2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

# Efficient Navigation Based on the Landmark-Tree Map and the $Z_\infty$ Algorithm Using an Omnidirectional Camera

Bastian Jäger*, Elmar Mair*, Christoph Brand, Wolfgang Stürzl, and Michael Suppa
**The authors assert equal contribution and joint first authorship.*

***Abstract*— Map based navigation is a crucial task for any mobile robot. On many platforms this problem is addressed by applying Simultaneous Localization and Mapping (SLAM) based on metric grid-maps. Such solutions work well on robots with adequate resources and limited workspaces. Platforms with limited payload which operate in unbounded workspaces, do often have insufficient resources to keep a metric world representation. Nevertheless, many applications demand that the robot can autonomously navigate between different operation areas. In this work the Landmark-Tree map (LT-map), a resource efficient topological map concept, is for the first time applied to a mobile robotic platform equipped with an omnidirectional camera. It enables the robot to efficiently adapt the acquired map online to the available memory. During map acquisition and navigation the motion is estimated by the $Z_\infty$-algorithm. Both methods are based on similar concepts, which results in a mutual benefit. An efficient navigation strategy based on the LT-map allows the robot to reliably follow previously recorded paths. The presented approach is evaluated on a mobile robot in indoor and outdoor scenarios. The experiments prove its feasibility and show that pruning the map just smooths the trajectories, which is the expected and desired behaviour.**

## I. Motivation

Autonomous navigation is a highly complex task, which often requires most resources on mobile robots. In general, robotic platforms build metric maps during exploration and localize themselves within these maps. A large variety of approaches for this so called *Simultaneous Localization and Mapping* (SLAM) concept exist in literature [1]. They differ in the way the map is organized and the localization is performed, but they, in general, have in common to rely on a metric representation of the environment. While such a metric map eases many computation steps, *e.g.* accurate planning of new trajectories, it requires a significant amount of memory to represent the environment.

Many applications require resource limited robots to cover long distances, which connect different workspaces: a Micro Aerial Vehicle (MAV), which starts at the rescue team and flies in a specific direction to detect people who require help, or a rover on a foreign planet which has to find back to the base station after some time for analysing the collected probes. Such applications do not require a complete metric representation of the environment, but rather local metric maps at the locations of operation and resource efficient roadmaps which connect these workspaces and guarantee that the robot can reliably switch between them. Adaptive tree data structures, like quadtrees, octrees or k-d trees [2], suffer from computation overheads if the space-usage is not balanced or by high map maintenance costs for re-balancing. Hence, they are also not adequate to represent such stretched passages.

In contrast to these approaches, biological systems, like insects, animals, or humans, do not seem to rely on metric maps for navigation [3]. They would, in general, not even possess accurate senses for metric navigation. Their behaviour suggests a topological navigation based on so called cognitive maps, where the visual perception often plays the most important role. Inspired by these insights and aiming for a similar robustness and efficiency, we introduced the Landmark-Tree map (LT-map) and illustrated its functionality in simulations [4]. It represents a topological map, which relies on the measurements of a goniometer where no metric distance information is available. The algorithm is designed to be highly scalable and to dynamically adapt to the provided memory. The hierarchical structure of the landmark tree and its advantages are discussed in more detail in Section III.

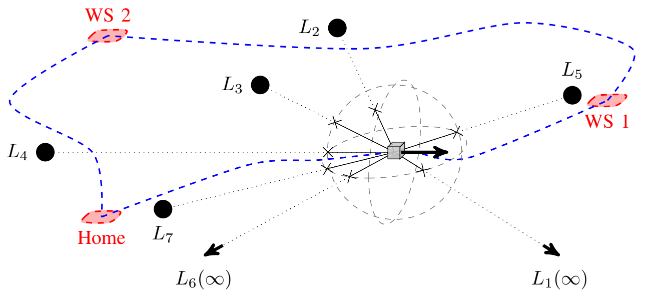


Fig. 1. This graphic depicts the problem we address in this work. A mobile robot, represented by the grey box, needs to move between different workspaces, denoted by the red areas. While for these regions high resolution metric maps may be required, the limited resources on some mobile robotic platforms may prevent a full metric representation of the complete operation space. We present a roadmap navigation, which efficiently adapts to the available memory and allows a mobile robot to move between different workspaces, solely based on bearing only measurements denoted by the crosses on the surrounding sphere pointing at the landmarks $L$.

In this work we realized an LT-Map based navigation on a mobile platform. The autonomous navigation is solely based on the measurements of an omnidirectional camera and enables an efficient switching between workspaces as

This work was supported by the Institute of Robotics and Mechatronics of the German Aerospace Center (DLR).

All authors are with the German Aerospace Center (DLR), Institute of Robotics & Mechatronics, D-82234 Wessling, Germany
elmar.mair at dlr.de

illustrated in Fig. 1. The motion is computed using the $Z_\infty$-algorithm which allows an efficient and robust estimation [5]. The LT-Map and the $Z_\infty$-algorithm mutually benefit from each other due to a similar clustering of the data into close and far distant landmarks. We show that simple control strategies are sufficient to realize the behaviour required for route following. We will also discuss the modifications necessary to deal with noisy sensor measurements and the effect of map pruning in case of memory shortage.

The remainder of this paper is structured as follows. First we describe the LT-Map and $Z_\infty$ concepts in Sections III and IV. The combination of these methods, their application to the measurements of a catadioptric camera as well as the implemented navigation strategies are explained in Section V. Finally, we evaluate the approach in Section VI, based on results of indoor and outdoor experiments with a mobile robot.

## II. Related Work

On many mobile platforms conventional metric SLAM algorithms are realized, which allow for straight-forward map updates and trajectory planning. The PTAM algorithm [6], *e.g.*, realizes parallel tracking and mapping for a monocular camera in real-time. It has been designed for augmented reality workspaces and creates a quite dense map of the environment. In order to use such an approach on resource limited systems, which are able to cover long distances, the map can be implemented as rolling map [7]. 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. For many applications it is crucial to return to a certain location and, thus, in case of limited memory, a global connectivity of different locations should be preferred over local information.

Topological maps have a better spatial scaling than grid based metric maps, but the missing geometric information bears also some drawbacks, *e.g.* it prevents the computation of shortcuts. Hence, most topological methods still contain and make use of metric information [8]. Another popular approach is to use sub-maps, where several small metric maps are connected by a topological graph [9], [10]. Such hybrid approaches combine the strengths of both paradigms. In our work we focus on how to connect such sub-maps without any metric information. We solely rely on bearing only measurements for the path which links different workspaces. 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.

Several approaches in the literature use omnidirectional cameras for topological navigation. Winters et. al. introduce a topological map, which triggers a visual servoing mechanism at crucial locations, like narrow passages [11]. Lui and Jarvis present a mobile robot which is equipped with two omnidirectional cameras [12]. The system builds a metric topological map based on Haar-wavelet signatures of unwarped images. The metric pose is computed by a three degree of freedom (DOF) visual odometry module which uses stereo correspondences from the cameras, processed on a GPU. In [13], Murillo et al. present a combination of topological and metric localization, where a pyramidal kernel of a large set of descriptors is used to detect the rough location and the 1D trifocal tensor is then used to compute an accurate metric localization estimate. 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, like histograms [14], linear PCA [15], or Haar wavelet coefficients [16]. In other implementations, landmarks are extracted and a set of feature descriptors is used to define a specific location, like in the bag-of-words models [17]. However, none of these approaches allows an easy pruning of the information in case of memory shortage. To identify 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.

The robot control concept which we adopt in this work is similar to [18], but instead of estimating the Essential matrix, we make use of the robust $Z_\infty$ based pose estimation to implement an efficient but reliable velocity control paradigm.

## III. The Landmark-Tree Map

The LT-map algorithm has been designed to enable an efficient online scaling of a roadmap to the available memory on a system [4]. A non-metric representation of the route is used, which requires only the measurements of a goniometer without any distance information. The landmarks are stored in a tree like structure, providing a hierarchy of global and local landmarks. This is achieved by extending each landmark descriptor with the bearing angle under which it is measured. This angle has to be compensated for the rotational motion of the robot, *e.g.* with a compass, in order to contain only translational information. In that way, far distant features, like landmarks on the horizon, consist of the same extended descriptor even after long trajectories. Local landmarks, which are close to the sensor, change their viewing angle already with small translational motions due to the motion parallax. Pure rotational motion does not change the descriptor-angle tuples and, hence, does not affect the map. In the following we denote these tuples as landmarks, with $L_i = (\boldsymbol{d}_i, \boldsymbol{\phi}_i)$, where $\boldsymbol{d}_i$ is the descriptor vector and $\boldsymbol{\phi}_i$ the vector containing azimuth and elevation under which the landmark $i$ is measured after compensation for the rotational motion. A viewframe $V$ is defined as a set of landmarks as perceived from a certain location and, thus, describes that location.

During map generation, *i.e.* when a path is traversed for the first time, landmarks are computed for each new measurement. Each time the weighted average angle, $\delta_t$, between two corresponding landmarks exceeds a specified threshold, a new viewframe is defined by the current set of landmarks. This similarity measure between two landmark sets of size $N$ is computed using a Pseudo-Huber cost function, such that

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

where $\boldsymbol{l}_i$ and $\boldsymbol{l}_i'$ denote the unit vectors pointing to the same landmark $i$ at different locations. The unit vector can be easily computed from $\boldsymbol{\phi}_i$. This cost function includes a control parameter $b$ to weight small errors quadratically, but large errors linearly with slope $2b$. Only new landmarks, compared to the previous viewframe, are added as a new leaf at the latest branch. Hence, the persistent landmarks lie in the upper nodes, close to the root, and contain translation invariant information, which we denote as global information in the following. The so called local, translation dependent information, is represented by the landmarks in the lower nodes, down to the leaves. The stepwise transition from global to local information is as fine as the environment requires and implicitly realized by the different layers. At this point we want to emphasize that, in practice, the landmarks are not only ordered based on their distance, but also according to their stability. The nodes close to the root contain only really stable landmarks, whereas the more unstable ones, which cannot be tracked continuously, are located closer to the leaves. Each complete branch in the tree, from the root to the leaf, represents a viewframe. The structure of such a tree is illustrated in Fig. 2. For a more detailed explanation please refer to [4].

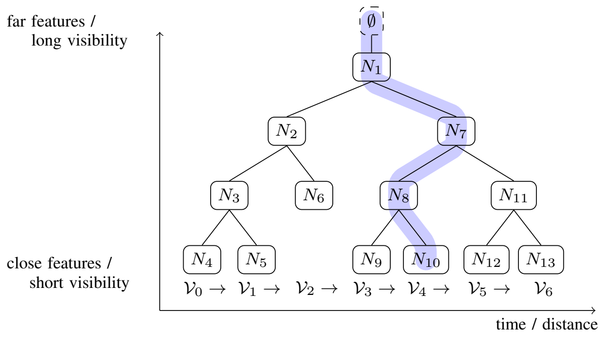

far features / long visibility
close features / short visibility
time / distance
$\emptyset$
$N_1$
$N_2$
$N_7$
$N_3$
$N_6$
$N_8$
$N_{11}$
$N_4$
$N_5$
$N_9$
$N_{10}$
$N_{12}$
$N_{13}$
$\mathcal{V}_0 \rightarrow \mathcal{V}_1 \rightarrow \mathcal{V}_2 \rightarrow \mathcal{V}_3 \rightarrow \mathcal{V}_4 \rightarrow \mathcal{V}_5 \rightarrow \mathcal{V}_6$

Fig. 2. The two dimensions of the landmark tree represent the travel distance (time) and the distance of the landmarks to the robot. Each branch of the tree is a viewframe (as highlighted in blue for $V_4$). The nodes $N$ along the branch contain the landmarks acquired at a certain location.

## IV. The $Z_\infty$ Algorithm

The LT-map requires rotationally aligned landmarks to build the map. The algorithm is designed for long-distance navigation and, hence, for outdoor scenarios. The $Z_\infty$-algorithm, an efficient feature-based motion estimation technique, also greatly benefits from such environments and only relies on the measurements of a goniometer [5], [19]. It uses a RANSAC [20] framework to separate far distant, translation-invariant landmarks from local, translation-variant ones and rejects features on dynamic objects which do not fit a common rigid motion model. In that way, the rotation estimation is separated from the translation estimation which allows to apply efficient closed-form solutions for both motion components as described in [5]. However, the translation estimation presented there is based on a projective camera. In order to estimate the translation based on omnidirectional measurements, we minimize the following cost function over the current landmark set:

$$\mathrm{E}(\boldsymbol{t}) \;=\; \sum_{i=0}^{N} w_i \cdot \left( \left(\boldsymbol{l}_i' \times \boldsymbol{l}_i\right)^T \boldsymbol{t} \right)^2 - \lambda \left( \|\boldsymbol{t}\|^2 - 1 \right) , \quad (2)$$

where $\boldsymbol{l}_i$ and $\boldsymbol{l}_i'$ denote the rotation compensated direction vectors to landmark $i$ of the reference and the current measurement. The translation $\boldsymbol{t}$ is forced to have unit length by the Lagrangian multiplier $\lambda$ and each landmark vector can be weighted by a factor $w_i$, *e.g.* if different tracking accuracies apply. After replacing $\boldsymbol{m}_i = \boldsymbol{l}_i' \times \boldsymbol{l}_i$ we get

$$\begin{aligned}\mathrm{E}(\boldsymbol{t}) &= \sum_{i=0}^{N} w_i \left(\boldsymbol{m}_i^T \boldsymbol{t}\right)^2 - \lambda \left( \|\boldsymbol{t}\|^2 - 1 \right) \\ &= \sum_{i=0}^{N} w_i \boldsymbol{t}^T \boldsymbol{m}_i \boldsymbol{m}_i^T \boldsymbol{t} - \lambda \left( \|\boldsymbol{t}\|^2 - 1 \right) \\ &= \boldsymbol{t}^T \left( \sum_{i=0}^{N} w_i \boldsymbol{m}_i \boldsymbol{m}_i^T \right) \boldsymbol{t} - \lambda \left( \|\boldsymbol{t}\|^2 - 1 \right) \quad (3)\end{aligned}$$

Substituting $\boldsymbol{M} = \sum_{i=0}^{N} w_i \boldsymbol{m}_i \boldsymbol{m}_i^T$ and setting the derivative of $\mathrm{E}(\boldsymbol{t})$ to zero results in an eigenvector problem, defined by

$$\begin{aligned} 0 &= 2\boldsymbol{M}\boldsymbol{t} - 2\lambda \boldsymbol{t} \\ \lambda \boldsymbol{t} &= \boldsymbol{M}\boldsymbol{t} \, . \quad (4)\end{aligned}$$

Hence, up to sign, $\boldsymbol{t}$ is the eigenvector corresponding to the smallest eigenvalue of the matrix $\boldsymbol{M}$ and can be computed by a singular value decomposition (SVD), such that

$$\tilde{\boldsymbol{t}} \;=\; \boldsymbol{V}_{\boldsymbol{M};:,3} \quad \text{with} \quad \mathrm{SVD}\left(\boldsymbol{M}\right) \;=\; \boldsymbol{U}_{\boldsymbol{M}} \boldsymbol{\Sigma}_{\boldsymbol{M}} \boldsymbol{V}_{\boldsymbol{M}}^T , \quad (5)$$

whereas the subscripts in $\boldsymbol{V}_{\boldsymbol{M};:,3}$ refer to the last column of the matrix. The sign is determined by

$$\boldsymbol{t} \;=\; \begin{cases} -\tilde{\boldsymbol{t}} & \text{if } \sum_{i=0}^{N} \left(\boldsymbol{l}_i' - \boldsymbol{l}_i\right)^T \tilde{\boldsymbol{t}} < 0 \\ \tilde{\boldsymbol{t}} & \text{else} \end{cases} . \quad (6)$$

To ensure a proper rotation estimation we added another constraint for a set of landmarks to be selected by RANSAC as best inlier set. We reject all sets for which the maximum angle between two neighbouring landmarks is larger than 180°. This prevents, that a translational motion is misinterpreted as rotation. We also evaluated the use of a visual compass for rotation estimation. However, it is known that in unbalanced environments the visual compass achieves only poor results [21]. In presence of much closer objects on one side, compared to the opposite side, the visual compass fails and is greatly outperformed by the model based rotation estimation of the $Z_\infty$-algorithm.

However, in order to apply the $Z_\infty$-algorithm, far translation-invariant features have to be identified. On one hand, the LT-map allows to easily access these landmarks by simply extracting the ones stored in the upper nodes of the tree. This gives an initial set of landmarks for efficient rotation estimation and reduces the number of random sampling steps as required by RANSAC. On the other hand, the set of translation-invariant features, which is returned by the $Z_\infty$-algorithm, can be used to find correspondences in the upper nodes of all viewframes. The resulting matches allow to identify loop closures or viewframes where some far distant landmarks could not be tracked and, thus, are missing. If such gaps are detected, they can be filled by shifting the respective landmark to a common node in a higher level of the tree. At the end, both algorithms, the LT-map and the $Z_\infty$-algorithm, split the landmarks into far-distant and local ones. The LT-map does this stepwise, whereas the $Z_\infty$-algorithm provides a binary splitting.

## V. LT-Map Based Navigation Using an Omnidirectional Camera

First off, we would like to motivate our sensor choice. All algorithms used in the presented navigation framework are based on the measurements of a goniometer. 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 fulfils these criteria and allows to realize large aperture angles. Increasing the field of view while keeping the pixel resolution constant, one can see that the accuracy has an optimum around 100° [19]. Nonetheless, we chose to use a catadioptric camera, as presented in [22], because large field of views significantly increase the robustness. Especially in case of roadmap navigation algorithms, where the paths are traversed in both directions, it is crucial to look not only to the front, but also to the back, in order to keep the heading of other sensors in driving direction. Furthermore, an omnidirectional camera minimizes the risk to lack far distant features for rotation estimation. This allows to operate also in narrow passages and in front of walls.

We evaluated different global feature trackers, ASIFT [23], SIFT [24], SURF [25], and BRISK [26], on omnidirectional images. While there was not much difference between SIFT, SURF, and BRISK, the most reliable tracking was achieved using ASIFT. However, it requires the most processing time and, hence, we chose to use BRISK, due to its high efficiency. The experiments also revealed an improved performance of all trackers if the omnidirectional images are unwarped to a panorama image before processing. This allowed us to use U-BRISK, the rotational-variant version of the algorithm, to increase performance.

### A. Roadmap Learning

If a robot explores an environment for the first time, the path is learned as LT-map. The starting point and the initial orientation of the robot define the origin. The optical flow vectors which represent a pure translational motion are estimated by the $Z_\infty$-algorithm and used to determine whether a new viewframe should be added according to $\delta_t$ introduced in Eq. 1. If a new viewframe is triggered, the features of the current frame are matched with the ones of the previous image in order to reject volatile landmarks before adding them as new viewframe.

One advantage of the tree structure in the LT-map is, that removing local or unstable information comes at low cost. Once the memory limits are reached, the lowest level of the tree is simply truncated and memory is freed. Therefore, the robot does not have to know in advance how long the path will be, which it is going to acquire. It can simply start with a small value for $\delta_t$ and just 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. In case of a completed map, the landmarks in the nodes of the first and last branch are not discarded, since we would like to have higher accuracy for finding the goal or start location. The idea is, that during path navigation, the accuracy to stay exactly on the trajectory is not so much of interest. However, we would like to have a high accuracy when finding the goal location, because it could be a key location within the aspired workspace. Also the starting point should provide as much information as possible in order to reliably hook onto the trail, in cases where the starting location is not accurately known within the current workspace. Hence, the landmarks of the nodes in the first and the last branch are not discarded, but shifted upwards into the parent node as illustrated in Fig. 3. If the map is pruned during acquisition, also the last branch is trimmed, because it does not require a higher accuracy. After pruning, we check all branches whether they

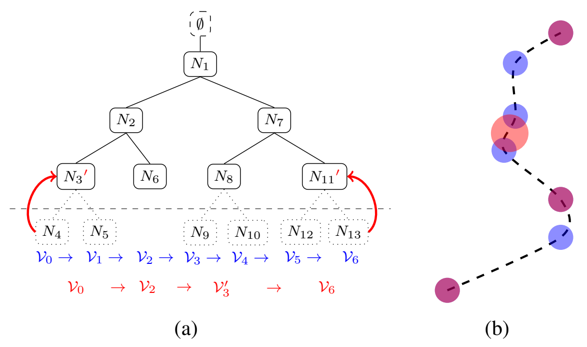

$\emptyset$
$N_1$
$N_2$
$N_7$
$N_3{}'$
$N_6$
$N_8$
$N_{11}{}'$
$N_4$
$N_5$
$N_9$
$N_{10}$
$N_{12}$
$N_{13}$
$\mathcal{V}_0 \rightarrow \mathcal{V}_1 \rightarrow \mathcal{V}_2 \rightarrow \mathcal{V}_3 \rightarrow \mathcal{V}_4 \rightarrow \mathcal{V}_5 \rightarrow \mathcal{V}_6$
$\mathcal{V}_0 \rightarrow \mathcal{V}_2 \rightarrow \mathcal{V}_3' \rightarrow \mathcal{V}_6$
(a)
(b)

Fig. 3. If the tree has to be trimmed to free some space, the nodes in the lowest level are removed as illustrated in the left drawing. If the outer branches represent an endpoint of the path, the landmarks are shifted into the parent node, as illustrated by the red arrows, such that $N_3' = N_3 \cup N_4$ and $N_{11}' = N_{11} \cup N_{13}$. The right drawing sketches the corresponding exemplary path with the viewframe locations denoted as circles. The less local information is available, the larger the circle, which depicts the caption area of a viewframe. The blue circles correspond to the original viewframes and the red ones to the trimmed tree.

contain sufficient landmarks for reliable homing, otherwise the new leaves of these branches are deleted and the tree is compressed.

### B. Roadmap Following

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. These landmark sets are used as reference for the pose estimation, which is computed by applying again the $Z_\infty$-algorithm. Hence, outliers in the landmark set are filtered by the RANSAC framework. The resulting translation vector $\boldsymbol{t}$ is applied as input for the motion control. In order to suppress erroneous translation estimates, we apply a median filter for the azimuth and elevation angle over $K$ most recently acquired vectors $\boldsymbol{t}$. The circular problem for the median rotation calculation of setting the 360° incision properly is solved as follows:

$$\hat{\boldsymbol{\phi}} = \operatorname{med}\Big(\big\{\operatorname{mod}(\boldsymbol{\phi}_k - \bar{\boldsymbol{\phi}} + \pi, 2\pi)\big\}_{k=0}^{K}\Big) + \bar{\boldsymbol{\phi}} - \pi \quad (7)$$

with $\bar{\boldsymbol{\phi}} = \frac{1}{K}\sum_{k=0}^{K}\boldsymbol{\phi}_k$, where $\boldsymbol{\phi} = (\phi_a, \phi_e)^T$ consist of the azimuth and elevation of the vector $\boldsymbol{t}$. The median and the modulo function are applied elementwise. Finally, the median 2D heading vector can be computed by

$$\hat{\boldsymbol{t}} = \left(\cos\left(\hat{\phi}_a\right),\ \sin\left(\hat{\phi}_a\right),\ \sin\left(\hat{\phi}_e\right)\right)^T. \quad (8)$$

Assuming that more measurements result in a higher confidence, we scale the vector $\hat{\boldsymbol{t}}$ by the number of elements in the median filter, $K$, and the number of landmarks used for its estimation, $N$, such that

$$\boldsymbol{v} = \frac{K}{K_{\max}} \min\left(\frac{N}{N_{\max}}, 1\right) \hat{\boldsymbol{t}}, \quad (9)$$

where $N_{\max}$ denotes the specified number of landmarks which is sufficient for a reliable motion estimation and $K_{\max}$ is an empirically chosen value for the maximum filter size which depends on the sensor, feature tracker, and the environment. The resulting vector $\boldsymbol{v}$ is used to control the velocity of the robot with a simple PID controller. It is interpreted as scaled velocity, which is multiplied with a constant maximum velocity. Thus, the robot continuously adapts its direction to follow the path and moves faster if it is confident about the direction estimate and slower if the uncertainty increases. This behaviour allows to acquire more measurements, *e.g.*, in case of poor conditioning of the environment.

The switching to the next viewframe is triggered in two cases. Either the translational flow vectors are small enough according to Eq. 1, meaning that the robot is close to the reference viewframe, or the estimated direction vector changes by more than 90°, meaning that the robot just passed the viewframe and wants to head backwards. The second constraint may be problematic if the orientation between consecutive viewframes changes by more than 90°. The robot would immediately find the second viewframe located in its back after switching to the first one, which would trigger another viewframe switch. Hence, it requires some time to turn to the new heading direction after viewframe switching. This can be easily realized by a short timeout after each viewframe switch or by waiting until the heading direction matches.

## VI. Experiments

As platform for our experiments we use the Pioneer-3DX robot as illustrated in Fig. 4. The platform is equipped with different ground truth sensors and markers. For our indoor experiments we rely on the ART tracking system[1] and outdoors we use the measurements of a tachymeter[2], which provide the most accurate measurements. Navigation commands are sent to the velocity interface of the robot's motor controller board via serial interface. As middleware, for the communication between the different modules, we use ROS[3]. All the processing is done on-board and can be monitored via WiFi. To stop the robot in an emergency or for manual operation, it can be controlled via a gamepad. The exploration trajectories are controlled manually.

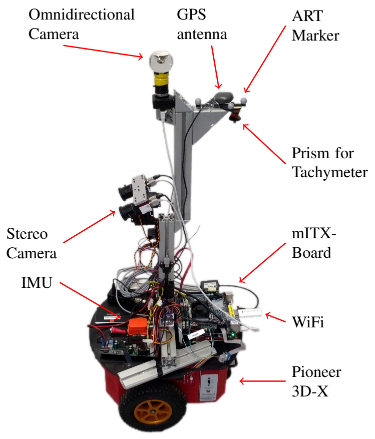


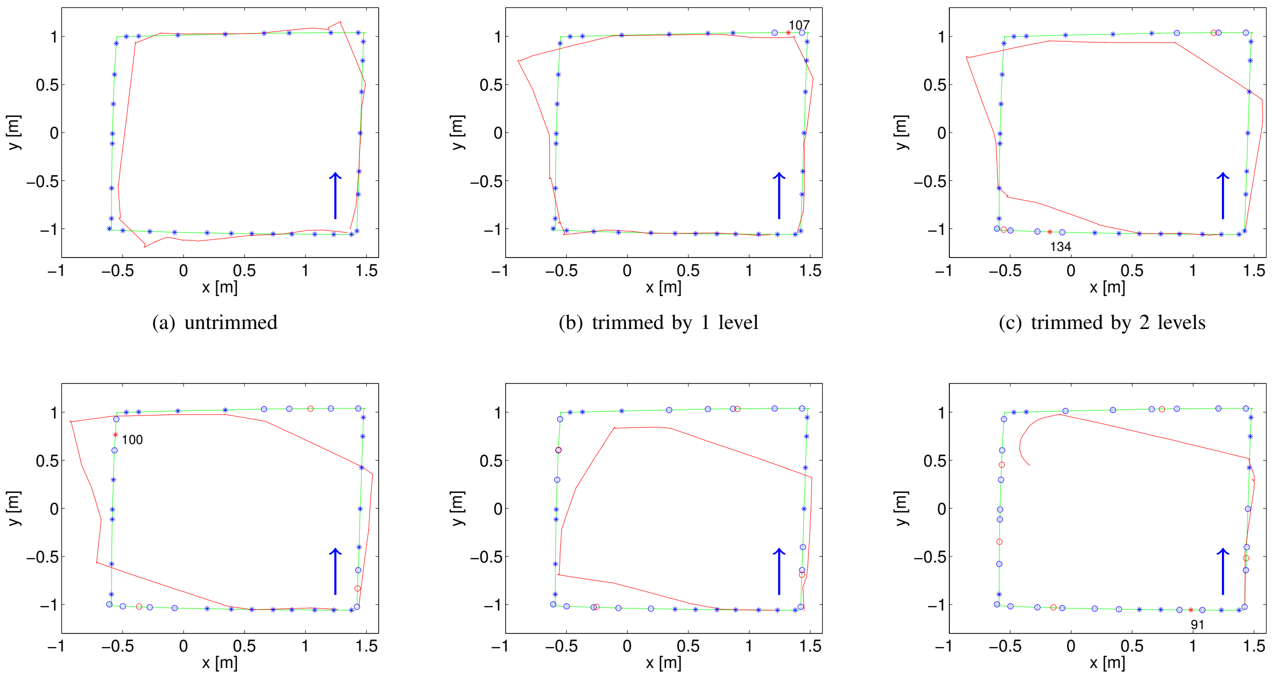
Fig. 4. In our experiments we used a Pioneer-3DX as mobile platform. The robot is equipped with a GPS antenna, a reference prism for the tachymeter and an ART marker for ground truth. A further reference is computed by fusing the measurements of an IMU and stereo odometry as described in [27], whereas the Semi Global Matching (SGM) stereo processing runs on a Xilinx Spartan-6 FPGA. A Kontron KTQM67/mITX embedded motherboard is used as processing platform, equipped with an Intel Core i7 Processor, 8 GB DDR3, and a SSD storage device. All the processing is done on-board and a wireless network is used to monitor the data on a host computer.

We acquired and processed images at 5 Hz with a resolution of $474 \times 474$ pixels which can also be realized on much smaller platforms. The images were then unwarped to a panorama with $0.5°/\text{px}$ resolution in both dimensions, resulting in a usable image of size $720 \times 172$ pixels. We continuously adapted the BRISK detector to find about 400 landmarks in the indoor scenario and 500 in the outdoor experiments, which yielded about 200-300 valid matches. The unwarped omnidirectional images have been extended to include a small overlap which compensates for the descriptor size of U-BRISK and prevents that features get lost at the borders. Fig. 5 shows two examplary images acquired by the catadioptric camera.

### A. Indoor Evaluation

The first evaluation of the presented navigation strategy has been performed in our lab. The robot was controlled

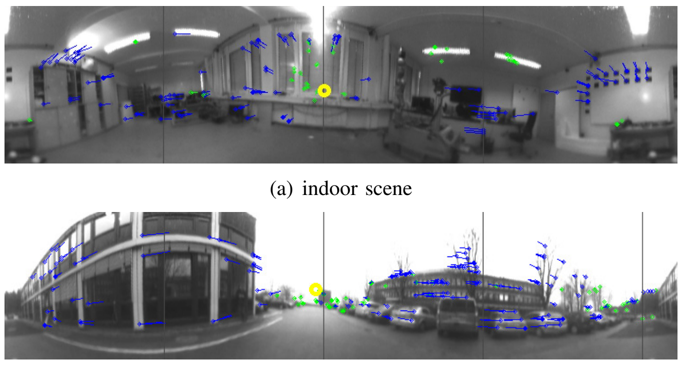
[1]4 ARTtrack1 cameras from ART, http://www.ar-tracking.com
[2]TCRP1201 from Leica, http://www.leica-geosystems.com
[3]http://www.willowgarage.com/pages/software/ros-platform



(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

Fig. 6. These plots show the path following trajectories of the indoor experiments for the complete and the trimmed trees. The green line denotes the mapping trajectory, the blue asterisk represent the locations where viewframes were acquired and the red trajectory illustrates the path followed by the robot. The blue arrow depicts the travelling direction and the start point. The blue circles are the viewframes which were pruned and red markers represent the remaining viewframe branches, whereas their location is chosen as the center of gravity of the pruned viewframes. As marker we use a red asterisk, if enough landmarks for homing are still available, and a red circle otherwise. As threshold for this decision we chose 50 landmarks in our experiments. The numbers next to the red asterisks indicate the amount of remaining landmarks.



(a) indoor scene

(b) outdoor scene

Fig. 5. These screenshots show two examples where the translational optical flow vectors (blue), the landmarks used for rotation estimation (green) and the estimated direction of translation (yellow) are visualized for the indoor and the outdoor scenario. The grey lines depict the angles in 90° steps.

for building a LT-map, acquiring 34 viewframes. For this experiment we simplified the control paradigm in the way that the robot evaluates the images, turns into the new heading direction and goes straight for 15 cm, before it stops again to acquire new images. In that way the results are not smoothed by the controller and more insights into the actual motion estimation can be gathered. Fig. 6 shows the path following performance for different pruning levels. The tree is stepwise trimmed by one level and the resulting trajectories prove that for each pruning step, the accuracy decreases. The robot starts cutting edges but reliably finds its goal even with only four out of eight tree levels. With only three remaining levels, the robot gets lost after some time. It cannot find enough features to reliably estimate the navigation direction and starts a search pattern[4] as illustrated in Fig. 6(f).

TABLE I
MAPPING STATISTICS FOR THE INDOOR SCENARIO

| Height (trim) | View-frames | Land-marks | Memory usage [%] | Viewframe error [m] | Path error [m] |
|---|---|---|---|---|---|
| 8 (0) | 34 | 10171 | 100 | 0.057 | 0.055 |
| 7 (-1) | 33 | 9732 | 95.7 | 0.052 | 0.057 |
| 6 (-2) | 28 | 8322 | 81.8 | 0.133 | 0.099 |
| 5 (-3) | 23 | 7193 | 70.7 | 0.145 | 0.108 |
| 4 (-4) | 18 | 5832 | 57.3 | 0.208 | 0.1127 |
| 3 (-5) | 11 | 3551 | 34.9 | - | 0.1547 |

Table I shows some statistics of the resulting LT-map, like the number of viewframes, the total number of elements and the percentage of memory occupied by the trimmed trees. With almost half the memory usage, the goal could be reliably reached. Furthermore, the performance of the path following is evaluated by computing the average error between the location where the robot assumes to have reached a viewframe and the true location of viewframe acquisition.

[4]As search pattern we implemented a “lying 8” with 120 cm diameter.

In the last column we compare the average distance of each measurement location to the closest point on the mapped trajectory. Both values increase with the number of trimmed levels, which proves a reduced accuracy due to less local information. At this point we want to emphasize that this algorithm does not aim at reproducing a path with high accuracy, but to reliably reach a goal using only limited memory.

### B. Outdoor Evaluation

In our outdoor experiments we used the same setup as for the indoor experiments, beside the application of the velocity controller as described in Section V-B and the use of a tachymeter as ground truth instead of the ART tracking system. The robot was manually steered along a path, which was recorded as LT-map, and afterwards the robot was commanded to return to its origin using three differently trimmed maps. Fig.7 illustrates the trajectories, showing that the path becomes smoother by pruning the tree. As soon as we trimmed five levels, the robot was not able to find enough landmarks to reliably compute the direction of motion and, hence, started to execute the search pattern.

TABLE II
MAPPING STATISTICS FOR THE OUTDOOR SCENARIO

| Height (trim) | View-frames | Land-marks | Memory usage [%] | Viewframe error [m] | Path error [m] |
|---|---|---|---|---|---|
| 9 (0) | 24 | 8493 | 100 | 0.777 | 0.551 |
| 7 (-2) | 23 | 7862 | 92.6 | 0.615 | 0.499 |
| 5 (-4) | 18 | 6229 | 73.3 | 0.872 | 0.408 |

In Table II we list again some statistics of the acquired LT-map and the achieved accuracy. Interestingly, the path accuracy slightly improves by pruning the tree, which can be explained by the fact that the configuration of the velocity controller was rather conservative and, hence, the reaction of the robot to motion changes was quite slow.

Compared to the indoor experiments we acquired less viewframes, which can be explained by a larger average distance of the landmarks. Hence, the similarity measure $\delta_t$ triggers viewframes at larger intervals and, thus, automatically adapts to the information available in the scene.

Fig. 8 shows the computed azimuth direction over time of the run illustrated in Fig. 7(c). The angles are expressed in the robot frame and directly used by the controller. As one can see, the angle to the next viewframe converges to zero after a switch as the robot turns in the new heading direction. If the angle exceeds $\pm 90°$ the robot assumes to have passed the viewframe and it switches to the next one. The other criterion for a viewframe switch is if the similarity measure $\delta_t$ is below the specified threshold, which is illustrated by the green asterisks. As motivated in Section V-B, a five seconds timeout ensures that the robot can align to the new viewframe before the angle criterion triggers the next switch. This is shown by the dashed gaps in the black horizontal lines. The timeout is chosen according to the rotation velocity of the robot to allow for a $180°$ turn.

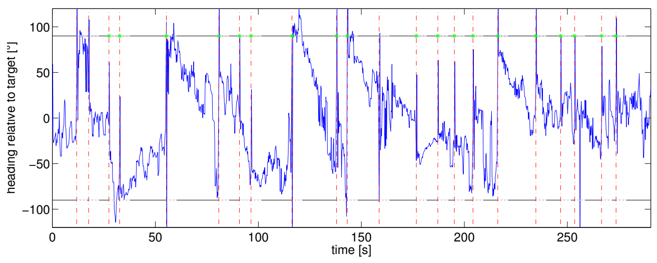


Fig. 8. This plot shows the computed azimuth angles (blue line) for the trajectory illustrated in Fig. 7(c). The red vertical lines denote viewframe switches and the black horizontal lines depict the timeout for viewframe switching. During the periods depicted by two close dashed lines, any viewframe switching which is triggered by the angular constraint is blocked.

## VII. Conclusion and Future Work

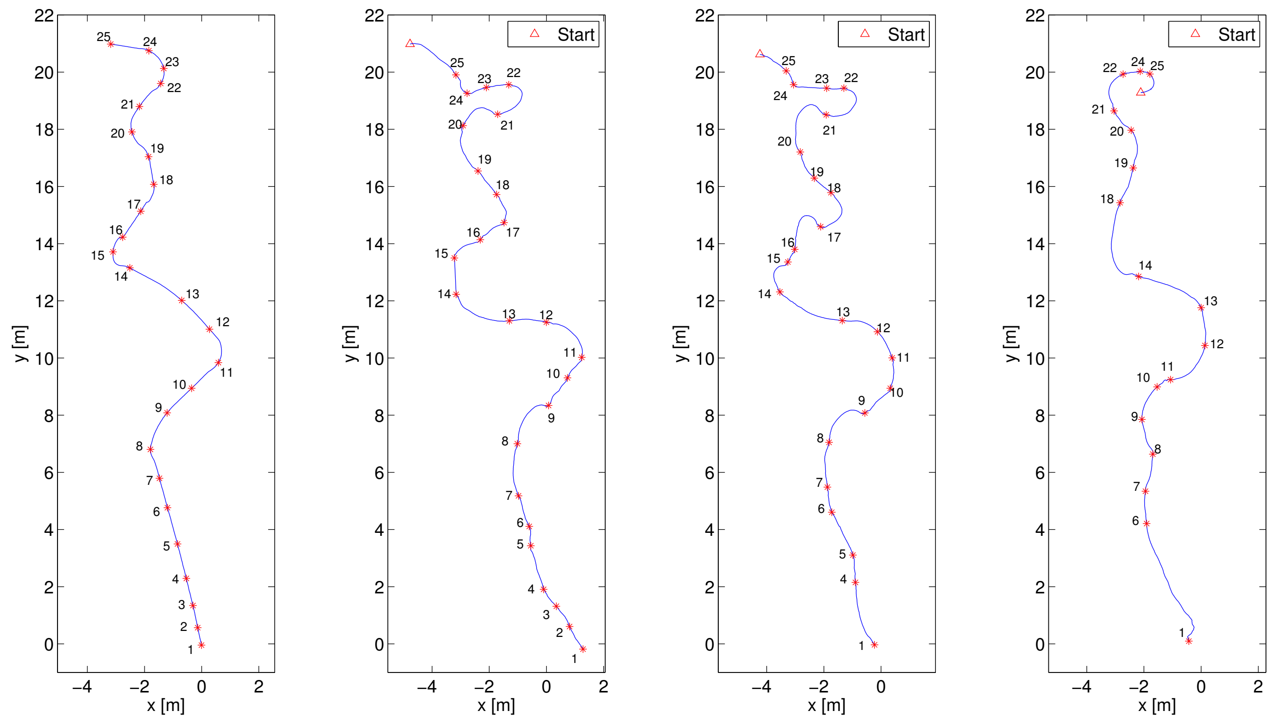
In this work we have presented an efficient and scalable roadmap approach, which can be applied on resource limited platforms to connect distant workspaces. It allows to efficiently adapt the resolution of the map to the provided memory at runtime. No metric information is required – the presented navigation solely relies on the bearing only measurements of a camera. The motion is estimated applying the $Z_\infty$-algorithm to omnidirectional measurements. Both methods benefit from each other, due to similar underlying concepts.

In our experiments we have shown that the navigation algorithm can be realized on small robotic systems, by relying only on a low resolution omnidirectional camera at low framerate. By pruning the tree of the LT-map to free some memory, local information is discarded. This results in smoother trajectories, which, in combination with a local obstacle detection, completely satisfies the requirements it is designed for. Such an obstacle avoidance can be easily realized by directly evaluating the time-to-collision from the optical flow [5].

As a next step, we would like to introduce a measure, which allows us to determine up to which level we can trim the tree and still have a sufficiently high probability to successfully follow the path. We also want to apply the algorithm to a 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 we would like to improve to be able to track landmarks over longer distances.

## References

[1] S. Thrun et al. Robotic mapping: A survey. *Exploring artificial intelligence in the new millennium*, pages 1–35, 2002.
[2] Krzysztof Koperski, Junas Adhikary, and Jiawei 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*. Citeseer, 1996.
[3] R.F. Wang and E.S. Spelke. Human spatial representation: Insights from animals. *Trends in cognitive sciences*, 6(9):376–382, 2002.
[4] M. Augustine, E. Mair, A. Stelzer, F. Ortmeier, D. Burschka, and M. Suppa. Landmark-tree map: a biologically inspired topological map for long-distance robot navigation. In *Robotics and Biomimetics (ROBIO), 2012. IEEE International Conference on*. IEEE, 2012.



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

Fig. 7. The plots show the mapping and the three homing trajectories (blue lines) using differently trimmed trees. The red asterisks with the numbers represent the locations where the respective viewframe was acquired (mapping, (a)) or where the robot switched to the next viewframe (homing, (b-d)). The red triangle denotes the starting point for the homing runs.

[5] E. Mair and D. Burschka. *Mobile Robots Navigation*, chapter $Z_\infty$ - Monocular Localization Algorithm with Uncertainty Analysis for Outdoor Applications, pages 107 – 130. In-Tech, March 2010.

[6] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *ISMAR*, pages 1–10. IEEE, 2007.

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

[8] S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

[9] B. Lisien, D. Morales, D. Silver, G. Kantor, I. Rekleitis, and H. Choset. Hierarchical simultaneous localization and mapping. In *Intelligent Robots and Systems, 2003 (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 1, pages 448–453. IEEE, 2003.

[10] K. Konolige, E. Marder-Eppstein, and B. Marthi. Navigation in hybrid metric-topological maps. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3041–3047. IEEE, 2011.

[11] N. Winters, J. Gaspar, G. Lacey, and J. Santos-Victor. Omni-directional vision for robot navigation. In *Omnidirectional Vision, 2000. Proceedings. IEEE Workshop on*, pages 21–28. IEEE, 2000.

[12] W.L.D. Lui and R. Jarvis. A pure vision-based topological slam system. *The International Journal of Robotics Research*, 31(4):403–428, 2012.

[13] A.C. Murillo, C. Sagüés, J.J. Guerrero, T. Goedemé, T. Tuytelaars, and L. Van Gool. From omnidirectional images to hierarchical localization. *Robotics and Autonomous Systems*, 55(5):372–382, 2007.

[14] I. Ulrich and I. Nourbakhsh. Appearance-based place recognition for topological localization. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, pages 1023–1029. IEEE, 2000.

[15] B. Krose, R. Bunschoten, N. Vlassis, Y. Motomura, et al. Appearance based robot localization. *IJCAI 99 Workshop: Adaptive Spatial Representations of Dynamic Environments*, 1999.

[16] N. Ho and R. Jarvis. Vision based global localisation using a 3d environmental model created by a laser range scanner. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 2964–2969. IEEE, 2008.

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

[18] R. Basri, E. Rivlin, and I. Shimshoni. Visual homing: Surfing on the epipoles. *International Journal of Computer Vision*, 33(2):117–137, 1999.

[19] E. Mair, M. Suppa, and D. Burschka. Error propagation in monocular navigation for zinf compared to eightpoint algorithm. In *Intelligent Robots and Systems (IROS), 2013. IEEE/RSJ International Conference on*. IEEE, 2013.

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

[21] Frédéric Labrosse. The visual compass: Performance and limitations of an appearance-based method. *Journal of Field Robotics*, 23(10):913–941, 2006.

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

[23] J.M. Morel and G. Yu. Asift: A new framework for fully affine invariant image comparison. *SIAM Journal on Imaging Sciences*, 2(2):438–469, 2009.

[24] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[25] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. *Computer Vision–ECCV 2006*, pages 404–417, 2006.

[26] S. Leutenegger, M. Chli, and R.Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *Proc. IEEE Int. Conf. on Computer Vision*, 2011.

[27] A. Stelzer, H. Hirschmüller, and M. Görner. Stereo-vision-based navigation of a six-legged walking robot in unknown rough terrain. *The International Journal of Robotics Research*, 31(4):381–402, 2012.