To appear in *Advanced Robotics*
Vol. 28, No. 5, March 2014, 1–18

FULL PAPER

A Biologically Inspired Navigation Concept Based on the Landmark-Tree Map for Efficient Long-Distance Robot Navigation

Elmar Mair[a,*], Marcus Augustine[b], Bastian Jäger[a], Annett Stelzer[a], Christoph Brand[a], Darius Burschka[c], Michael Suppa[a]

[a] *German Aerospace Center (DLR), Institute of Robotics & Mechatronics, Wessling, Germany.*
[b] *Otto-von-Guericke Universität, Workgroup Computer Systems in Engineering, Magdeburg, Germany.*
[c] *Technische Universität München (TUM), Dept. of Computer Science, Garching b. München, Germany.*

(*received March 2013*)

Map-based navigation is a crucial task for any mobile robot. Usually, in an unknown environment this problem is addressed by applying Simultaneous Localization and Mapping (SLAM) based on metric grid-maps. However, such maps are in general rather computational expensive and do not scale well. Insects are able to cover large distances and reliably find back to their nests, although they are quite limited in their resources. Inspired by theories on insect navigation, we developed a data structure which is highly scalable and efficiently adapts to the available memory during run-time. Positions in space are memorized as snapshots, which are unique configurations of landmarks. Unlike conventional snapshot or visual map approaches, we do not simply store the landmarks as a set, but we arrange them in a tree-like structure according to the relevance of their information. The resulting navigation solely relies on the direction measurements of arbitrary landmarks. In this work we present the concept of the Landmark-Tree map and apply it to a mobile platform equipped with an omnidirectional camera. We verify the reliability and robustness of the LT-map concept in simulations as well as by experiments with the robotic platform.

Keywords: topological navigation; roadmap; landmark-tree; LT-map; bio-inspired

1. Motivation

Autonomous navigation is a highly complex task, which often requires most resources on mobile robots. The navigation problem can in general be divided into *local navigation*, where the robot moves within its close surroundings relative to a local frame and solves a specific task, and *global navigation*, where the robot travels between task-related workspaces, *e.g.* from its base to the location of interest.

Typical local navigation problems include obstacle avoidance, collecting probes and assembling components. To accomplish the respective task, the robot needs to have detailed geometrical knowledge of its workspace for precise localization and for planning accurate trajectories. If no a priori maps are available, the robot has to generate a complete environment model itself.

Since robotic systems tend to get smaller and more agile and the number of applications in which robots need to cover long distances is growing, the global navigation problem gains importance. To name a few examples, think of a Micro Aerial Vehicle (MAV), which starts at a rescue team and flies in a specific direction to search for people who require help, or a rover on a foreign planet which has to find back to the base station for analyzing the collected probes. These robots must be able to reliably reach a previously visited, possibly far distant location.

[*]Corresponding author. Email: `elmar.mair@dlr.de`

Exact knowledge of their positions is not required, neither is an accurate representation of the environment. The robot's map of the environment only needs to contain as much information as necessary for reaching the goal location.

As should have become clear, the demands on local and global maps are not the same. While local navigation aims for an accurate localization within a specific, small-sized workspace, which requires a high metric resolution, a global navigation strategy tries to expand the workspace dimensions and, hence, focuses on a representation of the world which is as sparse as possible but still allows a reliable guidance. Solving both tasks with a single map inherently leads to trade-offs and an inferior performance of both tasks, especially on resource limited systems.

Powerful methods have been presented in literature how to solve the navigation task in mobile robotics. Most of them are based on metric maps which are used for both, local and global navigation. The use of metric grid-maps is wide-spread, since they provide geometric understanding of the scene and allow for an accurate trajectory planning. The drawback is that they are often expensive to calculate, because of the lack of a natural high level environment discretization. Furthermore, metric approaches suffer from high memory consumption for spacious environments because they do not scale satisfiably. Hence, tree data structures, like quadtrees, octrees, or k-d trees are used to provide an efficient representation of the metric space [1]. However, they are affected by computation overheads if the space-usage is not balanced or by high map maintenance costs for re-balancing. A special form of metric maps are the so-called rolling maps, which have a constant size and spatial resolution, but move along with the robot [2]. In that way an accurate local positioning can be provided, but the inherent drift of the global pose may prevent the robot from finding back to its origin, once the limits of the map are exceeded.

The poor scalability makes metric grid maps unfeasible in global navigation problems. Thinking of biological navigation behaviors, insects, animals, or humans do not seem to rely on metric maps for navigation [3]. In general, they do not even possess accurate senses for metric navigation. Their behavior suggests a navigation based on topological maps, the so-called cognitive maps, where the visual perception often plays the most important role. Topological maps resemble graphs and do not require to put the information in a metric context. Distinct places are represented as nodes, while edges denote the adjacency between different locations [4]. Because of this sparse representation of the environment, a high degree of memory efficiency is achievable in relation to the covered terrain and the corresponding map size. This results in a good scalability of such maps [5]. However, while the positions in metric maps are clearly defined by their precise coordinates specified in a common reference frame, topological maps do not allow this kind of accurate localization, what makes exact position control, *e.g.* for manipulation tasks, difficult.

Hence, in this work we stress to split the navigation problem and come up with separate, specialized solutions for both tasks: small metric maps for the local workspaces, but large topological roadmaps which interconnect them and allow the robot to efficiently switch between these areas. As illustrated in Fig. 1, delimiting the metric maps to the required areas of operation allows for a much higher spatial resolution of the workspace maps compared to using a conventional metric grid-map for covering the full operation area of the robot.

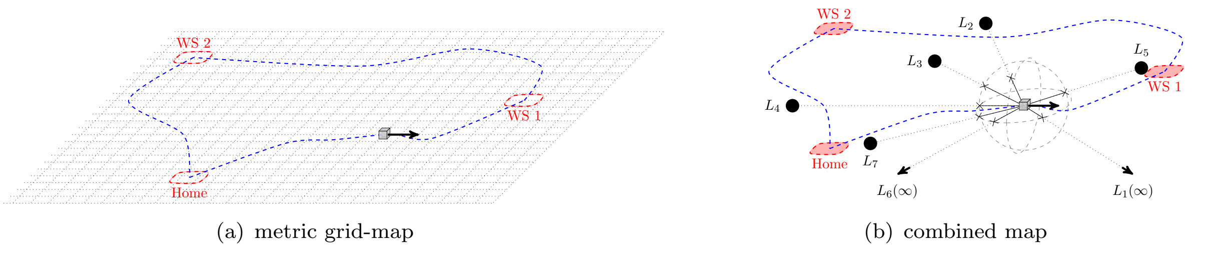


Figure 1. (a) Metric map covering the whole operation area of the robot. (b) Topological roadmap for connecting high-resolution metric maps of the single workspaces (WS1, WS2). The navigation between the workspaces is solely based on bearing-only measurements denoted by the crosses on the surrounding sphere pointing at the landmarks L.

This paper focuses on a mapping method for global navigation. Inspired by biological navigation concepts, we developed the Landmark-Tree map (LT-map), a novel topological mapping strategy which is designed to be highly scalable and to dynamically adapt to the available memory [6, 7]. The LT-map uses a hierarchical tree structure to resemble the dominance of the information in the environment. For this, it solely relies on bearing-only measurements of landmarks and does not require any metric distance information. The concept of the LT-map is applied to a mobile platform equipped with an omnidirectional camera. The reliability of the concept and its robustness on memory restrictions are verified in simulations and experiments.

The remainder of this paper is structured as follows. First, we want to motivate our work with some insights from biology and discuss the related work in Section 2, before we introduce the LT-map and the navigation based on it in Section 3. The map generation and its usage is evaluated in Section 4 based on synthetic data and experiments on a robotic platform. We conclude the work in Section 5, discussing the advantages and the limitations of the presented approach and providing an outlook on future work.

2. Related Work

Insects only have access to a restricted nervous system, but still manage the global navigation task with high robustness and reliability. Ants for example, which have a 0.1 milligram sized brain, are capable of locomotion, sensing and reasoning tasks in unpredictable, complex conditioned and often extremely changing habitats [8]. A vast amount of experiments with insects have been conducted, backing the existence and functionality of their navigational toolkit [8–12]. Experiments show that insects rely heavily on visual cues [13]. Their surrounding panorama plays an important role for global navigation [13]. In order to orient themselves in their environment, insects use view-dependent learning of visual scenes from particular vantage points [8]. They seem to employ a retinotopically organized image matching. This means that insects store their retinal image – the image they visually perceive at a distinguished place – and most likely, those memories are internally linked with each other [8]. In order to recognize a place and navigate themselves to a food source or their nest, they consequently compare their currently perceived retinal image with their memories [14]. However, it is not clear which information insects store [15] – whether they use the full image or another signature to memorize a location.

Based on experiments with bees, Cartwright and Collett [9, 10] developed the *snapshot-model*, which was one of the first concepts in biology that presented the relationship between insect navigation and their ability to memorize positions in space by the configuration of surrounding objects. This model does not store the complete image but the perceived configuration of landmarks at a certain location in a so-called snapshot. This contains a circular projection of the surrounding landmarks including the angle configurations and the sizes at which the landmarks currently appear on the insect's retina. Navigation is performed by moving in a direction so that the currently perceived landmark configuration gets aligned with the stored snapshot.

In literature on robotics, various denotations are used for concepts similar to the snapshot model. Dai introduced the term *viewframe*, which describes a discrete place as a set of landmarks that are observable from a location and their corresponding relative angles [16]. Thus, a viewframe is a circular projection of the surrounding landmarks – a landmark panorama. The conceptionally broader and more abstract approach by Levitt et al. [17], which additionally uses range estimates, employs the term *orientation regions*. Kawamura [18] transfers the viewframe concept into 3D space. A distinct place is described by a configuration of landmarks and their relative angles on a dome-like structure surrounding the robot.

When the snapshot approach is used for navigation, the robot computes motion commands by comparing its current observation with a snapshot of its goal location, similar to visual servoing [19]. The Average Landmark Vector (ALV) model [20] computes motion vectors between two distinct snapshots from bearing-only landmark measurements in a very efficient way.

The biological concept of navigation using snapshots is also related to approaches in the field of topological mapping. Various navigational approaches that use solely topological maps can be found in literature [21–23]. The method proposed by Franz et al. [24] creates nodes in a topological graph that represent distinct places only in case the environment has changed significantly. Winters et al. introduce a topological map, which triggers a visual servoing mechanism at crucial locations, *e.g.* narrow passages [25]. Like in these approaches, topological maps are in general used for loop closure detection, rough localization or to trigger some behaviours. The locations are defined by signatures derived from the images, *e.g.* histograms [26] or landmarks [27]. However, none of these approaches allows an easy pruning of the information in case of memory shortage. Identifying the information in the map which is redundant or not crucial for localization is computationally expensive and requires a cumbersome evaluation of the complete map. Hence, in general, whole nodes (locations) are discarded or the map is equally thinned out in the metric space, if such information is available.

There are also hybrid approaches that enrich the relating edges in topological maps with additional metric information. The advantage is that the stated idiothetic information error is bounded and reset when a node is reached [5]. Thrun [28] presents such a hybrid approach, using occupancy grid-based maps to build a metric map, serving for high precision movement planning. Tomatis et al. [29] use a hierarchical approach that consists of a global topological map with embedded local metric maps which are associated to each topological node. Stachniss et al. [30] create a metric map using probabilistic methods and, based on the generated map, estimate the topology of the environment.

In our work we focus on how to connect workspaces without any metric information, solely using bearing-only measurements of landmarks. This comes at the cost of not having a complete representation of the environment, *i.e.* a trajectory between two arbitrary points cannot be generated. Instead, the path is stored as a topological roadmap, which offers not as much navigation flexibility as a grid-based representation, but it provides the least overhead and, thus, the most efficient way to store the required information.

The method presented in this paper is inspired by the snapshot model developed by Cartwright and Collett [9, 10] and uses Dai's viewframe concept [16]. Thus, we use the term *viewframe* for describing the landmark configuration observed at a certain location in space. Different to Dai's presented method, our work employs alternate, cost-function-based techniques to compare viewframes. As far as the topological map is concerned, the presented method is also similar to work by Franz et al. [24]. However, in contrast to conventional visual maps, we present a novel approach for organising the landmarks of the viewframes in a tree structure that implicitely orders the landmarks by their distance from the robot without ever directly measuring it. That allows easy pruning in case of memory shortage and, thus, significantly improves scalability.

3. The Landmark-Tree map algorithm

Let us first define a viewframe $\mathcal{V}$ as a representation of a distinct location in the three-dimensional Euclidean space[1] $\mathbb{R}^3$ by a unique configuration of landmarks. Each landmark L_i is identified by a unique tuple $L_i = (\boldsymbol{d}_i, \boldsymbol{\phi}_i)$, consisting of its descriptor $\boldsymbol{d}_i$ and its bearing angle $\boldsymbol{\phi}_i$ containing azimuth and elevation under which the landmark i is seen. We assume the viewframes to be all rotationally aligned with each other, *e.g.* by using compass information. The relation between $\boldsymbol{l}_i$, the unit vector pointing in the direction of landmark i, and $\boldsymbol{\phi}_i$ can be described by

$$\boldsymbol{l} = (\cos(\phi_a) \ \sin(\phi_a) \ \sin(\phi_e))^T , \quad (1)$$

[1] Without loss of generality for other dimensional spaces.

where $\boldsymbol{\phi} = (\phi_a \ \phi_e)^T$, and ϕ_a and ϕ_e are the azimuth and elevation angles, respectively.

Considering bearing-only measurements, a translational motion results in large bearing changes of close landmarks, whereas more distant ones hardly change their bearing. Therefore, the closer a landmark, the more accurate translational information can be derived. In context of global navigation, a high position accuracy, which comes at the cost of a large number of landmarks, is in general not required and discarding dispensable features saves a significant amount of memory.

A straightforward solution for building a global navigation map would store each viewframe as a node of a topological map. However, that means that distant landmarks, which are present in several viewframes, are stored redundantly. Furthermore, in case of memory shortage, there is no easy way to identify and discard the close landmarks carrying the accurate position information. In our method, we introduce a hierarchy in the landmark structure which allows us to overcome these problems.

3.1 *Tree-based map representation*

The core idea is to arrange the detected landmarks into a tree-like structure (the *Landmark-Tree map*), sorting them from *global* to *local* information. Considering that the angles of landmarks change slower the greater the distance of the landmark is, we will use this information as sorting criterion.

The construction of the LT-map is best explained by an example. Fig. 2 illustrates a scenario where a robot moves through an outdoor scenery and shows the resulting LT-maps. There are local landmarks, like stones (L_3, L_4, L_8, L_9), far distant landmarks, like mountains (L_6, L_7), and landmarks in between, like trees (L_1, L_2, L_5). First, the LT-map is initialized with an empty root node. The angles and descriptors of the landmarks of the first view $\mathcal{V}_1$ are stored in the first child of the root as shown in Fig. 2(b) (the landmarks L_8 and L_9 are not yet visible). If the robot starts moving to $\mathcal{V}_2$, the local landmarks change their bearing angles and are stored in a new leaf on the right side, generating the tree in Fig. 2(c). The upper node contains all landmarks whose bearing angles remained within a specified limit. Once the robot reaches $\mathcal{V}_3$, also the angles under which L_2 and L_5 are seen change. The local landmarks L_3 and L_4 are not visible anymore, which results in the tree depicted in Fig. 2(d). The robot keeps on moving and, finally, also L_1 changes in $\mathcal{V}_4$, as denoted in Fig. 2(e). At $\mathcal{V}_5$ the robot starts measuring the local landmarks L_8 and L_9 and adds them, together with the changed landmark L_5, in a new leaf, as shown in Fig. 2(f). The final tree in Fig. 2(g) contains a new set of leaves, which result from the bearing angle changes of the local landmarks L_8 and L_9. The algorithm keeps adding nodes or full branches at the same side of the tree to maintain the temporal order.

As a consequence, all nodes along a complete branch, from the root to a leaf, represent a certain viewframe and, thus, a specific location. The landmarks in the upper nodes did not change their bearings within a certain threshold for long parts of the exploration trail, which means they are translation-invariant and, therefore, correspond to far distant objects. The landmarks in the lower nodes and leaves changed their bearings quickly, which means they belong to close objects. In the lower levels, we also find the landmarks which are only visible for a short time, due to volatile descriptors or occlusions. Hence, the horizontal axis of the tree expresses the non-metric distance of the viewframes from the endpoints of the route, which are on either sides of the tree. The leaves are the access points to the viewframes containing the landmarks of consecutive locations on the traveled path. The vertical axis reflects the visibility and the distance of the landmarks from the route and splits them stepwise into long-term visible, far distant as well as stable landmarks on the top and short-term visible, close or volatile ones in the leaves. These relationships are illustrated in Fig. 3.

By generating an LT-map to represent a certain route, we achieve both stated aims:

(1) The memory consumption is reduced, because the landmarks which are shared by con-

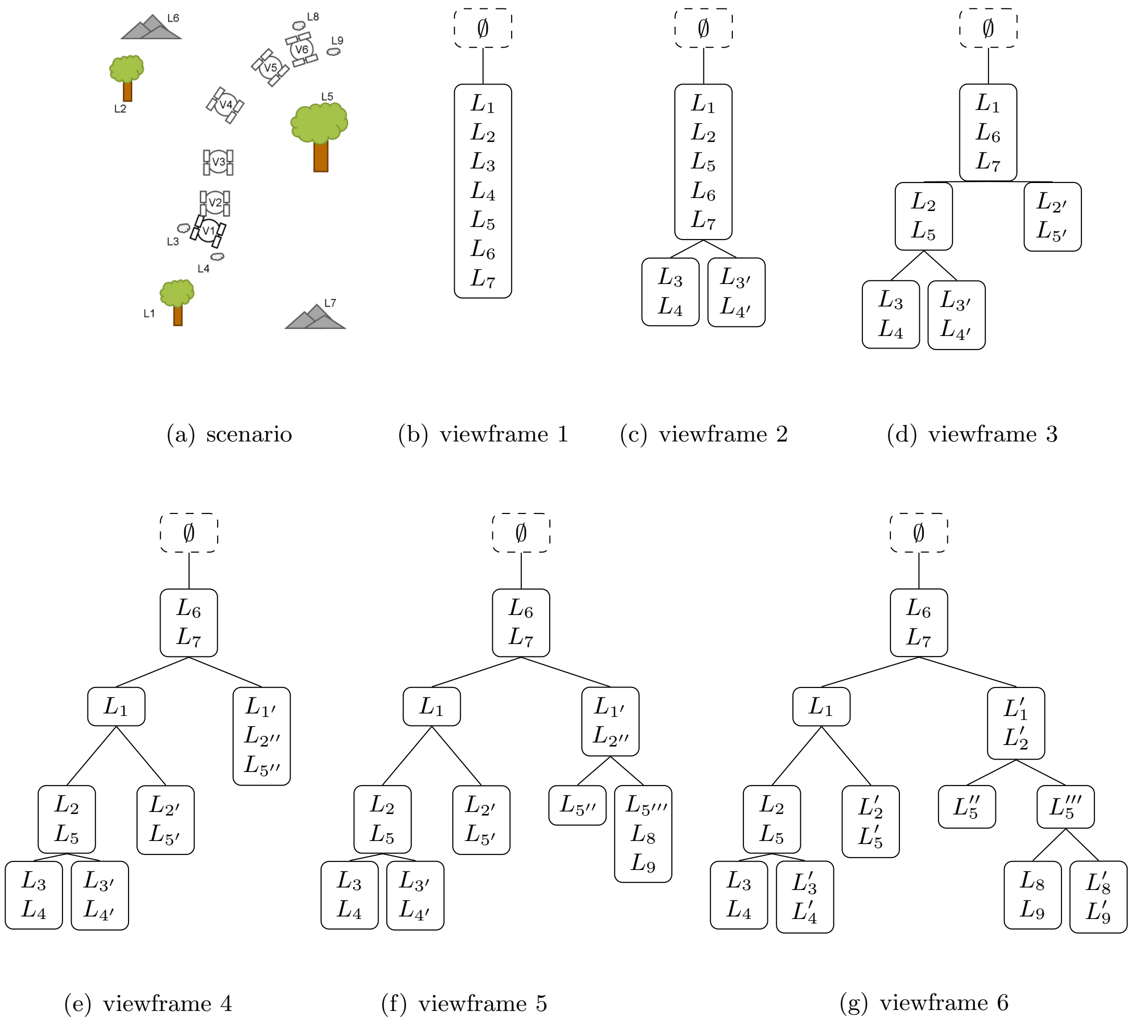


Figure 2. Construction of the LT-map. (a) Robot moving through an outdoor scene. (b)-(g) Configurations of the tree at the specified viewframe locations.

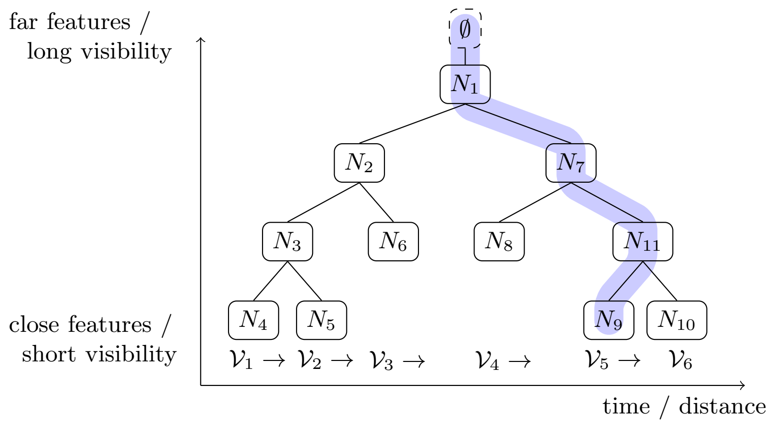

far features / long visibility
close features / short visibility
time / distance
$\emptyset$
N_1
N_2
N_7
N_3
N_6
N_8
N_{11}
N_4
N_5
N_9
N_{10}
$\mathcal{V}_1 \rightarrow \mathcal{V}_2 \rightarrow \mathcal{V}_3 \rightarrow \mathcal{V}_4 \rightarrow \mathcal{V}_5 \rightarrow \mathcal{V}_6$

Figure 3. The two dimensions of the landmark tree represent the travel distance (time) and the distance of the landmarks from the viewframe locations. Each branch of the tree is a viewframe (as highlighted for V_5). The nodes N along the branch contain the landmarks acquired at a certain location.

secutive viewframes and appear under a similar angle are only stored once. However, in case the angle changes quickly, several instances of a landmark will be stored. Thinking of how the appearance of features in, *e.g.*, cameras changes when a landmark is seen from different angles, this is actually an eligible feature. The algorithm uses the new descriptor when inserting a new landmark and, thus, it does not suffer from mismatches due to affine transformations and virtual features.

(2) In case of memory shortage one can easily prune the leaves of the tree and discard the local, short-term information while sticking to the more dominant global, long-term information. If the lower levels of the tree are cut, the robot does not follow the exact trajectory anymore, but takes shortcuts wherever the local information is missing. This does not pose a problem as long as each viewframe is in the catchment area of the neighboring ones.

Hence, *the size of a memory limited map can theoretically increase infinitely while exploring and only the probability of the robot to get lost between two node locations increases over time.* The remaining landmarks in the map inherently represent the best possible guidance-information acquired during exploration.

The hierarchical structure of the tree does not only help to save resources and expand the map dynamically, but it also eases loop-closure. A loop detector would only need to compare the landmarks in the upper level nodes, which represent dominant, translation invariant objects, and only in case of a match it has to proceed to the respective children. Hence, only a few landmarks need to be tested instead of all possible viewframes. Furthermore, the upper landmarks can be used to estimate the rotation like a visual compass, which eliminates the need for a magnetic compass to align the viewframes before processing.

The question remains how to decide when to acquire a new viewframe or which matching criterion to use for landmarks. The latter can easily be realized by applying two thresholds, one for the feature descriptor and one for the bearing angle tolerance, which reflects the tracking accuracy. The boolean function $\mathrm{c}(L_i, L_j)$ reveals, whether a landmark L_i corresponds to landmark L_j, and can be computed by

$$\mathrm{c}(L_i, L_j) = \begin{cases} 1 & \text{if } \|\, \mathrm{d}(\boldsymbol{d}_i,\ \boldsymbol{d}_j) \,\| < \zeta_{\boldsymbol{d}} \quad \bigwedge \quad \|\, \arccos\big(\boldsymbol{l}_i^T \boldsymbol{l}_j\big) \,\| < \zeta_{\boldsymbol{\phi}} \\ 0 & \text{else} \end{cases} , \tag{2}$$

where $\mathrm{d}(\cdot)$ depicts the distance measure used for the landmark descriptor and $\zeta_{\boldsymbol{d}}$ and $\zeta_{\boldsymbol{\phi}}$ denote the applied thresholds.

A new viewframe is acquired when the dissimilarity measure $\delta_{\boldsymbol{t}}$ between the landmark sets associated with the previously stored viewframe and the current observation exceeds a threshold ζ_{δ_t}. The value of $\delta_{\boldsymbol{t}}$ is computed as the weighted average angle between two corresponding landmark sets of size N by using a Pseudo-Huber cost function, such that

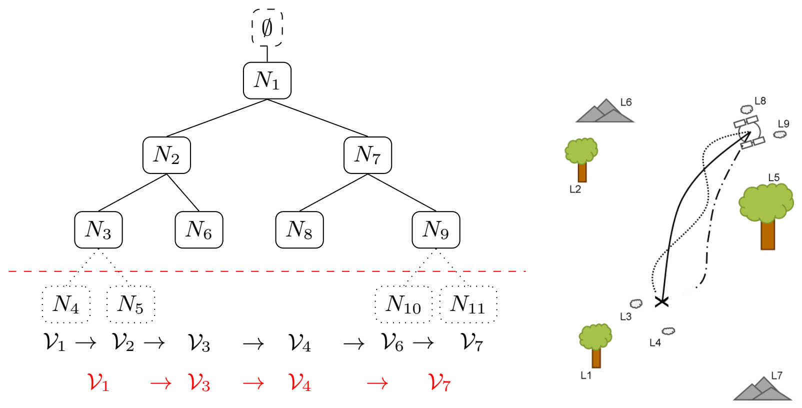
$$\delta_{\boldsymbol{t}} = \frac{1}{N} \sum_{i=1}^{N} 2b^2 \left(\sqrt{1 + \frac{\psi^2}{b^2}} - 1 \right) \quad \text{with} \quad \psi^2 = 2 - 2\, \boldsymbol{l}_i^T \boldsymbol{l}_i' \,, \tag{3}$$

where $\boldsymbol{l}_i$ and $\boldsymbol{l}_i'$ denote the unit vectors (as computed by Eq. 1) pointing to the same landmark i from two different, rotationally aligned viewframe locations. The cost function includes a control parameter b to weight small errors quadratically, but large errors linearly with slope $2b$. We compute ψ instead of simply the scalar product of the unit vectors to get the proper scaling of the cost function for small angles. If $\delta_{\boldsymbol{t}}$ is above the threshold ζ_{δ_t}, the robot assumes to have traveled enough distance and acquires a new viewframe. Due to the fact, that $\delta_{\boldsymbol{t}}$ relies on bearing measurements only, the algorithm automatically adapts to its environment, in the sense that less viewframes are acquired if few local landmarks with translational information are provided. Furthermore, in narrow passages, where a high accuracy is required, many viewframes are created, which allows for a safe navigation.

3.2 *Pruning the Landmark-Tree*

As illustrated in Fig. 4, the LT-map is pruned by simply trimming the leaves of the tree, which discards local, short-term information. After pruning, all branches are checked whether they contain sufficient landmarks for reliable navigation, otherwise the new leaves of these branches are deleted and the tree is compressed. For a resource limited system, the depth of the LT-map changes over time from a rather detailed representation of the route with a lot of local information to a wide tree with less depth, but which can span a long path. Due to the hierarchical structure of the tree, the pruning step is not computationally expensive and, hence, does in general not influence the performance of other tasks running on the robot.

Another nice effect of the efficient scalability of the LT-map is, that the robot does not have to know in advance how long the path will be which it is going to explore. It can just start acquiring viewframes at a high spatial resolution and prune the tree as soon as it is required. In that way, the map dynamically adapts its spatial resolution in a non-metric way to the available memory.


$\emptyset$
N_1
N_2
N_7
N_3
N_6
N_8
N_9
N_4
N_5
N_{10}
N_{11}
$\mathcal{V}_1 \rightarrow \mathcal{V}_2 \rightarrow \mathcal{V}_3 \rightarrow \mathcal{V}_4 \rightarrow \mathcal{V}_6 \rightarrow \mathcal{V}_7$
$\mathcal{V}_1 \rightarrow \mathcal{V}_3 \rightarrow \mathcal{V}_4 \rightarrow \mathcal{V}_7$
L6
L8
L9
L2
L5
L3
L4
L1
L7

(a) tree trimming by 1 level

(b) path following scenario

Figure 4. Tree pruning. (a) The lowest nodes are removed from the tree. (b) Exemplary scenario with three possible paths. cross: goal location; solid line: precisely followed trajectory based on the complete tree; dotted line: path followed after pruning the tree by one level as depicted in (a); dot-dashed line: trajectory resulting from the tree being trimmed by two levels.

3.3 *Roadmap navigation using the LT-map*

The route defined by an LT-map can be followed by simply moving from viewframe to viewframe and, thus, sequentially extracting the landmarks of each branch from the start to the goal viewframe. The direction vector to each such location can be computed based on the differences in the landmark bearings of the current measurement and the aspired reference viewframe stored in the tree. We will make use of the so-called *secant method* which was inspired by the *Average Landmark Vector* (ALV) model [20].

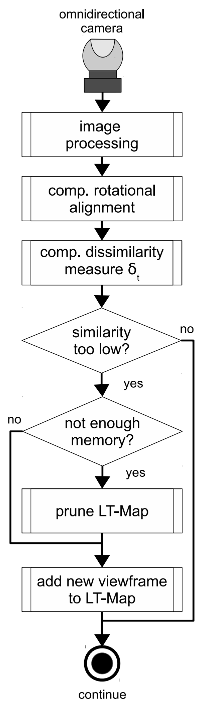
The basic concept of the secant method consists of the calculation of an average landmark vector by summing up the differences of all landmark correspondences after rotationally aligning them to the LT-map. The average landmark vector between the current landmark measurements and the reference viewframe represents the navigation direction $\boldsymbol{t}$ as

$$\boldsymbol{t} = \frac{1}{N} \sum_{i=1}^{N} \left(\boldsymbol{l}_i - \boldsymbol{l}'_i \right) \ , \tag{4}$$

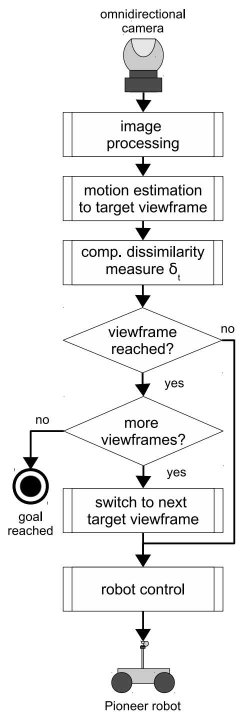
where N is the number of visible landmarks, $\boldsymbol{l}_i$ denotes the unit vector in the reference frame pointing to landmark $\boldsymbol{L}_i$ and $\boldsymbol{l}'_i$ the unit-length measurement vector pointing to the same landmark in the current frame. The advantages of the secant method compared to other methods, like the *tangential method* presented in [31], are that it estimates a direct navigation vector to the goal and its computation is highly efficient.

A viewframe is reached as soon as the dissimilarity criterion δ_t defined in Eq. 3 falls below a certain threshold. At that point, the next branch in the LT-map is chosen as new goal viewframe or the final destination is reached.

The mapping and the navigation algorithm are summarized as flow diagrams in Fig. 5.



(a) mapping



(b) roadmap navigation

Figure 5. These flow diagrams illustrate the mapping (a) and the roadmap navigation (b) algorithms.

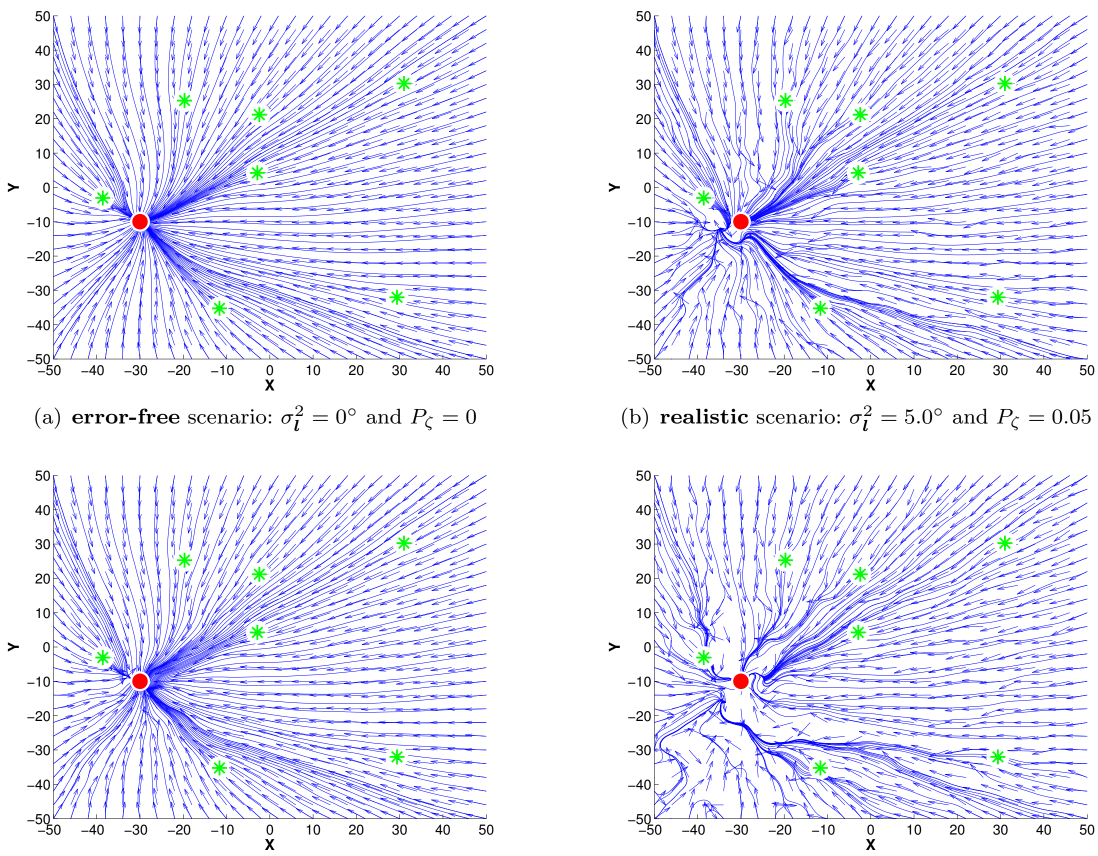
4. Experiments

In the following, we will show some experiments in a controlled 2D simulation environment and on a robotic platform. We evaluated the reliability of the method, its performance in case of memory shortage and in the presence of noise and outliers. The visualizations of the simulations do not include any units, so an arbitrary scale may be chosen.

4.1 *Robustness Evaluation*

To demonstrate the robustness of the method, we simulated two different noise terms: *angular measurement aberration* of the landmark angles and a percentage of *outliers*. The angular aberration was modeled as zero-mean white Gaussian noise with variance σ_l^2 and outliers were specified by a probability P_ζ that the actual measurement of a landmark is replaced by a random angle in the interval $[0; 360)^\circ$. We defined a realistic noise scenario to have an angular measurement noise of $\sigma_l^2 = 5.0^\circ$ and $P_\zeta = 0.05$.

Fig. 6 shows the movement directions expressed as unit vectors that were calculated for every position in a 50×50 example environment. The lines represent the streamlines from the border positions: they start at the marginal area of the grid and follow the calculated vectors (represented as arrows) contained in the vector field. A good homing performance is observed when the lines meet at the goal position, *i.e.*, the robot reaches the goal.



(a) **error-free** scenario: $\sigma_l^2 = 0°$ and $P_\zeta = 0$

(b) **realistic** scenario: $\sigma_l^2 = 5.0°$ and $P_\zeta = 0.05$

(c) **noise** scenario: $\sigma_l^2 = 30.0°$ and $P_\zeta = 0.0$

(d) **outlier** scenario: $\sigma_l^2 = 0.0°$ and $P_\zeta = 0.1$

Figure 6. 2D secant method vector fields: the homing vectors calculated with the secant method based on error-free and error-prone measurements. The asterisks denote the landmarks and the circle the goal location.

Fig. 6 shows the calculated homing vectors in different error-free and error-prone scenarios. Even though the noise terms obviously were influencing the calculated homing vector, a strong tendency towards the marked goal position is observable. Since the secant method computes the translation vector as an average over the landmark measurements, the quality of the estimate is more affected by outliers than by noise. In our simulations we did not apply any robustification methods. However, when applying the approach to real data in Section 4.3, we compute the direction of translation within a RANSAC [32] framework to suppress outliers.

4.2 *Simulation in 2D*

In the following, we show the simulation results of an LT-map based navigation in the 2D space with a virtual agent. In total, 500 random landmarks within the X- and Y-axis interval $[-200; 200]$ were chosen. The exploration track for the virtual agent is defined by 15 waypoints selected within an 80×80 square. Along this trajectory 53 locations for viewframe acquisition were chosen manually. At each location all landmarks are visible. The resulting LT-map consists of 23464 landmark entries, and 100 node relations. For all measurements we used $\sigma_l^2 = 5°$ and

$P_\zeta = 0.05$ as noise and outlier characteristics. The corresponding landmark tree is sketched in Fig. 7, showing the number of landmarks in the different nodes.

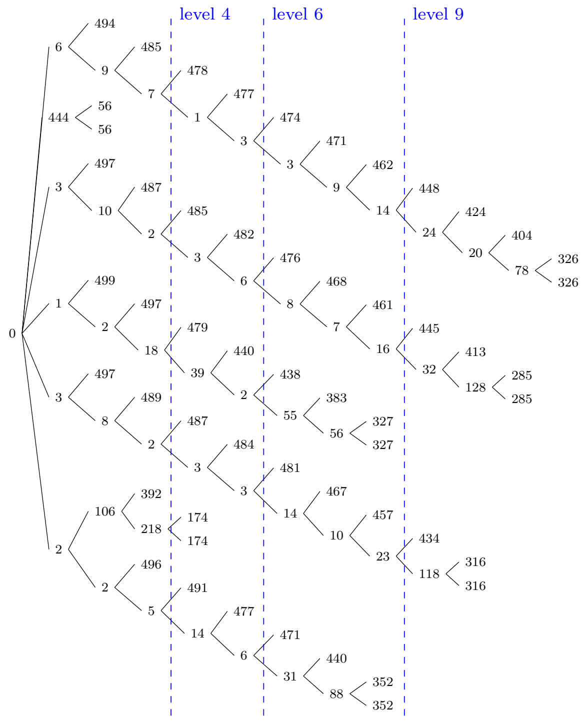


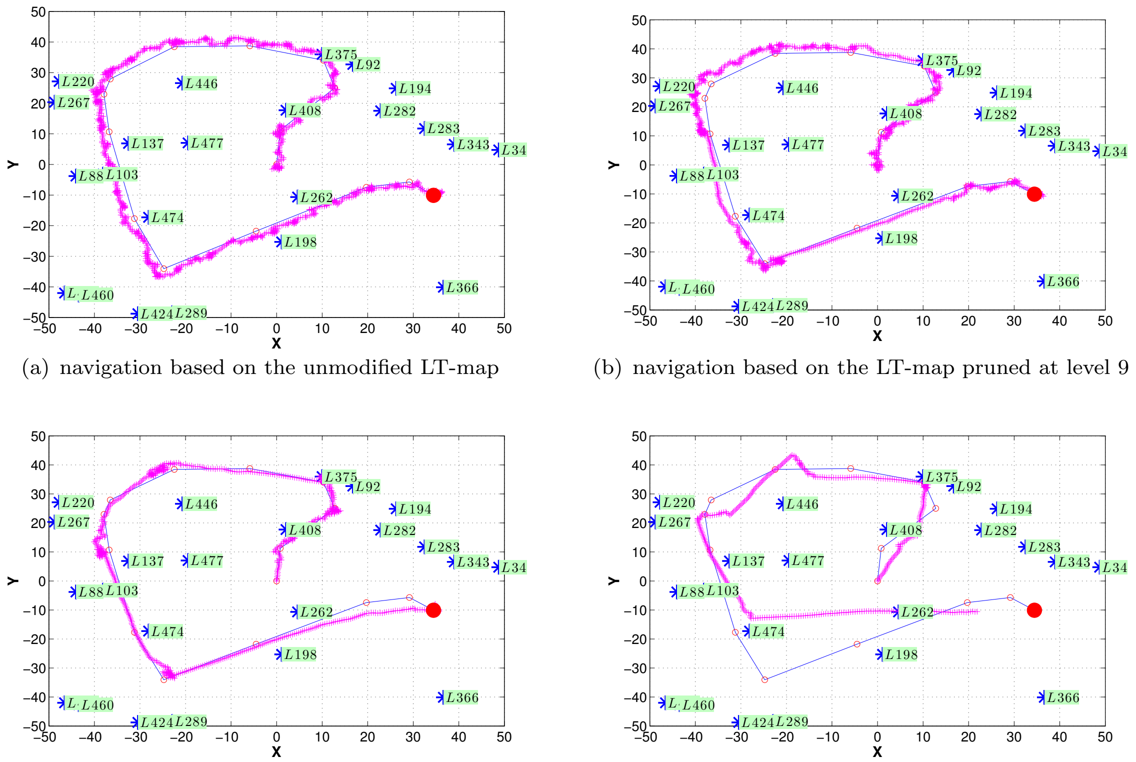
Figure 7. Original landmark tree and the pruning levels. The nodes in this representation contain only the number of landmark angles that are stored in the nodes.

The agent was commanded to follow the previously acquired trajectory using the created LT-map. For that, landmark measurements were simulated and the agent moved in the estimated direction with a step-size of 1. The path which relies on the complete tree provides a high congruency with the simulated learning path as shown in Fig. 8(a). The jitter in the trajectory is due to the simulated measurement aberrations.

Next, we pruned the landmark tree and evaluated the navigation performance, simulating less available memory. The trajectory depicted in Fig. 8(b) is based on 79.4% of the landmarks in the original LT-map. As information was purged from the tree and less intermediate viewframes were available, shortcuts were taken and the track was not followed with such a high precision anymore. This becomes even more apparent, if the tree is pruned at level 6, containing only 46.9% of the information, as shown in Fig. 8(c). The last run in Fig. 8(d) is only based on 24.7% of the original data, but the agent still achieves a feasible navigation performance and reaches the goal reliably. More statistic details to these four runs are shown in Table 1.

Tree height (trim)	Viewframes	Landmarks	Node relations
13 (0)	53 (100 %)	23464 (100 %)	100
9 (-4)	44 (83.0 %)	18642 (79.4 %)	82
6 (-7)	29 (54.7 %)	11001 (46.9 %)	52
4 (-9)	18 (34.0 %)	5793 (24.7 %)	30

Table 1. Mapping statistics for the 2D simulation



(a) navigation based on the unmodified LT-map

(b) navigation based on the LT-map pruned at level 9

(c) navigation based on the LT-map pruned at level 6

(d) navigation based on the LT-map pruned at level 4

Figure 8. Simulation of the navigation in 2D space using the secant method on the full and three pruned landmark trees, where $\sigma_l^2 = 5°$ and $P_\zeta = 0.05$. The recorded reference track is depicted as polygonal chain, whereas the waypoints (viewframes) are marked as circles. The navigation path performed by the agent is indicated by crosses.

4.3 *Experiments on a Robotic Platform*

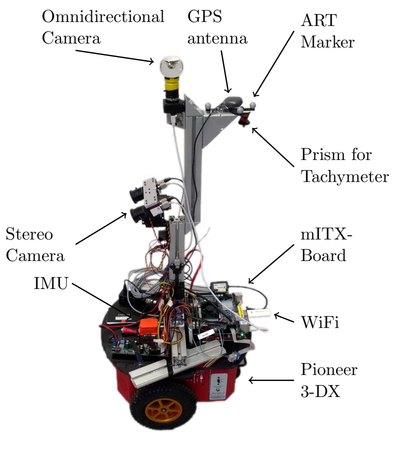
We used the Pioneer 3-DX robot illustrated in Fig. 9(a) as mobile platform for the experiments. The robot is equipped with several ground truth sensors and markers. For external referencing, we used an infrared tracking system[1] in the indoor experiments and a tachymeter in the outdoor scenario. The motion control is implemented on the Pioneer's embedded controller and commanded via serial interface. A Kontron KTQM67/mITX embedded motherboard is used as processing platform, equipped with an Intel Core i7 processor. All processing takes place on-board and can be monitored via WiFi.

We are focusing on an efficient algorithm which can run on resource limited platforms. For that, also the sensor should be compact and lightweight with low power consumption. A camera fulfills these criteria and allows to realize large aperture angles, which provides a better-posed computation of the navigation direction and more robustness. Hence, we chose to use an omnidirectional catadioptric camera as presented in [33].

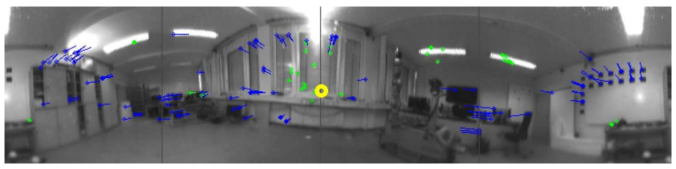
The image acquisition and processing runs at 5 Hz at a resolution of 474 × 474 pixels which can also be realized on much smaller platforms. The images are then unwarped to a panorama with 0.5°/px resolution in both dimensions, resulting in a usable image of size 720 × 172 pixels (see Fig. 9(c)).

We used BRISK [34] to extract and track image features, due to its high efficiency. Since our robot will not experience any tilt motion, we can apply U-BRISK, the rotational variant version of the algorithm, to further increase the efficiency. In our experiments, we leveled the output for the detector to 400 landmarks per image by a time delayed control. This resulted in about 200-300 valid landmark correspondences after matching. The unwarped omnidirectional images

[1] ARTtrack1 cameras from ART, http://www.ar-tracking.com



(a) mobile platform


(b) unwarped indoor image showing optical flow vectors

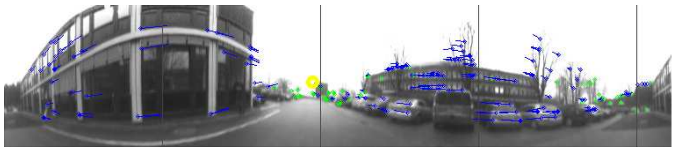

(c) unwarped outdoor image showing optical flow vectors

Figure 9. (a) Pioneer 3-DX mobile platform, equipped with ART marker and other sensors for ground truth. (b-c) unwarped images as used for processing. The blue optical flow vectors are used for translation and the green ones for rotation estimation. The yellow circle denotes the computed direction of translation.

are extended to include a small overlap, which compensates the descriptor size of U-BRISK and prevents that features get lost at the panoramic incision.

The rotation and the direction of translation between two landmark sets is computed by applying the Z_∞-algorithm [35] and, thus, separating both motion components in a RANSAC framework as illustrated in Fig. 9(c).

4.3.1 Indoor Experiments

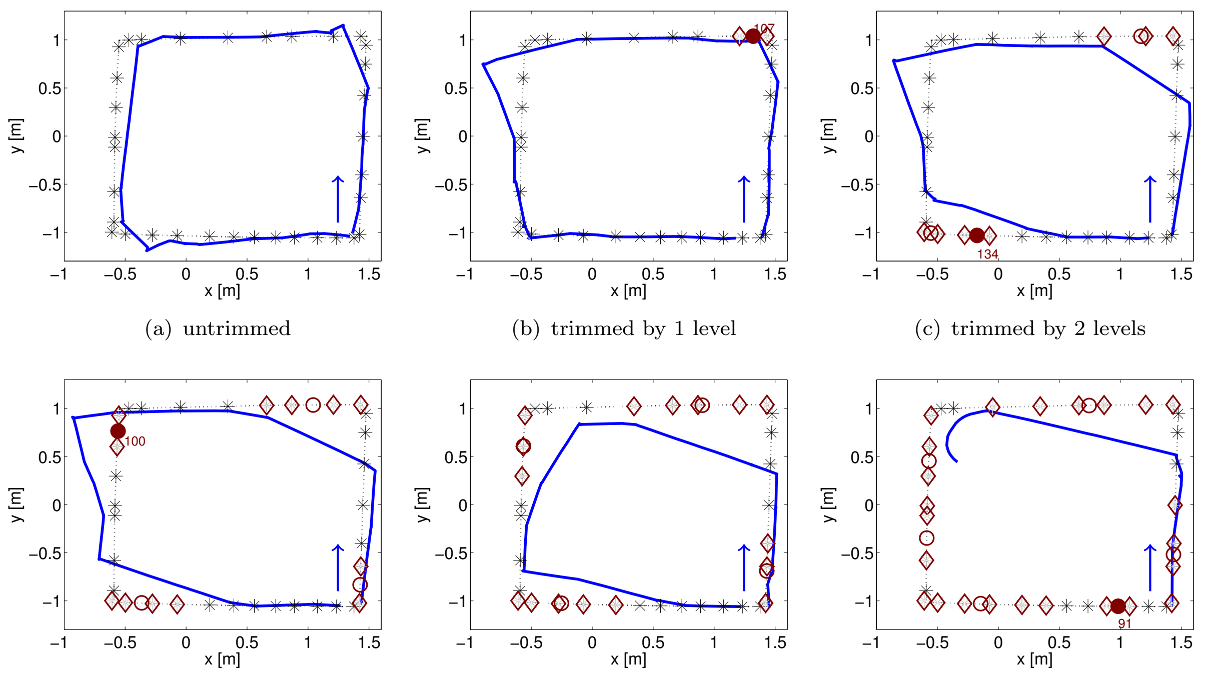
For creating the LT-map, the robot was commanded along a square path with an edge length of 2 m in an indoor laboratory environment. We chose a threshold ζ_{δ_t} which yielded a new viewframe after approximately 0.25 m. Hence, in total 34 viewframes were acquired along the trajectory.

During path following, the robot was commanded to evaluate an image, turn towards the new heading and go straight for 0.15 m before stopping again for acquiring the next image. The roadmap navigation performance was evaluated by an iteratively pruned LT-map. Fig. 10 shows the trajectories of five successfully completed path followings where up to 4 levels were pruned. After trimming the tree by 5 levels, the navigation failed due to the lack of matches. When truncating the tree, the trajectories became smoother and the robot cut corners due to the lack of local landmarks.

Tree height (trim)	Viewframes	Landmarks	Viewframe error [m]	Path error [m]
8 (0)	34 (100%)	10171 (100%)	0.057	0.055
7 (-1)	33 (97.1%)	9732 (95.7%)	0.052	0.057
6 (-2)	28 (82.3%)	8322 (81.8%)	0.133	0.099
5 (-3)	23 (67.6%)	7193 (70.7%)	0.145	0.108
4 (-4)	18 (52.9%)	5832 (57.3%)	0.208	0.113
3 (-5)	11 (32.4%)	3551 (34.9%)	-	0.155

Table 2. Mapping statistics for the indoor mobile robot experiment

In Table 2 some statistics on the pruned maps and the resulting path following precisions are presented. With only 57.3% of the landmarks, the robot was still able to reliably find its goal. The viewframe error indicates the average deviation between the location of the reference viewframe and the position where the robot switched to the next viewframe due to the dissimilarity measure. The path error was computed as the average distance between each measured location during



(a) untrimmed (b) trimmed by 1 level (c) trimmed by 2 levels

(d) trimmed by 3 levels (e) trimmed by 4 levels (f) trimmed by 5 levels

Figure 10. Trajectories for the original and the trimmed tree. Dotted line: mapping trajectory; asterisks: locations where viewframes were acquired; solid line: path followed by the robot; arrow: traveling direction from the start point; diamonds: viewframes which were pruned; circles: remaining viewframe branches, whose locations are chosen as the center of gravity of the pruned viewframes. A filled circle is used, if enough landmarks (at least 50) for homing are still available, and an empty circle otherwise. The numbers next to the filled circles indicate the amount of remaining landmarks.

the path following and the closest point on the map trajectory. As expected, the viewframe error as well as the path error increase by consecutively trimming the tree and, thus, discarding local information.

4.3.2 Outdoor Experiments

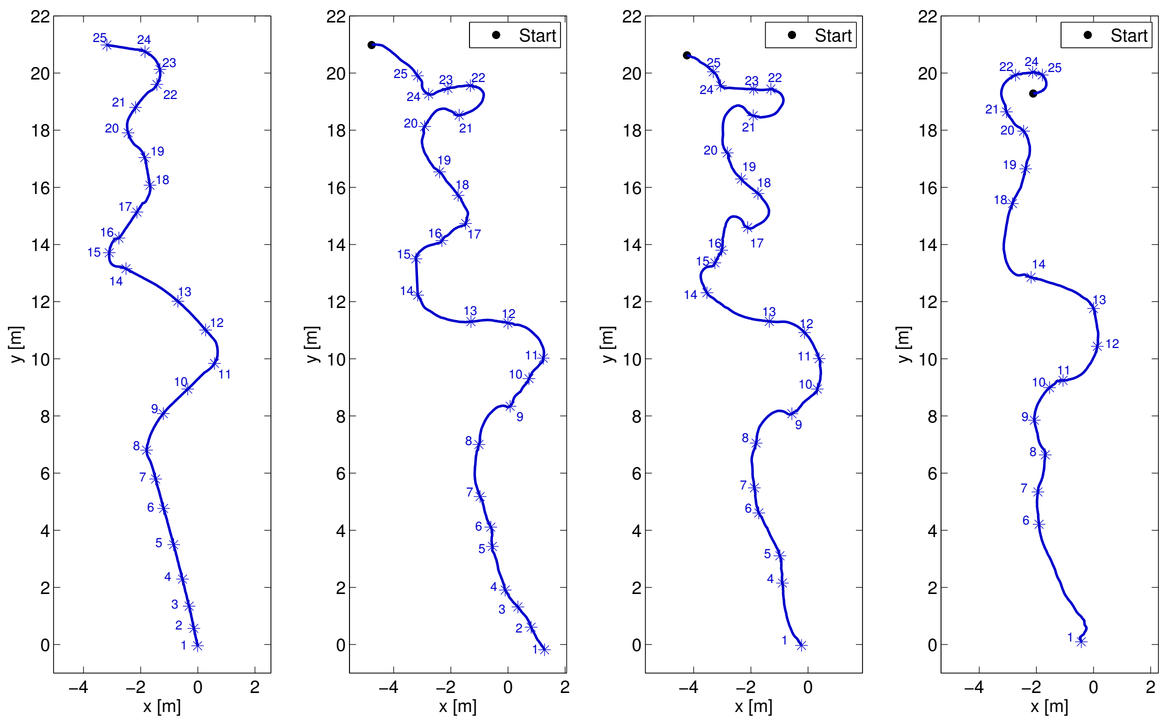
For the outdoor runs we changed the control paradigm of the robot. The robot continuously moves while the heading direction gets smoothly adapted to the direction estimate computed from the LT-map. The uncertainty of the estimate, *i.e.* the number of tracked landmarks and their consistency, is used to control the velocity of the robot.

Again, the robot was manually commanded to move along a certain trajectory, recording an LT-map. Afterwards, the recorded path was followed based on differently trimmed trees. The reference trajectory and three navigation results are illustrated in Fig. 11. As for the indoor scenario, we can observe that the path becomes smoother by pruning the tree. As soon as we trimmed the tree by 5 levels, the tracking was unable to find sufficient landmarks and, hence, the motion estimation failed.

Tree height (trim)	Viewframes	Landmarks	Viewframe error [m]	Path error [m]
9 (0)	24 (100%)	8493 (100%)	0.777	0.551
7 (-2)	23 (96%)	7862 (92.6%)	0.615	0.499
5 (-4)	18 (75%)	6229 (73.3%)	0.872	0.408

Table 3. Mapping statistics for the outdoor mobile robot experiment

Table 3 shows the statistics of the outdoor experiments. The path error decreases slightly by pruning the tree. This can be explained by the fact, that the applied velocity control was rather conservative and, hence, better supports the slow motion changes on the smoothed trajectory of the trimmed tree. Compared to the indoor experiments, less viewframes were stored in the map, which can be explained by a larger average distance of the landmarks. As explained in Section 3.1, the dissimilarity measure δ_t triggers viewframes at larger intervals if the average distance of the landmarks in the environment increases and, thus, it automatically adapts to the information available in the scene.



(a) mapping (b) untrimmed (c) trimmed by 2 levels (d) trimmed by 4 levels

Figure 11. The plots show the mapping and the three homing trajectories using differently trimmed trees of the outdoor experiment. The numbered asterisks represent the locations of the viewframes (mapping) or where the robot assumed to have reached a reference viewframe and switched to the next one (homing) respectively. The circle denotes the starting point for the homing runs.

5. Conclusion and Outlook

In this paper, we presented a novel mapping strategy which due to its non-metric and hierarchic nature allows for a flexible adaption to memory limitations. Thus, it is especially suited for resource limited mobile robots in outdoor environments, which need to cover long distances. It combines the power of tree data structures as used for metric maps, and the information clustering of topological maps. Unlike in conventional maps, the tree structure is not built in the metric domain but in a domain which splits the landmarks into close and far distant landmarks. This allows us to easily adapt the tree to the available memory, by simply pruning the local information, which is negligible for long distance navigation.

We ran simulations and performed experiments on a robotic platform to evaluate the performance of the presented approach. Thereby, we have shown that a reliable navigation can be achieved using only a small percentage of the available landmarks. The algorithm has proven to be robust against noise and outliers, like mismatches and occlusions. The mobile robot was able to record a trajectory and follow it, solely based on the measurements of an omnidirectional camera at low framerate and at low resolution. Pruning the tree of the LT-map and, thus, discarding local information results in smoother trajectories. In combination with a local obstacle detection, this completely satisfies the requirements for which it is designed, like connecting wide spread workspaces efficiently.

The approach is limited to path following due to the missing metric distance information. It can be assumed, that an acquired path is in general free of obstacles, because it has already been taken during exploration. Nonetheless, a local obstacle avoidance should be provided due to the possibility of changing environments and the smoothing effect when pruning.

As a next step, we want to come up with a heuristic to measure the quality and, thus, the probability with which the robot can follow the path based on the (pruned) map. We also want to apply the algorithm to an MAV, which allows us to validate the approach in 3D space. Furthermore, we think that the current bottleneck for the approach is the feature tracker, which does not allow to track landmarks over large distances. Therefore, we would like to come up with an algorithm which is able to track landmarks over a longer time.

Acknowledgments

The authors would like to thank Frank Ortmeier for his helpful comments and general support of this work.

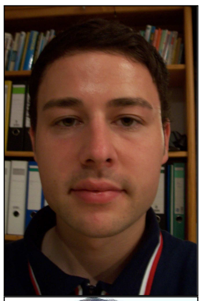
Notes on contributors

Elmar Mair received his Doctoral degree in computer science (CS) in 2012 and his M.Sc. in CS in 2007 from the Technische Universität München (TUM), Germany. Since 2012, he is researcher at the Institute of Robotics and Mechatronics at the German Aerospace Center (DLR). His main research interests are visual navigation and sensor fusion for mobile robotic systems. Focusing on solutions for resource limited systems, he is looking for inspiration in biology and tries to close the gap between efficiency and reliability.

Marcus Augustine received a B.Sc. degree in Computer Science from the University of Augsburg, Germany, in 2009. He graduated with a M.Sc. degree in Computer Science specialized on Robotics and AI as well as Software Engineering at the Technical University of Munich (TUM) in 2012. He is currently affiliated with the Computer Systems in Engineering Working Group at the Otto-von-Guericke University Magdeburg, Germany, and his main research topics are bio-inspired concepts in robotics and autonomous navigation.

Bastian Jäger recieved his engineering degree in mechanical engineering with focus on mechatronics and information technology in 2013 from the Technische Universität München (TUM), Germany. He is currently employed as research assistant at the Institute of Robotics and Mechatronics at the German Aerospace Center (DLR), where he works in the field of visual navigation.

Annett Stelzer received her Diploma in Mechatronics Engineering from the Technical University of Ilmenau, Germany, in 2008. Since 2009 she is a research assistant at the Institute of Robotics and Mechatronics of the German Aerospace Center (DLR). Her research interests include computer vision, sensor data fusion, mapping and vision based robot navigation.



Christoph Brand received his B.S. Degree and Diploma in Electrical Engineering and Information Technology from the Technical University in Munich (TUM), Germany, in 2010 and 2011. He joined the Institute of Robotics and Mechatronics of the German Aerospace Center (DLR) in 2011. His research areas include autonomous navigation and mapping in unknown unstructured environments and dynamic navigation.

Darius Burschka received his PhD degree in Electrical and Computer Engineering in 1998 from the Technische Universität München in the field of vision-based navigation and map generation with binocular stereo systems. Currently, he is an Associate Professor in Computer Science at the Technische Universität München, Germany, where he heads the computer vision and perception group. His areas of research are sensor systems for mobile and medical robots and human computer interfaces. The focus of his research is on vision-based navigation and three-dimensional reconstruction from sensor data.

Michael Suppa received his PhD "summa cum laude" in mechanical engineering from the University of Hannover in 2007. Since 2000, he is with DLR where he is heading the Department of Perception and Cognition at Institute of Robotics and Mechatronics since 2009. His research interests include 3Dmodeling, object recognition and scene analysis, optical navigation, and sensorbased exploration for autonomous mobile and fixed based systems. He has published more than 40 papers in conferences and journals.

References

[1] Koperski K, Adhikary J, Han J. Spatial data mining: progress and challenges survey paper. In: Proc. ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery. Montreal, Canada. 1996.

[2] Weiss S, Achtelik M, Lynen S, Chli M, Siegwart R. Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments. In: Proc. IEEE International Conference on Robotics and Automation (ICRA). 2012. p. 957–964.

[3] Wang R, Spelke E. Human spatial representation: Insights from animals. Trends in cognitive sciences. 2002;6(9):376–382.

[4] Kuipers B. Modeling Spatial Knowledge. Cognitive Science. 1978;2(2):129–153.

[5] Filliat D, Meyer JA. Map-based Navigation in Mobile Robots. I. A Review of Localization Strategies. Cognitive Systems Research. 2003;4(4):283–317.

[6] Augustine M, Mair E, Stelzer A, Ortmeier F, Burschka D, Suppa M. Landmark-Tree Map: a Biologically Inspired Topological Map for Long-Distance Robot Navigation. In: IEEE International Conference on Robotics and Biomimetics (ROBIO). 2012.

[7] Jäger B, Mair E, Brand C, Stürzl W, Suppa M. Efficient Navigation Based on the Landmark-Tree Map and the Zinf Algorithm Using an Omnidirectional Camera. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2013.

[8] Wehner R. Desert Ant Navigation: How Miniature Brains Solve Complex Tasks. Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology. 2003;189(8):579–588.

[9] Cartwright BA. Landmark Learning in Bees: Experiments and Models. Journal of Comparative Physiology. 1983;151(4):521–543.

[10] Cartwright BA, Collett TS. Landmark Maps for Honeybees. Biological Cybernetics. 1987;57:85–93.

[11] Collett M, Collett TS. Insect Navigation: No Map at the End of the Trail? Current Biology. 2006; 16(2):R48–R51.

[12] Collett TS, Graham P, Harris RA, Hempel De Ibarra N. Navigational Memories in Ants and Bees: Memory Retrieval when Selecting and Following Routes. Advances in the Study of Behavior. 2007; 36:123–172.

[13] Collett M. Spatial Memories in Insects. Current Biology. 2009;19(24):R1103–R1108.

[14] Zeil J, Boeddeker N. Visual Homing in Insects and Robots. In: Floreano D, editor. Flying Insects and Robots. chap. 7. Berlin: Springer Verlag. 2009. p. 87–100.

[15] Möller R. A Biorobotics Approach to the Study of Insect Visual Homing Strategies. 2002.

[16] Dai D, Lawton DT. Range-free Qualitative Navigation. Proc IEEE International Conference on Robotics and Automation (ICRA). 1993;1:783–790.

[17] Levitt TS, Lawton DT, Chelberg DM. Qualitative Landmark-Based Path Planning and Following. Proc AAAI. 1987;2:689–694.

[18] Kawamura K, Koku AB, Wilkes DM, Sekmen A. Toward Egocentric Navigation. International Journal of Robotics and Automation. 2002;17(4):135–145.

[19] Malis E. Survey of vision-based robot control. In: ENSIETA European Naval Ship Design Short Course. 2002.

[20] Lambrinos D, Labhart T, Pfeifer R. A Mobile Robot Employing Insect Strategies for Navigation. Robotics and Autonomous Systems. 2000;30(1-2):39–64.

[21] Booij O, Terwijn B, Zivkovic Z. Navigation Using an Appearance Based Topological Map. In: Proc. IEEE International Conference on Robotics and Automation (ICRA). Rome, Italy. 2007. p. 3927–3932.

[22] Angeli A, Doncieux S, Meyer J, Filliat D. Incremental Vision-Based Topological SLAM. In: Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2008. p. 1031–1036.

[23] Goedemé T, Nuttin M, Tuytelaars T, Van Gool L. Omnidirectional Vision Based Topological Navigation. International Journal of Computer Vision. 2007;74(3):219–236.

[24] Franz MO, Schölkopf B, Mallot HA, Bülthoff HH. Learning View Graphs for Robot Navigation. Autonomous Robots. 1998;5(1):111–125.

[25] Winters N, Gaspar J, Lacey G, Santos-Victor J. Omni-directional vision for robot navigation. In: Proc. IEEE Workshop on Omnidirectional Vision. 2000. p. 21–28.

[26] Ulrich I, Nourbakhsh I. Appearance-Based Place Recognition for Topological Localization. In: Proc. IEEE International Conference on Robotics and Automation (ICRA). 2000. p. 1023–1029.

[27] Cummins M, Newman P. Fab-map: Probabilistic localization and mapping in the space of appearance. The International Journal of Robotics Research. 2008;27(6):647–665.

[28] Thrun S. Probabilistic Algorithms in Robotics. AI Magazine. 2000;21(4):93–109.

[29] Tomatis N, Nourbakhsh I, Siegwart R. Hybrid Simultaneous Localization and Map Building: a Natural Integration of Topological and Metric. Robotics and Autonomous Systems. 2003;44(1):3–14.

[30] Stachniss C, Grisetti G, Martínez-Mozos O, Burgard W. Efficiently Learning Metric and Topological Maps with Autonomous Service Robots. it - Information Technology. 2007;49(4):232–238.

[31] Weber K, Venkatesh S, Srinivasan M. Insect-Inspired Robotic Homing. Adaptive Behavior. 1998 Jan; 7(1):65–97.

[32] Fischler M, Bolles R. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM. 1981;24(6):381–395.

[33] Stürzl W, Soccol D, Zeil J, Böddeker N, Srinivasan M. Rugged, obstruction-free, mirror-lens combination for panoramic imaging. Applied optics. 2008;47(32):6070–6078.

[34] Leutenegger S, Chli M, Siegwart R. Brisk: Binary robust invariant scalable keypoints. In: Proc. IEEE International Conference on Computer Vision. 2011.

[35] Mair E, Burschka D. Mobile Robots Navigation. chap. Z_∞ - Monocular Localization Algorithm with Uncertainty Analysis for Outdoor Applications. In-Tech. 2010. p. 107 – 130.