Landmark-Tree Map: a Biologically Inspired Topological Map for Long-Distance Robot Navigation

Marcus Augustine[*†‡] marcus.augustine@ovgu.de
Elmar Mair[*‡§] elmar.mair@dlr.de
Annett Stelzer[§] annett.stelzer@dlr.de
Frank Ortmeier[†] frank.ortmeier@ovgu.de
Darius Burschka[‡] burschka@cs.tum.edu
Michael Suppa[§] michael.suppa@dlr.de

[†]Otto-von-Guericke University, Workgroup Computer Systems in Engineering, D-39106 Magdeburg, Germany.
[‡]Technische Universität München (TUM), Department of Computer Science, D-85748 Garching bei München, Germany.
[§]German Aerospace Center (DLR), Institute of Robotics & Mechatronics, D-82234 Wessling, Germany.
[*]*The authors assert equal contribution and joint first authorship.*

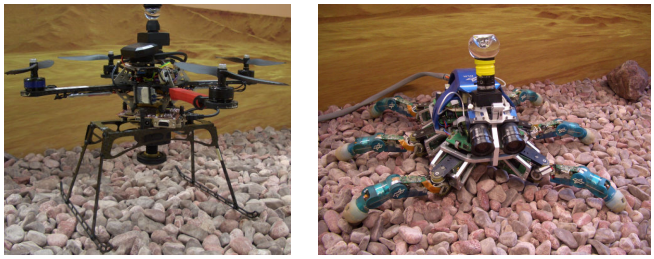
***Abstract*—Metric maps provide a reliable basis for mobile robot navigation. However, such maps are in general quite resource expensive and do not scale very well. Aiming for a highly scalable map, we adopt theories of insect navigation to develop an algorithm which builds a topological map for global navigation. Similar to insect conduct, positions in space are memorized as snapshots, which are unique configurations of landmarks. Unlike conventional snapshot approaches, we do not simply store the landmarks as a set, but we build a landmark tree which enables us to easily free memory in case of a continuously growing map while still preserving the dominant information. The resulting navigation is not sensor specific and solely relies on the directions of arbitrary landmarks. The generated map enables a mobile robot to navigate between defined locations and let it retrace a previously pursued path. Finally, we verify the reliability of the Landmark-Tree Map (LT-Map) concept and its robustness on memory limitations.**

***Index Terms*—map, topological, navigation, landmarks, tree, bio-inspired, efficient, scalable**

I. MOTIVATION AND RELATED WORK

A crucial task of any mobile robot is to be aware of its position and motion. It needs to localize itself within its close surrounding to detect obstacles, feed the control loop or solve a specific task (*local navigation*). However, there is also the requirement to move between task-related workspaces in a larger context, *e.g.*, to find the path back to its starting point – the so-called home location (*global navigation*). In robotics, there are two major concepts how the perception of the environment can be stored: either using metrical mapping strategies or topological ones.

Metrical maps are wide-spread and are suitable for path planning with a high degree of accuracy. The positions in metrical maps are unambiguous definitions afforded by their precise coordinates specified in a common reference frame. The drawback is that they are often expensive to calculate, because of the lack of a natural high level environment discretization, and they suffer from high memory consumption for stretched environments [1]. As soon as outdoor scenarios are considered, which are often huge, unstructured and dynamic, most metrical approaches do not scale satisfiably. Hence, the overall navigation performance of *Simultaneous Localization and Mapping* (SLAM) algorithms that are based on metrical maps suffers especially from increasing map sizes [2]. Tree data structures, like quadtrees, octrees or k-d trees, are used to provide an efficient representation of the metric space [3]. They are affected by computation overheads if the space-usage is not balanced or by high map maintenance costs for re-balancing.



(a) flying robot with a catadioptric (up) and fisheye (down) camera (b) crawling robot with a catadioptric camera

Figure 1. Resource limited platforms are in general confined by a rather small workspace due to memory limitations. Equipped with omndirectional cameras and using novel navigation strategies their operation space can be expanded significantly.

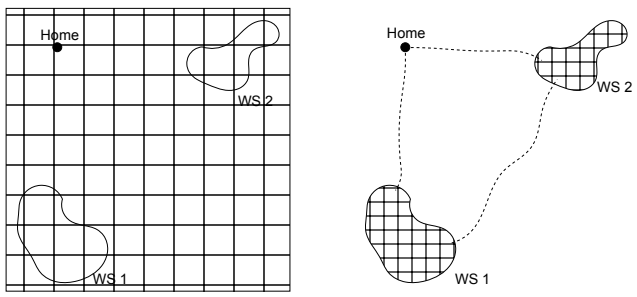
Topological maps, however, resemble graphs and do in general not put the information in a metrical context. Distinct places are represented as nodes, while edges denote the adjacency between different locations [4]. Because of this sparse representation of the environment, a high degree of memory efficiency is achievable as far as the covered terrain and the corresponding map size is taken into account. This results in a good scalability behavior of such maps [5].

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. However, the constraints on a local and a global map are not the same. Local navigation aims for an accurate localization within a specific, small-size workspace, which requires a large number of landmarks and a high metric resolution. A global navigation strategy tries to increase

978-1-4673-2127-3/12/$31.00 © 2012 IEEE

the dimensions of the workspace and, hence, focuses on a representation of the world which is as sparse as possible. Only the most dominant landmarks should be preserved, which still allow 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 like the ones depicted in Fig. 1.

In [6], *e.g.*, the navigation system for a micro aerial vehicle (MAV) is presented. It employs a scrolling metric map with a reduced number of landmarks, which leads to a trade-off between localization accuracy and workspace dimensions. The scrolling map allows the robot to move without limitations, but it can lead to the loss of crucial information necessary to find back, *e.g.*, to its home location. To overcome these problems, we encourage a separation of local and global navigation like motivated in [7]–[9]. The idea is that local metric maps are represented as nodes of a scalable topological map, which represents the spatial relation between them. Such a spatial semantic hierarchy [7] eases loop closure, higher level reasoning and memory management. However, general topological map concepts do not offer a solution for an efficient map thinning in case of memory limitations. Another interesting solution to the global navigation problem is *RatSLAM* [10], which builds a topological map with metric information, by separating the topological and the metric layer. This approach requires a proper scaling of the map in advance, because it cannot be changed efficiently at runtime. All these concepts are either based on dense meshes of metric maps or they connect the local maps by metric information. The first case is not resource efficient, because it might not be necessary to store detailed metric information of some locations, like transitional pathways. The latter case is error prone, due to drifts of the odometry, and may fail over long distances. A method which allows to store navigation information for long distances in an efficient way and, thus, enables the transitions between specific workspaces (as depicted in Fig. 2) would be of great interest.



(a) metric map (b) sparse map

Figure 2. The left drawing denotes a conventional metric map covering the full operation area of the robot. Especially, if the distances between specific workspaces (denoted as WS1 and WS2) are much longer than the dimensions of the workspaces, it makes more sense to delimit the metric maps to the required area of operation and store the trajectories in between in a more efficient way as depicted in the right picture. This allows also for a much higher spatial resolution of the workspace maps.

Insects have only access to a restricted nervous system with a small brain, but still manage the global navigation task with unbelievably 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 [11]. A vast amount of experiments with insects have been conducted, backing the existence and functionality of their navigational-toolkit [11]–[16]. Experiments show that the insects rely heavily on visual cues [17], [18]. Ants appear to limit the use of landmark memories solely to recognize goals or to trigger procedural movement commands in order to follow their route [14], [19]. Further experiments prove that the surrounding panorama plays an important role for global navigation [18]. In order to orient themselves in their environment, the insects use view-dependent learning of visual scenes from particular vantage points [11]. They seem to employ a retinotopically organized image matching. This means that the 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 [11]. 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 [20].

Based on these experiments, several models for insect navigation have been proposed in biology and were used as a basis for technical realisations. However, it is not clear which information insects store [21] – whether they use the full image [22] or another signature [23]–[25] to memorize a location as a so called *snapshot* or *viewframe*. The model in [26], *e.g.*, requires an a-priori map of the approximate locations of the major landmarks, whereas the approach described in [27] relies on range information which would require the use of range sensors.

In the following, we will present a novel approach which was especially inspired by the *snapshot model* [12], [13] and *viewframe concept* [28] as well as the topological map presented in [29]. They store a so called snapshot or viewframe as the projection of the surrounding landmarks on a unit sphere. We also rely only on the angles of the different landmarks without any metric information. However, we do not simply want to store the landmark sets as a characterization of a specific location, but we propose a different storing strategy which allows for more efficient memorization and memory management. The landmarks are arranged in a tree, ordered by their distance, without ever measuring it directly. This simplifies the removal of landmarks which are only measurable for a short time and, thus, enables an easy adaptation to memory limitations. We also do not assume a specific sensor, like a camera. In our work we use an abstract notion of the term landmarks. Every feature that is distinguishable and expressable under a certain angle relative to the robot may serve as a landmark, irrespective whether it may be perceived acustically, visually etc. However, for the sake of readability but without loss of generality we will focus only on visual landmarks in our explanations.

The remainder of this paper is structured as follows. In the next section, we will present the Landmark-Tree Map (LT-Map) concept, how to build and maintain such a map and how a robot can navigate based on it. It is followed by an evaluation of the approach in Section III. Finally, we will conclude with a short outlook on future work and summarize the strengths and limitiations of the proposed method.

II. THE LANDMARK-TREE MAP ALGORITHM

It is not feasible on resource limited platforms to consider the whole image for snapshot memorization. The information has to be preprocessed and only dominant, discriminative landmarks should be used for path guidance. The question remains how to store such a set of landmarks which defines a location. Most landmarks will be observed for a longer period of time and, thus, be present in several nodes of a topological map. This leads to redundancies and is not efficient at all. Furthermore, if the memory limits of the map are reached, it is difficult to decide which snapshots can be removed in order to free memory and enable a further extension of the map. In our algorithm, we introduce a hierarchy in the landmark structure which allows us to overcome these problems. In the following we will describe how to build such a map and how a robot can use it to navigate.

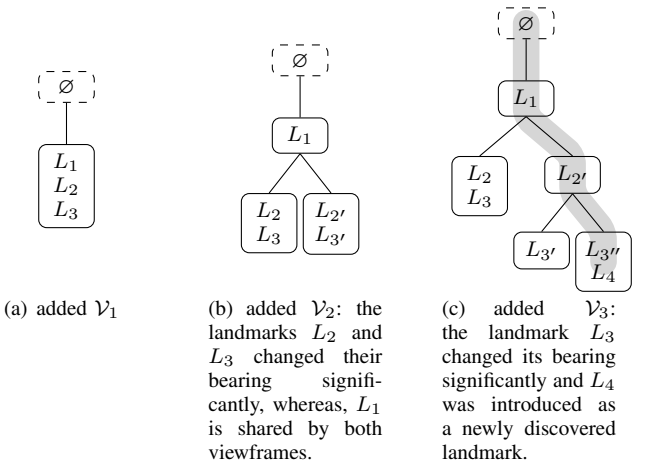
Let us first define a viewframe $\mathcal{V}$ as a representation of a distinct location in the three dimensional Euclidean space[1] $\mathbb{R}^3$ by a unique configuration of landmark bearings defined as vectors $\boldsymbol{l}_i$ pointing on the unit sphere. Each landmark is identified by a unique id, which can most simply be its descriptor. We assume the viewframes to be all rotationally aligned with each other, either by using compass information or by using the upper-level features of the map itself, as described in the next section.

A. Tree-Based Map Representation

The core idea is to reassemble the detected landmarks into a tree-like structure (the *Landmark-Tree Map*), sorting them from *global* to *local* information. Considering, that the angles of far distant landmarks remain static as the robot advances, we can use this information for our purpose.

The LT-Map is always initialized with an empty root node. The landmarks of the first view $\mathcal{V}_1$ acquired during exploration are stored in the first child of the root as shown in Figure 3(a). If the robot starts moving and former landmarks vanish or their angles change significantly, these landmarks are put into a new leaf of the tree as seen in Figure 3(b). If new landmarks appear, these landmarks are also stored in a new leaf of the tree (see Figure 3(c)), whereas the order of the leaves is important. In that way, the landmarks, whose bearings remain constant while the robot is moving, are stored in the upper nodes, because they are shared by more viewpoints, while more volatile landmarks are pushed into the lower level nodes or even the leaves. The root node containing the empty set is necessary to model views that do not share any landmarks with their predecessor. The exact algorithm is shown in detail in Algorithm 1.

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



(a) added $\mathcal{V}_1$

(b) added $\mathcal{V}_2$: the landmarks L_2 and L_3 changed their bearing significantly, whereas, L_1 is shared by both viewframes.

(c) added $\mathcal{V}_3$: the landmark L_3 changed its bearing significantly and L_4 was introduced as a newly discovered landmark.

Figure 3. Constructing the LT-Map by successively inserting viewframes. Each path from the root to a leaf represents a specific viewframe (as illustrated for $\mathcal{V}_3$). Hence, the order of the leaves is important.

As a consequence, all nodes along a path from one leaf to the root represent a certain viewframe and, thus, a specific location. This is indicated with the grey line in Figure 3(c) for $\mathcal{V}_3$, the third viewframe that has been inserted in the LT-Map. The landmarks in the upper nodes did not change their bearing 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 did change their bearing quickly, which means they are part of close objects. Actually, the landmarks in the leaves appeared only in a single viewframe with the respective bearing.

In this way we achieve both stated aims:

1) The memory consumption is reduced, because the landmarks which are shared by consecutive 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 transformation and virtual features.
2) In case of memory shortage one can easily prune the leaves of the tree and forget 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 is not further problematic as long as the navigation to each leaf is in the catchment area of the neighbouring leaves.

Algorithm 1: Append Viewframe To Tree

```
input       : a viewframe V, a Landmark-Tree T
output      : a Landmark-Tree T'
precondition : V is not empty
postcondition: |T| ≤ |T'|
if T.root not set then
    T.root = new Node(∅)
Node_current = T.root
while V has elements do
    {SameElements} = shared elements in
        Node_current.latestChild and V
    {RemainingElementsNode} =
        Node_current.latestChild \ {SameElements}
    {RemainingElementsViewframe} = V \ {SameElements}
    if {SameElements} is empty then
        Node_current.children ← new Node(V)
        V = ∅
    else
        if Node_current.latestChild ⊂ V then
            if Node_current.latestChild has children then
                Node_current = Node_current.latestChild
                V = {RemainingElementsViewframe}
            else
                if {RemainingElementsViewframe} ⊈ ∅ then
                    Node_current.latestChild.children ← new
                        Node(∅)
                    Node_current.latestChild.children ← new
                        Node ({RemainingElementsViewframe})
                V = ∅
        else
            TemporaryNode = new
                Node({RemainingElementsNode})
            move children from Node_current.latestChild to
                TemporaryNode
            Node_current.latestChild.children ← TemporaryNode
            Node_current.latestChild.children ← new
                Node({RemainingElementsViewframe})
            Node_current.latestChild = {SameElements}
            V = ∅
return T'
```

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. For a resource limited system, the depth of the LT-Map changes over time from a rather deep representation with a lot of local information into a wide tree with less depth, but which can span a long path.

The hierarchical structure of the tree does not only help to save resources and expand the map dynamically, but it also eases loop-closure or allows for an efficient pose estimation. The upper landmarks represent dominant, translation invariant objects. Hence, a loop detector would only need to compare the landmarks in the upper level nodes and only in case of a match it has to proceed to the respective child nodes. Only a few landmarks need to be tested instead of all possible snapshots. Furthermore, an inherent separation into translation-invariant and translation-dependent features enables a highly efficient pose estimation by the Z_∞-algorithm [30]. Accordingly, 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.

B. Navigating Using the LT-Map

The question remains how to navigate based on the constructed map. The leaves of the tree denote the different viewframes, which holds true also after a pruning operation. Thus, to navigate from a specific node of the performed trajectory to the desired node, one just has to follow each intermediate location, defined by the respective viewframe. The direction vector to each location can be computed by the difference in the landmark bearings of the measured snapshot and the reference viewframe stored in the tree. Different methods have been proposed in literature to compute the direction vector to a reference viewframe based on two landmark sets. We will make use of the so called *secant method* which was inspired by the *Average Landmark Vector* (ALV) model [31]. As the name implies, the basic concept is the calculation of an average landmark vector as the sum of all visible landmarks. The difference between this average vector of the reference snapshot and the measured snapshot represents the navigation direction. Let $\boldsymbol{l}_i$ denote the unit vector in the reference frame pointing to landmark $\boldsymbol{L}_i$ and $\tilde{\boldsymbol{l}}_i$ the unit-length measurement vector pointing to the same landmark. For each tuple $\left(\boldsymbol{l}_i, \tilde{\boldsymbol{l}}_i\right)$ the (secant) correction vector is computed and the sum of all correction vectors results in the navigation vector $\hat{\boldsymbol{v}}$, such that

$$\hat{\boldsymbol{v}} = \frac{1}{N} \sum_{i=1}^{N} \left(\boldsymbol{l}_i - \tilde{\boldsymbol{l}}_i\right) , \tag{1}$$

with N denoting the number of visible landmarks. The advantages of the secant method compared to other methods, like the *tangential method* presented in [17], are that it estimates a direct navigation vector to the goal and its computation is highly efficient.

As soon as the landmarks correspond, the next viewframe in the tree is used as a reference. The correspondence of two snapshots is determined by a similarity value δ, which is weighted by the Pseudo-Huber cost function [32], resulting in

$$\delta = \frac{1}{N} \sum_{i=1}^{N} 2b^2 \left(\sqrt{1 + \left(\frac{\boldsymbol{l}_i^T \tilde{\boldsymbol{l}}_i}{b} \right)^2} - 1 \right) . \tag{2}$$

This cost function weights small errors quadratically, but large errors linearly with slope $2b$ to suppress outliers. The resulting classifier is also used during the exploration phase to determine whether a new viewframe should be acquired or not in order to suppress too many intermediate leaves.

At this point we want to emphasize the importance of wide field sensors, like omnidirectional cameras, for landmark

detection. It is obvious, that the deviation from the original path in a certain direction can best be observed by landmarks which are seen in the perpendicular direction to it, whereas the landmarks on the line of deviation do not change their bearing angles at all, like illustrated in Fig. 4. A wide aperture angle allows not only for a better-posed computation of the navigation direction but also for more robustness. The aim of this algorithm is not to achieve a high accuracy when following a previously explored path, but to gain a high efficiency, flexibility and robustness. Hence, the accuracy, which could be increased by a smaller field of perception is of less interest than the possibility to stick to dominant landmarks in arbitrary directions.

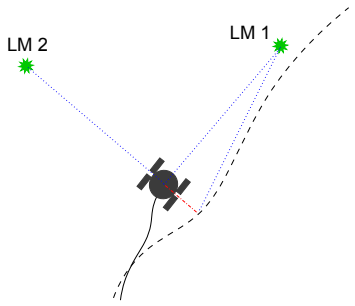


Figure 4. This drawing should motivate the use of omnidirectional sensors. The dashed line represents the trajectory which should be followed and the continuous line depicts the path travelled by the robot. Landmarks perpendicular to the path deviation (red line), like LM 1, are most sensitive to the error, whereas landmarks in the direction of the deviation, like LM 2, do not measure the error at all.

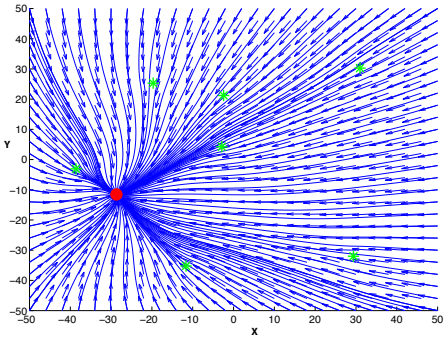
III. EVALUATION

In the following, we will show some experiments in a controlled simulation environment to evaluate the reliability of the algorithm and its performance in case of memory shortage. The simulations were performed in the two dimensional as well as in the three dimensional space to meet the requirements of both classes, ground or naval and air or submarine robots. The visualisations do not include any units, so an arbitrary scale may be chosen.

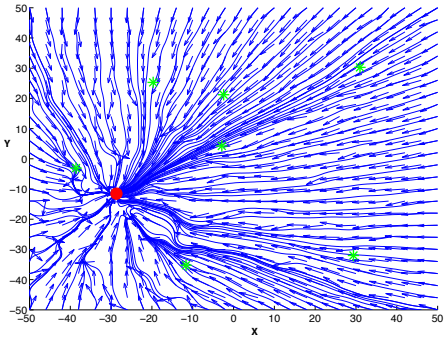
To demonstrate the robustness of the algorithm, we simulated two different noise terms: *angular measurement aberration* modeling a measurement error with landmark angles and a percentage of *outliers*. The angular aberration was modeled as zero-mean white Gaussian noise and outliers were specified by a probability P_{out} that the actual measurement of a landmark is replaced by a random angle in the interval $[0; 360)°$. We defined a realistic noise-scenario, with an angular measurement noise of $\sigma^2_{\text{angle}} = 5.0°$ and $P_{\text{out}} = 0.05$.

A. Navigation-Vector Calculation Performance

First we want to illustrate the navigation vector computed by the secant method described in Section II-B. Fig. 5 shows the movement directions expressed as unit vectors that were calculated for every position in a 50 × 50 example environment. The green asterisks depict the landmarks, the red filled circle denotes the goal position. The blue lines represent the streamlines from the border positions: they start at the marginal area of the grid and follow the calculated vectors contained in the vector field. A good homing performance was observed when the lines meet at the goal position, *i.e.*, the robot reaches the goal. To stick back to the biological motivation, they can also be considered as the trail of, *e.g.*, ants released at the frame of the diagram and walking homebound from the foraging ground.



(a) **error-free** scenario: $\sigma^2_{\text{angle}} = 0°$ and $P_{\text{out}} = 0$



(b) **realistic** scenario: $\sigma^2_{\text{angle}} = 5.0°$ and $P_{\text{out}} = 0.05$

Figure 5. 2D secant method vector fields: the homing vectors calculated with the secant method based on error-free and error-prone measurements.

Fig. 5(a) shows the calculated homing vectors in an error-free environment, *i.e.*, without any angular measurement aberrations or outliers. Fig. 5(b) shows the realistic case. Even though the noise terms obviously were influencing the calculated homing vector, a strong tendency towards the red marked goal position is visible.

B. Evaluation in 2D

Next, we performed navigation experiments in the two dimensional space. An exemplary navigation run is shown in Fig. 7. In total, 500 random landmarks within the X- and Y-axis-value interval $[-200; 200]$ were selected and represented as green asterisks. The robot learning track depicted as blue line is defined by 15 waypoints that are marked as red circles.

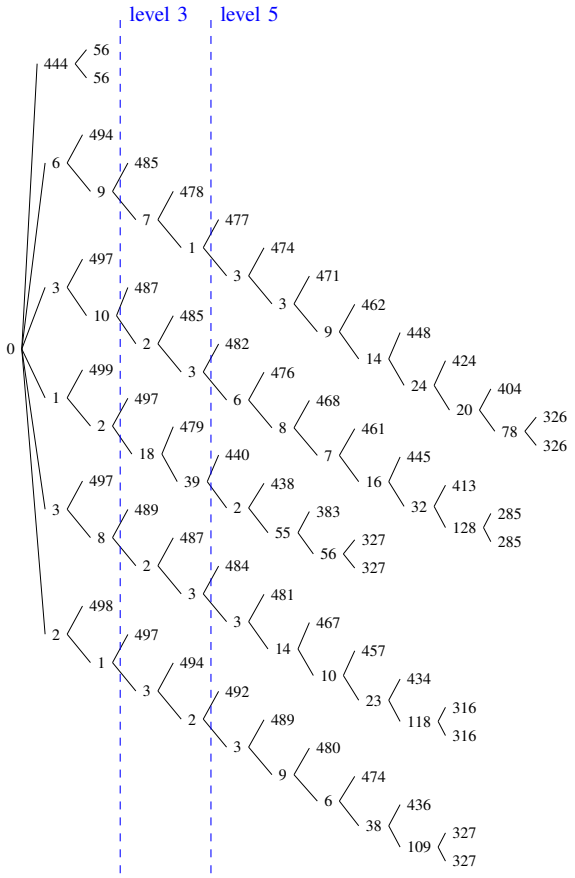


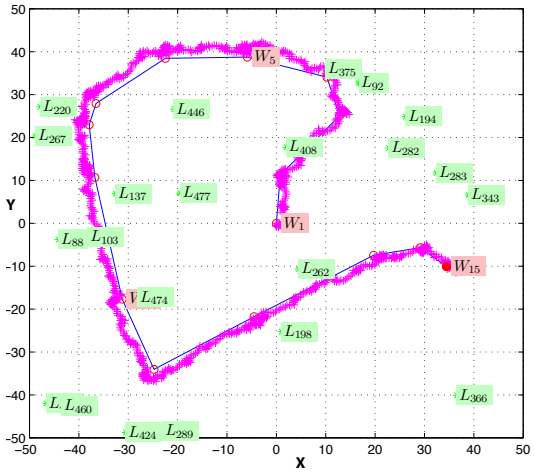
Figure 6. Visualisation of the original landmark tree and the pruning levels. The nodes in this representation contain only the number of landmark angles that are stored in the nodes of the landmark tree.

The navigation path of the robot (the calculated movement vectors) is indicated by magenta crosses.

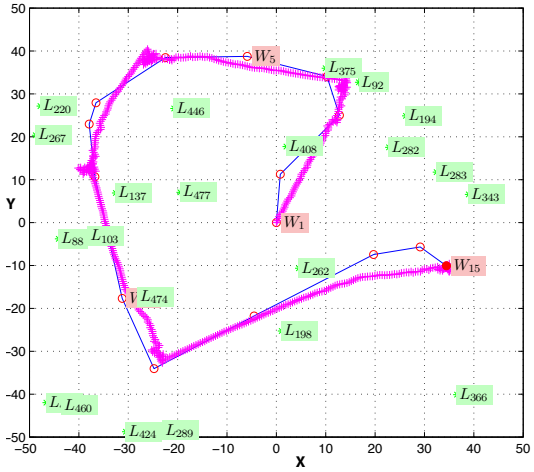
The final, unmodified LT-Map consists of 53 viewframes, 23860 landmark entries, and 100 node relations. The corresponding landmark tree is sketched in Fig. 6. The resulting navigation path, which relies on all this information, provides a high congruency with the simulated learning path as shown in Fig. 7(a). The jitter in the trajectory is due to the simulated measurement aberrations.

Next, we pruned the landmark tree and evaluated the navigation performance provided that only less memory resources would be available. In Fig. 7(b), the LT-Map consisted only of 33.7 % of the original data, expressing 22 viewframes that hold 8044 landmark data entries and were linked with 38 edges. As information was purged from the tree and less intermediate viewframes were available, shortcuts were taken and the track was not followed in such a high precision as the previous run.

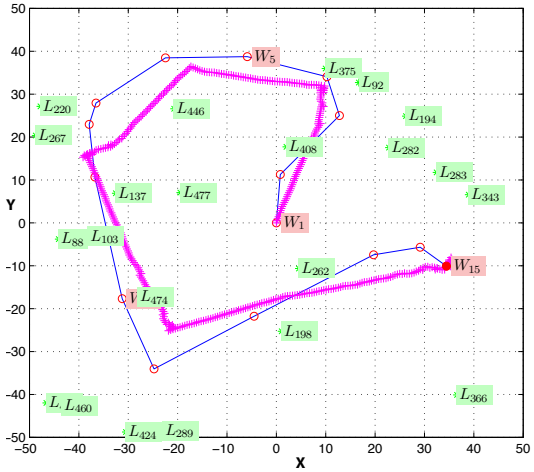
Finally, in Fig. 7(c) only 12 viewframes have been encoded by the LT-Map, holding 3086 landmark data entries organised with 18 edges between the tree nodes. This corresponds to a memory reduction of 87.1 % compared to the initial LT-Map, but the robot still achieves a feasible navigation performance and reaches the goal reliably.



(a) navigation based on the unmodified LT-Map



(b) navigation based on the LT-Map pruned at level 5



(c) navigation based on the LT-Map pruned at level 3

Figure 7. Simulation of the navigation in 2D space using the secant method on the full and two pruned landmark trees, where $\sigma^2_{\text{angle}} = 5°$ and $P_{\text{out}} = 0.05$.

The relation between pruning level and path following accuracy is illustrated in Fig. 8. It shows that several leaves can be cut without significantly losing navigation accuracy and only at a certain level the accuracy drops. In our simulation the robot moves a certain length and then stops to acquire a new measurement and compute the new direction of motion. This discrete navigation together with the simulated measurement noise prevents the robot from achieving a higher accuracy using more (local) landmarks. However, the reduction of the memory consumption, by pruning the first levels, is crucial as it is also shown in Fig. 8. Hence, these diagrams underline the efficiency improvements achieved by using the LT-Map, which are gained without loosing significant navigation performance.

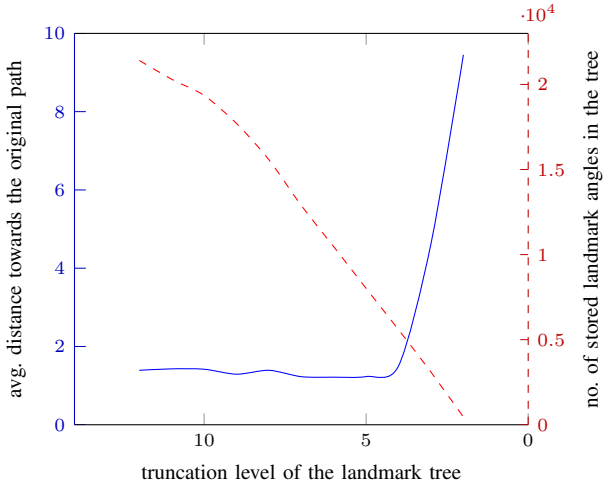


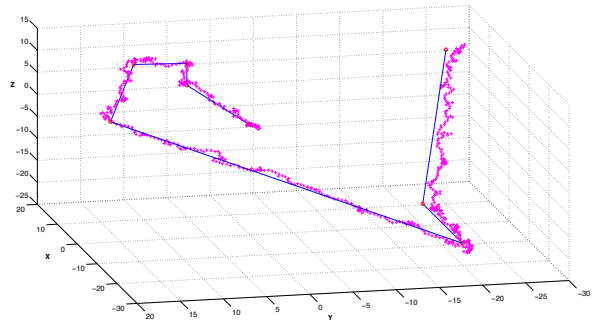
Figure 8. Average distance of the steps towards the original path and landmark angles that are stored within the LT-Map.

C. Evaluation in 3D

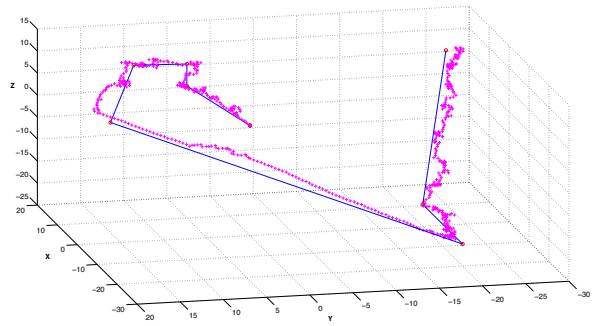
In the following we show some experiments of the LT-Map in 3D navigation scenarios. We simulated 100 randomly selected landmarks within the X-, Y- and Z-axis interval $[-100; 100]$. For the sake of clarity, the landmarks are not displayed in the visualisation. In total, 8 waypoints (red circles) have been selected to specify the track of the robot (blue line). The full LT-Map consisted of 32 viewframes and the tree had a maximum depth of 7 nodes. An experimental navigation was conducted based on that tree as shown in Fig. 9(a).

Fig. 9(b) shows the same navigation run after pruning the landmark tree up to level 3 with only 19 viewframes remaining. The stored landmarks have been reduced from 2673 to 1311 and the number of edges has been decreased from 58 to 34. Even though this small number of viewframes, the initial robot track could be followed and the goal position was reached reliably.

These experiments prove that the presented approach can also be applied to more than two dimensional spaces and still provide the previously seen efficiency and reliability characteristics.



(a) navigation based on the unmodified landmark tree



(b) navigation based on the pruned landmark tree

Figure 9. Simulation of the navigation in 3D space using the secant method on the full and a pruned landmark tree, where $\sigma^2_{\text{angle}} = 5°$ and $P_{\text{out}} = 0.05$.

IV. DISCUSSION AND CONCLUSION

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 resources and, at the same time, provides an ordering of the landmarks which enables features like efficient motion estimation by the Z_∞-algorithm or the realization of a visual compass. In our experiments, we have shown that a reliable navigation could also be achieved using only a small percentage of the available landmarks. Furthermore, the algorithm has proven to be robust against noise and outliers, like mismatches and occlusions.

The approach is limited to path following due to the missing metric information and does not provide a high accuracy, which could only be achieved if many near landmarks would be stored with slightly different bearing information. This would result in many redundantly stored landmarks and, thus, inflate the tree. The algorithm has not been designed for

accurate navigation in unknown areas, but to connect wide spread workspaces efficiently. For that it does not require a high accuracy, but reliability and robustness. Even though it can be assumed, that an acquired path is in general free of obstacles, because it has already been taken during exploration, a local obstacle avoidance should be provided due to the lack of accuracy and changing environments.

As motivated earlier, a wide field of perception improves the performance of the algorithm. Hence, our research platforms are equipped with different omnidirectional cameras as shown in Fig. 1. We currently evaluate these cameras as well as various image-feature detectors and descriptors, aiming for a high detection repeatability and good matching properties in the large field of view of such cameras. Further steps are finding loop closure mechanisms and heuristics to measure the probability with which the robot can follow the path based on the (pruned) map. The latter would enable the robot to autonomously determine each time before pruning the tree whether it should continue exploring or turn around because of too high risk to get lost.

REFERENCES

[1] S. Thrun, "Robotic Mapping: A Survey," Carnegie Mellon University, Computer Science Department, Pittsburgh, PA, Tech. Rep., 2002.

[2] B. Siciliano and O. Khatib, Eds., *Springer Handbook of Robotics*. Springer Verlag, Berlin, 2008.

[3] K. Koperski, J. Adhikary, and J. Han, "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.

[4] B. Kuipers, "Modeling spatial knowledge," *Cognitive science*, vol. 2, no. 2, pp. 129–153, 1978.

[5] D. Filliat and J.-A. Meyer, "Map-based Navigation in Mobile Robots. I. A Review of Localization Strategies," *Cognitive Systems Research*, vol. 4, no. 4, pp. 283–317, 2003.

[6] S. Weiss, M. Achtelik, S. Lynen, M. Chli, and R. Siegwart, "Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments," 2012.

[7] B. Kuipers, J. Modayil, P. Beeson, M. MacMahon, and F. Savelli, "Local metrical and global topological maps in the hybrid spatial semantic hierarchy," in *Proceedings of the IROS*, vol. 5. IEEE, 2004, pp. 4845–4851.

[8] J. Modayil, P. Beeson, and B. Kuipers, "Using the topological skeleton for scalable global metrical map-building," in *Proceedings of the IROS*, vol. 2. IEEE, 2004, pp. 1530–1536.

[9] H. Chang, C. Lee, Y. Hu, and Y. Lu, "Multi-robot slam with topological/metric maps," in *Proceedings of the IROS*. IEEE, 2007, pp. 1467–1472.

[10] M. Milford, G. Wyeth, and D. Prasser, "Ratslam: a hippocampal model for simultaneous localization and mapping," in *AAAI*, vol. 1, april-1 may 2004, pp. 403 – 408 Vol.1.

[11] R. Wehner, "Desert Ant Navigation: How Miniature Brains Solve Complex Tasks," *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, vol. 189, no. 8, pp. 579–588, Aug. 2003.

[12] B. A. Cartwright, "Landmark Learning in Bees: Experiments and Models," *Journal of Comparative Physiology*, vol. 151, no. 4, pp. 521–543, 1983.

[13] B. A. Cartwright and T. S. Collett, "Landmark Maps for Honeybees," *Biological Cybernetics*, vol. 57, pp. 85–93, 1987.

[14] M. Collett and T. S. Collett, "Insect Navigation: No Map at the End of the Trail?" *Current Biology*, vol. 16, no. 2, pp. R48–R51, Jan. 2006.

[15] T. S. Collett, P. Graham, R. A. Harris, and N. Hempel De Ibarra, "Navigational Memories in Ants and Bees: Memory Retrieval when Selecting and Following Routes," *Advances in the Study of Behavior*, vol. 36, pp. 123–172, 2007.

[16] "Invertebrate Colour Vision," in *Invertebrate Vision*, 1st ed., E. Warrant and D.-E. Nilsson, Eds. Cambridge University Press, 2006, pp. 250–290.

[17] K. Weber, S. Venkatesh, and M. Srinivasan, "Insect-Inspired Robotic Homing," *Adaptive Behavior*, vol. 7, no. 1, pp. 65–97, Jan. 1998.

[18] M. Collett, "Spatial Memories in Insects," *Current Biology*, vol. 19, no. 24, pp. R1103–R1108, Dec. 2009.

[19] R. Wehner, M. Boyer, F. Loertscher, S. Sommer, and U. Menzi, "Ant Navigation: One-Way Routes Rather Than Maps," *Current Biology*, vol. 16, no. 1, pp. 75–79, Jan. 2006.

[20] J. Zeil and N. Boeddeker, "Visual Homing in Insects and Robots," in *Flying Insects and Robots*, D. Floreano, Ed. Springer Verlag, Berlin, 2009, ch. 7, pp. 87–100.

[21] R. Möller, "A Biorobotics Approach to the Study of Insect Visual Homing Strategies," p. 71, 2002.

[22] H. A. Mallot, H. H. Bülthoff, M. O. Franz, and B. Schölkopf, "Where Did I Take That Snapshot? Scene-Based Homing By Image Matching," *Biological Cybernetics*, vol. 79, pp. 191–202, 1998.

[23] L. Gerstmayr, F. Röben, and R. Möller, "From Insect Visual Homing to Autonomous Robot Cleaning," 2008.

[24] A. Vardy and F. Oppacher, "Low-Level Visual Homing," in *Advances in Artificial Life - Proceedings of the 7th European Conference on Artificial Life*. Springer Verlag, Berlin, 2003, pp. 875–884.

[25] T. Goedemé, M. Nuttin, T. Tuytelaars, and L. Van Gool, "Omnidirectional Vision Based Topological Navigation," *International Journal of Computer Vision*, vol. 74, no. 3, pp. 219–236, Jan. 2007.

[26] K. Kawamura, A. B. Koku, D. M. Wilkes, and A. Sekmen, "Toward Egocentric Navigation," *International Journal of Robotics and Automation*, vol. 17, no. 4, pp. 135–145, 2002.

[27] T. S. Levitt, D. T. Lawton, and D. M. Chelberg, "Qualitative Landmark-Based Path Planning and Following," *Proceedings of the AAAI*, vol. 2, pp. 689–694, 1987.

[28] D. Dai and D. T. Lawton, "Range-free Qualitative Navigation," *Proceedings of the ICRA*, vol. 1, pp. 783–790, 1993.

[29] M. O. Franz, B. Schölkopf, H. A. Mallot, and H. H. Bülthoff, "Learning View Graphs for Robot Navigation," *Autonomous Robots*, vol. 5, no. 1, pp. 111–125, 1998.

[30] E. Mair and D. Burschka, "Monocular Localization Algorithm with Uncertainty Analysis for Outdoor Applications," in *Mobile Robots Navigation*, A. Barrera, Ed. In-Tech, 2010, pp. 107–130.

[31] D. Lambrinos, T. Labhart, and R. Pfeifer, "A Mobile Robot Employing Insect Strategies for Navigation," *Robotics and Autonomous Systems*, vol. 30, no. 1-2, pp. 39–64, 2000.

[32] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.