbf{h} = \sum_{i=1}^N |\phi_i - \phi'_i| \mathbf{l}'_{i\perp} \quad \text{with} \quad \mathbf{l}'_{i\perp} = \begin{cases} \mathbf{R}(90^\circ) \mathbf{l}'_i & \text{if } \phi_i < \phi'_i \\ \mathbf{R}(-90^\circ) \mathbf{l}'_i & \text{if } \phi_i \geq \phi'_i \end{cases} \quad (8)$$

and $\mathbf{R}(\alpha) = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix}$.

This homing vector calculation method leads to a landmark avoiding behavior. This is beneficial when the landmarks are also obstacles, which is not the case when using image features as landmarks, since image features can also be detected on flat, but textured ground.

Liu et al. (2010) introduced an image-based visual servoing method for homing vector calculation based on bearing-only landmark information in 3D. It uses the angles β_i formed between the unit vectors \mathbf{l}'_i and \mathbf{l}'_{i+1} to the landmarks L_i and L_{i+1} to generate homing vector components along the bisecting lines of these angles. The homing vector is then calculated as

$$\mathbf{h} = \sum_{i=1}^N \mathbf{v}_{P_i} \mathbf{l}'_i \quad (9)$$

$$\mathbf{v}_{P_i} = -2 \begin{pmatrix} \cos(\frac{\beta'_i}{2}) & 0 & 0 & \dots & 0 & \cos(\frac{\beta'_N}{2}) \\ \cos(\frac{\beta'_1}{2}) & \cos(\frac{\beta'_2}{2}) & 0 & \dots & 0 & 0 \\ 0 & \cos(\frac{\beta'_2}{2}) & \cos(\frac{\beta'_3}{2}) & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & \cos(\frac{\beta'_{N-1}}{2}) & \cos(\frac{\beta'_N}{2}) \end{pmatrix} \begin{pmatrix} \beta'_1 - \beta_1 \\ \beta'_2 - \beta_2 \\ \beta'_3 - \beta_3 \\ \dots \\ \beta'_N - \beta_N \end{pmatrix}$$

$$\beta_i = \text{acos}(\mathbf{l}_i^T \mathbf{l}_{i+1}) \quad \text{and} \quad \beta'_i = \text{acos}(\mathbf{l}'_i{}^T \mathbf{l}'_{i+1}) \quad \text{for} \quad i < N$$

$$\beta'_N = \text{acos}(\mathbf{l}'_N{}^T \mathbf{l}'_1).$$

To compare the performance of the discussed homing vector calculation methods, we defined a reference scenario consisting of an environment with a general landmark distribution that is neither isotropic nor totally clustered. We added angular measurement noise with a standard deviation of 1° , 10% landmark occlusions and 10% false landmark matches as outliers. Then, we generated homing vector streamline plots and computed the angular deviations from the direct path to the home position. We define a homing vector computation method as robust, if all streamlines from any direction reach the home location. Furthermore, the angular deviation from the direct path to the home location should be as small as possible. Table 1 gives an overview of the discussed homing vector calculation methods, their robustness and their angular deviations from the direct path. Fig. 6 and Fig. 7 visualize the streamlines and angle deviations. As can be seen, only the normalized difference vector model and the normalized improved difference vector model are robust according to our definition, which means that all streamlines reach the home location. Furthermore, the normalized improved difference vector method has the lowest angle deviations from the direct path to the home position. Hence, in a general environment with outliers and noise, the normalized improved difference vector method should be chosen.

Except for the tangential correction vector method (Weber et al. (1999)), all of the described methods for calculating homing vectors also work in 3D. Flying robots can make use of three-dimensional homing vectors, but wheeled or legged robots are restricted to the ground, so that they can only change the yaw component of their orientation actively. That means, the z-component of the homing vector or the homing elevation angle can be ignored, and only the azimuth component of the homing angles or the x- and y-components of the homing vector are considered. However, in environments with large height changes, the elevation angle can help to ensure that the robot is on the correct path, which for example leads uphill.

Table 1. Comparison of homing vector calculation methods for reference scenario

	robust	mean angle deviation	standard deviation of angle deviation	maximum angle deviation
Difference vector model (Fig. 6a) (Lambrinos et al. (2000))	-	14.1°	21.2°	176.7°
Normalized difference vector model (Fig. 6b)	✓	12.2°	10.5°	60.7°
Improved difference vector model (Fig. 6c)	-	11.1°	17.8°	178.9°
Normalized improved difference vector model (Fig. 7a)	✓	7.3°	5.7°	38.6°
Tangential correction vector method (Fig. 7b) (Weber et al. (1999))	-	25.9°	23.1°	177.6°
IBVS based on bisecting components (Fig. 7c) (Liu et al. (2010))	-	36.4°	31.5°	180.0°

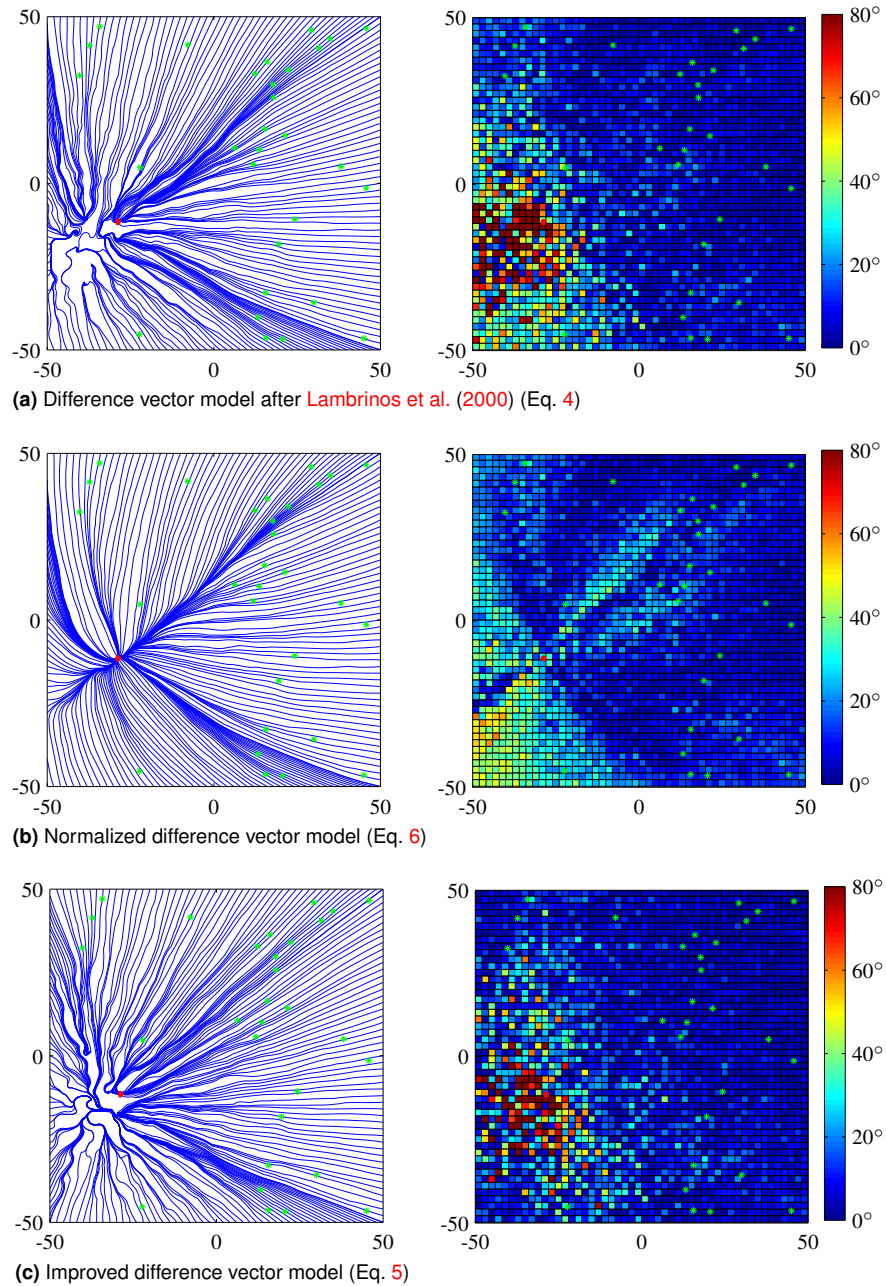
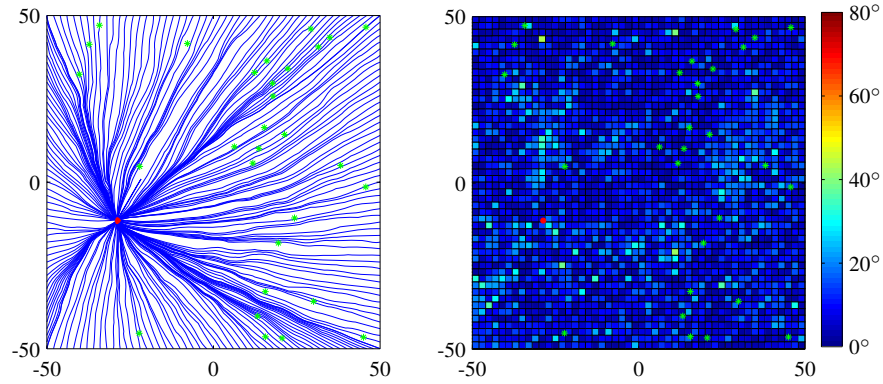
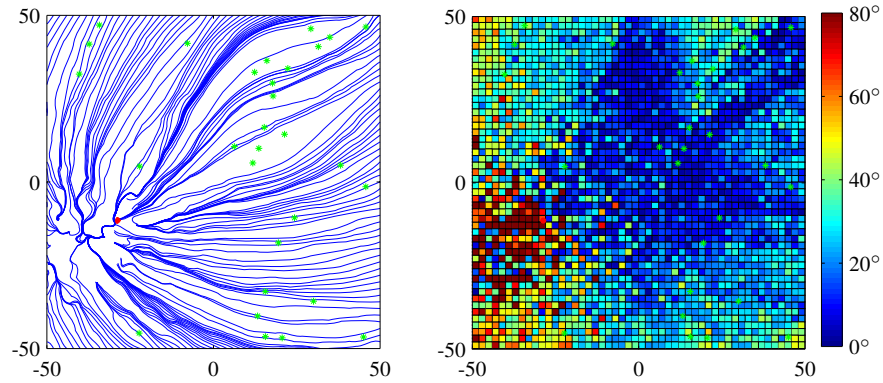


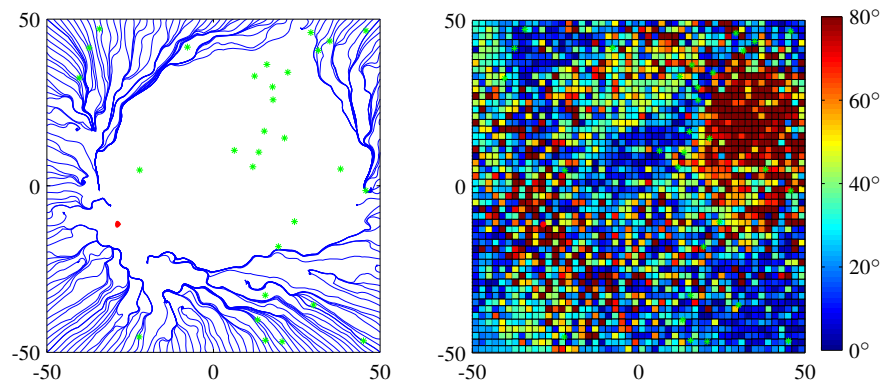
Figure 6. Homing vector streamlines (left) and deviations from the direct path (right) for the reference scenario (general landmark configuration (green asterisks), noise (1.0°), outliers (10%), occlusions (10%)). Red circle: home location. The x and y coordinates are given in units.



(a) Normalized improved difference vector model (Eq. 7)



(b) Tangential correction vector method after Weber et al. (1999) (Eq. 8)



(c) Image based visual servoing after Liu et al. (2010) (Eq. 9)

Figure 7. Homing vector streamlines (left) and deviations from the direct path (right) for the reference scenario (general landmark configuration (green asterisks), noise (1.0°), outliers (10%), occlusions (10%)). Red circle: home location. The x and y coordinates are given in units.

3 The Trail-Map

While moving through the environment, the robot has to record a new viewframe every time the dissimilarity measure exceeds the defined threshold. The bearing angle to far landmarks and landmarks in the direction of travel hardly changes between successive viewframes, which makes them translation invariant. In contrast, close landmarks are translation variant since they are perceived at significantly different angles between two viewframes. Thus, simply storing the sequence of viewframes would lead to many redundancies. Rather, only one instance of each landmark view should be stored as long as its bearing angle does not change more than a threshold δ_{ang} . This was the key idea behind the LT-Map introduced by (Augustine et al. (2012)). In their work, landmark views are organized in a tree such that the translation invariant landmark views are at the top of the tree, so that they are shared among several viewframes. The quickly changing translation variant landmark views are stored in the leaves. This data structure not only avoids redundancies, but it also allows scaling the stored information in case of memory shortage by simply pruning the leaves of the tree. Thus, only quickly changing volatile information is discarded but the long-term stable landmark views are preserved.

However, as already shown in our previous work (Stelzer et al. (2014)), the LT-Map is not the best data structure for storing a sequence of viewframes. We will explain the shortcomings of the LT-Map using the simplified scenario in Fig. 8. Here, the robot path leads through an environment with three landmarks and we assume that a landmark view changes significantly when the bearing angle difference to its previous instance in the map is more than $\delta_{\text{ang}} = 45^\circ$. That results in eight δ_{ang} -sectors originating from each landmark in the environment*. As long as the robot is within the same sector of a landmark, it perceives the landmark with a bearing angle difference of less than δ_{ang} . This is visualized by connected path segments in the corresponding color of the landmark. Assuming that two places are sufficiently different as soon as one landmark view changes significantly, the intersecting regions of the landmark sectors form the viewframe locations. That means, every time the robot crosses the border of a landmark sector, it records a new viewframe. The first three viewframes are highlighted in Fig. 8.

The LT-Map which the robot would create by following the path in Fig. 8 is shown in Fig. 9. From viewframe 1 to 2, only the bearing angle to landmark L_1 changes significantly. From viewframe 2 to 3, only landmark L_2 changes more than δ_{ang} . However, caused by the structure of the tree, also a new instance of the landmark view to L_1 has to be stored, because the tree does not permit overlaps between neighbouring branches. As soon as a landmark view in a higher level of the tree changes significantly, new instances of the landmark views in all child branches have to be stored. During the traverse of the path more of these redundancies occur, which are indicated by connecting bars in Fig. 9. In the end, the LT-Map consists of 20 landmark views.

To find a better data structure for the sequence of viewframes, we projected the connected path segments for each landmark in a table as shown in Fig. 10. A landmark

*For the sake of simplicity we assume that the δ_{ang} intervals are aligned with each other and do not start at the first observation of each landmark.

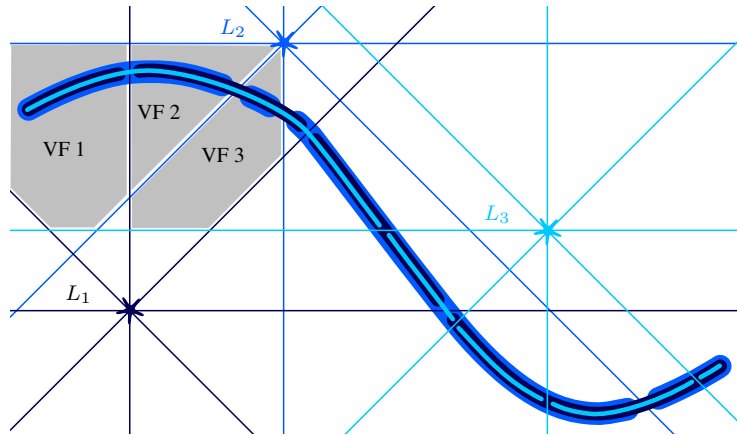


Figure 8. Scenario with 11 viewframes VF1 to VF11 and three landmarks L_1 to L_3 (adapted from [Stelzer et al. \(2014\)](#))

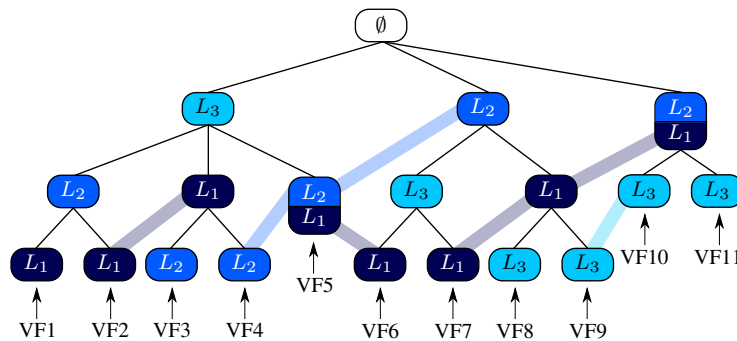


Figure 9. Resulting LT-Map. The connecting bars indicate redundancies caused by the tree data structure.

view is shared among several viewframes as long as its bearing angle does not change significantly (i.e. more than δ_{ang}). To introduce a hierarchy for scaling the map, we order the landmark views by the number of viewframes in which they stay constant. This corresponds to the level of translation invariance. For this reason, we call the new data structure Translation Invariance Level Map (Trail-Map). Now, each level contains all the landmark views which span exactly the number of viewframes corresponding to the level number. Level 4 is empty since no landmark view stays within the angle threshold δ_{ang} for exactly four viewframes. The Trail-Map has 13 landmark views, and, thus, already saves 35% compared to the number of landmark views in the LT-Map for the same simple scenario, only by avoiding redundancies. When the map has to be scaled, the lower levels in the Trail-Map can be deleted and only the landmark views which change quickly along the path are discarded.

This data structure can be implemented without much overhead as a linked list of linked lists as shown in Fig. 11. A landmark view (LV) in this structure consists of the

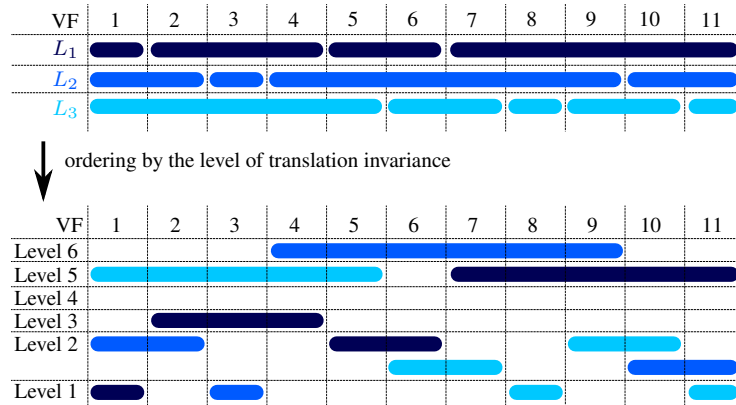


Figure 10. Derivation of the Trail-Map data structure (adapted from Stelzer et al. (2014))

landmark's ID, its descriptor, its bearing angle as a unit vector and the number of the viewframe when the LV was added to the map. The latter is required to derive for each landmark view which viewframes it belongs to. A level consists of the level number and the list of landmark views in the order in which they were added to the map. The list of levels is also sorted by the level number. For adding more viewframes to the Trail-Map, an *open list* has to be maintained, which contains pointers to all landmark views of the latest recorded viewframes, also sorted by the number of the corresponding level.

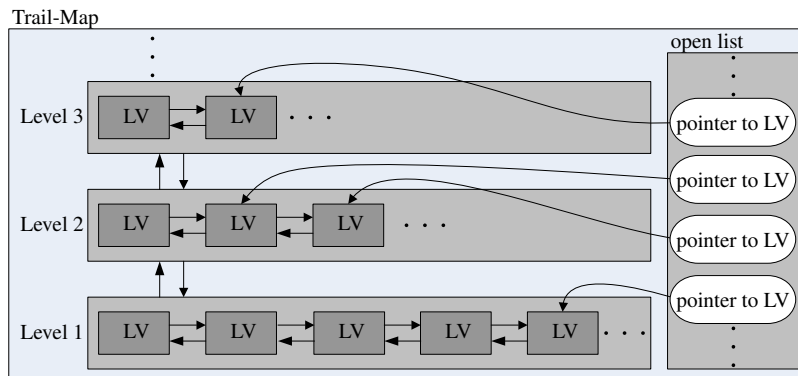


Figure 11. Implementation of the Trail-Map (adapted from Stelzer et al. (2014))

Pseudocode for appending a new viewframe to the map is given in Algorithm 1. As an extension of the basic algorithm proposed in Stelzer et al. (2014), we added code lines for assigning a *waiting* flag to unobserved landmark views to account for possibly occluded landmarks. Initially, the Trail-Map consists of the empty level 1. The open list is also empty. Thus, all landmark views in first viewframe are added to level 1 of the Trail-Map and pointers to these entries are stored in the open list (lines 39-42). Before the second viewframe is added to the map, a new level is appended, because now level 1

is not empty anymore (lines 1-3). After that, the algorithm loops through the open list. For each landmark view carrying the *waiting* flag it is checked how long it could not be observed. If it has not been visible for more than the number of viewframes specified by the *bufferSize* variable, it is removed from the open list (lines 5-10). The remaining landmark views in the open list are compared to all landmark views in the viewframe to be added. If matching descriptors are found, the bearing angle difference is checked. If the bearing angle of the landmark view in the open list has not changed significantly, the landmark view is deleted from its current level in the map and appended to the next higher level (lines 22-23). Its pointer in the open list is updated (line 24). In case the landmark view is marked as *waiting*, it is checked in which level the landmark view would have been if it had been visible all the time (line 16), the corresponding levels are added to the map (lines 17-19) and the waiting flag is removed (line 20). Thus, the landmark view is stored in a way as if it had been visible all the time. If the bearing angle of the landmark view in the open list has changed significantly, its pointer is removed from the open list, the new landmark view is appended to level 1 and a pointer to this new landmark view is appended to the open list (lines 26-28). A landmark view which could be matched with the open list is removed from the viewframe and the search for matches terminates (lines 30-32). If a landmark in the open list could not be matched with the current viewframe, it is assigned the *waiting* flag (lines 35-37). In the end, all landmark views in the viewframe that were not found in the open list are added to level 1 of the map and pointers to these landmarks are appended to the open list (lines 39-42).

This method for adding a viewframe to the Trail-Map automatically yields an open list which is sorted by the level number of the landmark views. Additionally, the landmark view lists are sorted by the number of the viewframe when the landmark view was inserted in the map. This helps for retrieving a viewframe from the map, when all landmark views belonging to the demanded viewframe number have to be collected from the levels. Furthermore, since for homing the viewframes are extracted in reverse order of their first visit, only the neighboring landmark views on the landmark view lists in the levels have to be considered instead of searching through the whole lists.

4 Simulations of Trail-Map-Based Homing

To show the superior performance of the Trail-Map compared to the LT-Map, we created the simulation environment shown in Fig. 12 with 100 uniquely identifiable randomly distributed landmarks in an area of 200×200 units and commanded a learning path of about 130 units length starting at $(0, 0)$. The simulated robot was equipped with a noise-free omnidirectional landmark sensor, which was able to observe and identify all landmarks in the environment correctly. During the traverse of the path, the robot recorded a Trail-Map. For acquiring a new viewframe, we chose the k th maximum dissimilarity measure $\delta_{\text{diss}}^{\text{max}}$ (Eq. 3) with $k = 1$, because no outliers or noise were simulated. The dissimilarity threshold was set to $\xi_{\delta_{\text{diss}}^{\text{max}}}^{\text{map}} = 5^\circ$. That means, every time one landmark changed its bearing angle by more than this value, a new viewframe was created. The Trail-Map was also created with the parameter $\delta_{\text{ang}} = 5^\circ$. That means, a new instance of a landmark view was only stored if it changed more than 5° compared

Input : viewframe VF ; $trailMap$; $openList$
Output: $trailMap$; $openList$

```

1 if ! $trailMap$ .GetHighestLevel.IsEmpty then
2   |  $trailMap$ .addLevel;
3 end
4 foreach  $l$  in  $openList$  do
5   | if  $l.isWaiting$  then
6     | if  $trailMap$ .getViewframeCount - ( $l.insertionFrame$  +  $l.getLevel$ -1) >  $bufferSize$  then
7       |  $openList$ .removePointerTo( $l$ );
8       | continue;
9     | end
10  | end
11  | foreach  $f$  in  $VF$ .LandmarkViews do
12    | if  $f$ .descriptor matches  $l$ .descriptor then
13      | if  $|f$ .angle- $l$ .angle| <  $\delta_{ang}$  then
14        |  $l.insertionLevel$ := $l$ .getLevel;
15        | if  $l.isWaiting$  then
16          |  $l.insertionLevel$ := $trailMap$ .getViewframeCount -  $l$ .InsertionFrame;
17          | while  $trailMap$ .getHighestLevel.Number  $\leq$   $l.insertionLevel$  + 1 do
18            |  $trailMap$ .addLevel;
19          | end
20          |  $l.isWaiting$ :=false;
21        | end
22        |  $trailMap$ [ $l$ .getLevel].remove( $l$ );
23        |  $trailMap$ [ $l.insertionLevel$ +1].append( $l$ );
24        |  $openList$ .updatePointerTo( $l$ );
25      | else
26        |  $openList$ .removePointerTo( $l$ );
27        |  $trailMap$ [1].append( $f$ );
28        |  $openList$ .appendPointerTo( $f$ );
29      | end
30      |  $VF$ .remove( $f$ );
31      | found:=true;
32      | break;
33    | end
34  | end
35  | if !found then
36    |  $l.isWaiting$ :=true;
37  | end
38 end
39 foreach  $f$  in  $VF$ .LandmarkViews do
40   |  $trailMap$ [1].append( $f$ );
41   |  $openList$ .appendPointerTo( $f$ );
42 end

```

Algorithm 1: Appending a viewframe to the Trail-Map (extended from Stelzer et al. (2014))

to its current instance in the map. We ran the simulation 100 times for each parameter set and randomly changed the landmark positions after each run.

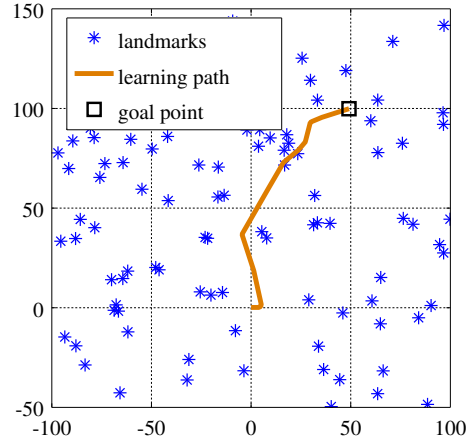


Figure 12. Simulation environment with learning path. The robot should retrace the path to reach the goal location.

First, the robot built a Trail-Map and an LT-Map during the traverse of the learning path. On average, the recorded LT-Maps had 3315 landmark views, while the Trail-Maps only consisted of 1278 landmark views, which means memory savings of more than 60%. This shows, how many redundancies were present in the LT-Map, which were avoided in the Trail-Map data structure.

To compare the homing performances using both data structures, we implemented homing using the difference vector model (Eq. 6). In this simulation, we placed the robot back to the starting position after learning the path and let the robot compute homing vectors to retrace the path in the same direction as during learning. Since the robot has a perfect omnidirectional sensor, the direction of retracing the learned path has no influence on the performance. We used the same dissimilarity measure as in the mapping phase $\xi_{\delta_{\text{diss}}^{\text{max}}}^{\text{hom}} = \xi_{\delta_{\text{diss}}^{\text{max}}}^{\text{map}} = 5^\circ$. That means, every time the dissimilarity measure of the currently perceived view and the reference viewframe dropped below the threshold $\xi_{\delta_{\text{diss}}^{\text{max}}}^{\text{hom}}$, the robot assumed to have reached the reference viewframe and started homing to the next viewframe in the Trail-Map. Homing finished when the robot reached the viewframe corresponding to the goal position of the robot. We simulated homing for different pruning levels of the LT-Map and the Trail-Map and compared the resulting path error as the area between the learning trajectory and the homing trajectories. To ensure that the goal viewframe could always be reached, we never pruned landmark views belonging to the goal viewframe. For each pruning level, the simulation was run 100 times and the landmark configuration was randomly changed after each run.

Fig. 13 shows that the Trail-Map always has a significantly lower number of landmark views than the LT-Map. The most noticeable plot is shown in Fig. 14, which visualizes the path error based on the percentage of remaining landmark views in both maps. It can be seen that the Trail-Map can be pruned by about 50% without significantly losing path accuracy, while the homing performance using the LT-Map already degrades after deleting less than 10% of the landmark views. Keeping in mind

that the 100% mark in this plot corresponds to 3315 landmark views for the LT-Map, but only to 1278 landmark views for the Trail-Map, this advantage of the Trail-Map becomes even more remarkable.

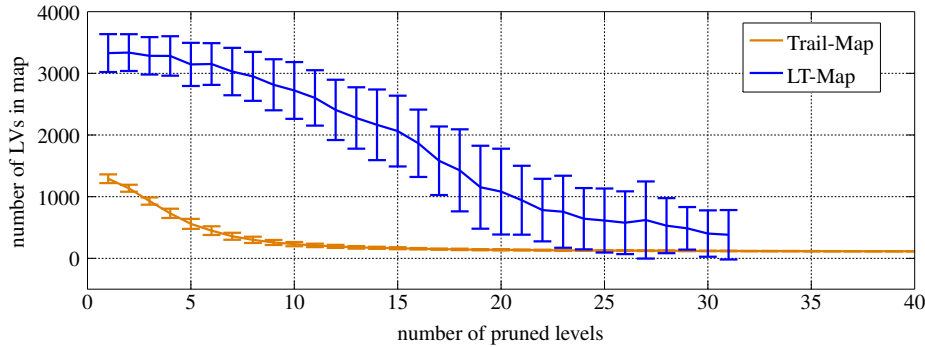


Figure 13. Means and standard deviations of the number of landmark views in the LT-Map and the Trail-Map depending on the number of pruned levels (adapted from Stelzer et al. (2014))

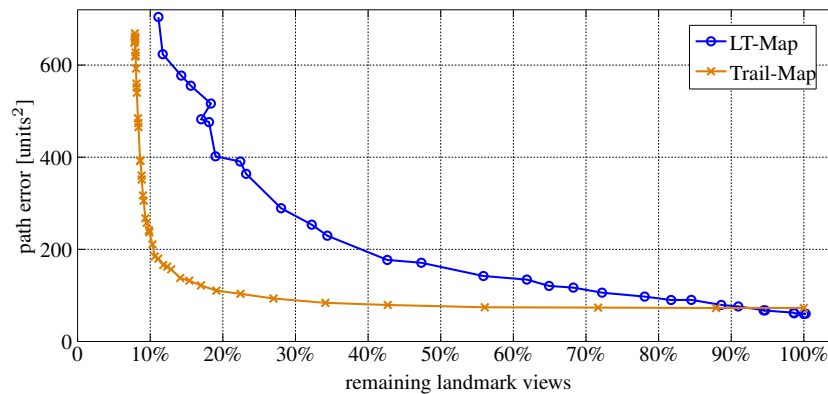


Figure 14. Comparison of path errors of LT-Map and Trail-Map for different pruning ratios (adapted from Stelzer et al. (2014))

To get an impression of what the homing trajectories using the LT-Map and the Trail-Map look like, Fig. 15 shows some trajectories recorded at different pruning levels for one sample environment. Comparing the path errors using the full Trail-Map (ref. Fig. 15d) and the full LT-Map (ref. Fig. 15a), the trajectory using the LT-Map is slightly more accurate, which can also be observed in Fig. 14. The reason for that is the large number of landmark views in the LT-Map. Since not only new instances of the landmark views that changed significantly are stored, but also new instances of other landmark views are inserted in the LT-Map to comply with the tree structure, the resulting resolution of the LT-Map is higher than the angle threshold of $\delta_{\text{ang}} = 5^\circ$. That results in slightly more accurate homing paths using the full LT-Map. However, the

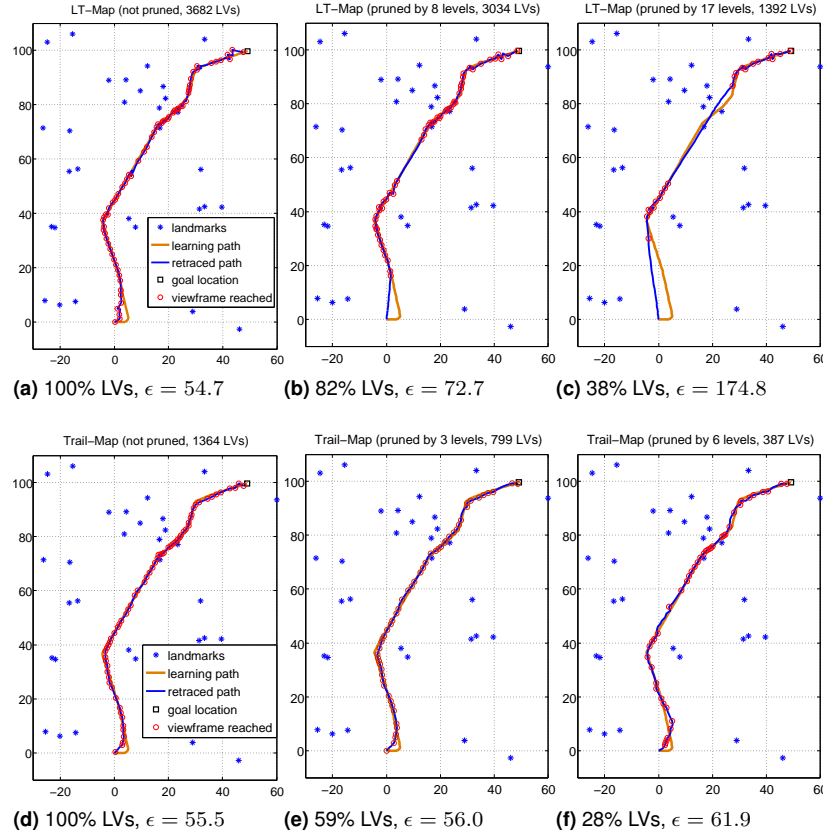


Figure 15. Comparison of Trail-Map and LT-Map pruning behavior (adapted from [Stelzer et al. \(2014\)](#)). Top row: LT-Map, Bottom row: Trail-Map. ϵ : path error in units².

accuracy degrades quickly when the LT-Map is pruned. After pruning only 18% of the landmark views, the path error increases visibly (ref. Fig. 15b). In contrast, the accuracy of Trail-Map-based homing hardly decreases after pruning 3 levels, which corresponds to deleting more than 40% of all landmark views (ref. Fig. 15e). Pruning the LT-Map to an approximately similar number of viewframes as in the full Trail-Map already results in a path error of $\epsilon = 174.8$, as shown in Fig. 15c. In contrast, Fig. 15f shows that the even after pruning more than 70% of the landmark views in the Trail-Map, the homing performance is still good.

In [Stelzer et al. \(2014\)](#) we showed that the memory savings of the Trail-Map do not come at the expense of runtime. Instead, adding viewframes to the map, retrieving viewframes in the same or reverse order of their acquisition and map pruning are faster than using the LT-Map, assuming the number of landmark views for each viewframe is less than 1000. Especially the pruning operation is magnitudes faster for the Trail-Map. The runtimes for adding viewframes to the map and retrieving subsequent

viewframes does not depend on the length of the travelled path but only on the number of landmark views per viewframe. This is an important property for robots with limited computational resources.

5 Application to Real Data

Bringing the homing method to the real world poses several challenges that were not present in the simulations. This section will address these issues.

5.1 Landmark Detection and Matching

The main challenge of the real world implementation of Trail-Map-based homing is the landmark detection and matching process, which was neglected in the simulations, where we assumed known data association. Landmarks are extracted from the omnidirectional camera images which are unwarped to panorama images. We decided to use BRISK (Leutenegger et al. (2011)) for this task, because it is computationally efficient. To reject outliers from the set of matches, we make use of the method suggested by Jäger et al. (2013), which is based on a RANSAC algorithm. First, RANSAC is applied to the feature matches using a translation invariant model for estimating the rotation between the two image acquisition points. Then, RANSAC is applied to the remaining feature matches using a translation variant model to estimate the translation between the images. Fig. 16 shows the result of landmark detection and matching for two sample panorama images taken at nearby locations in our laboratory test environment. Around 900 landmarks were detected in each image, but only about 240 landmarks could be matched, of which 30 were detected as outliers.



Figure 16. Landmark matches: Black: valid matches, white: rejected outliers.

5.2 Rotational Alignment of the Viewframes

As previous simulations (Stelzer et al. (2015)) have shown, the homing algorithm is very sensitive to rotational misalignments of the viewframes. If no reliable compass is available, an estimated yaw angle by fusing inertial and visual odometry data will drift over time. That does not pose a problem during the mapping phase, because

the viewframes are recorded sequentially, so that the errors between two subsequent viewframes are small. However, during homing the error in the yaw estimate can have become large, especially for homing towards early viewframes close to the home position. For this reason, we use the rotation that is estimated by the RANSAC algorithm in the outlier rejection step for aligning the viewframes before dissimilarity measures or homing vectors are computed. Thus, we can entirely omit a compass or any other yaw estimate. Only in case the robot moves on hilly ground, roll and pitch angle measurements are additionally required to align the landmark unit vectors. These angles can be estimated from an IMU, for example.

5.3 Homing Vector Smoothing

Since the single homing vectors computed from the current panorama image and the goal viewframe using Eq. 5 are noisy, we use a method described in (Jäger et al. (2013)) for smoothing the homing vectors. In this method, the median out of the last n homing vectors to the current goal viewframe is computed, where n was empirically chosen to be 7. The longer this buffer, the more robust is the result, but also the larger is the delay in the control. To avoid waiting times when the robot reaches a viewframe and starts homing to the next one, the robot computes homing vectors to the current and the following goal viewframe simultaneously for each panorama image. Thus, a median homing vector for the next goal viewframe is already available when the robot starts homing to it.

5.4 Robot Motion Control

To make the robot follow the computed homing vector, we applied a very simple motion strategy. A homing vector is calculated continuously as the robot moves through the environment. At every time step, the robot computes its forward velocity v_x and its turn velocity ω_z depending on the difference between its current yaw angle α and the current homing vector direction ϕ_h as

$$v_x = v_{\max} \max \left(0, 1 - \frac{|\phi_h - \alpha|}{\phi_t} \right), \quad (10)$$

$$\omega_z = \omega_{\max} \max \left(-1, \min \left(1, \frac{\phi_h - \alpha}{\phi_t} \right) \right), \quad (11)$$

where v_{\max} and ω_{\max} are the maximal forward and turn speeds. If the difference between the current yaw angle and the current homing vector direction is higher than a threshold ϕ_t , the robot performs pure turning. Otherwise, the robot combines forward and turn motions, while the forward velocity increases and the turning velocity decreases the closer the robot's yaw angle gets to the desired homing direction.

6 Experiments

To experimentally evaluate Trail-Map based homing, we used a Pioneer 3-DX robot equipped with an omnidirectional camera, a stereo camera and an inertial measurement unit (IMU). The omnidirectional camera was a catadioptric system with a PointGrey

USB camera providing images of 896×896 pixels, which were unwarped to panorama images of 1440×280 pixels and downscaled to 30%, i.e. 432×84 pixels, for landmark detection. The stereo camera and the IMU are used for pose estimation by fusing visual odometry and inertial data as presented by Chilian et al. (2011). However, only the estimated roll and pitch angles are used for computing aligned landmark unit vectors. The computational hardware is an Intel Core i7-3740QM CPU with 2.70GHz and a Spartan 6 LX75 FPGA Eval Board to perform dense stereo matching using SGM (Hirschmüller (2008)) at a rate of more than 10Hz. The next section will show the results of experiments conducted in an indoor laboratory environment. After that, in Section 6.2 we will analyze the long-range performance based on experiments in outdoor terrain.

6.1 Indoor Laboratory Experiments

The first set of experiments was conducted in an indoor laboratory environment on flat ground without obstacles. Ground truth for the experiments was provided by an optical tracking system based on several infrared cameras that track a reflecting target body mounted on the robot.

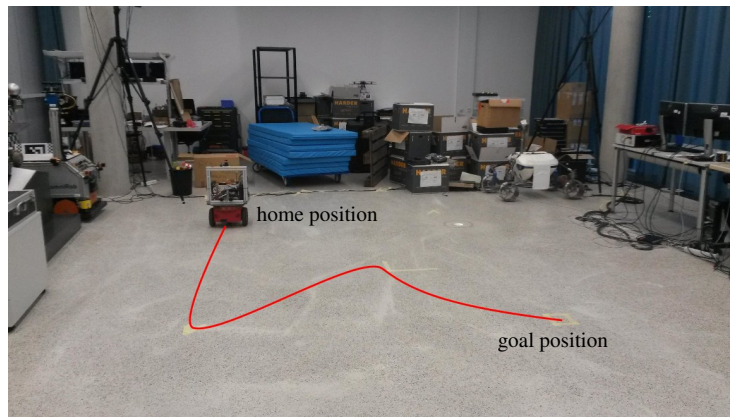


Figure 17. Experimental setup for the indoor laboratory experiment with home position, path and goal position

6.1.1 Mapping We remotely controlled the robot along a trajectory of about 5.1 meters length to a goal point in the laboratory environment. The setup and the path are shown in Fig. 17. During the traverse the robot recorded a Trail-Map, using an angle threshold of $\delta_{\text{ang}} = 10^\circ$ and a dissimilarity threshold of $\xi_{\delta_{\text{ang}}}^{\text{map}} = 0.05\text{rad} \approx 2.9^\circ$ for mapping. The viewframe dissimilarities were computed using the average angle error method in Eq. 2. The resulting Trail-Map consisted of 17 viewframes which contained 882 landmark views on average. Thus, in total 15000 landmark views were recorded for this path. However, the Trail-Map had only 10992 landmark views, because slowly changing landmark views are not stored redundantly. Thus, the Trail-Map saved more than 26% of memory compared to

straightforward viewframe storage. The trajectory with the recorded viewframes, along with the number of landmark views in each viewframe and the landmark matches between the viewframes is shown in Fig. 18. It can be seen that the viewframes VF11 to VF15 in the center of the free area are further apart from each other than the viewframes which are closer to the border of the free driving area. This shows that the distances between the viewframes are smaller in regions with close landmarks and grow if only distant landmarks can be observed. The landmark views were spread over 12 levels (ref. Table 2), where nearly 75% of the landmark views were located in level 1. The reason for the large amount of landmark views in level 1 can already be inferred from the number of matches between the viewframes, which is on average only 261 compared to an average of 882 landmark views per viewframe. That means that many landmarks in the viewframes cannot be matched with the landmarks in the neighboring viewframes, which leaves them in level 1.

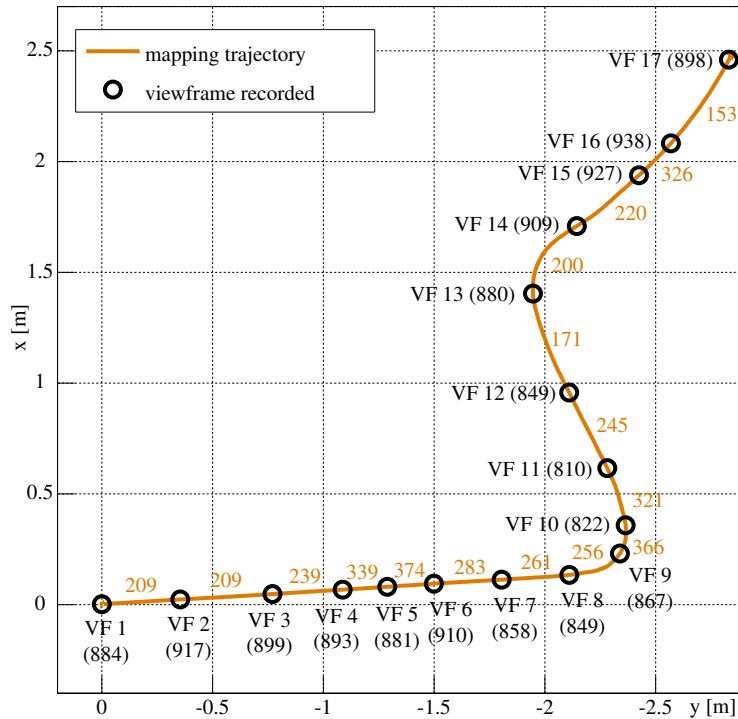
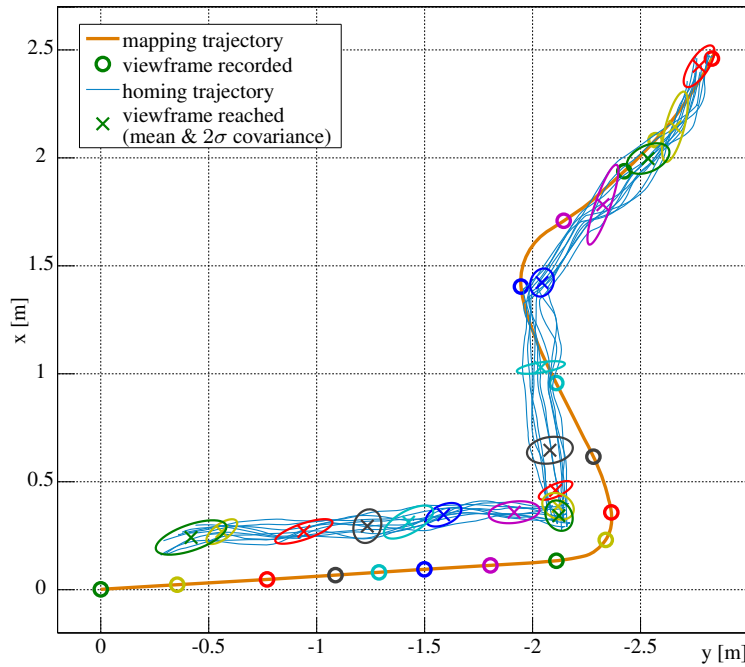


Figure 18. Path with viewframes (VF) for the indoor laboratory experiment. Numbers in brackets: number of landmark views in the viewframe. Orange numbers: number of landmark matches between the viewframes

6.1.2 Validation of Homing Next, the robot performed homing using the recorded Trail-Map. We placed the robot back to the goal position after each homing run and recorded 10 homing trajectories. For homing we used a dissimilarity threshold of $\xi_{\text{diss}}^{\text{hom}} = 0.03\text{rad} \approx 1.7^\circ$ average angle error for detecting when a viewframe is reached.

Table 2. Distribution of landmark views (LV) over the levels in the indoor laboratory experiment

level	1	2	3	4	5	6	7	8	9	10	11	12
LVs	8171	1430	634	328	185	108	79	31	16	6	3	1
ratio	74.3%	13.0%	5.8%	3.0%	1.7%	1.0%	0.7%	0.3%	0.1%	0.05%	0.03%	0.01%

**Figure 19.** Mapping and homing trajectories using the full Trail-Map in the indoor laboratory experiment

The robot computed the homing vectors using the improved difference vector model in Eq. 5 at a rate of about 2.5Hz. Fig. 19 shows the resulting trajectories with the centers and covariance ellipses of the coordinates where the robot reached a viewframe. It can be seen that all homing paths were close together and the robot followed the mapping trajectory. The deviation from the recorded trajectory can be explained by the area in which the dissimilarity measure for a viewframe falls below the threshold ξ_{diss}^{hom} . The shape and size of this area depends on the landmark configuration in the environment. This relation is illustrated in Fig. 20, which shows a mapping path with two viewframes VF1 and VF2 in an environment with three landmarks L1, L2 and L3. Two landmarks on one side are close to the mapping path, the other landmark on the opposite side is far away. That represents the landmark distribution in our experiment for the viewframes VF1 to VF9, where the robot had close objects to its right and only distant objects to its left hand side (ref. Fig. 17). The areas in which the dissimilarity measures for the

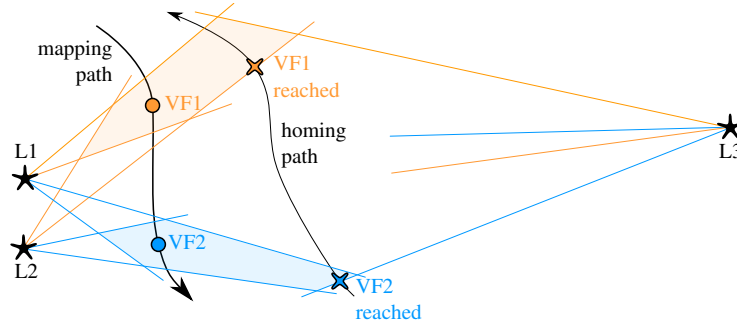


Figure 20. Shape of the areas in which the dissimilarity measures for the corresponding viewframes fall below the threshold $\xi_{\delta_{diss}^{ang}}^{hom}$ depending on the landmark distribution

corresponding viewframes fall below the threshold are the shaded polygons that result from the intersections of sectors originating at the landmarks. The opening angle of the sectors is $2\xi_{\delta_{diss}^{ang}}^{hom}$ and the viewframes are in the centers of the sectors[†]. It can be seen that the shape of the shaded area is stretched in the direction of the far landmark. Now, as soon as the robot enters the shaded area of a viewframe, it detects the viewframe as reached and switches to the next viewframe. That explains, why the deviation from the mapping path in Fig. 19 had an offset in the direction of the far landmarks.

We analyzed the homing paths by computing the mean path error and the corresponding standard deviation, the maximal path error and the endpoint error. We computed the path error at a certain point of the homing trajectory as the shortest distance of this point to the mapping trajectory. The path error statistics were then computed from the path errors of all points of the homing trajectory. The statistics of the 10 homing runs are shown in Fig. 21. The average of all mean path errors was 0.14m, the maximal path deviation was 0.27m on average and the average endpoint error was about 0.48m. These values, especially the endpoint error seem to be large compared to the path length of 5.1m. However, these values do *not* depend on the length of the path, but only on the configuration of landmarks in the environment and the chosen dissimilarity threshold. Thus, also for longer paths with the same home position, the endpoint error would be approximately 0.5m. For this reason, we will not give any errors as percentage of the driven path length, but will only state the absolute errors.

6.1.3 Landmark Matching Fig. 22 shows how the landmark views in a viewframe were distributed among the levels and how many of the landmark views in each level were matched during the homing process. It becomes obvious that only 8% of the landmarks in level 1 could be matched, while in higher levels this ratio was about 20%. This fact is caused by many *spurious* landmarks being added to the map.

[†]Please note that this is a simplification. The dissimilarity is also lower than the threshold in some border areas, in which the angle error to one landmark is higher than $\xi_{\delta_{diss}^{ang}}^{hom}$, but which is compensated by the other landmarks having errors lower than the threshold.

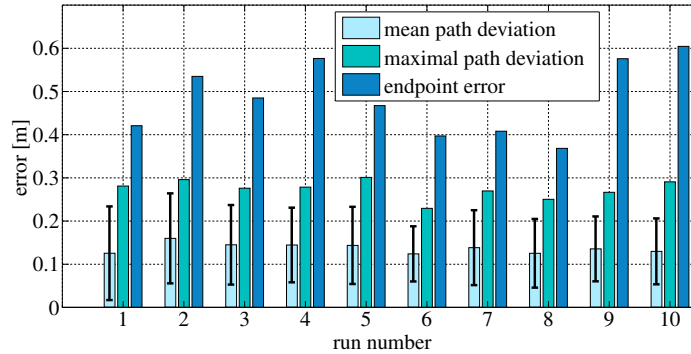


Figure 21. Path errors and endpoint errors for the single homing runs in the indoor laboratory experiment

A spurious landmark is a landmark which does not correspond to a specific object in the environment but appears as characteristic point in the image at object borders due to the high contrast between the object in the foreground and the background. Thus, it is only visible within a relatively small angle and changes or disappears if the viewpoint shifts. For this reason, the landmarks cannot be matched from other viewpoints, which explains the low matching ratio in level 1 and also the high number of landmark views in this level. However, instead of applying prefiltering methods for selecting only stable landmarks, we can exploit the Trail-Map structure and discard these landmark observations by pruning the lower levels of the Trail-Map.

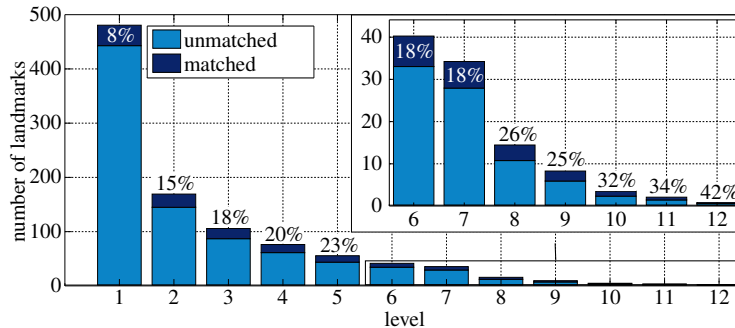


Figure 22. Average number of matched and unmatched landmarks per viewframe and their distribution over the levels in the indoor laboratory experiment. The percentage corresponds to the ratio of matched landmarks.

Fig. 23 shows how the landmark matches of the different levels were distributed in the panorama image. It can be seen that the landmarks in the higher levels were located in the direction of motion and on distant objects.

6.1.4 Pruning Behavior To evaluate the homing performance when pruning the recorded Trail-Map at different levels, we performed 10 homing runs for each pruning level. To ensure that the robot reached the home position, the home viewframe

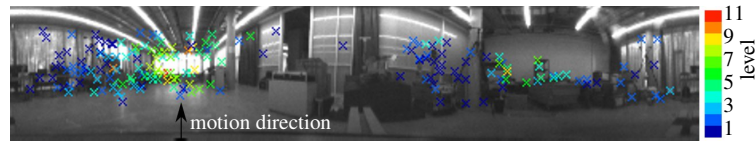


Figure 23. Distribution of the landmark matches of the different levels in the image for the indoor laboratory experiment

was never pruned. The resulting statistics are shown in Fig. 24. When pruning one level, already 2/3 of all landmark views were deleted. However, the homing performance degraded only slightly. When pruning the Trail-Map to less than 20% of its original size, also only slightly higher path errors occurred. The endpoint errors were independent from the pruning level since the home viewframe was not pruned. We stopped pruning after 7 levels, because the robot frequently entered a spiral search pattern that we implemented when not enough landmark matches can be found. However, it still reached the home position in all trials.

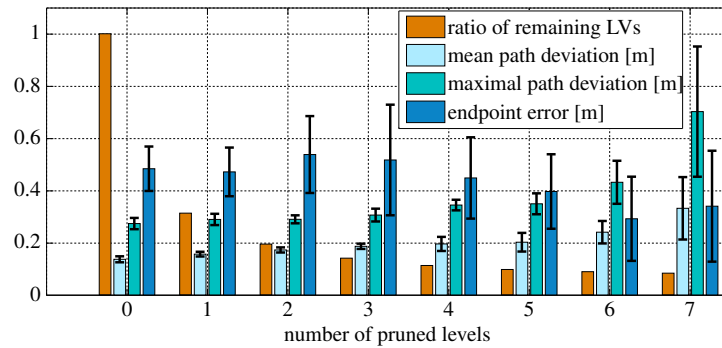


Figure 24. LVs, path and endpoint errors for different pruning levels in the indoor laboratory experiment

To further show how the homing paths change depending on the pruning level, we recorded another Trail-Map for a meandering path of 17.7m length. The Trail-Map contained 64 viewframes and 29586 landmark views spread over 23 levels. Then, we let the robot perform single homing runs at different pruning levels and recorded the homing trajectories. The results are shown in Fig. 25. It can be observed that the robot took shortcuts the more the Trail-Map was pruned. However, with only 975 landmark views left (which is only 3.3% of the full Trail-Map size), the robot still found the home position. Of these 975 landmark views, 711 corresponded to viewframe 1, which was not pruned to ensure that the robot reached the home position.

For a better understanding of how much memory is saved, we should note that one landmark view needs about 100 bytes of memory. In detail, a landmark view consists of the landmark's ID (4 bytes integer), the landmark's descriptor (64 bytes for BRISK),

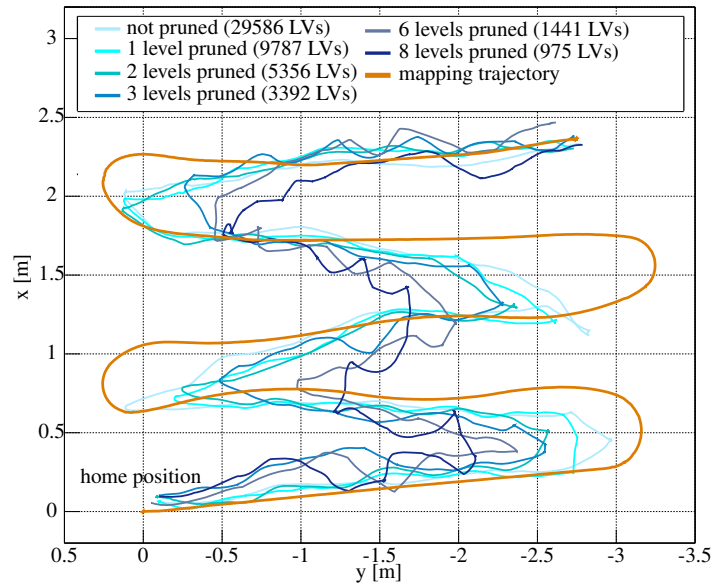


Figure 25. Homing paths for different pruning levels in the indoor meandering experiment

a 3D unit vector (24 bytes for 3 double numbers[‡]) and the number of the viewframe when the landmark view was added to the Trail-Map (4 bytes integer). Thus, the landmark descriptor is the main memory consumer. The Trail-Map consists of a list of levels, in which each level contains a list of landmark views and the level number (4 bytes integer). In the list of landmark views, each node has some additional memory requirements, depending on the programming environment. Thus, the full Trail-Map of the meandering path of 17m length requires about 3MB, which would extrapolate to about 18MB for 100m. This value is in the region of the memory consumption reported for Visual Teach and Repeat (Furgale and Barfoot (2010b)) (35MB for 100m) and the method introduced by Krajník et al. (2010) (10MB per 100m). However, by pruning the map by 2 levels, the path deviation is still acceptable, but the memory requirements decrease to about 500kB for 17m, which corresponds to approximately 3MB per 100m. Furthermore, when pruning 8 levels of the map, only 100kB are sufficient to still reach the home position – where most of the required memory stems from the first viewframe that has not been pruned. This corresponds to less than 600kB per 100m. Whether this kind of extrapolation holds for longer paths will be discussed in the next section on long-range outdoor experiments.

6.2 Long-Range Outdoor Experiments

To show the long-range performance of Trail-Map-based homing, we performed outdoor experiments on untraveled roads on the DLR Oberpfaffenhofen site, using the

[‡]By knowing that it is a unit vector, only 2 elements would also be sufficient.

same Pioneer 3-DX robot as in the indoor laboratory experiments. Ground truth was obtained by a tachymeter, which automatically tracked a prism mounted on the robot and recorded the x, y and z coordinates using an infrared laser beam.

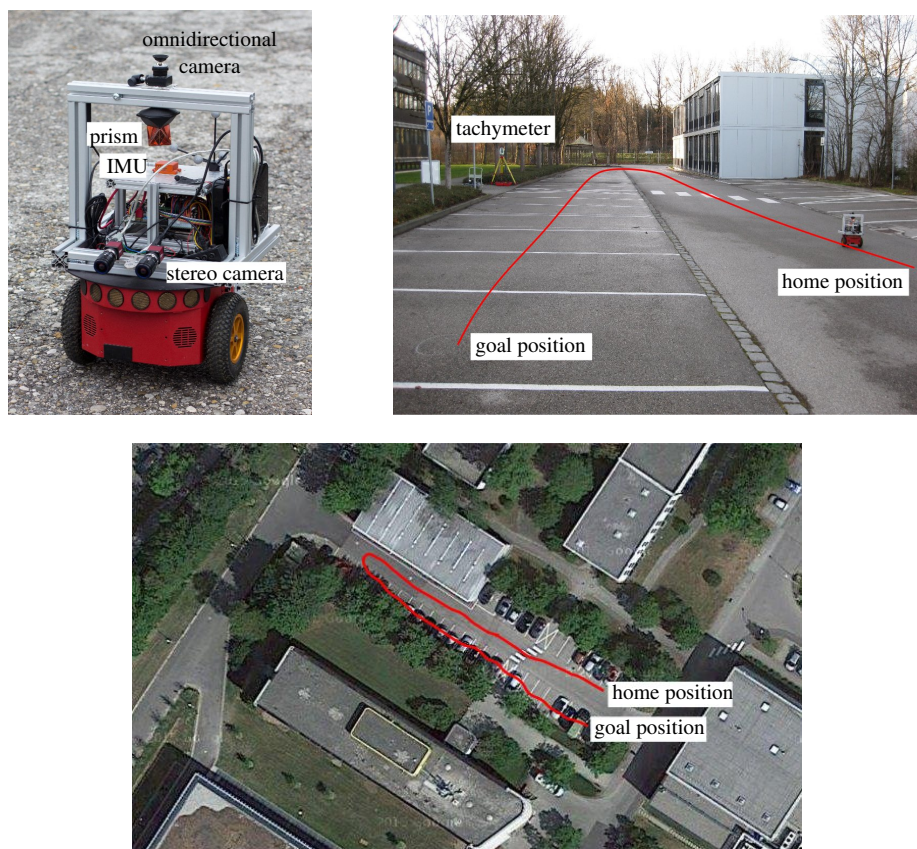


Figure 26. Pioneer robot and experimental setup with path for the outdoor carpark experiment (satellite imagery ©2015, DigitalGlobe, GeoBasis-DE/BKG)

6.2.1 Carpark Experiment In a first experiment, we remotely controlled the robot along a U-shaped path of 98.7m length in an empty carpark between two buildings. Fig. 26 shows the experimental setup and visualizes the robot path in a satellite image.

For mapping, we chose the average angle error dissimilarity measure (ref. Eq. 2) and a dissimilarity threshold of $\xi_{\delta_{\text{diss}}^{\text{map}}} = 0.12\text{rad}$ (approx. 6.9°) for creating a new viewframe. The angle threshold for creating the Trail-Map was $\delta_{\text{ang}} = 10^\circ$. With these parameters, the robot recorded 49 viewframes and 36024 landmark views in total. The resulting Trail-Map had 31323 landmark views spread over 17 levels, which means memory savings of 15% compared to storing all landmark views. Again, the majority of landmark views (88%) was in level 1 of the Trail-Map. Although each viewframe had 735 landmark views on average, only a mean of 122 of them could be matched

between successive viewframes. Thus, all the unmatched landmarks remained in level 1 of the map, additionally to the landmark views that changed their bearing by more than δ_{ang} compared to the last viewframe. Statistics of the mapping process are summarized in Table 3. The mapping trajectory is shown in Fig. 27.

Table 3. Mapping statistics of the carpark experiment

Trail-Map angle threshold δ_{ang}	10°
mapping dissimilarity threshold $\xi_{\delta_{\text{ang}}}^{\text{map}}$	$0.12\text{rad} \approx 6.9^\circ$
path length	98.7m
observed LVs	36024
recorded viewframes	49
levels in Trail-Map	17
LVs in in full Trail-Map	31323
avg. LVs per viewframe	735 ± 25.5
avg. matches between viewframes	122 ± 24.7
LVs in Level 1	27681 (88.4%)
LVs in Level 2	1967 (6.3%)
LVs in Level 3	720 (2.3%)
LVs in Level 4	383 (1.2%)
LVs in Level 5	253 (0.8%)

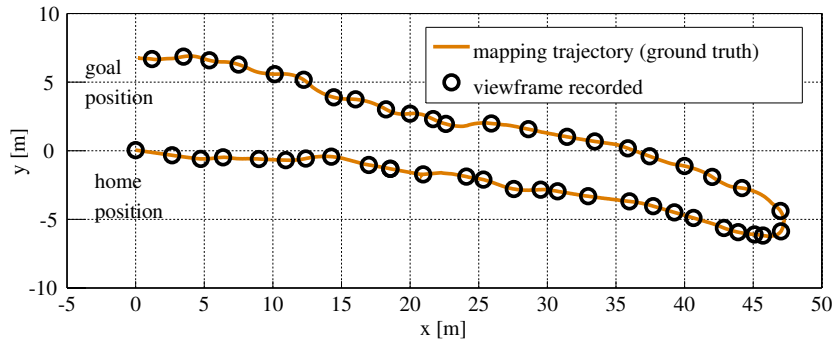


Figure 27. Mapping trajectory for the outdoor carpark experiment

For homing, we pruned the Trail-Map – except for the home viewframe – by 2 levels, which proved to be a good trade-off between memory consumption and path accuracy in the indoor experiments. As a result 2355 landmark views remained in the Trail-Map, which corresponds to approximately 250kB of memory. On average, each viewframe had 155 remaining landmark views after pruning. The dissimilarity measure for detecting a viewframe as reached was set to $\xi_{\delta_{\text{ang}}}^{\text{hom}} = 0.05\text{rad}$ average angle error (approx. 2.9°) and the robot computed homing vectors using the normalized improved difference vector model in Eq. 7, which is robust against outliers. The homing process

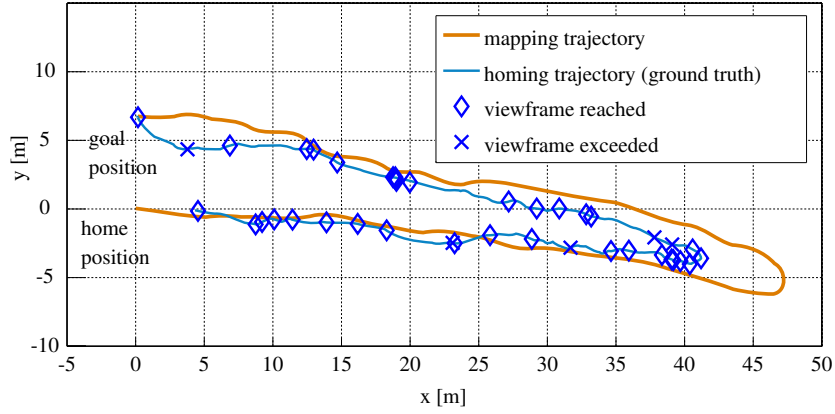


Figure 28. Homing trajectory for the Trail-Map pruned by 2 levels in the outdoor carpark experiment

took 14.4 minutes and the robot could drive at a maximal velocity of 0.2m/sec. The robot's homing trajectory is shown in Fig. 28. This figure shows the locations where the robot assumed to have reached or exceeded a viewframe. The robot assumes to have exceeded a viewframe when the homing vector jumps by more than 90° (Jäger et al. (2013)). As the figure shows, the robot performed homing successfully, but took a short-cut near the curve. That could be caused by the bigger size of the viewframes after pruning two levels of the map. We observed a similar behavior in our indoor experiments (ref. Section 6.1.4). During homing, the robot could match 20 landmark views on average, which is only 12.9% of the available landmark views in each viewframe. The robot's homing trajectory had an average path error of $0.88 \pm 0.60\text{m}$ and a maximal deviation from the mapping path of 2.5m. The endpoint error when the robot assumed to have reached the home position was 4.33m. This large deviation from the home position could be caused by the lack of nearby landmarks. The homing statistics are summarized in Table 4.

Table 4. Homing statistics of the carpark experiment

homing dissimilarity threshold $\xi_{\delta_{\text{diss}}}^{\text{hom}}$	$0.05\text{rad} \approx 2.9^\circ$
pruned levels	2
remaining LVs in Trail-Map	2355
memory requirements	$\approx 250\text{kB}$
average number of LVs per viewframe	155 ± 29.0
average number of matched LVs per viewframe	20 ± 6.4 (12.9%)
average path deviation	$0.88 \pm 0.60\text{m}$
maximal path deviation	2.5m
endpoint error	4.33m

This experiment shows that the robot is able to retrace a path of nearly 100m length with a Trail-Map size of about 250kB. However, the robot moved through a bounded environment and was theoretically able to detect the majority of landmarks it observed at the home position along the whole path. Since this might have an effect on the memory consumption, we will present another experiment in a segmented environment.

6.2.2 Urban Experiment In another experiment, we remotely controlled the robot along a curved path of 87.4m length on a road between buildings, so that the landmarks which were visible from the home position could not be observed from the goal position. Fig. 29 shows the commanded path in a satellite image. The robot used the same mapping parameters as in the previous experiment ($\xi_{\delta_{\text{diss}}^{\text{map}}}^{\text{map}} = 0.12\text{rad}$ average angle error and $\delta_{\text{ang}} = 10^\circ$) and recorded 49 viewframes with an average of 601 landmark views per viewframe. The resulting Trail-Map had 16 levels and 25130 landmark views. Again, 87% of all landmark views were in level 1 of the Trail-Map, and only an average of 107 landmarks could be matched between successive viewframes. Table 5 shows the statistics of the urban mapping results. The mapping trajectory with the viewframes is shown in Fig. 30. As can be seen, the viewframes were closer together near the curve in the area where the robot left the carpark, because in this region the landmarks were closer to the robot trajectory than in the rest of the environment.

Table 5. Mapping statistics of the urban experiments

Trail-Map angle threshold δ_{ang}	10°
mapping dissimilarity threshold $\xi_{\delta_{\text{diss}}^{\text{map}}}^{\text{map}}$	$0.12\text{rad} \approx 6.9^\circ$
path length	87.4m
observed LVs	29437
recorded viewframes	49
levels in Trail-Map	16
LVs in in full Trail-Map	25130
avg. LVs per viewframe	601 ± 82.2
avg. matches between viewframes	107 ± 29.5
LVs in Level 1	21845 (86.9%)
LVs in Level 2	1780 (7.1%)
LVs in Level 3	725 (2.9%)
LVs in Level 4	301 (1.2%)
LVs in Level 5	189 (0.8%)

For homing, we first pruned the Trail-Map by 2 levels but did not prune the home viewframe, so that the resulting Trail-Map had 2195 landmark views. This corresponds to 220kB of memory. The homing parameters were also equal to the carpark experiment ($\xi_{\delta_{\text{diss}}^{\text{hom}}}^{\text{hom}} = 0.05\text{rad}$ average angle error). The homing vectors were computed using the normalized improved difference vector model in Eq. 7 at a rate of about 3.4Hz parallel to the pose estimation process on the Pioneer's Intel Core i7 CPU with 2.7GHz. The homing trajectory is shown in Fig. 31. We had a tracking drop out of a few meters, when the tachymeter lost the connection to the prism. We filled the gap in the plot



Figure 29. Experimental setup and robot path for the outdoor urban experiments (satellite imagery ©2015, DigitalGlobe, GeoBasis-DE/BKG)

with the pose estimate that the robot obtained by fusing IMU data and visual odometry. The robot successfully retraced the path with an average path error of $0.77 \pm 0.51\text{m}$, a maximal path error of 2.3m and an endpoint error of 1.74m. On average, the robot could match 22 landmark views in each image with the landmark views in the corresponding goal viewframe. Fig. 32 presents some sample images of the homing run with color-coded landmark matches. It can be seen that after pruning 2 levels mainly landmarks in the direction of motion were left. For this reason, it is important to choose a homing vector calculation method can deal with landmark clusters located in the direction of motion, such as the proposed improved difference vector model or its normalized version. Furthermore, the figure shows that the red landmarks, which are in the highest levels of the Trail-Map, could be observed over a large range of viewframes.

To evaluate if homing is still possible with even less memory, we pruned the same Trail-Map by 3 levels, resulting in 1484 remaining landmark views. The robot performed homing for nearly 60 meters but then failed to compute stable homing vectors. On average, the robot matched only 17 landmarks in each image with the corresponding goal viewframe. The homing vectors started jumping, so that the robot

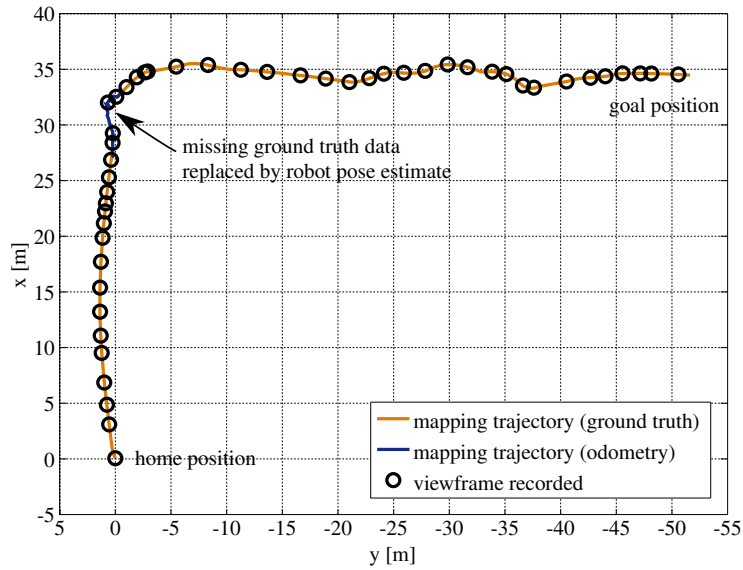


Figure 30. Mapping trajectory for the outdoor urban experiment

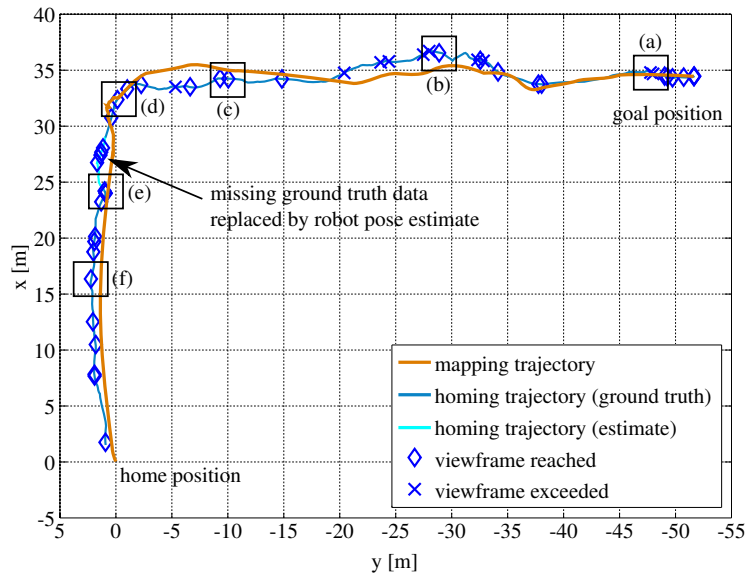


Figure 31. Homing trajectory for the Trail-Map pruned by 2 levels (2195 LVs) with marked image locations for Fig. 32 in the outdoor urban experiment

assumed it had exceeded the viewframes and started homing to the next one. In the end, the current goal viewframe was too far away from the robot's current position, so that not enough landmark matches could be found. We stopped the homing process at this

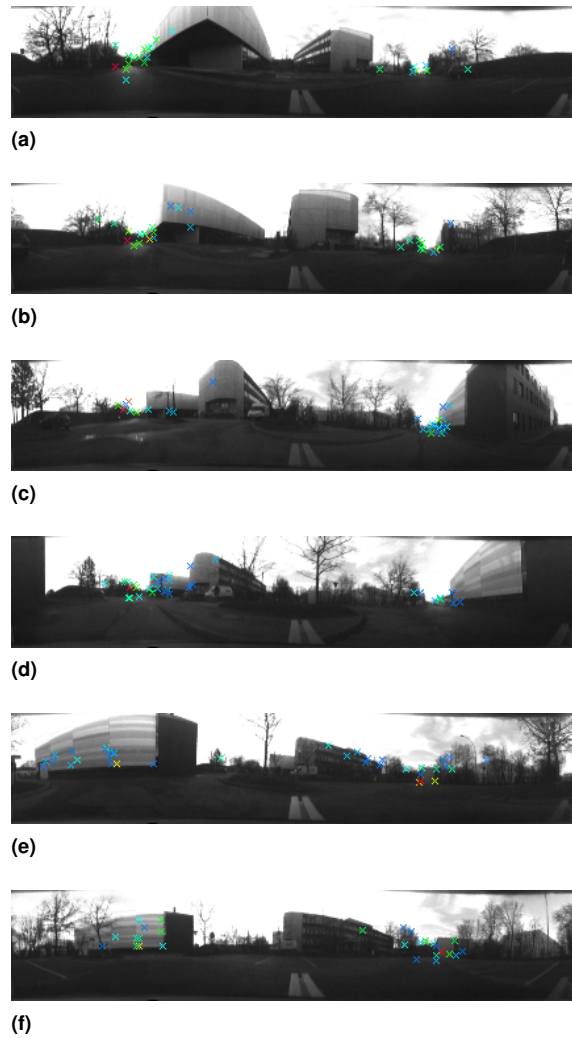


Figure 32. Sample panorama images with matched landmarks during homing in the urban experiment with the Trail-Map pruned by 2 levels for robot positions shown in Fig. 31. Color code: blue-green-yellow-red corresponds to landmarks in levels 3-16 in this order.

point. Fig. 33 shows the robot trajectory. Table 6 summarizes the statistics for the two homing experiments.

These experiments show that a Trail-Map size of about 220kB for 87m length is sufficient for the robot to retrace a path through a segmented environment, in which the visibility of landmarks is restricted to certain parts of the trajectory. Thus, we can assume that these memory requirements scale approximately linearly with the length of the traversed path, even for longer traverses of more than a kilometer in length, for example. The memory requirements in this experiment also correspond to about 250kB

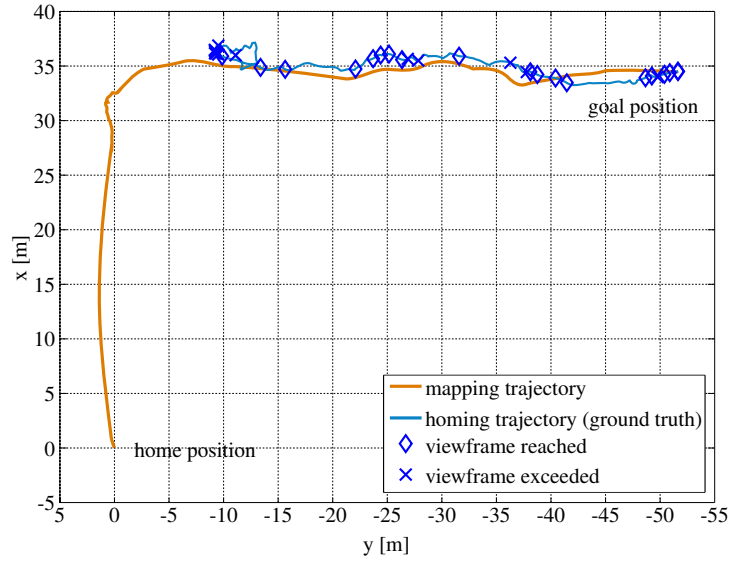


Figure 33. Homing trajectory for the Trail-Map pruned by 3 levels (1484 LVs) in the outdoor urban experiment. In this experiment, homing failed because the number of landmark matches was not sufficient to compute stable homing vectors.

Table 6. Homing statistics of the urban experiments

homing dissimilarity threshold $\xi_{\text{diss}}^{\text{hom}}$	$0.05\text{rad} \approx 2.9^\circ$	$0.05\text{rad} \approx 2.9^\circ$
pruned levels	2	3
remaining LVs in Trail-Map	2195	1484
memory requirements	$\approx 220\text{kB}$	$\approx 150\text{kB}$
avg. number of LVs per viewframe	137 ± 22.7	116 ± 29.7
avg. matched LVs per viewframe	22 ± 7.1 (16.0%)	17 ± 7.2 (14.7%)
avg. path deviation	$0.77 \pm 0.51\text{m}$	homing failed
maximal path deviation	2.3m	homing failed
endpoint error	1.74m	homing failed

for 100m as in the previous experiment in the bounded carpark environment. However, stronger pruning of the Trail-Map can lead to a failure of the homing process.

The real-world experiments have shown that a robust landmark matching method is the key component for robust homing using the Trail-Map. Nevertheless, we could demonstrate in different environments that successful homing is possible with less than 300kB of memory for a path of 100 meters length. Extrapolating the memory requirements for longer traverses results in average memory requirements of about 3MB per kilometer. Compared to the teach-and-repeat approach described by [Furgale and Barfoot \(2010a\)](#), which is based on overlapping metric submaps and requires about 350MB of data per kilometer, the Trail-Map-based homing approach requires two orders of magnitudes less memory and the computational complexity is also constant during mapping and homing. The Trail-Map also significantly outperforms the method proposed by [Krajník et al. \(2010\)](#), who reported memory requirements of 848MB for a path of 8km.

7 Conclusion

In this paper we presented a landmark-based navigation method for autonomous visual homing, which is solely based on landmark bearing angles and does not use metric distance information. Landmark observations and their angular configurations are stored in so-called viewframes at certain locations along the robot's path. The robot retraces this path by computing homing vectors, which point in the direction to align the robot's current landmark configuration with the one stored in the goal viewframe. In this work, we gave an overview of several existing homing vector computation methods and introduced new methods, which outperform the existing ones in case of landmark outliers and non-isotropic landmark distributions. The main contribution was the development of the Trail-Map, which is a novel data structure for scalable and non-redundant viewframe storage that also allows constant-time mapping and homing. It is based on the insight that the bearing angles to distant landmarks and landmarks in the direction of travel hardly change during motion (translation invariant landmarks), while the bearing angles of nearby landmarks change significantly as the robot passes them (translation variant landmarks). Thus, the Trail-Map stores the landmark observations in a hierarchical order of their level of translation invariance and avoids redundant storage of landmark observations that do not change significantly between subsequent viewframes. In case of memory shortage, the Trail-Map can be pruned by deleting the landmark views in the lower levels of the map. By doing so, only the information of quickly changing and possibly unstable landmarks and outliers is discarded, and the information of stable, translation invariant landmarks is preserved. In simulations we could show the superior performance of the Trail-Map compared to the LT-Map, which is the only existing scalable data structure for viewframe-based homing. We validated the approach in indoor and outdoor experiments with a robot using natural landmarks extracted from omnidirectional camera images, where we could show the scalability and memory efficiency of the method. In particular, we demonstrated in long-range outdoor experiments that a Trail-Map pruned by 2 levels with less than 300kB is sufficient to store and retrace a path of 100 meters length, which saves 97% of memory compared to state-of-the-art methods, while still offering constant runtimes for mapping and homing. Since the method achieved these values also in a segmented environment, in which the home and the goal location had no common landmarks, we

can assume that the memory requirements can be approximately linearly extrapolated for longer traverses of more than a kilometer in length, for example. Due to these properties, we also claim that a Trail-Map-based method is a promising alternative to state-of-the-art metric approaches for the task of robot homing.

The bottleneck of the whole method is the landmark detection and matching process, because Trail-Map-based homing heavily depends on correct and reliable landmark matches. Unstable landmarks unnecessarily inflate the Trail-Map, and false matches affect the correctness of the homing vector. Using a more robust landmark detector and matcher would improve the homing performance and decrease the memory requirements of the Trail-Map. Furthermore, the robot should be able to automatically identify and adjust the parameters of the homing method, for example the dissimilarity thresholds and the number of levels to be pruned. Additionally, the homing method should be combined with local obstacle avoidance to ensure that the robot always stays on a safe path.

Acknowledgements

We would like to thank Wolfgang Stürzl for providing the omnidirectional camera, its calibration and the image remapping functions, and Franz Andert and Thomas Jost for providing the outdoor ground truth measurement equipment.

References

- A. A. Argyros, K. E. Bekris, S. C. Orphanoudakis, and L. E. Kavraki. Robot homing by exploiting panoramic vision. *Autonomous Robots*, 19(1):7–25, 2005.
- M. Augustine, E. Mair, A. Stelzer, F. Ortmeier, D. Burschka, and M. Suppa. Landmark-Tree Map: A biologically inspired topological map for long-distance robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO) 2012, Guangzhou, China, 2012*.
- B. A. Cartwright and T. S. Collett. Landmark learning in bees: Experiments and models. *Journal of Comparative Physiology*, 151(4):521–543, 1983.
- B. A. Cartwright and T. S. Collett. Landmark maps for honeybees. *Biological Cybernetics*, 57(1-2):85–93, 1987.
- A. Cherubini and F. Chaumette. Visual navigation of a mobile robot with laser-based collision avoidance. *The International Journal of Robotics Research*, 32(2):189–205, 2013.
- A. Chilian, H. Hirschmüller, and M. Görner. Multisensor data fusion for robust pose estimation of a six-legged walking robot. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS) 2011*, pages 2497–2504, 2011.
- D. Dai and D. T. Lawton. Range-free qualitative navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) 1993*, pages 783–790, 1993.
- M. O. Franz, B. Schölkopf, H. A. Mallot, and H. H. Bülthoff. Where did I take that snapshot? Scene-based homing by image matching. *Biological Cybernetics*, 79(3):191–202, 1998.
- P. Furgale and T. Barfoot. Stereo mapping and localization for long-range path following on rough terrain. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) 2010*, pages 4410–4416, 2010a.

- P. Furgale and T. D. Barfoot. Visual teach and repeat for long-range rover autonomy. *Journal of Field Robotics*, 27(5):534–560, 2010b.
- T. Goedemé, T. Tuytelaars, L. Van Gool, G. Vanacker, and M. Nuttin. Feature based omnidirectional sparse visual path following. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2005*, pages 1806–1811, 2005.
- H. Hirschmüller. Stereo processing by semi-global matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341, February 2008.
- B. Jäger, E. Mair, C. Brand, W. Stürzl, and M. Suppa. Efficient navigation based on the landmark-tree map and the Z^∞ algorithm using an omnidirectional camera. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2013*, pages 1930–1937, 2013.
- K. Kawamura, A. B. Koku, D. M. Wilkes, R. A. Peters, and A. Sekmen. Toward egocentric navigation. *International Journal of Robotics and Automation*, 17(4):135–145, 2002.
- J. Kosecka, L. Zhou, P. Barber, and Z. Duric. Qualitative image based localization in indoors environments. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages II–3, 2003.
- T. Krajník, J. Faigl, V. Vonásek, K. Košnar, M. Kulich, and L. Přeučil. Simple yet stable bearing-only navigation. *Journal of Field Robotics*, 27(5):511–533, 2010.
- D. Lambrinos, T. Labhart, and R. Pfeifer. A mobile robot employing insect strategies for navigation. *Robotics and Autonomous Systems*, 30(1-2):39–64, 2000.
- S. Leutenegger, M. Chli, and R. Y. Siegwart. BRISK: Binary robust invariant scalable keypoints. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV) 2011*, pages 2548–2555, 2011.
- M. Liu, C. Pradalier, Q. Chen, and R. Siegwart. A bearing-only 2D/3D-homing method under a visual servoing framework. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) 2010*, pages 4062–4067, 2010.
- E. Mair, M. Augustine, B. Jäger, A. Stelzer, C. Brand, D. Burschka, and M. Suppa. A biologically inspired navigation concept based on the Landmark-Tree Map for efficient long-distance robot navigation. *Advanced Robotics*, 28(5):289–302, 2014.
- Y. Matsumoto, M. Inaba, and H. Inoue. Visual navigation using view-sequenced route representation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) 1996*, pages 83–88, 1996.
- R. R. Murphy. *Disaster robotics*. MIT Press, 2014.
- S. Šegvić, A. Remazeilles, A. Diosi, and F. Chaumette. Large scale vision-based navigation without an accurate global reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2007*, pages 1–8, 2007.
- A. Stelzer, E. Mair, and M. Suppa. Trail-Map: A scalable landmark data structure for biologically inspired range-free navigation. In *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO) 2014, Bali, Indonesia*, pages 2138–2145, 2014.
- A. Stelzer, M. Suppa, and W. Burgard. Trail-Map-based homing under the presence of sensor noise. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2015*, pages 929–936, 2015.
- A. Tapus, F. P. Battaglia, and R. Siegwart. The hippocampal place cells and fingerprints of places: Spatial representation animals, animats and robots. In *Proceedings of the 9th Intelligent*

-
- Autonomous Systems Conference (IAS)*, pages 104–113, 2006.
- A. Vardy. Long-range visual homing. In *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO) 2006*, pages 220–226, 2006.
- K. Weber, S. Venkatesh, and M. Srinivasan. Insect-inspired robotic homing. *Adaptive Behavior*, 7(1):65–97, 1999.
- J. Zeil, M. I. Hofmann, and J. S. Chahl. Catchment areas of panoramic snapshots in outdoor scenes. *Journal of the Optical Society of America A*, 20(3):450–469, 2003.
- A. M. Zhang and L. Kleeman. Robust appearance based visual route following for navigation in large-scale outdoor environments. *The International Journal of Robotics Research*, 28(3): 331–356, 2009.