Robotics and Autonomous Systems 61 (2013) 497–516

Contents lists available at SciVerse ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Dense topological maps and partial pose estimation for visual control of an autonomous cleaning robot

L. Gerstmayr-Hillen [a,b], F. Röben [a], M. Krzykawski [a], S. Kreft [a], D. Venjakob [a], R. Möller [a,b,*]

[a] Computer Engineering Group, Faculty of Technology, Bielefeld University, 33594 Bielefeld, Germany

[b] Center of Excellence 'Cognitive Interaction Technology', 33594 Bielefeld, Germany

ARTICLE INFO

Article history:
Received 5 July 2012
Received in revised form
12 December 2012
Accepted 21 December 2012
Available online 17 January 2013

Keywords:
Topological mapping
Local visual homing
Omnidirectional vision
Cleaning robot control
Complete coverage

ABSTRACT

We present a mostly vision-based controller for mapping and completely covering a rectangular area by meandering cleaning lanes. The robot is guided along a parallel course by controlling the current distance to its previous lane. In order to frequently compute and – if necessary – correct the robot's distance to the previous lane, a dense topological map of the robot's workspace is built. The map stores snapshots, i.e. panoramic images, taken at regular distances while moving along a cleaning lane. For estimating the distance, we combine bearing information obtained by local visual homing with distance information derived from the robot's odometry. In contrast to traditional mapping applications, we do not compute the robot's full pose w.r.t. an external reference frame. We rather rely on partial pose estimation and only compute the sufficient and necessary information to solve the task. For our specific application this includes estimates of (i) the robot's distance to the previous lane and of (ii) the robot's orientation w.r.t. world coordinates. The results show that the proposed method achieves good results with only a small portion of overlap or gaps between the lanes. The dense topological representation of space and the proposed controller will be used as building blocks for more complex cleaning strategies making the robot capable of covering complex-shaped workspaces such as rooms or apartments.

1. Introduction

Autonomous cleaning robots should cover the whole accessible area, while keeping the portion of repeated coverage as small as possible [1,2]. Especially for domestic cleaning robots, this aspect is essential because of their limited battery and computational power. In order to keep the portion of repeated coverage as small as possible and to avoid uncleaned areas, cleaning robots need to be precisely guided along their cleaning trajectory and need to be capable of distinguishing between cleaned places and places which still need to be cleaned. This distinction requires a map of the robot's environment [1,2].

This paper describes a navigation strategy for an autonomous floor-cleaning robot mapping its workspace while moving along parallel and meandering cleaning lanes. The map is not only extended but also concurrently used by the proposed trajectory controller to obtain parallel lanes at a predefined distance. Due to the limited computational power and memory capacities of cleaning robots, all methods proposed in this paper were developed keeping these hardware limitations in mind. In the following, we will briefly outline the two essential aspects of this paper, namely *map building* and *trajectory control* for autonomous cleaning robots.

The *map* has to store all the information about the robot's environment necessary for accomplishing the robot's task at the current or later points in time, and the type of map has to be suitable for the robot's application as well as for the robot's sensory equipment. In our case, the robot (Section 6.2) is equipped with an omnidirectional vision system (textbook: [3]) as main source of information, and local visual homing (review: [4]) is applied for deriving information about spatial relations. Local visual homing can be considered as a special case of ego-motion estimation comparing two panoramic images to compute the change of the robot's orientation (referred to as compass) and the bearing (but not the metrical distance) between the images (please refer to Section 4.1 for details). These particular constraints suggest building a topological map of the environment (reviews: [5,6], textbook: [7]). Such maps are graph-based representations of the environment characterizing places by sensory data, in the case of omnidirectional vision by a panoramic camera image, perceived at that place and linking places if they are directly reachable from each other. We think that such maps are the best choice for our purpose because (i) building such maps from omnidirectional images is possible in real-time even with limited computational

* Corresponding author at: Computer Engineering Group, Faculty of Technology, Bielefeld University, 33594 Bielefeld, Germany.
E-mail addresses: LHillen@ti.uni-bielefeld.de (L. Gerstmayr-Hillen), Moeller@ti.uni-bielefeld.de (R. Möller).
URL: http://www.ti.uni-bielefeld.de (R. Möller).

0921-8890/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2012.12.006

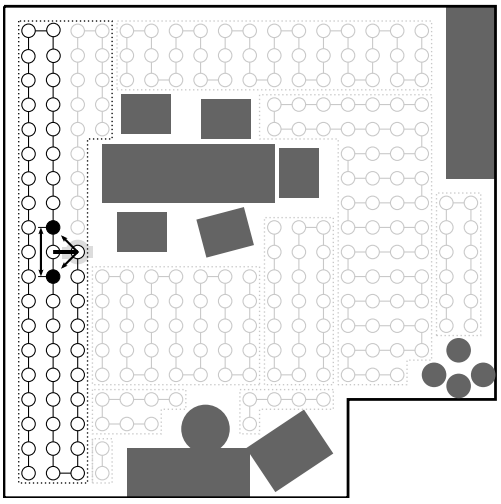

Fig. 1. Key ideas of the proposed trajectory controller. By taking the bearing to snapshots stored along the previous lane (black filled circles) and by combining the angular information (arrows) with an estimate of the distance between the considered snapshots (double-tipped arrow), the distance (thick black line) to the previous lane can be estimated. Complex-shaped workspaces can be incrementally covered completely by combining several cleaning segments (dotted areas) of parallel and meandering lanes as resulting from the control strategy proposed in this paper.

resources, (ii) the trajectory controller can use our existing local visual homing algorithms (Section 4.1) for estimating spatial relations between place nodes, and (iii) additional (and non-visual) task-relevant information can be attached to the place nodes. In their purest form, topological maps do not contain metrical position information and are a sparse representation of space applied for coarse navigation (Section 2.2). However, our particular task requires both position information to keep the robot at a predefined distance to its previous lane and precise navigation. To resolve these problems, a (partial) position estimate is attached to the place node when adding it to the map (leading to topo-metric maps; Section 2.1.2) and place nodes are added at a finer spatial resolution (leading to dense topological maps; Section 4.2) allowing for frequent controller updates. At later stages of the cleaning process (which are beyond the scope of this paper), the map will provide the spatial information required (i) to detect and to approach areas which still need to be cleaned and (ii) to visually detect areas which already have been cleaned (loop-closure problem, e.g. [6,8,9]) at a fine spatial resolution.

The *trajectory controller* proposed in this paper uses the map to keep the robot on parallel lanes while concurrently extending the map. By this means, a part of the robot's entire workspace is covered by meandering and parallel lanes (Fig. 1, black dotted area). At the beginning of a cleaning run, the robot moves straight forward and successively adds place nodes (circles) to its dense topological map. At the end of the lane, the robot turns if possible and starts a new lane. From the second lane on, the controller not only stores place nodes while moving, but also uses neighboring snapshots (filled circles) stored along the previous lane to estimate its current distance to the previous lane. By applying a local visual homing algorithm, the bearing and the compass information to at least two snapshots (arrows) are computed. Furthermore, the spatial distance between the considered snapshots (double-tipped arrow) is determined from the robot's wheel odometry. Both sorts of information are then combined in order to estimate the robot's distance to the previous lane (thick black line). This estimate is passed to a controller generating a movement command keeping the robot at a constant distance to the previous lane (and hence on a course parallel to this lane).

To estimate the robot's distance to the previous lane, we do not determine the robot's full pose w.r.t. world coordinates as is the case for most related approaches (Sections 2 and 3.2). Rather, we only compute the information which is sufficient and necessary to fulfill the task. In our case, this includes estimates of the robot's distance to the previous lane and of its orientation. With only one of these estimates lacking, the robot would not be able to fulfill the task. For example, in case the distance estimate was missing, the robot would not be able to keep its distance to the previous lane constant resulting (i) in a course parallel to the previous lane but following it at an arbitrary distance and (ii) in inter-lane distances varying from lane to lane. We refer to our approach as partial pose estimation in order to distinguish it from standard approaches estimating the robot's full pose (Section 2). The advantage of partial pose estimation over full pose estimation is that the former is computationally less demanding—which we think is an important aspect if one considers the limited computational power of an autonomous cleaning robot. Due to this aspect, we also do not correct the partial pose estimates after adding the corresponding place node to the dense topological map. This aspect distinguishes our approach from related work on visual simultaneous localization and mapping (SLAM; Sections 2, 3.2.2, and 3.2.3).

By the proposed controller, a single segment of parallel and meandering lanes can be covered. The robot's entire workspace can be completely covered by combining several of these cleaning segments (Fig. 1, light-gray); the strategies required for this purpose are subject of future work (Section 9). Thus, the presented control strategy and mapping algorithm can be understood as the basis of a system making an autonomous floor-cleaning robot relying on omnidirectional vision capable of cleaning complex areas.

The remainder of the paper is structured as follows: Sections 2 and 3 review the fields of map-based navigation relying on omnidirectional images and of autonomous cleaning robots. The main building blocks of our trajectory controller, namely local visual homing and dense topological maps, are described in Section 4 by introducing existing work and outlining how it is extended or applied by the proposed navigation strategy. On the basis of these principles, the proposed trajectory controller is presented in Section 5. Experiments and results are reported in Sections 6 and 7. The article is summarized in Section 8, and future working directions are pointed out in Section 9. The trajectory controller proposed in this paper was developed in a feasibility study [10] and was – without detailed mathematical description – presented at conferences [11,12] and described in a dissertation [13].

2. Map-based navigation with omnidirectional images

This section reviews existing work on visual long-range navigation (Sections 2.1 and 2.2) and relates these approaches to the navigation strategies proposed in this paper (Section 2.3). Due to the large amount of work published in this field, we restrict our review to navigation methods (i) based on omnidirectional vision and (ii) published since 2000.

Approaches to mobile robot mapping can be partitioned into *quantitative* and *qualitative* methods (Fig. 2). Quantitative maps (Section 2.1) include an estimate of the robot's position with respect to an external frame of reference (usually referred to as world coordinate system) and in some cases also position estimates of the visible features. In contrast, qualitative maps (Section 2.2) do not contain metrical position information, but rather model spatial interrelations and characterize places by the sensory information, i.e. the camera image, perceived at the place. These methods use a graph-based representation of space. Known positions are represented by nodes and linked by edges if they are directly reachable from each other. Both approaches have in common that reliable navigation requires consistent maps. This can be achieved by correct data association, by clever sensor-data integration, by reliable loop-closure detection, and by avoiding both spatial aliasing and perceptual variability [6,14].

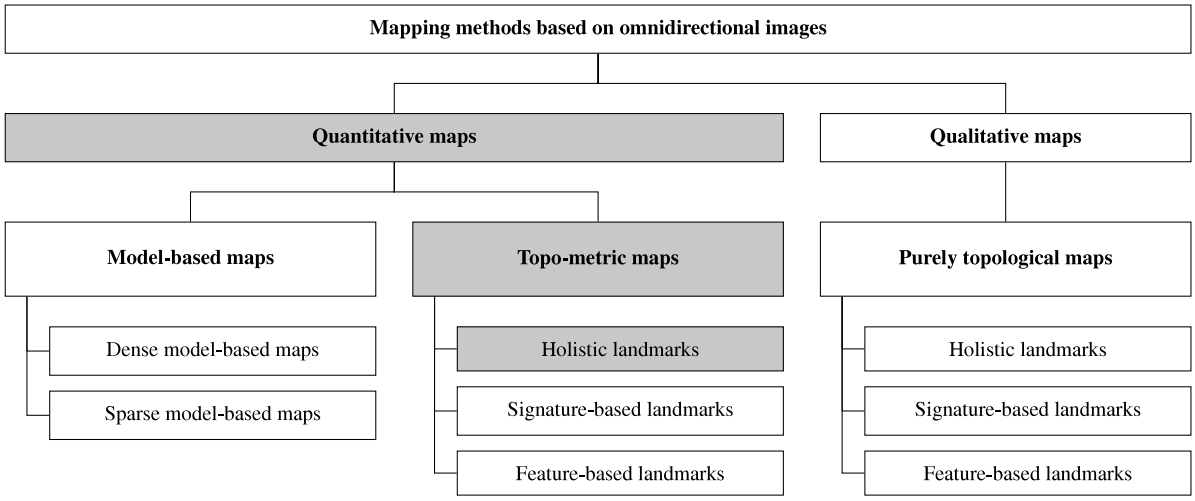

Fig. 2. Taxonomy of mapping methods based on omnidirectional vision. The proposed trajectory controller is a *quantitative method* relying on a *topo-metric map and holistic landmarks* environmental representation.

2.1. Quantitative mapping

Quantitative maps (also referred to as geometrical maps) build a geometrical representation of space involving position information of the robot and in some cases of the visible landmarks. Hence, places are characterized by their position with respect to a frame of reference [15]. The difficulties of quantitative mapping are (i) spatiotemporal sensor-data fusion and (ii) to relate the robot's current sensor data to a position estimate. The majority of methods use Bayesian filters or constrained optimization techniques, which both require correct data association and efficient correction steps to prevent the position estimate from drifting from the robot's true position [6,15]. Most quantitative approaches rely on a *single frame of reference*. However, methods relying on *multiple frames of reference* exist and are referred to as hybrid maps or hierarchical methods. For theoretical details on these methods please refer to the reviews [16,17] or the textbooks [6,18]. Examples relying on omnidirectional vision include [19–25]. Quantitative maps can be further categorized into *model-based maps* (Section 2.1.1) and *topo-metric maps* (Section 2.1.2).

2.1.1. Model-based maps

Model-based maps are detailed and geometrical representations of the robot's environment containing the positions of the mapped landmarks w.r.t. an external reference frame. Hence, mapping methods of this class are closely related to structure-from-motion methods [26]. In order to update the map or in order to localize the robot within the map, the robot's current sensor data has to be transformed into a local map. For this purpose, external sensor information and internal sensor data are fused. In a second step, the obtained local map has to be matched with the existing global map in order to derive a position estimate [27]. Model-based maps can be further divided into *dense maps* and *sparse maps* [22].

Dense model-based maps. Dense model-based maps represent the robot's environment by a set of 3D surfaces or by a 2D occupancy grid. However, they are computationally demanding and cannot cope with feature-free or non-textured environments [22]. Examples of such maps include applications for full 3D environment reconstruction [28–32], for building 2D grid maps [33–37], or for localization in 2D footprint maps [38–40].

Sparse model-based maps. Methods building sparse model-based maps extract features from the omnidirectional images and estimate both the robot's position and the features' positions with respect to the world coordinate system. Features are usually detected by point-of-interest detectors and represented by feature descriptors (reviews: [41–43]). As the map consists only of a set of features in space, it requires a small amount of storage, but the resulting map is more difficult to interpret for humans than other quantitative maps. Landmark matching is done by matching feature descriptors. This type of map is especially suited for feature-based simultaneous localization and mapping (SLAM) techniques (review: [8]). Such methods iteratively correct both the robot's position as well as the landmarks' positions in order to improve the accuracy of the map over time. To this end, they are closely related to bundle adjustment [44,45]. As features, point features [46–48] and line features [49] are used.

2.1.2. Topological maps with metrical information

Topological maps with metrical information, or shorter topo-metric maps, combine the advantages of quantitative and qualitative maps [14]. Like qualitative methods (Section 2.2), they rely on a graph-based representation. Places are nodes in the graph and are characterized by the raw camera image (or by information directly derived from the image) taken at that place in combination with an additional estimate of the robot's position at the time of image acquisition. Due to the graph-based representation, all places stored in the map are former robot positions, which are used as landmarks for deriving spatial relations or for correcting the map. Traditional applications aim at building a *sparse* graph reducing the memory requirements for storing the map, although *dense* topo-metric map are – like in our case – possible (Section 4.2). Like for quantitative maps (Section 2.1), the position estimate allows for disambiguating places with identical visual appearance and for approaching arbitrary positions (and not only snapshot positions as it is the case for qualitative methods). Depending on how places or landmarks are characterized, the following three subcategories exist: *holistic landmarks*, *signature-based landmarks*, and *feature-based landmarks*.

Holistic landmarks. Holistic methods use the entire panoramic image to characterize a place. The main drawback of using images as landmarks is that localization or place recognition usually requires a pixel-by-pixel comparison of the images. Navigation methods using maps of this class often fuse odometry information and visual bearing information obtained by local visual homing (Section 4.1). The graph-based SLAM system presented by [50]

estimates the robot's full pose and iteratively updates the map based on odometry and bearing information.

The method presented in this paper also relies on holistic landmarks and on a visual homing method operating on such landmarks to estimate angular relations between landmarks. In contrast to the method of [50], our approach builds a dense map of the environment, performs only a partial pose estimation, and keeps the partial pose information stored in each place node constant without correcting it at a later point in time.

Signature-based landmarks. To reduce the computational complexity of the image comparisons required to accomplish the robot's task (e.g. localization or loop-closure detection), these methods derive a lower-dimensional description from the entire camera image, which is then used for comparison. A Monte-Carlo localization method for a robot navigating in a do-it-yourself store using signatures based on color statistics is described by [51,52].

Feature-based landmarks. The methods of this class characterize a place by a set of visible features described by a feature descriptor instead of using the entire image or a lower-dimensional description of the entire image. In contrast to sparse model-based maps (Section 2.1.1), the spatial positions of the features in the world are not known. Topo-metric maps and feature-based landmarks are used for Monte-Carlo localization [19,20,23,53–59] and for trajectory-based SLAM [56,60–64]. As trajectory-based SLAM methods correct the map by fusing odometry and bearing information, these methods are closely related to the described trajectory controller.

2.2. Qualitative mapping

In contrast to quantitative maps, qualitative maps model the robot's workspace without position information. To this end, such navigation methods also lack the steps of fusing sensor data into a common frame of reference and to infer an estimate of the robot's position w.r.t. that frame of reference. Quantitative methods rather operate on the raw sensor data and therefore are (in many cases) computationally less demanding than quantitative methods [5,14,15]. All qualitative methods rely on a graph-based representation of space referred to as purely topological maps. In case the graph is not branched, the representation is also referred to as route map [5].

2.2.1. Purely topological maps

Like topo-metric maps (Section 2.1.2), purely topological maps are usually sparse, graph-based representations of space using former robot positions as landmarks. As they do not contain an estimate of the robot's position, it is not possible to distinguish different places with identical visual appearance (spatial aliasing, Section 2.1), and it is not possible to accurately approach arbitrary positions in space. Due to this, topological maps are often used for coarse navigation. For path planning and other applications operating on the topological map, standard graph algorithms (textbook: [65]) can be used. Like for topo-metric maps (Section 2.1.2), one can further distinguish purely topological maps with *holistic landmarks*, with *signature-based landmarks*, and with *feature-based landmarks*. For a more detailed discussion of the different landmark types please refer to Section 2.1.2.

Holistic landmarks. Maps belonging to this category characterize places by the entire camera image. Such maps are used for route following [66–69] and for topological mapping [70].

Signature-based landmarks. Instead of storing and comparing the entire panoramic image, signature-based methods rely on a global image signature. This signature is derived from the entire image and has a much lower dimensionality than the number of pixels. For localization or place recognition, these methods apply histogram-based signatures [71–73], Fourier-based signatures [73,74], statistical signatures [73], or eigenspace representations [75–77].

Feature-based landmarks. Feature-based approaches characterize places by a set of visible features as detected by point-of-interest operators. Examples include route following [78], localization [79,80], online mapping [81], and offline mapping [82–84].

2.3. Relevance for our work

As outlined in the introduction (Section 1), our approach for systematically covering an area requires a map which (i) can be efficiently built from omnidirectional images, (ii) allows for smooth interaction with local visual homing, and (iii) stores metrical position information required for keeping the robot at a predefined distance to its previous lane. Aspects (ii) and (iii) exclude the usage of *qualitative maps* (Section 2.2) because such maps do not contain metrical position information at all. Hence, although the robot could be guided along a parallel course the distance to the previous lane could not be adjusted. Among *quantitative maps*, we consider *model-based maps* (Section 2.1.1) unsuitable because (i) they are among the methods requiring the most computational power and memory capacity, (ii) they are often built in an off-line process after exploring the robot's environment, and (iii) applying local visual homing is very difficult or even impossible with such a representation of space. We rather consider *topo-metric maps* (Section 2.1.2) to be the most appropriate choice due to fully meeting our requirements. The particular combination with the used homing method (min-warping; Section 4.1.1) suggests using a map relying on *holistic landmarks*.

Trajectory-based visual SLAM methods (e.g. [50,56,60–64]) are closely related to our approach because they integrate odometry and bearing information for building a topo-metric map. However, they use newly obtained sensor information for a posterior correction of former pose estimates. With the presented approach, we avoid updating position estimates after adding the place node to the map thus circumventing the computationally demanding correction steps. While not necessary at the current state of the work, we also hope that future extensions of this method do not require posterior pose corrections; we rather aim at covering the robot's entire workspace by several locally consistent segments (for details please refer to Section 7.4.4) as obtained by the described trajectory controller.

3. Autonomous floor-cleaning robots for household usage

Cleaning robots are considered a sub-domain of mobile service robots assisting humans in monotonous and tedious tasks [85,86]. The following review is limited to the field of domestic floor-cleaning robots; for a review on other application domains, the reader is referred to [87]. As domestic floor-cleaning robots are both commercially available and subject of academic research, Sections 3.1 and 3.2 cover these two fields; the relevance of the related works for the presented navigation strategy is discussed in Section 3.3.

3.1. Commercial floor-cleaning robots

Commercial floor-cleaning robots are available on the market since the late 1990s. Depending on their navigation strategy, two generations of cleaning robots can be distinguished: first generation robots rely on preprogrammed movement patterns or

on random-walk strategies. Hence, achieving complete coverage requires a relatively large amount of time and results in a large proportion of repeated coverage [88,89]. Because of their limited sensor equipment, these robots are not capable of building a map of their workspace and cannot distinguish covered areas from areas which still need to be cleaned [2]. The series of Roomba robots produced by iRobot are the most well known examples of this generation and also the most frequently sold cleaning robots in the world [87].

Second generation robots can cover their workspace more efficiently and with a smaller proportion of repeated coverage. For this purpose, they rely on systematic exploration strategies (usually parallel and meandering lanes), and they can recognize and directly approach uncleaned areas. To accomplish this task, these robots are equipped with better sensors, and it is likely that they build a map of their workspace. Examples of second generation robots relying on a 360° laser range finder include the Neato XV series and the Vorwerk Kobold VR100. The LG Hom-Bot 2.0, the Samsung NaviBot series, and the Philips HomeRun FC9910 all use a monocular camera directed towards the ceiling in combination with a gyroscope as main source of sensory information.

3.2. Academic research on floor-cleaning robots

Most academic research on autonomous cleaning robots focuses on certain aspects of cleaning robots rather than on entire systems. The most common of these aspects include planning of cleaning paths or simultaneous localization and mapping (SLAM) based on vision and based on other sensor modalities. These aspects will be briefly discussed in the following.

3.2.1. Planning of cleaning paths

For planning of cleaning paths, complete coverage planning algorithms are applied. These algorithms aim at completely covering the robot's workspace while keeping the proportion of repeated coverage as small as possible [90]. Coverage algorithms can be categorized into algorithms assuming the map of the environment to be a priori known (e.g. [91]) and algorithms including the mapping process (e.g. [92–94]). For details on these methods, the reader is referred to the reviews by [89,90,94]. Frontier-based planning methods lift the constraint of physically visiting every accessible place. These methods rather aim at complete sensor coverage for mapping the workspace, and thus plan paths to approach freespace at the border of explored areas (e.g. [95–97]). We are currently not aware of algorithms for the planning of cleaning paths relying on (omnidirectional) vision as main source of sensor information.

3.2.2. Visual SLAM

SLAM algorithms (Section 2) concurrently localize the robot within a map while extending this map and correct the map whenever new sensor data is available. Most SLAM algorithms in the context of cleaning robot navigation rely on a monocular camera directed towards the ceiling. In terms of the hierarchy proposed in Section 2, these methods belong to the class of sparse model-based maps (Section 2.1.1). The methods by [98,99] use an extended Kalman filter (EKF) for estimating the robot's pose and the positions of the landmarks. The approach was extended by [100,101] to multi-robot SLAM using a particle filter for state estimation. All these papers focus on state estimation and map building and do not consider further aspects of cleaning robot navigation. Nevertheless, figures and supplemental videos suggest that the Samsung robots (Section 3.1) rely on these navigation strategies.

The SLAM method proposed by [102] relies on a monocular camera directed towards the robot's movement direction. It estimates the robot's current state and the 3D positions of the known feature points with an EKF framework. New features are added without delayed measurement by assuming a large initial uncertainty along the feature's viewing direction and later on refining this uncertainty.

A low-dimensional feature descriptor for corner features is proposed by [103], and the method is demonstrated in the context of robot cleaning. For this purpose, a database of known features on the ceiling is used, and each feature is associated with its position for Monte-Carlo localization. Although not described in [103], the method could also be applied for mapping or visual SLAM.

3.2.3. SLAM with other sensor modalities

A SLAM algorithm referred to as vector-field SLAM is described by [104,105]. For position estimation by an extended Kalman filter, by an exactly sparse extended information filter (ESEIF), or by graph-based optimization techniques, the method learns the spatial variation of a continuous signal. As external reference signal, the robot currently derives bearing information from static IR spots projected onto the ceiling by the Northstar system [106].

A similar approach referred to as magnetic field-based SLAM is used by [107,108]: the robot learns a map of anomalies of the ambient magnetic field arising in indoor environments, and a particle filter is used for localization of the robot and for estimating its state.

3.3. Relevance for our work

In the following sections, we discuss how the works reviewed in Sections 3.1 and 3.2 relate to our work with respect to the robot's sensory equipment (Section 3.3.1) and navigation strategies (Section 3.3.2).

3.3.1. Sensory equipment

The long-term goal of our work is to develop a set of navigation strategies making an autonomous cleaning robot capable of systematically and completely covering its workspace. As this objective involves mapping the robot's environment [1,2], our robot should rely on sensors facilitating this task. This suggests using more elaborated sensors like those used by other second generation robots such as vision or laser range finders. Although omnidirectional vision is currently not used by comparable commercial or academic floor-cleaning robots, we think it is an appropriate sensor for this task. The advantages of such a camera setup are (i) that the robot is capable of capturing a 360° view of its environment with only a single camera image, (ii) the visible image content is independent of the robot's orientation, (iii) that rotational and translational motion components can be easily separated, which is important for ego-motion estimation [109], and (iv) that a dense three-dimensional signal (two-dimensional pixel grid plus intensity information) is obtained. The fourth aspect distinguishes vision from laser range finders sensing a dense one-dimensional pixel grid with distance and reflectance information and from IR distance sensors returning sparse direction and distance information with a very limited range only. Compared to a monocular camera directed upwards, an omnidirectional vision setup mainly perceives the walls of the robot's workspace and therefore offers more visible structure than the ceiling. Because of their 360° field of view and their viewing direction, visible objects do not vanish from panoramic images due to robot movements but only due to environmental properties such as occlusions. We think that this is an important advantage over standard cameras. However, these are exactly the properties making omnidirectional methods more prone against image disturbances (e.g. caused by dynamic scene changes or changes of the illumination; Section 7.4.3)

Fig. 3. Local visual homing. The robot moves from its start position (light gray) to its current position (gray). By comparing the current view (CV) with the snapshot, i.e. camera image acquired at the start position (SS), the home direction α as well as the relative orientation change ψ between the considered snapshots (also referred to as visual compass) can be estimated w.r.t. the robot's current orientation (figure: black thick bars). The panoramic images show the laboratory in which experiments were conducted; images were obtained with the omnidirectional vision setup and the parameters used for the experiments (Section 6).

than methods using standard cameras facing the ceiling, especially if these cameras only have a narrow field of view. Regarding our particular application, the main drawback of omnidirectional vision setups is that they cannot be completely enclosed by the robot's housing (as is the case for monocular cameras directed upwards), and therefore protrude. Nevertheless, by relying on compact omnidirectional vision setups like panoramic annular lenses (as used for our experiments; Section 6.2), the mirror-lens combination by [110,111], or wide-angle fish-eye lenses, this effect can be reduced, and it is possible to build flat robots capable of cleaning underneath obstacles such as beds.

3.3.2. Navigation strategy

Using omnidirectional images as main sensory information makes our approach closer related to omnidirectional navigation methods building topo-metric or purely topological maps (Sections 2.1.2 and 2.2.1) than to the methods reviewed in Section 3.2. Additionally, it limits the range of applicable navigation strategies. Due to being tailored for usage with their particular sensor modalities, vector-field SLAM [104,105] and magnetic field SLAM [107, 108] could not be applied with our robot (Section 6.2). Although the SLAM methods relying on a standard camera facing upwards (Section 3.2.2) could be extended for usage with an omnidirectional vision setup, we expect them to be computationally more demanding than our method because these methods include computationally demanding steps of detecting and matching features. Even if they could be executed on a robot's on-board computer, we expect them to require a more powerful (and hence more expensive) computer than our method. The reviewed methods for planning of cleaning paths (Section 3.2.1) cannot be applied with our robot setup because (i) they do not rely on (omnidirectional) vision but on range information, (ii) they assume an estimate of the robot's full pose to be known (often without describing how this estimate is obtained, and (iii) at least some of the reviewed works do not include a mapping stage but assume the map to be a priori known.

4. Essential building blocks

This section introduces the two essential building blocks for the proposed trajectory controller, namely local visual homing (Section 4.1) and dense topological maps (Section 4.2), and outlines how these concepts are applied or extended by our navigation strategy.

4.1. Local visual homing

Local visual homing (Fig. 3) is the ability to return to a previously visited place based solely on visual information. As it is a qualitative navigation method, it does not rely on metrical position estimates. Visual homing methods are strongly influenced by the snapshot hypothesis of insect navigation [112]: it states that places are solely characterized by images taken at the corresponding positions and that homing can be achieved by comparing the current camera image (referred to as current view, CV) and the image stored at the home position (referred to as snapshot, SS). By comparing the two images, a home vector pointing from the current position to the snapshot position is computed. To return to the snapshot position, the robot follows the direction of the home vector and – due to erroneous estimates – repeats the homing step until it reaches its goal position. As local visual homing methods are purely local navigation methods, their navigation range is restricted to the robot's sensory horizon [5]. Due to this, every snapshot position is surrounded by a catchment area. For positions within this area, homing to the snapshot position is possible. The navigation range can be extended by integrating several snapshots into a graph-based representation of space (Sections 2.1.2 and 2.2.1).

For the proposed navigation strategy (Section 5), the robot does not physically approach the snapshot position. Rather, visual homing is used to take the bearing from the current robot position to a former robot position represented by the corresponding snapshot stored in the topological map. Thus, like other long-range navigation methods building on graph-based representations of space, our approach uses former robot positions as landmarks and applies local visual homing to determine angular relations between them. However, these angular relations only define the spatial arrangement of the considered snapshots up to scale. For this reason, it is not possible to control the robot's distance to the previous lane solely based on angular relations. To solve the task, we use a topological map with partial pose information (Section 2.1.2) and estimate the distance between consecutive snapshots along the lane based on the robot's odometry (Section 5.3).

4.1.1. Min-warping with compass acceleration

Among all homing methods (recent review: [4]) image-warping methods [4,113,114] have proven to be accurate and robust. Furthermore, warping methods not only compute an estimate $\hat{\alpha}$ of the home direction but also of the azimuthal orientation difference $\hat{\psi}$ between the two images (visual compass, Fig. 3). For the proposed navigation strategy, we use a variant of 2D-warping called min-warping with compass acceleration because it offers a good trade-off between homing accuracy and computational complexity [4]. The method uses an explicit search for discrete combinations of the home direction α and the orientation change ψ. Depending on α and ψ, the shift and scale change of an image feature (in the case of min-warping the complete image column of a panoramic image) can be computed. The search space of orientation changes can be restricted to a small fraction of likely orientations by applying a visual compass method implicitly contained in the min-warping method (in [4] referred to as compass acceleration). The parameters α and ψ resulting in the best match between snapshot and current view columns are used as estimates $\hat{\alpha}$ and $\hat{\psi}$ of the home direction and of the orientation change. Since warping methods use the entire image to derive the home vector and compass estimates, they are well suited for long-range navigation methods relying on topological maps (with or without position information) using holistic landmarks to characterize places (Sections 2.1.2 and 2.2.1).

4.2. Dense topological map

For a robot using omnidirectional vision as main sensory input and relying on local visual homing to determine the spatial arrangement of landmarks, it is straightforward to build a topological representation of space. As described in Section 2, traditional

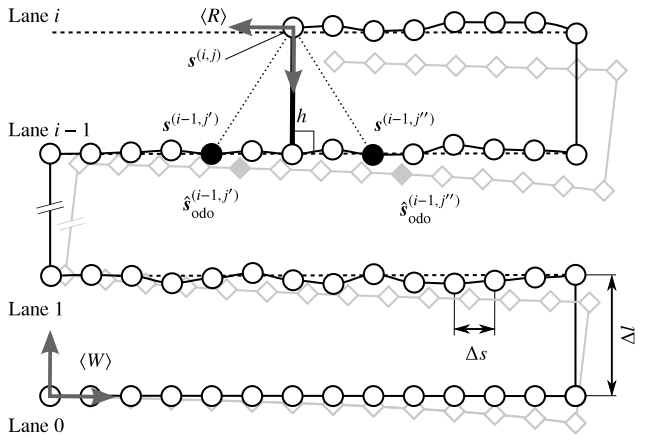

Fig. 4. Positions and coordinate systems involved in the proposed navigation strategy. The diagram shows the true robot trajectory (black line with circles), the trajectory based on odometry estimates (gray line with diamonds), and the ideal cleaning trajectory (black dashed line). Circles depict robot positions $\boldsymbol{s}^{(i,j)}$ at which snapshots $\boldsymbol{S}^{(i,j)}$ were taken; diamonds mark the corresponding odometry estimates $\hat{\boldsymbol{s}}^{(i,j)}_{\text{odo}}$. The robot's current distance h to the previous lane equals the height of the triangle defined by $\boldsymbol{s}^{(i,j)}$, $\boldsymbol{s}^{(i-1,j')}$, and $\boldsymbol{s}^{(i-1,j'')}$.

topological maps (both with and without metrical information) are sparse representations of space. We are of the opinion, that the spatial resolution of such a sparse map is not suitable for our application because a certain number of snapshots is required to (i) control the robot's distance to its previous cleaning lane and (ii) to determine areas which still need to be cleaned and to find loop closures on a relatively fine level. These problems can be circumvented by sampling the robot's workspace more densely, thereby increasing the number of place nodes stored in the map.

In our application, a dense topological map is used to guide a floor-cleaning robot along parallel and meandering lanes. While moving along a cleaning lane i, a new place node $\mathcal{P}^{(i,j)}$ is added to the map whenever the robot's distance to the previous place node $\mathcal{P}^{(i,j-1)}$ exceeds the inter-snapshot distance Δs. For this purpose, the distance traveled since adding $\mathcal{P}^{(i,j-1)}$ is measured based on the robot's odometry. By this means, a regular and grid-like representation of the robot's workspace is built. Each place node

$$\mathcal{P}^{(i,j)} = \left\{ \boldsymbol{S}^{(i,j)}, \hat{\boldsymbol{s}}^{(i,j)}_{\text{odo}}, \hat{\theta}^{(i,j)}_{\text{vis}} \right\} \tag{1}$$

contains the sensory information required for the further processing steps of the trajectory controller: the robot's current snapshot $\boldsymbol{S}^{(i,j)}$, an odometry-based estimate of the robot's position

$$\hat{\boldsymbol{s}}^{(i,j)}_{\text{odo}} = \left(\hat{x}^{(i,j)}_{\text{odo}}, \hat{y}^{(i,j)}_{\text{odo}} \right)^{\top}, \tag{2}$$

and a visually determined estimate $\hat{\theta}^{(i,j)}_{\text{vis}}$ of the robot's orientation. Lanes and snapshots along a lane are counted by $i = 0, 1, \ldots, i_{\max}$ and $j = 0, 1, \ldots, j_{\max}$, respectively. The following paragraph will describe the information stored in the place nodes in more detail.

The panoramic snapshot $\boldsymbol{S}^{(i,j)}$ is acquired at the unknown snapshot position

$$\boldsymbol{s}^{(i,j)} = (x, y, \theta)^{\top} \tag{3}$$

and is obtained by capturing an omnidirectional camera image, unfolding it to a cylindrical image, and preprocessing it (Section 6.2). As we build a topological map relying on holistic landmarks (Sections 2.1.2 and 2.2.1), we use the entire image to characterize the position $\boldsymbol{s}^{(i,j)}$. The odometry estimates $\hat{\boldsymbol{s}}^{(i,j)}_{\text{odo}}$ (Fig. 4, gray diamonds) are given w.r.t. the world coordinate system $\langle W \rangle$, which is defined as a right-handed coordinate system at the initial robot position with its x-axis being aligned with the robot's initial movement direction. Due to the inaccuracy of the robot's odometry, the estimates $\hat{\boldsymbol{s}}^{(i,j)}_{\text{odo}}$ can deviate from the true but unknown robot positions $\boldsymbol{s}^{(i,j)}$. Because of this, true robot positions $\boldsymbol{s}^{(i,j)}$ and corresponding odometry estimates $\hat{\boldsymbol{s}}^{(i,j)}_{\text{odo}}$ are strictly separated in the description of the proposed controller (Section 5). The odometry estimates $\hat{\boldsymbol{s}}^{(i,j)}_{\text{odo}}$ are used for selecting neighboring snapshots (Section 5.3.1) and for estimating the base of the triangle for triangulation (Section 5.3.3). The visual orientation estimate $\hat{\theta}^{(i,j)}_{\text{vis}}$ is obtained by fusing the compass information obtained from local visual homing with the orientation estimates of the previous lane (Section 5.4.2).

At the current stage, we do not add links to the topological map because the proposed controller operates solely on the graph's nodes. Nevertheless, links could for example be added depending on the neighbors or triangulation sets computed by the trajectory controller (Sections 5.3.1 and 5.3.2).

In contrast to related work on trajectory-based SLAM, where the robot's full pose is determined for visually correcting the estimate of the robot's state (Section 2.1.2), we do not correct the estimates stored in the map at a later point in time, and we solely compute the robot's orientation $\hat{\theta}_{\text{vis}}$. We refer to the latter aspect as partial pose estimation. Although we do not compute the robot's full pose, we consider the map built by our navigation strategy to be a topometric map because all orientation estimates $\hat{\theta}^{(i,j)}_{\text{vis}}$ are computed w.r.t. an external reference coordinate system. In future work, the place nodes can be extended by further task-relevant information such as local obstacle information.

5. Trajectory controller

In this section, the proposed trajectory controller is described in detail. Fig. 5 visualizes the sequence of processing steps required to guide the robot along parallel and meandering lanes.

5.1. First lane

While the robot moves along the first lane (lane counter $i = 0$), it successively adds place nodes $\mathcal{P}^{(0,j)}$ to its topological map. The first lane is considered a special case because it does not have a predecessor, and therefore all processing steps requiring snapshots stored along the previous lane cannot be applied. These steps include (i) the lane-distance estimation and (ii) the estimation of the robot's current orientation. Both steps are required to compute motion commands for correcting the deviation from the desired inter-lane distance. Instead of estimating and controlling the robot's distance to the previous lane, the first lane has to be kept straight by beacon aiming or – as in the case of our experiments (Section 6) – by wall following. As the robot's orientation cannot be estimated, all orientation estimates $\hat{\theta}^{(0,j)}_{\text{vis}}$ are assumed to be zero:

$$\hat{\theta}^{(0,j)}_{\text{vis}} = 0. \tag{4}$$

As a simplification for our experiments, the first lane is ended after traveling for a fixed distance. If the proposed controller were integrated in a framework of further cleaning strategies, the first lane would end if an obstacle were detected or if the robot approached an already cleaned area. All subsequent lanes end if an obstacle is encountered or if the robot reaches the end of the previous lane. Since the proposed strategy relies on images stored along the previous lane for taking the bearing, the current lane cannot exceed its direct predecessor. In case the current lane is not blocked by an obstacle, reaching the end of the current lane is detected by simply counting the snapshots stored along this lane and stopping the robot if the number equals that of the previous lane. In principle, the snapshot counter j could be used to estimate the robot's

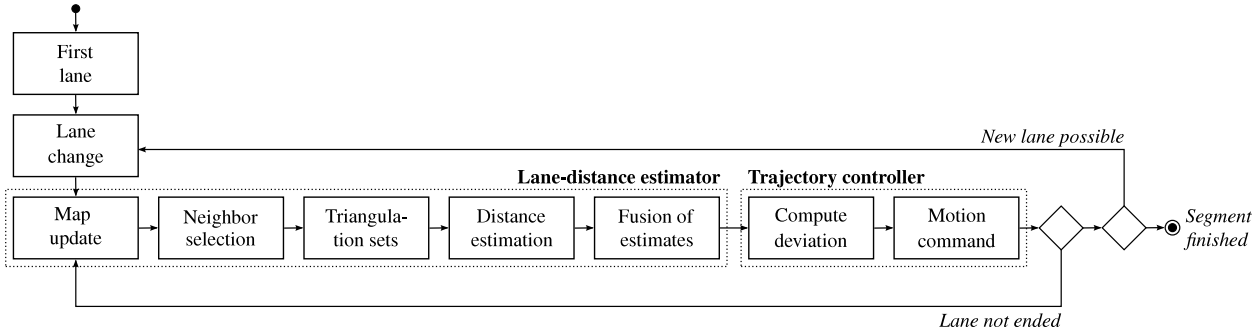

Fig. 5. Flow chart of the proposed trajectory controller.

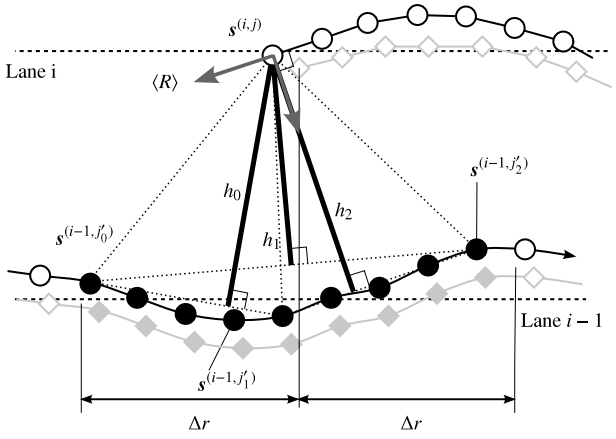

Fig. 6. Determining neighbors and triangulation sets. At the current robot position $\boldsymbol{s}^{(i,j)}$, a set $\mathcal{N}^{(i,j)}$ of neighbors (filled circles) along the previous lane is computed. The search radius Δr influences how many snapshots are selected as neighbors according to Eq. (5). Among all possible triangles with apex $\boldsymbol{s}^{(i,j)}$, three different triangles (dotted lines), referred to as triangulation sets $\mathcal{T}_k$, are used. The triangles' heights h_k are then used to compute an estimate $\hat{h}$ of the robot's current distance to the previous lane.

traveled distance along the current lane or for deriving the robot's full position. However, because this estimate relied not on external sensor measurements, we expect it to be rather imprecise and we would rely on the robot's odometry instead.

5.2. Lane changes

At the end of each lane, the robot performs a lane change. For the lane change, the robot turns by 90°, moves forward for the inter-lane distance Δl, turns again by 90°, and starts a new lane. The rotation of the robot is controlled by the visual compass method proposed by [115]. During the lane change, no place nodes are added to the topological map, and the robot solely relies on its odometry without using any visual navigation method. In order to achieve optimal coverage, the inter-lane distance Δl should be chosen as the width of the robot's cleaning unit.

5.3. Lane-distance estimator

Along the second and all subsequent lanes (lane counter $i \geq 1$), the robot not only adds place nodes $\mathcal{P}^{(i,j)}$ to the map, but also computes the distance estimate $\hat{h}$ to the previous lane whenever a new node is added to the map. The processing steps required for this purpose will be described in the following.

5.3.1. Selection of neighbors

After adding the current place node $\mathcal{P}^{(i,j)}$ to the map, its direct neighbors taken along the previous lane $i-1$ (Fig. 6, filled circles) are determined. These include the set

$$\mathcal{N}^{(i,j)} = \left\{ \mathcal{P}^{(i-1,j')} \mid \text{abs}\left(\hat{x}_{\text{odo}}^{(i,j)} - \hat{x}_{\text{odo}}^{(i-1,j')}\right) < \Delta r \right\}, \tag{5}$$

i.e. the place nodes $\mathcal{P}^{(i-1,j')}$ taken along the previous lane for which the distance between $\hat{x}_{\text{odo}}^{(i,j)}$ and $\hat{x}_{\text{odo}}^{(i-1,j')}$ is smaller than Δr. For sake of simplicity, we only consider snapshots stored along the previous lane for the selection of neighbors. Since we assume that the robot initially moves into the positive x-direction of the world coordinate system $\langle W \rangle$ (Section 4.2), the selection relies solely on the x-components of the position estimates $\hat{\boldsymbol{s}}$ obtained by the robot's odometry. Due to being only a coarse search for neighboring snapshots, the selection is the only part of the proposed algorithm relying on absolute position estimates obtained by the robot's odometry (light-gray diamonds). The choice of the search radius Δr influences how many snapshots are selected as neighbors and by this means also the accuracy of the estimate $\hat{h}_k$ of the inter-lane distance computed in the subsequent processing steps (Sections 5.3.2–5.3.3). If Δr is chosen too small, the snapshots considered along the previous lane for taking the bearing are too close together, and the quality of the resulting home vectors can be decreased due to local properties of the environment. Otherwise, if Δr is chosen too large, the estimate $\hat{h}_k$ becomes more sensitive to small deviations from the true home vector due to the tangents being involved in Eq. (23). According to our experience, good results are obtained if the resulting triangles are approximately right-angled.

5.3.2. Triangulation sets

Based on the set $\mathcal{N}^{(i,j)}$ of neighbors, a subset of place nodes is selected, which will in the following step be used to compute the robot's current distance to the previous lane. Among all neighboring place nodes in $\mathcal{N}^{(i,j)}$, we use the outermost nodes $\mathcal{P}^{(i-1,j'_0)}$ and $\mathcal{P}^{(i-1,j'_2)}$ as well as an intermediate place node $\mathcal{P}^{(i-1,j'_1)}$ with

$$j'_0 = \arg\min_{j'} \left\{ \hat{x}_{\text{odo}}^{(i-1,j')} \in \mathcal{N}^{(i,j)} \right\}, \tag{6}$$

$$j'_2 = \arg\max_{j'} \left\{ \hat{x}_{\text{odo}}^{(i-1,j')} \in \mathcal{N}^{(i,j)} \right\}, \quad \text{and} \tag{7}$$

$$j'_1 = \text{floor}\left(\frac{j'_0 + j'_2}{2}\right). \tag{8}$$

The snapshots $\boldsymbol{S}^{(i-1,j'_0)}$, $\boldsymbol{S}^{(i-1,j'_1)}$, and $\boldsymbol{S}^{(i-1,j'_2)}$ associated with these place nodes are in subsequent steps used to take the bearing from the current robot position $\boldsymbol{s}^{(i,j)}$ to each of the former robot positions $\boldsymbol{s}^{(i-1,j'_k)}$, at which these snapshots were acquired. For this purpose, three homing angles $\hat{\alpha}_k$ and changes of the orientation $\hat{\psi}_k$ are computed by local visual homing:

$$(\hat{\alpha}_k, \hat{\psi}_k)^\top = \text{lvh}\left(\boldsymbol{S}^{(i,j)}, \boldsymbol{S}^{(i-1,j'_k)}\right) \quad \text{with } k \in \{0, 1, 2\}. \tag{9}$$

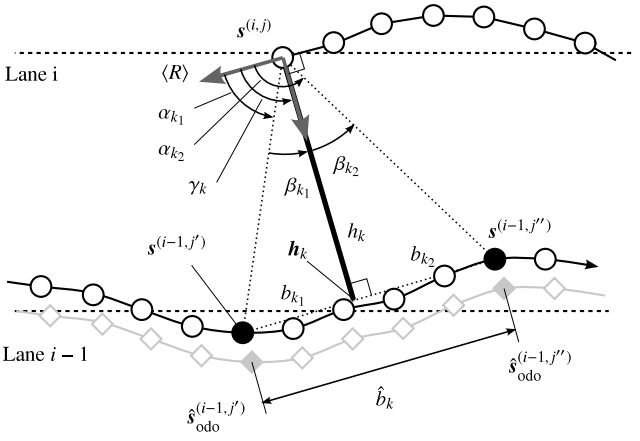

Fig. 7. Visual triangulation used to estimate the robot's current distance to the previous lane. An estimate $\hat{h}_k$ of the triangle's height can be obtained because the two angles α_{k_1} and α_{k_2} can be determined by visual homing, and because the base length $b_k = b_{k_1} + b_{k_2}$ can be approximated by the relative distance $\hat{b}$ based on the robot's odometry. For details please refer to Section 5.3.

Since the results of local visual homing have to be considered as a noisy and potentially error-prone measurement rather than an exact computation [4], we refer to them as estimates $\hat{\alpha}_k$ and $\hat{\psi}_k$.

Together with the current place node $\mathcal{P}^{(i,j)}$, the place nodes $\mathcal{P}^{(i-1,j'_0)}$, $\mathcal{P}^{(i-1,j'_1)}$, and $\mathcal{P}^{(i-1,j'_2)}$ are combined to three triangulation sets (Fig. 6, dotted lines)

$$\mathcal{T}_0 = \left\{ \mathcal{P}^{(i,j)}, \mathcal{P}^{(i-1,j'_0)}, \mathcal{P}^{(i-1,j'_1)} \right\}, \tag{10}$$

$$\mathcal{T}_1 = \left\{ \mathcal{P}^{(i,j)}, \mathcal{P}^{(i-1,j'_0)}, \mathcal{P}^{(i-1,j'_2)} \right\}, \quad \text{and} \tag{11}$$

$$\mathcal{T}_2 = \left\{ \mathcal{P}^{(i,j)}, \mathcal{P}^{(i-1,j'_1)}, \mathcal{P}^{(i-1,j'_2)} \right\}. \tag{12}$$

Each triangulation set will in the following step be used to compute an estimate $\hat{h}_k$ (with $k \in \{0, 1, 2\}$) of the robot's distance h to the previous lane (thick black lines). In case the robot's true trajectory deviates from the ideal trajectory with its parallel and straight lanes (dashed lines), different triangulation sets will also result in different estimates $\hat{h}_k$. In order to improve the estimate $\hat{h}$ of the robot's true distance h to the previous lane, the estimates $\hat{h}_k$ of several triangles will be computed and fused. Using three triangulation sets has proven to be a good trade-off between computational complexity and estimation accuracy because experiments have shown that further increasing the number of considered triangles could not improve the performance of the algorithm. Nevertheless, further triangulation sets could be determined by using a similar scheme for more than three triangles.

5.3.3. Distance estimation

For each triangulation set $\mathcal{T}_k$ ($k \in \{0, 1, 2\}$) determined in the previous step, an estimate $\hat{h}_k$ of the robot's true distance h to the previous lane is computed. The following derivation relies on the fact that a triangle is completely specified by two angles and the length of one of its sides (Fig. 7). In our case, we use the angles

$$\beta_{k_1} = \measuredangle\left(\boldsymbol{s}^{(i-1,j')}, \boldsymbol{s}^{(i,j)}, \boldsymbol{h}_k\right) \quad \text{and} \tag{13}$$

$$\beta_{k_2} = \measuredangle\left(\boldsymbol{h}_k, \boldsymbol{s}^{(i,j)}, \boldsymbol{s}^{(i-1,j'')}\right), \tag{14}$$

where $\boldsymbol{h}_k$ is the foot of the triangle's height. The base is the line segment bounded by the snapshot positions $\boldsymbol{s}^{(i-1,j')}$ and $\boldsymbol{s}^{(i-1,j'')}$. As side length, we rely on the triangle's base length b_k. With these quantities, the triangle's height h_k can be obtained by resolving the relation

$$b_k = b_{k_1} + b_{k_2} = h_k(\tan\beta_{k_1} + \tan\beta_{k_2}) \tag{15}$$

for h_k:

$$h_k = \frac{b_k}{\tan\beta_{k_1} + \tan\beta_{k_2}}. \tag{16}$$

However, none of these quantities are known or can be derived from the available information, and thus further assumptions are required for approximating them.

As angular information, the homing angles α_{k_1} and α_{k_2} between the robot's current heading and the snapshot positions $\boldsymbol{s}^{(i-1,j')}$ and $\boldsymbol{s}^{(i-1,j'')}$ can be obtained by taking the bearing from the robot's current position $\boldsymbol{s}^{(i,j)}$ to the triangle's base points $\boldsymbol{s}^{(i-1,j')}$ and $\boldsymbol{s}^{(i-1,j'')}$ (Eq. (9)). The homing angles α_{k_1} and α_{k_2} are related to β_{k_1} and β_{k_2} by the unknown angle γ_k between the robot's heading and the triangle's height h_k:

$$\beta_{k_1} = \gamma_k - \alpha_{k_1} \quad \text{and} \tag{17}$$

$$\beta_{k_2} = \alpha_{k_2} - \gamma_k. \tag{18}$$

By assuming $\gamma_k = \hat{\gamma} \in \{-90°, 90°\}$ (with the sign depending on the robot's current direction of travel and the direction of meandering) and by considering that taking the bearing only provides estimates $\hat{\alpha}_{k_1}$ and $\hat{\alpha}_{k_2}$ of the exact homing angles α_{k_1} and α_{k_2}, we can approximate β_{k_1} and β_{k_2} by

$$\hat{\beta}_{k_1} = \hat{\gamma} - \hat{\alpha}_{k_1} \quad \text{and} \tag{19}$$

$$\hat{\beta}_{k_2} = \hat{\alpha}_{k_2} - \hat{\gamma}. \tag{20}$$

The assumption $\hat{\gamma} = \pm 90°$ holds exactly if and only if the robot is moving parallel to its previous lane (Figs. 7 and 6, triangulation set $\mathcal{T}_2$). In other cases, the assumption is violated (Fig. 6, triangulation sets $\mathcal{T}_0$ and $\mathcal{T}_1$).

Furthermore, the true base length b_k is unknown because the positions of image acquisitions $\boldsymbol{s}^{(i-1,j')}$ and $\boldsymbol{s}^{(i-1,j'')}$ are unknown. As the robot's odometry obtains good estimates for distances between snapshots taken along a lane, we approximate b_k by the Euclidean distance

$$\hat{b}_k = \left\| \hat{\boldsymbol{s}}^{(i,j')}_{\text{odo}} - \hat{\boldsymbol{s}}^{(i,j'')}_{\text{odo}} \right\| \tag{21}$$

between the odometry estimates $\hat{\boldsymbol{s}}^{(i-1,j')}_{\text{odo}}$ and $\hat{\boldsymbol{s}}^{(i-1,j'')}_{\text{odo}}$. Based on these assumptions, the estimate $\hat{h}_k$ for the robot's current distance h to the previous lane is then obtained by

$$\hat{h}_k = \frac{\hat{b}_k}{\tan\hat{\beta}_{k_1} + \tan\hat{\beta}_{k_2}} \tag{22}$$

$$= \frac{\hat{b}_k}{\tan\left(\hat{\gamma} - \hat{\alpha}_{k_1}\right) + \tan\left(\hat{\alpha}_{k_2} - \hat{\gamma}\right)}. \tag{23}$$

5.3.4. Fusion of several estimates

To fuse the estimates $\hat{h}_k$ of the robot's current distance to the previous lane, the median $\hat{h}$ over all estimates is computed:

$$\hat{h} = \underset{k}{\text{median}}\ \hat{h}_k \quad \text{with } k \in \{0, 1, 2\}. \tag{24}$$

The result is used by the control algorithm (Section 5.4) to derive a motion command keeping the robot on a course parallel to the previous lane.

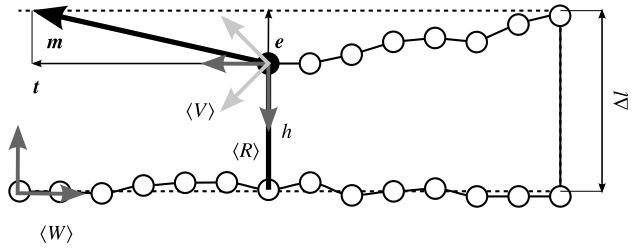

Fig. 8. Computation of the motion vector. The proposed controller computes a motion vector $\boldsymbol{m}$ which reduces the deviation $e = \|\boldsymbol{e}\|$ from the robot's current distance h to the previous lane and the desired lane distance Δl. The robot is supposed to return to the desired lane within some moving target distance $t = \|\boldsymbol{t}\|$. To determine the wheel speeds for controlling the robot, $\boldsymbol{m}$ is transformed from world coordinates $\langle W\rangle$ to the robot coordinate system $\langle R\rangle$ and further to an auxiliary coordinate system $\langle V\rangle$.

5.4. Trajectory controller

The trajectory controller uses the estimate $\hat{h}$ of the robot's current distance h to its previous lane to keep the robot at the desired distance Δl from the previous lane. To reduce the deviation from the desired lane, a motion vector $\boldsymbol{m}$ is computed, which is mapped to wheel speeds in order to control the robot. Keeping the robot's distance to the previous lane constant implies that the robot's desired lane is specified w.r.t. its previous lane.

5.4.1. Motion vector

The motion vector $\boldsymbol{m}$ is supposed to compensate the deviation

$$e = \Delta l - \hat{h}, \tag{25}$$

and is defined as the vector sum

$${}^{\langle W\rangle}\boldsymbol{m} = \boldsymbol{e} + \boldsymbol{t} = \begin{pmatrix} 0 \\ s_e \cdot e \end{pmatrix} + \begin{pmatrix} s_t \cdot t \\ 0 \end{pmatrix} \tag{26}$$

of the deviation vector $\boldsymbol{e}$ and the moving target vector $\boldsymbol{t}$ (Fig. 8). The vector $\boldsymbol{e}$ is chosen parallel to the y-axis of the world coordinate system $\langle W\rangle$ pointing always into the direction compensating the deviation e. The moving target vector $\boldsymbol{t}$ points parallel to the x-axis of $\langle W\rangle$ into the current movement direction. Its length $t = \|\boldsymbol{t}\|$ is the moving target distance within which the robot is supposed to completely reduce the deviation from its desired lane. With small choices of t, the controller is more reactive and capable of quickly reducing the deviation; larger choices of t make the controller slower but result in smoother trajectories. The multipliers s_e and s_t, both from the set $\{-1, 1\}$, adjust the directions of the vectors $\boldsymbol{e}$ and $\boldsymbol{t}$ depending on the robot's direction of travel and its direction of meandering, i.e. the directions of its initial turn after finishing the first lane.

5.4.2. Transformation to wheel speeds

In Eq. (26), the motion vector $\boldsymbol{m} = {}^{\langle W\rangle}\boldsymbol{m}$ was defined w.r.t. world coordinates $\langle W\rangle$. In order to map the motion vector $\boldsymbol{m}$ to wheel speeds correcting the robot's deviation from the desired lane distance Δl, ${}^{\langle W\rangle}\boldsymbol{m}$ has to be transformed from the world coordinate system $\langle W\rangle$ to the robot coordinate system $\langle R\rangle$ and further to an auxiliary system $\langle V\rangle$.

As the robot's position in world coordinates is unknown due to partial pose estimation, only the rotational component of the coordinate transformation from $\langle W\rangle$ to $\langle R\rangle$ can be computed. However, this is sufficient for the following processing steps, and we do not have to rely on further assumptions. To determine the rotation between world coordinate system $\langle W\rangle$ and robot coordinate system $\langle R\rangle$, an estimate of the robot's orientation $\hat{\theta}_{\text{vis}}^{(i,j)}$ is required. It is derived by angular averaging [116]

$$\hat{\theta}_{\text{vis}}^{(i,j)} = \underset{k}{\text{mean}}\ \hat{\theta}_{\text{vis},k}^{(i,j)} \tag{27}$$

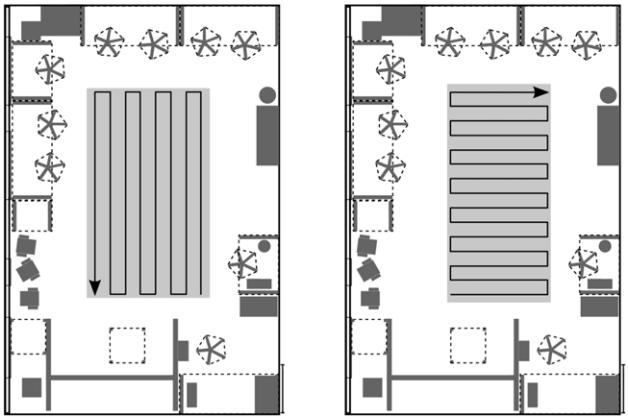

Fig. 9. Areas ideally covered by experiment 1 (left) and experiment 2 (right). Depicted are ground-level objects (e.g. legs of chairs and desks or closets; dark gray areas), obstacles above ground level (e.g. surfaces of chairs or desks; dashed lines), the ideal cleaning trajectories (black arrows), and the area ideally covered by the cleaning runs (light gray areas). The area covered by the cleaning trajectories is approximately the field of view of the visual tracking system (Section 6.4); the light-gray area was used as reference for computing the cleaning performance (Section 6.5). The measuring line at the lower right corner shows a length of 1 m.

the orientation estimates $\hat{\theta}_{\text{vis},k}^{(i,j)}$ ($k \in \{0, 1, 2\}$) obtained for each of the computed home vectors. These are obtained by the angular summation

$$\hat{\theta}_{\text{vis},k}^{(i,j)} = \psi_k + \hat{\theta}_{\text{vis}}^{(i-1,j'_k)} \tag{28}$$

of the relative change of orientation ψ_k computed in equation Eq. (9) and the vision-based orientation estimate $\hat{\theta}_{\text{vis}}^{(i-1,j'_k)}$ associated with place node $\mathcal{P}^{(i-1,j'_k)}$ taken along the previous lane. Based on the orientation estimate $\hat{\theta}_{\text{vis}}^{(i,j)}$, the vector ${}^{\langle R\rangle}\boldsymbol{m}$ is obtained by rotating ${}^{\langle W\rangle}\boldsymbol{m}$ by $-\hat{\theta}_{\text{vis}}^{(i,j)}$ around the robot's z-axis.

To transform the motion vector ${}^{\langle R\rangle}\boldsymbol{m}$ into velocities $\boldsymbol{v} = (v_L, v_R)$ for the left and right wheel of the robot, the mapping proposed by [117] is applied. Therefore, the movement vector ${}^{\langle R\rangle}\boldsymbol{m}$ is further transformed into the coordinate system $\langle V\rangle$ which is rotated by -45° around the z-axis of the robot coordinate system $\langle R\rangle$. To obtain wheel speeds, the resulting vector ${}^{\langle V\rangle}\boldsymbol{m}$ is normalized and scaled with the desired velocity v:

$$\boldsymbol{v} = \begin{pmatrix} v_L \\ v_R \end{pmatrix} = \frac{v}{\|{}^{\langle V\rangle}\boldsymbol{m}\|} \cdot {}^{\langle V\rangle}\boldsymbol{m}. \tag{29}$$

The resulting velocities v_L and v_R are passed to the robot's motion controller and kept constant until the next place node is added.

6. Experiments and setup

In order to test the proposed navigation strategy, we conducted experiments (Section 6.1) with a custom-built cleaning robot (Section 6.2). To demonstrate the capabilities of the proposed controller, the motion commands were disturbed with a strong systematic error to be compensated by the controller (Section 6.3). The experiments were recorded with a vision-based tracking system (Section 6.4), and the resulting trajectories were analyzed qualitatively and quantitatively (Section 6.5).

6.1. Experiments

We performed cleaning runs from two different start positions in our lab (Fig. 9). In experiment 1, the target trajectory consisted of eight lanes each of 4 m length; in experiment 2, it consisted of 15 lanes with a lane length of 2 m. In both experiments, the robot was

meandering to the left, and a total of 10 trials per experiment were recorded. The size and the position of the workspace within our lab were limited by the field of view of our visual tracking system, which was used to record the robot's trajectories (Section 6.4). Due to properties of the tracking system, experiments had to be conducted under constant and diffuse illumination conditions and in a static environment (see Section 6.4 for details).

The robot's motion controller was disturbed by a systematic error of 5%, which had to be compensated by the controller in order to keep the robot parallel to the previous lane (Section 6.3). We consider the specific choice to be a reasonable error because its effect is clearly visible if the robot is moving relying solely on its odometry (Fig. 12) and because – if moving under visual control – the performance is not considerably decreased in comparison to a smaller or a zero error (however, errors larger than 10% can cause the method to fail). To get an impression of the disturbances, please see also the supplemental videos (Appendix B). Half of the trials were disturbed with a systematic error causing a trajectory curved to the left; the other half was performed with an error causing a trajectory curved to the right. For data analysis (Section 6.5), the results of both types of error were pooled.

Along a cleaning lane, the robot was continuously moving with $v = 7.5$ cm/s, the inter-snapshot distance was $\Delta s = 10$ cm, the moving-target distance was $t = 45$ cm, and the radius of the neighbor search was $\Delta r = 45$ cm. As our robot (Section 6.2) is not equipped with a suction unit, we used an inter-lane distance of $\Delta l = 30$ cm, which is approximately the robot's diameter. Since we observed in simulation studies that the system achieves similar performance for different parameter combinations, we only present results for this parameter set (rather then presenting data for systematically varying the moving target distance t, the search radius r, the odometry error k, and the number of triangulation pairs).

The first lane was kept straight by following an artificial wall relying on the robot's IR range sensors. The artificial wall has a height of approximately 9 cm and is not visible for the robot because it is completely below the horizon of its omnidirectional vision system (see also Fig. 3 for omnidirectional images acquired during our experiments). An artificial wall was used (i) because our lab is too cluttered and does not offer a free wall sufficiently long for our experiments (Fig. 9) and (ii) because the experiments had to be conducted in an area which is observable by the used tracking system (Section 6.4).

For all experiments, min-warping with compass acceleration was used as homing method (Section 4.1.1, for details [4]). In the search process, the home direction α and the orientation change ψ were varied in discrete steps of 5°. The compass acceleration restricted the search space to 30% of the possible orientation changes, image columns were compared by the Euclidean distance, and we used 9 different scale planes in the range [0.2, 1.8].

6.2. Robot

Experiments were conducted with a custom-built differential-drive robot (Fig. 10). With a diameter of 33 cm and a height of 12 cm, its dimensions are comparable to those of commercially available cleaning robots for domestic usage (Section 3.1). In contrast to such robots, our robot is not equipped with a cleaning unit. For obstacle detection and wall following, the robot is equipped with IR distance sensors (Sharp GP2D12 and GP2D120). As an omnidirectional vision sensor, the robot relies on a camera (IDS Imaging UI-2220SE-M) with a panoramic annular lens (PAL, Tateyama S25G2817-27C, patent: [118]). Since the PAL is much more compact than standard catadioptric sensors [3], it sticks out of the housing by only 3 cm. The panoramic camera

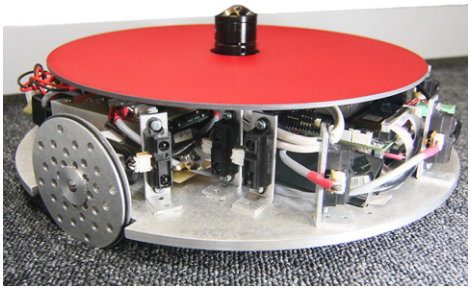

Fig. 10. Custom-built cleaning robot with panoramic annular lens (PAL) as omnidirectional vision sensor. For recording the robot's trajectory, the robot is equipped with a red disk which is used as tracking target by the visual tracking system (Section 6.4). Figure best viewed in color.

images were unfolded to cylindrical images by applying a custom-developed model-free mapping method. Unfolding included a histogram equalization followed by low-pass filtering (binomial filter with kernel size 5). The resulting images were sized 360×48 pixels and covered a vertical field of view from 0° to 38° above the horizon (see Fig. 3 for two example images). Image unfolding was done on the robot's on-board PC (Intel Atom Z530), and the unfolded images were transmitted to an external host computer (Intel Core i7 920XM) via wireless network connection. There, the computations required for the proposed navigation strategy (namely computing three home vectors, estimating the inter-lane distance, and deriving a motion command) were executed, and the resulting motion command was sent back to the robot. Among these processing steps, computing a single home vector requires approximately 70 ms [4] allowing for real-time control of the robot; in comparison to local visual homing, the other operations of the visual controller are negligible. The off-board computations and the client–server architecture were necessary because the proposed method was implemented relying on the prototyping software framework maintained by our group. Nevertheless, because our method was designed keeping in mind the limited computational power of autonomous cleaning robots, we expect it to be executable in real-time on a robot's on-board computer if the current implementation is optimized for this computer hardware. Although we are aware of its importance for a fully autonomous cleaning robot, such an implementation is left for future work.

6.3. Systematic error

In order to verify that the visual trajectory controller (Section 5.4) can keep the robot on the desired trajectory, the robot's motion controller is in all experiments biased by a strong systematic error. The error simulates differences in the robot's wheel diameters causing the robot to move on a circular path, which has to be compensated by the proposed navigation strategy. In case of differences in the wheel diameters, the robot's odometry cannot measure that the robot is moving on a curved trajectory. Thus, the resulting position estimates $\hat{\boldsymbol{p}}_{\text{rob}}$ lie on a straight line.

Adding a systematic error to the robot's desired wheel speeds $\boldsymbol{v} = (v_L, v_R)^\top$ by disturbing the velocities with a constant disturbance factor k proportional to the systematic error

$$\tilde{\boldsymbol{v}} = \begin{pmatrix} \tilde{v}_{\text{L}} \\ \tilde{v}_{\text{R}} \end{pmatrix} = \begin{pmatrix} \left(1 + \frac{k}{2}\right) v_{\text{L}} \\ \left(1 - \frac{k}{2}\right) v_{\text{R}} \end{pmatrix} \tag{30}$$

causes the robot to move on a circular path. With this equation, we assure that the error symmetrically influences both wheels without influencing the overall robot velocity. However, the robot's on-board odometry can measure that the robot moves on a curved

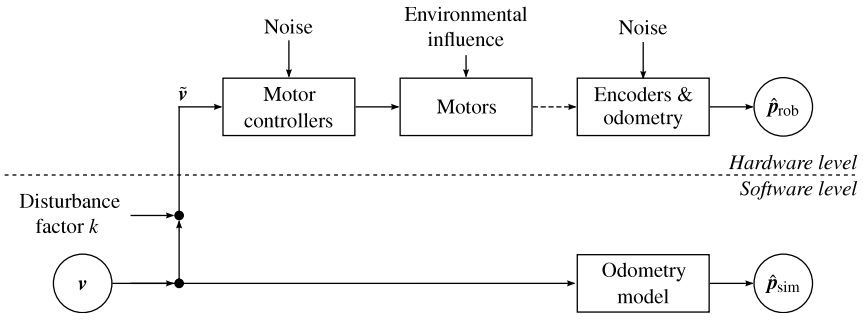

Fig. 11. Artificial disturbance of the robot's motion controller. The velocities $\boldsymbol{v}$ passed to the robot's motion controller are disturbed by a systematical error k resulting in biased velocities $\tilde{\boldsymbol{v}}$ causing the robot to move on a circular path. As the robot's on-board odometry can measure this disturbance, the position estimate $\hat{\boldsymbol{p}}_{rob}$ should not be used for the trajectory controller. Instead, the robot's on-board odometry is bypassed by a secondary odometry computing the robot's position $\hat{\boldsymbol{p}}_{sim}$ based on unbiased velocities $\boldsymbol{v}$.

path. Thus, relying on the odometry estimates $\hat{p}_{rob}$ would influence the controller, e.g. when the base length $\hat{b}$ of the considered triangle is estimated (Eq. (21)).

In order to eliminate these effects, we bypass the robot's on-board odometry (Fig. 11). For this purpose, a position estimate $\hat{\boldsymbol{p}}_{sim}$ is computed by applying a standard odometry model of differential-drive robots (e.g. [7], Ch. 5.2.4). As the computation of this estimate is based on the unbiased wheel speeds $\boldsymbol{v}$, the simulated odometry cannot measure the influences of the disturbance k. Whenever the odometry is read out, the request is not directed to the robot's on-board odometry, but the simulated odometry is updated, and its position estimate $\hat{\boldsymbol{p}}_{sim}$ is returned. Since the simulated odometry does not consider environmental influences (such as motor noise or wheel slippage), and since the simulated odometry assumes instantaneous changes of the wheel speeds, the position estimates of the simulated and the robot's on-board odometry without systematic error ($k = 0$) differ. However, we think that these differences are negligible.

Fig. 12 visualizes the effects of a 5% systematic error (i.e. $k = \pm 0.05$) as used for the experiments (Section 6.1). In the shown case, the error was $k = -0.05$, thus simulating that the robot's right wheel has a larger diameter than the left one. The robot was driven along meandering lanes of length 1 m without visual correction. As the robot's true trajectory (continuous black line) is curved whereas the trajectory obtained from the robot's odometry (dashed line) is parallel and meandering, the approach described in this section simulates differences in the diameter of the robot's wheels. In case the experiments reveal that the robot is guided along parallel lanes, these results therefore have to be due to the visual trajectory controller compensating for the strong systematic error introduced by k. Please note that these errors were only introduced for testing the trajectory controller; future work focusing on more elaborated cleaning strategies (Section 9) will rely on the robot's on-board odometry.

6.4. Tracking system

For external data analysis of the performed cleaning runs (Section 6.1), a custom-built visual tracking system was used. In order to track the robot, a red disk is mounted on top of the robot (Fig. 10), which is not visible in the robot's panoramic image. The robot's workspace is observed by two cameras mounted statically on the ceiling of the lab. For each camera, the red disk is detected in the camera image and tracked over time using the mean-shift algorithm [119]. The marker's center of gravity in the camera image is used to compute an estimate of the robot's world position by direct linear transformation [45]. To fuse the estimates of both cameras, a weighted average depending on the agent's distance to the camera is used. By this means, the robot's position but not its orientation can be determined. The resulting tracking accuracy is approximately 1 cm.

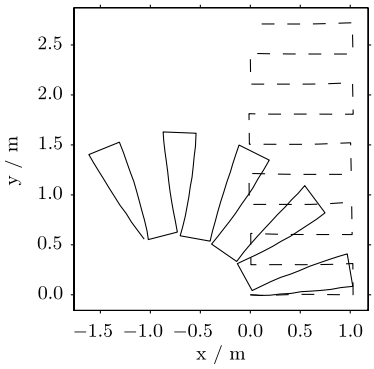

Fig. 12. Trajectory resulting from driving the robot along meandering lanes solely based on its disturbed odometry (i.e. without visual correction). The continuous black line depicts the robot's true course as recorded by the visual tracking system; the dashed line depicts the position estimates obtained by the disturbed odometry.

In its current implementation, the system cannot cope with certain illumination conditions and with occlusions of the tracking target. The first aspect includes transitions from sunlit areas to shadow areas thus requiring experiments to be conducted under diffuse and constant illumination. The second aspect subsumes occlusions caused by people moving in the lab or by obstacles if the robot moves too close to or underneath them (Fig. 9). Thus, experiments had to be conducted in a static environment and in the center of the lab's freespace.

6.5. Data evaluation

The trajectories recorded with the tracking system were analyzed qualitatively and quantitatively. For qualitative evaluation, the resulting trajectories were plotted (Figs. 13 and 14). For quantitative data analysis, piecewise polynomials of third order (Matlab function `spline`) were fitted to the recorded robot positions along a lane in order to interpolate between snapshot positions and to estimate the robot's orientation, which cannot be measured by our visual tracking system. Based on this, we computed measures for the inter-lane distances and the resulting cleaning performance. These measures will be described in the following.

6.5.1. Inter-lane distances

As the proposed trajectory controller (Section 5.4) is supposed to keep the robot at the desired lane distance Δl from the previous lane, we computed for every snapshot position the robot's true distance to the previous lane. For this purpose, we minimized the spatial distance between the considered snapshot position

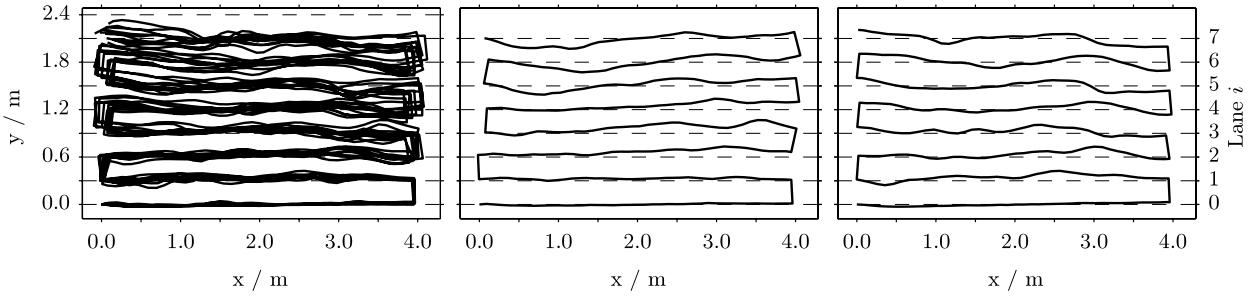

Fig. 13. Trajectories of experiment 1. Left: trajectories pooled over all ten trials. Middle/Right: two single trajectories with systematic error to the left and right, respectively.

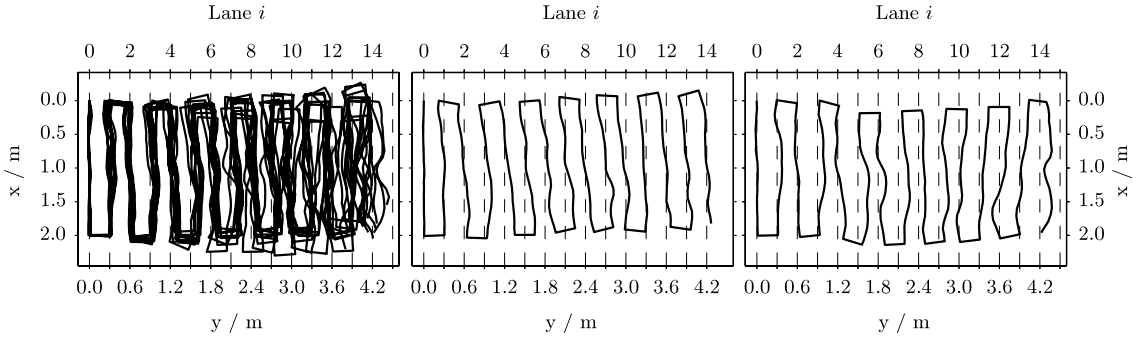

Fig. 14. Trajectories of experiment 2. Left: trajectories pooled over all ten trials. Middle/Right: two single trajectories with systematic error to the left and right, respectively.

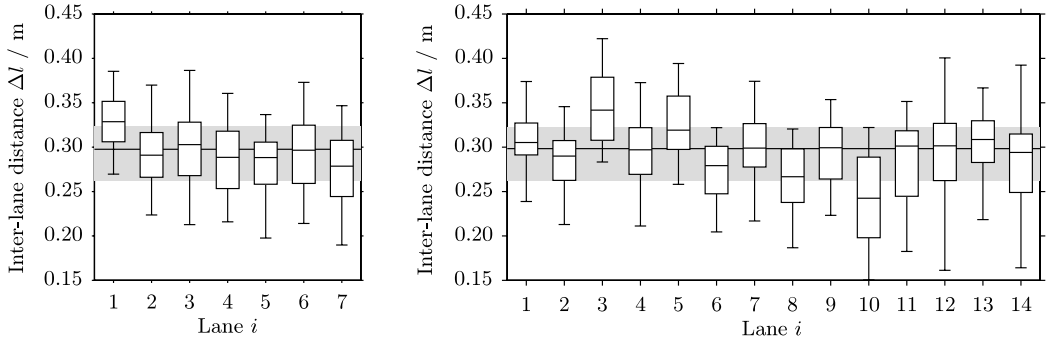

Fig. 15. True inter-lane distances of experiments 1 (left) and 2 (right). The boxes visualize for each lane i the range $I^{(i)}_{0.50}$ containing 50% of the obtained distance values; the median $P^{(i)}_{0.50}$ is depicted by the small horizontal line dividing each box. The whiskers span 90% of the inter-lane distance values ranging from the $P^{(i)}_{0.05}$ to the $P^{(i)}_{0.95}$ percentiles. The gray area in the figure's background marks the range $\bar{I}_{0.50}$ of inter-lane distances containing 50% of the values for pooling over all lanes; the overall median $\bar{P}_{0.50}$ is depicted by the horizontal line.

along the current lane and an arbitrary point along the previous lane approximated by piecewise polynomials. Thus, the search was not restricted to discrete positions of snapshots taken along the previous lane. The resulting inter-lane distances were pooled over all trials of an experiment and analyzed lane-by-lane as well as pooled over all lanes. As performance measures, percentiles and differences between percentiles of the resulting distance distributions were used. Results are shown in Fig. 15 and described in Section 7.2.

6.5.2. Cleaning performance

To assess the cleaning performance of the proposed navigation strategy, we analyzed the area covered by a simulated suction unit sized 30 cm × 5 cm, which was moved in small steps along the robot's trajectory. By this means, the area A_1 covered exactly once, the overlap between consecutive lanes A_2, and the uncovered area A_0 were computed. The percentages are measured w.r.t. the area covered by an ideal cleaning run (Fig. 9, light-gray areas). Hence, areas outside the ideal area are not considered for the evaluation. Lane changes are excluded from the evaluation assuming the robot's cleaning unit to be switched off. All percentages of A_0, A_1, and A_2 sum up to 100%; a perfect trial would yield $A_1 = 100.0\%$ and $A_0 = A_2 = 0.0\%$. Similar measures were also used in related work [88,89].

7. Results

The results obtained by experiments 1 and 2 were analyzed qualitatively (Section 7.1) and quantitatively (Sections 7.2 and 7.3). Thereafter, the results will be discussed in Section 7.4.

7.1. Qualitative analysis

Figs. 13 and 14 visualize the trajectories obtained for experiments 1 and 2, respectively. The left subfigures show all ten trials in one plot in order to analyze the method's repeatability. The middle and the right subfigures show a single trajectory with systematical error to the left (middle, $k = -0.05$) and to the right (right, $k = 0.05$).

Regarding the repeatability, the trajectories of both experiments are close together with only little variability of up to lane index $i = 3$. For $i > 3$, the repeatability of the lanes decreases especially at the beginning and the end of the lanes. In the middle of each lane (experiment 1: $1\,\text{m} < x < 3\,\text{m}$, experiment 2: $0.5\,\text{m} < x < 1.5\,\text{m}$), the lanes are still close together. Since the robot does not move straight and parallel to the x-axis, but rather oscillates around its desired lane, its orientation at the end of the lane is not parallel to the x-axis. After changing lanes, this deviation can be increased due to inaccuracies during the robot's rotation. At the beginning, the robot has to compensate for the deviation in order to return to a course parallel to the previous lane. At the end of the lane, the robot follows the deviations of the previous lane. Up to $i = 5$ and $i = 8$ for experiments 1 and 2, respectively, the trajectories of different lanes are clearly separable from each other; for larger i, trajectories of different lanes overlap. Thus, the repeatability decreases with increasing number of lanes i.

Analyzing single trajectories reveals that the robot moves on slightly curved lanes which are locally parallel to each other. As the robot keeps the distance to its previous lane constant, it follows the deviations which occurred along the previous lane. One would expect that this leads to controller errors accumulating over time because additional controller errors occur while moving parallel to the previous lane. However, the obtained trajectories do not reveal such controller errors accumulating from lane to lane. Only the last lane of each experiment is usually more curved than the other lanes. During all experiments, we have never observed that the current lane touches or even crosses the previous lane.

7.2. Inter-lane distances

The results for analyzing the robot's distance from the previous lane are summarized both in Fig. 15 and in Tables A.1 and A.2. For each lane i, the boxes mark the ranges $I^{(i)}_{0.50}$ from the lower quartile $P^{(i)}_{0.25}$ to the upper quartile $P^{(i)}_{0.75}$ containing 50% of the obtained distance values. The median inter-lane distance $P^{(i)}_{0.50}$ for each lane is depicted by the short horizontal lines dividing the boxes. The whiskers span 90% of the distance values ranging from $P^{(i)}_{0.05}$ to $P^{(i)}_{0.95}$. We refer to this interval as $I^{(i)}_{0.90}$. In the figure's background, the median inter-lane distances $\bar{P}_{0.50}$ for pooling over all lanes is visualized by the horizontal line. The gray area marks the range $\bar{I}_{0.50}$ between the lower quartile $\bar{P}_{0.25}$ and the upper quartile $\bar{P}_{0.75}$ for pooling over all lanes.

For both experiments, the median lane distance for pooling over all lanes and trials is $\bar{P}_{0.50} = 29.8$ cm. The lower and upper quartiles are $\bar{P}_{0.25} = 26.3$ cm and $\bar{P}_{0.75} = 32.4$ cm for experiment 1, and $\bar{P}_{0.25} = 26.2$ cm and $\bar{P}_{0.75} = 32.2$ cm for experiment 2. Thus, both experiments achieve almost identical results. With few exceptions, the median values and inter-quartile distances obtained for analyzing single lanes are similar to the overall values. Furthermore, the obtained results seem to be independent of the lane counter i (e.g. they do not increase with increasing lane counter).

For experiment 1 (Fig. 15, left), the minimum and maximum median inter-lane distances are $P^{(7)}_{0.50} = 27.9$ cm and $P^{(1)}_{0.50} = 32.9$ cm, respectively. The boxes span inter-quartile distances ranging from $I^{(1)}_{0.50} = 4.5$ cm to $I^{(6)}_{0.50} = 6.5$ cm; the whiskers span inter-percentile distances between $I^{(1)}_{0.90} = 11.5$ cm and $I^{(3)}_{0.90} = 17.3$ cm. For both measures, the range from the lower percentile to the median is larger than the range from the median to the upper percentile. The measures for even lanes (robot is moving in positive x-direction of $\langle W \rangle$, $i = 2, 4, \ldots$) and for odd lanes (moving in negative x-direction, $i = 1, 3, \ldots$) do not differ.

The median inter-lane distances obtained for the 14 lanes of experiment 2 (Fig. 15, right) vary between $P^{(10)}_{0.50} = 24.2$ cm and $P^{(3)}_{0.50} = 34.1$ cm. For the first 11 lanes, the median inter-lane distances of the even and odd lanes differ by several centimeters with the values of the odd lanes being larger. Most inter-quartile distances $I_{0.50}$ are approximately 6 cm, with $I^{(1)}_{0.50} = 3.6$ cm and $I^{(10)}_{0.50} = 9.1$ cm being the minimum and maximum inter-quartile distances. Most values of $I_{0.90}$ are approximately 15 cm; minimum and maximum are $I^{(12)}_{0.90} = 11.7$ cm and $I^{(6)}_{0.90} = 23.9$ cm, respectively.

7.3. Cleaning performance

For analyzing the cleaning performance, we computed for each of the 20 trials the percentages of the uncovered area A_0, of the area A_1 covered exactly once, and of the area A_2 covered twice due to overlap between consecutive lanes. As reference area, we used the area covered by ideal cleaning runs as depicted in Fig. 9; parts of the robot's trajectory outside these areas have not been considered. The complete data is given in Tables A.3 and A.4.

For experiment 1, the performance varies between $A_0 = 9.4\%$, $A_1 = 81.6\%$, and $A_2 = 9.0\%$ for the worst and $A_0 = 6.4\%$, $A_1 = 87.7\%$, and $A_2 = 5.9\%$ for the best trial with an average performance of $\bar{A}_0 = 7.9\%$, $\bar{A}_1 = 85.4\%$, and $\bar{A}_2 = 6.7\%$. The average performance of experiment 2 is slightly worse: we obtained a performance of $\bar{A}_0 = 10.3\%$, $\bar{A}_1 = 81.5\%$, and $\bar{A}_2 = 8.2\%$. This experiment also exhibits a larger variability between trials because the performance ranges from $A_0 = 17.2\%$, $A_1 = 71.7\%$, and $A_2 = 11.1\%$ to $A_0 = 5.9\%$, $A_1 = 89.3\%$, and $A_2 = 4.8\%$. Fig. 16 shows for each experiment the trajectory achieving the performance which is most similar to the average performance. The figure reveals that the uncovered area is mostly due to gaps between consecutive lanes because of the robot's distance to the previous lane being too large. Both the uncleaned area and the overlap between lanes do not increase with increasing lane number i. Rather, the method seems to achieve equal performance over all lanes.

7.4. Discussion and conclusions

The results presented in Sections 7.1–7.3 show that the proposed trajectory controller is capable of guiding the robot along parallel lanes while achieving a good coverage of the robot's workspace. Although several of the assumptions (Eqs. (4) and (19)–(21)) made for the controller's derivation are not completely satisfied, the obtained results reveal that the assumptions were chosen reasonably and that the controller exhibits a certain robustness against their violations. In the remainder of this section, aspects related to partial pose estimation, dense topological maps, image disturbances, and further cleaning strategies will be discussed in more detail.

7.4.1. Partial pose estimation

Keeping the robot's distance to its previous lane constant means that the robot's desired trajectory directly depends on its previous lane. This approach is potentially prone to controller errors accumulating from lane to lane: in case the robot does not exactly follow its current desired trajectory but oscillates around it, new controller errors occur, and the resulting trajectory will influence the desired trajectory of the following lane. Nevertheless, such accumulating errors did not occur during our experiments (Figs. 13–16). We think that this is due to the following two reasons: (i) the selection of different triangulation sets and the fusion of the estimates provided by each of these sets and (ii) the moving target distance t (Eq. (26)). Both aspects seem to have a dampening influence on the robot's trajectory causing the robot to follow the previous lane smoothly. From these results we conclude that our approach relying on partial pose estimation is sufficient to guide the robot along parallel and meandering lanes. Thus,

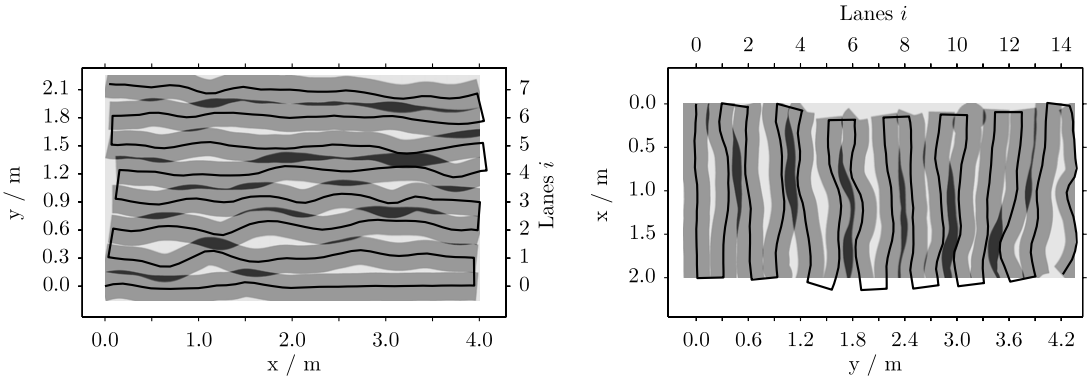

Fig. 16. Analysis of the cleaning performance for experiments 1 (left) and 2 (right). For each experiment, one trial with a performance close to the experiment's average performance is shown. The graphs depict the robot's trajectory (black line) together with the uncovered area A_0 (light gray), the area A_1 covered once (gray) and the area A_2 covered twice (dark gray).

a computationally more demanding full pose estimation is not required to solve the robot's task.

Accumulating controller errors could be reduced (i) by correcting the estimates attached to the corresponding place nodes at a later point in time, (ii) by selecting neighbors not along the previous lane but along its pre-predecessor, or (iii) by making the desired lane independent of the previous lane. Subsequent improvements of initial estimates are usually performed by SLAM methods (Sections 2, 3.2.2 and 3.2.3) and are essential for achieving globally consistent maps. However, they are also the computationally most demanding parts of SLAM algorithms. For our future work, we hope to avoid this step by completely covering the robot's workspace by several locally consistent cleaning segments rather than building a single globally consistent map.

Making the desired lane independent of the previous lane requires to specify the robot's desired lane with respect to world coordinates. However, this would require to estimate the robot's full pose $\hat{\boldsymbol{s}}^{(i,j)} = (\hat{x}^{(i,j)}, \hat{y}^{(i,j)}, \hat{\theta}^{(i,j)})$, which could be determined by intersecting the rays defined by the triangle's base points $\boldsymbol{s}^{(i-1,j')}$ and $\boldsymbol{s}^{(i-1,j'')}$ and by the corresponding homing angles α_{k_1} and α_{k_2} (Fig. 7). Even though computing an estimate of the robot's full pose would not require much overhead, we prefer the partial pose estimation. We are of the opinion that directly using the full position estimates computed by this means could not considerably improve the accuracy of our method. We would only expect improvements for fusing several estimates by a Bayesian filtering framework (such as the Kalman or the particle filter; textbook: [7]) which could use the sketched approach for its update step. However, such filtering frameworks are computationally more demanding than the proposed approach. Beyond that, even such frameworks could not completely prevent error accumulation from lane to lane. This problem is inherent to all methods using former robot positions as landmarks. The correction step of the filter would still refer to snapshot images stored along the previous lane. Thus, position errors and the corresponding uncertainties on the previous lane would affect the estimates on the current lane.

7.4.2. Dense topological maps

The results have also shown that the proposed spatial representation is useful for completely covering segments of the robot's entire workspace. By adding new snapshots at regular distances, a sufficiently large number of snapshots is available for correcting the robot's position while still preserving the advantages of topological maps (Section 4.2). We consider the choice of $\Delta s = 10$ cm to be sufficient for our application and do not expect that smaller choices of Δs could improve the performance of the proposed controller. This is due to the observation (unpublished data) that home vectors computed from a current view and a series of neighboring snapshots taken along the previous lane have correlated errors with the strength of the correlation depending on the spatial distance between the snapshots. Furthermore, smaller choices of Δs would also increase the memory requirements of the resulting topological map. Although not discussed in this paper, we expect that more complex cleaning strategies for completely covering complex-shaped workspaces can be developed based on dense topological maps.

7.4.3. Influence of image disturbances

Like any appearance-based method, local visual homing and hence also the proposed trajectory controller are prone to image disturbances as, among others, caused by dynamic scene changes, changes of the illumination, or a tilt of the robot if rolling over a cable or a carpet border. In its current implementation, the controller is not capable of detecting or reacting to such situations, even though a certain, but probably limited, robustness should arise from fusing several estimates (Section 5.3.4). We are of the opinion that – rather than extending the control algorithm – achieving robustness against image disturbances should be part of the underlying homing method therefore assuring correct home vectors to be passed to the controller. If the underlying homing method is not robust against such image disturbances, the computed home vectors will be erroneous and the estimated inter-lane distance will deviate. The same holds if the robot is operating in an environment with sparse visible image content making it difficult to establish column matches by the used warping method (Section 4.1). For such special cases, navigation strategies for completely covering complex-shaped workspaces (Section 9) should include backup strategies such as random walk.

The min-warping method (Section 4.1.1) used in this paper is capable of computing accurate home vectors over a wide range of different environments (as shown in [4] and observed in many unpublished real-robot experiments) and over a certain range of image disturbances (as observed but not yet published for illumination changes and dynamic scene changes). To this end, we expect the proposed method to achieve similar results than the ones presented here in different environments and under different operating conditions.

7.4.4. Implications for further cleaning strategies

In case only a single, rectangular segment is cleaned, the uncleaned area is due to gaps between consecutive cleaning lanes. Thus, the covered area could be further increased by reducing the desired inter-lane distance Δl. However, this would be at the cost of increasing the proportion of repeated coverage. The results presented in Section 7 show that relatively large areas can be covered by the proposed method. As real apartments are usually more

cluttered, we expect the maximum area which can be covered by a single cleaning segment to be much smaller. For this reason, good strategies for combining several cleaning segments and for keeping the first lane of these segments straight have to be developed. Based on these strategies, we hope to be able to circumvent the computationally demanding steps of subsequent position corrections by building a hierarchical representation of space with a purely topological representation on top of several locally consistent submaps each corresponding to a single cleaning segment.

Even if real apartments require the decomposition into several smaller cleaning segments, we think that a decomposition into rectangular segments of parallel lanes is the best choice for most of the cases. For special cases which cannot be covered efficiently by parallel and meandering lanes, alternative cleaning strategies or backup strategies could be provided by a more elaborated framework of cleaning strategies (Section 9).

8. Summary and conclusions

In this paper we have presented a mostly vision-based navigation strategy which guides the robot along meandering and parallel lanes in order to completely cover a rectangular segment of the robot's entire workspace. The proposed method will be used as essential building block for more elaborated cleaning strategies capable of completely covering complex-shaped workspaces (Section 9). While navigating, the robot builds up a dense topological map (Section 4.2) of its workspace by adding place nodes to the map at regular distances (in our case $\Delta s = 10$ cm). Due to this and due to the robot moving along parallel lanes (in our case $\Delta l = 30$ cm apart), a grid-like representation of the robot's environment is built. The dense topological map offers the advantages of topological maps, namely characterizing places by sensory information without extracting features and without integrating their spatial positions into a map with a common frame of reference. In contrast to standard topological maps, it represents the robot's environment with a finer resolution. At the current stage of our work, the dense topological representation is used to estimate the robot's current distance to the previous lane; in future work, the map will be used for deriving spatial information required to completely cover complex workspaces (e.g. to distinguish between cleaned and uncleaned areas).

The estimate of the robot's distance to the previous lane is derived by triangulation. For this purpose, the bearing to several snapshots stored along the robot's previous lane is taken, and the angular information is combined with odometry estimates of the relative distances between the considered snapshots along the previous lane (see Section 5.3 for details). By keeping the distance to the previous lane constant, the robot is guided along a lane which is parallel to its previous lane (Section 5.4). The results of the experiments have shown that the proposed method achieves good results with only a small portion of overlap or gaps between consecutive lanes.

For controlling the robot, only the robot's distance to the previous lane and the robot's current orientation in world coordinates is computed. Thus, we do not estimate the robot's position or the robot's full pose as it is usually done in current standard approaches. In contrast to standard approaches, we refer to our particular method as partial pose estimation. By computing the distance to the previous lane and the robot's current orientation, we solely rely on the necessary and sufficient information required to fulfill the task. With less information, the robot would no longer be capable of accomplishing its task. We think that avoiding unnecessary computations is an important aspect, especially if one has in mind that a potential cleaning robot will only be equipped with limited computational power and that relying on a less powerful computer can also reduce the costs of a possible product.

Table A.1
True inter-lane distances for experiment 1 as depicted in Fig. 15 (left). All results in centimeters. Pooled percentiles were obtained by pooling over all lanes and are in Section 7.2 referred to as $\bar{P}_x$.

Percentile	Lane i							Pooled
	1	2	3	4	5	6	7	
$P^{(i)}_{0.05}$	26.9	22.4	21.3	21.6	19.8	21.4	19.0	20.9
$P^{(i)}_{0.25}$	30.6	26.6	26.8	25.3	25.8	25.9	24.4	26.3
$P^{(i)}_{0.50}$	32.9	29.1	30.3	28.9	28.8	29.7	27.9	29.8
$P^{(i)}_{0.75}$	35.2	31.7	32.8	31.8	30.6	32.5	30.8	32.4
$P^{(i)}_{0.95}$	38.5	37.0	38.4	36.1	33.7	37.3	34.7	37.0

9. Future work

In future work, we will mainly work towards a complete framework of cleaning strategies required to completely cover complex-shaped workspaces by combining several segments of parallel and meandering lanes as resulting from the proposed trajectory controller. In contrast to some current approaches on planning of cleaning paths (Section 3.2.1), our strategy will incrementally cover the robot's workspace instead of precomputing the decomposition into cleaning segments based on an a priori known map. The resulting system will make use of the underlying dense topological map for detecting areas which still need to be cleaned, for loop-closure detection to avoid repeated coverage (e.g. by applying the methods proposed by [73]), for planning and approaching new lanes or new cleaning segments, and for planning and following paths back to the robot's charging station. With our work on navigation strategies of autonomous cleaning robots, we hope to develop a robot system relying on omnidirectional vision being capable of solving a real-world service task.

In order to compare the proposed method relying on partial pose estimation to standard methods relying on full pose estimation, future working directions will also include to apply Bayesian filter frameworks such as the extended Kalman filter or the particle filter to obtain a topological map with complete metrical pose information. Although the method was developed for indoor navigation of autonomous cleaning robots, it can also be applied to other tasks requiring complete coverage such as lawn mowing.

Acknowledgments

This work was funded by the German Research Foundation DFG under the grants MO 1037/2 and EXC 277. We are grateful to the two anonymous reviewers for comprehensive and constructive feedback, which helped to improve this manuscript.

Appendix A. Additional data

Tables A.1–A.4 summarize the results described in Section 7 in tabular form.

Appendix B. Supplementary data

Together with the manuscript, we submitted three supplemental videos: (i) a robot run without visual correction (i.e. only relying on its disturbed odometry, Fig. 12), (ii) a robot run under visual control without disturbed odometry, and (iii) a robot run under visual control with disturbed odometry (Figs. 13 and 14). The supplementary videos can be found online at http://dx.doi.org/10.1016/j.robot.2012.12.006.

Table A.2
True inter-lane distances for experiment 2 as depicted in Fig. 15 (right). All results in centimeters. Pooled percentiles were obtained by pooling over all lanes and are in Section 7.2 referred to as $\bar{P}_x$.

Percentile	Lane i														Pooled
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
$P^{(i)}_{0.05}$	23.9	21.3	28.3	21.1	25.8	20.5	21.7	18.7	22.3	15.0	18.3	16.1	21.8	16.4	19.4
$P^{(i)}_{0.25}$	29.1	26.3	30.8	26.9	29.7	24.7	27.8	23.8	26.4	19.8	24.5	26.2	28.3	24.9	26.2
$P^{(i)}_{0.50}$	30.5	29.0	34.2	29.7	32.0	27.9	29.9	26.7	29.9	24.2	30.1	30.1	30.9	29.4	29.8
$P^{(i)}_{0.75}$	32.7	30.7	37.9	32.2	35.8	30.1	32.7	29.8	32.2	28.9	31.8	32.7	33.0	31.5	32.2
$P^{(i)}_{0.95}$	37.4	34.6	42.2	37.3	39.4	32.3	37.4	32.0	35.4	32.2	35.1	40.1	36.7	39.2	37.7

Table A.3
Cleaning performance for experiment 1. All results in percent. A_0, A_1, and A_2 refer to the uncovered area, the area covered once, and the area covered repeatedly. The last column contains the average computed over all trials; these values are in Section 7.3 referred to as $\bar{A}_0$, $\bar{A}_1$, and $\bar{A}_2$.

	Trial										Avg.
	1	2	3	4	5	6	7	8	9	10	
A_0	7.4	9.4	7.8	7.3	6.4	7.8	7.6	7.9	8.1	8.8	7.9
A_1	85.9	81.6	86.4	86.4	87.7	85.1	85.3	84.8	85.8	85.2	85.4
A_2	6.7	9.0	5.9	6.3	5.9	7.1	7.1	7.3	6.1	6.0	6.7

Table A.4
Cleaning performance for experiment 2. All results in percent. A_0, A_1, and A_2 refer to the uncovered area, the area covered once, and the area covered repeatedly. The last column contains the average computed over all trials; these values are in Section 7.3 referred to as $\bar{A}_0$, $\bar{A}_1$, and $\bar{A}_2$.

	Trial										Avg.
	1	2	3	4	5	6	7	8	9	10	
A_0	5.9	14.6	9.7	9.6	6.6	9.6	8.4	10.4	17.2	11.3	10.3
A_1	89.3	75.5	82.6	84.6	86.9	81.1	82.6	79.4	71.7	81.5	81.5
A_2	4.8	9.8	7.7	5.9	6.6	9.3	9.0	10.2	11.1	7.2	8.1

References

[1] E. Prassler, A. Ritter, C. Schaeffer, P. Fiorini, A short history of cleaning robots, Autonomous Robots 9 (3) (2000) 211–226.
[2] E. Prassler, K. Kosuge, Domestic robotics, in: B. Siciliano, O. Khatib (Eds.), Springer Handbook of Robotics, Springer, 2008, pp. 1253–1281.
[3] R. Benosman, S.B. Kang, Panoramic Vision: Sensors, Theory, and Applications, 1st ed., Springer, 2001.
[4] R. Möller, M. Krzykawski, L. Gerstmayr, Three 2D-warping schemes for visual robot navigation, Autonomous Robots 29 (3) (2010) 253–291.
[5] M.O. Franz, H.A. Mallot, Biomimetic robot navigation, Robotics and Autonomous Systems 30 (1–2) (2000) 133–153.
[6] W. Burgard, M. Hebert, World modeling, in: B. Siciliano, O. Khatib (Eds.), Springer Handbook of Robotics, Springer, 2008, pp. 853–869.
[7] R. Siegwart, I.R. Nourbakhsh, Introduction to Autonomous Mobile Robots, MIT Press, 2004.
[8] Z. Chen, J. Samarabandu, R. Rodrigo, Recent advances in simultaneous localization and mapping using computer vision, Advanced Robotics 21 (3–4) (2007) 233–265.
[9] S. Thrun, J.J. Leonard, Simultaneous localization and mapping, in: B. Siciliano, O. Khatib (Eds.), Springer Handbook of Robotics, Springer, 2008, pp. 871–889.
[10] S. Kreft, Reinigungstrajektorien mobiler Roboter unter visueller Steuerung, Diploma thesis (in German), Computer Engineering Group, Faculty of Technology, Bielefeld University, 2007.
[11] R. Möller, A. Vardy, L. Gerstmayr, F. Röben, S. Kreft, Neuroethological concepts at work: insect-inspired methods for visual robot navigation, in: Biological Approaches for Engineering, 2008, pp. 91–94.
[12] L. Gerstmayr, F. Röben, M. Krzykawski, S. Kreft, D. Venjakob, R. Möller, A vision-based trajectory controller for autonomous cleaning robots, in: R. Dillmann, J. Beyerer, C. Stiller, J.M. Zöllner, T. Gindele (Eds.), Autonome Mobile Systeme 2009, Springer, 2009, pp. 65–72.
[13] F. Röben, Biologically inspired visual navigation in indoor and outdoor environments, Ph.D. Thesis, Faculty of Technology, Bielefeld University, 2010.
[14] J.-A. Meyer, D. Filliat, Map-based navigation in mobile robots: Part II. A review of map-learning and path-planning strategies, Cognitive Systems Research 4 (4) (2003) 283–317.
[15] D. Filliat, J.-A. Meyer, Map-based navigation in mobile robots: Part I. A review of localization strategies, Cognitive Systems Research 4 (4) (2003) 243–282.
[16] P. Buschka, A. Saffiotti, Some notes on the use of hybrid maps for mobile robots, in: Proceedings of the International Conference on Intelligent Autonomous Systems, IAS 04, 2004, pp. 547–556.
[17] T. Bailey, H. Durrant-Whyte, Simultaneous localization and mapping (SLAM): Part II. State of the art, IEEE Robotics and Automation Magazine 13 (3) (2006) 108–117.
[18] S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics, MIT Press, 2005.
[19] M. Saedan, C.W. Lim, M.H. Ang, Omnidirectional image matching for vision-based robot localization, in: Proceedings of the IEEE International Conference on Mechatronics, ICM 06, 2006, pp. 17–22.
[20] M. Saedan, C. Lim, M.H. Ang, Vision-based localization using a central catadioptric vision system, in: O. Khatib, V. Kumar, D. Rus (Eds.), Experimental Robotics, in: Springer Tracts in Advanced Robotics, vol. 39, Springer, Berlin, Heidelberg, 2008, pp. 397–406.
[21] T. Goedemé, T. Tuytelaars, L. van Gool, G. Vanacker, M. Nuttin, Feature based omnidirectional sparse visual path following, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 05, 2005, pp. 1806–1811.
[22] T. Goedemé, M. Nuttin, T. Tuytelaars, L. van Gool, Omnidirectional vision based topological navigation, International Journal of Computer Vision 74 (3) (2007) 219–236.
[23] A.C. Murillo, C. Sagüés, J.J. Guerrero, T. Goedemé, From omnidirectional images to hierarchical localization, Robotics and Autonomous Systems 55 (5) (2007) 372–382.
[24] F. Dayoub, T. Duckett, G. Cielniak, An adaptive spherical view representation for navigation in changing environments, in: Proceedings of the European Conference on Mobile Robots, ECMR 09, 2009, pp. 1–6.
[25] L. Maohai, S. Lining, H. Qingcheng, C. Zesu, P. Songhao, Robust omnidirectional vision based on mobile robot hierarchical localization and autonomous navigation, Information Technology Journal 10 (1) (2011) 29–39.
[26] K. Daniilidis, J.O. Eklundh, 3-D vision and recognition, in: B. Siciliano, O. Khatib (Eds.), Springer Handbook of Robotics, Springer, 2008, pp. 543–562.
[27] G.N. DeSouza, A.C. Kak, Vision for mobile robot navigation: a survey, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (2) (2002) 237–267.
[28] R. Bunschoten, B. Kröse, Robust scene reconstruction from an omnidirectional vision system, IEEE Transactions on Robotics and Automation 19 (2) (2003) 351–357.
[29] B. Kröse, R. Bunschoten, S.T. Hagen, B. Terwijn, N. Vlassis, Household robots look and learn: environment modeling and localization from an omnidirectional vision system, IEEE Robotics and Automation Magazine 11 (4) (2004) 45–52.
[30] S. Fleck, F. Busch, P. Biber, W. Straßer, H. Andreasson, Omnidirectional 3d modeling on a mobile robot using graph cuts, in: Proceedings of the

IEEE International Conference on Robotics and Automation, ICRA 05, 2005, pp. 1748–1754.
[31] S. Ikeda, J. Miura, 3d indoor environment modeling by a mobile robot with omnidirectional stereo and laser range finder, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 06, 2006, pp. 3435–3440.
[32] S. Fleck, F. Busch, P. Biber, W. Straßer, Graph cut based panoramic 3d modeling and ground truth comparison with a mobile platform—the wägele, Image and Vision Computing 27 (1–2) (2009) 141–152.
[33] F.R. Correa, J. Okamoto, Omnidirectional stereovision system for occupancy grid, in: Proceedings of the IEEE International Conference on Advanced Robotics, ICAR 05, 2005, pp. 628–634.
[34] C. Plagemann, C. Stachniss, J. Hess, F. Endres, N. Franklin, A nonparametric learning approach to range sensing from omnidirectional vision, Robotics and Autonomous Systems 58 (6) (2010) 762–772.
[35] J. Miura, Y. Negishi, Y. Shirai, Mobile robot map generation by integrating omnidirectional stereo and laser range finder, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 02, 2002, pp. 250–255.
[36] Y. Negishi, J. Miura, Y. Shirai, Vision-based mobile robot speed control using a probabilistic occupancy map, Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, MFI 03, 2003, pp. 64–69.
[37] Y. Negishi, J. Miura, Y. Shirai, Mobile robot navigation in unknown environments using omnidirectional stereo and laser range finder, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 04, Vol. 3, 2004, pp. 2737–2742.
[38] F.v. Hundelshausen, S. Behnke, R. Rojas, An omnidirectional vision system that finds and tracks color edges and blobs, in: RoboCup 2001: Robot Soccer World Cup V, Springer, 2002, pp. 374–379.
[39] E. Menegatti, A. Pretto, A. Scarpa, E. Pagello, Omnidirectional vision scan matching for robot localization in dynamic environments, IEEE Transactions on Robotics 22 (3) (2006) 523–535.
[40] P. Heinemann, J. Haase, A. Zell, A combined Monte-Carlo localization and tracking algorithm for RoboCup, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 06, 2006, pp. 1535–1540.
[41] T. Tuytelaars, K. Mikolajczyk, Local invariant feature detectors: a survey, Foundations and Trends in Computer Graphics and Vision 3 (3) (2008) 177–280.
[42] J. Li, N.M. Allinson, A comprehensive review of current local features for computer vision, Neurocomputing 71 (2008) 1771–1787.
[43] A. Gil, O.M. Mozos, M. Ballesta, O. Reinoso, A comparative evaluation of interest point detectors and local descriptors for visual SLAM, Machine Vision Applications 21 (6) (2010) 905–920.
[44] B. Triggs, P. McLauchlan, R. Hartley, A. Fitzgibbon, Bundle adjustment — a modern synthesis, in: Proceedings of the International Workshop on Vision Algorithms, 2000, pp. 298–372.
[45] R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, 2003.
[46] D. Scaramuzza, F. Fraundorfer, M. Pollefeys, Closing the loop in appearance-guided omnidirectional visual odometry by using vocabulary trees, Robotics and Autonomous Systems 58 (6) (2010) 820–827.
[47] T. Lemaire, S. Lacroix, SLAM with panoramic vision, Journal of Field Robotics 24 (1–2) (2007) 91–111.
[48] A. Levin, R. Szeliski, Visual odometry and map correlation, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 04, 2004, pp. 611–618.
[49] C. Sagüés, A.C. Murillo, J.J. Guerrero, T. Goedemé, Localization with omnidirectional images using the radial trifocal tensor, in: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 06, 2006, pp.551–556.
[50] W. Hübner, H.A. Mallot, Metric embedding of view graphs: a vision and odometry-based approach to cognitive mapping, Autonomous Robots 23 (3) (2007) 183–196.
[51] H.-M. Gross, A. König, H.J. Böme, C. Schröter, Vision-based monte carlo self-localization for a mobile service robot acting as shopping assistant in a home store, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2002, pp. 256–262.
[52] H.-M. Gross, A. König, C. Schröter, H.J. Böhme, Omnivision-based probabilistic self-localization for a mobile shopping assistant continued, in: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 03, 2003, pp. 1505–1511.
[53] H. Andreasson, A. Treptow, T. Duckett, Localization for mobile robots using panoramic vision, local features and particle filter, in: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, ICRA 05, 2005, pp. 3348–3353.
[54] H. Tamimi, H. Andreasson, A. Treptow, T. Duckett, A. Zell, Localization of mobile robots with omnidirectional vision using particle filter and iterative sift, Robotics and Autonomous Systems 54 (9) (2006) 758–765.
[55] H. Andreasson, A. Treptow, T. Duckett, Self-localization in non-stationary environments using omni-directional vision, Robotics and Autonomous Systems 55 (7) (2007) 541–551.
[56] H. Andreasson, T. Duckett, A.J. Lilienthal, Mini-SLAM: Minimalistic visual SLAM in large-scale environments based on a new interpretation of image similarity, in: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 07, 2007, pp. 4096–4101.
[57] M. Saedan, C.W. Lim, M.H. Ang, Appearance-based SLAM with map loop closing using an omnidirectional camera, in: Proceedings of the IEEE/ASME International Conference Advanced Intelligent Mechatronics, ICAIM 07, 2007, pp. 1–6.
[58] A.C. Murillo, C. Sagüés, J.J. Guerrero, T. Goedemé, Hierarchical localization by matching vertical lines in omnidirectional images, in: Proceedings of the IEEE/RSJ International Workshop From Sensors to Human Spatial Concepts, 2006, pp. 13–19.
[59] A.C. Murillo, J.J. Guerrero, C. Sagüés, Surf features for efficient robot localization with omnidirectional images, in: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 07, 2007, pp. 3901–3907.
[60] H. Andreasson, T. Duckett, A.J. Lilienthal, A minimalistic approach to appearance-based visual SLAM, IEEE Transactions on Robotics 24 (5) (2008) 991–1001.
[61] P.E. Rybski, F. Zacharias, M.L. Gini, N. Papanikolopoulos, Using visual features for building and localizing within topological maps of indoor environments, in: Innovations in Robot Mobility and Control, Springer, 2005, pp. 251–271.
[62] P.E. Rybski, S. Roumeliotis, M.L. Gini, N. Papanikopoulos, Appearance-based mapping using minimalistic sensor models, Autonomous Robots 24 (3) (2008) 229–246.
[63] I. Esteban, O. Booij, Z. Zivkovic, B. Kröse, Mapping large environments with an omnivideo camera, in: International Conference on Simulation, Modelling and Programming of Autonomous Robots, SIMPAR 08, Springer, 2008, pp. 297–306.
[64] O. Booij, Z. Zivkovic, B. Kröse, Efficient data association for view-based SLAM using connected dominating sets, Robotics and Autonomous Systems 57 (12) (2009) 1225–1234.
[65] T.H. Cormen, C. Stein, Rivest R L, C.E. Leiserson, Introduction to Algorithms, 2nd ed., McGraw-Hill, 2001.
[66] L. Smith, A. Philippides, P. Graham, B. Baddeley, P. Husbands, Linked local navigation for visual route guidance, Adaptive Behavior 15 (3) (2007) 257–271.
[67] L. Smith, A. Philippides, P. Graham, P. Husbands, Linked local visual navigation and robustness to motor noise and route displacement, in: Proceedings of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats, SAB 08, 2008, pp. 179–188.
[68] F. Labrosse, Short and long-range visual navigation using warped panoramic images, Robotics and Autonomous Systems 55 (9) (2007) 675–684.
[69] A. Vardy, Long-range visual homing, in: Proceedings of the IEEE International Conference on Robotics and Biomimetics, 2006, pp. 220–226.
[70] M.O. Franz, B. Schölkopf, H.A. Mallot, H.H. Bülthoff, Learning view graphs for robot navigation, Autonomous Robots 5 (1) (1998) 111–125.
[71] I. Ulrich, I. Nourbakhsh, Appearance-based place recognition for topological localization, in: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 00, 2000, pp. 1023–1029.
[72] F. Werner, J. Sitte, F.D. Maire, Automatic place determination using colour histograms and self-organising maps, in: Proceedings of the 13th International Conference on Advanced Robotics, ICAR 07, 2007, pp. 111–115.
[73] L. Gerstmayr-Hillen, O. Schlüter, M. Krzykawski, R. Möller, Parsimonious loop-closure detection based on global image-descriptors of panoramic images, in: Proceedings of the 15th International Conference on Advanced Robotics, ICAR 2011, 2011, pp. 576–581.
[74] E. Menegatti, T. Maeda, H. Ishiguro, Image-based memory for robot navigation using properties of omnidirectional images, Robotics and Autonomous Systems 47 (4) (2004) 251–267.
[75] L. Paletta, S. Frintrop, J. Hertzberg, Robust localization using context in omnidirectional imaging, in: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 01, Vol. 2, 2001, pp. 2072–2077.
[76] M. Jogan, A. Leonardis, Robust localization using panoramic view-based recognition, in: Proceedings of the International Conference on Pattern Recognition, ICPR 00, Vol. 4, 2000, pp. 136–139.
[77] M. Jogan, A. Leonardis, Robust localization using an omnidirectional appearance-based subspace model of the environment, Robotics and Autonomous Systems 45 (1) (2003) 51–72.
[78] A.A. Argyros, K.E. Bekris, S.C. Orphanoudakis, L.E. Kavraki, Robot homing by exploiting panoramic vision, Autonomous Robots 19 (1) (2005) 7–25.
[79] A. Ascani, E. Frontoni, A. Mancini, P. Zingaretti, Robot localization using omnidirectional vision in large and dynamic outdoor environments, in: Proceedings of the IEEE/ASME International Conference on Mechtronic and Embedded Systems and Applications, MESA 08, 2008, pp. 576–581.
[80] C. Valgren, A.J. Lilienthal, Sift, surf and seasons: appearance-based long-term localization in outdoor environments, Robotics and Autonomous Systems 58 (2) (2010) 157–165.
[81] C. Valgren, A.J. Lilienthal, T. Duckett, Incremental topological mapping using omnidirectional vision, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 06, 2006, pp. 3441–3447.

[82] Z. Zivkovic, B. Bakker, B. Kröse, Hierarchical map building using visual landmarks and geometric constraints, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 05, 2005, pp. 2480–2485.
[83] B. Kröse, O. Booij, Z. Zivkovic, A geometrically constrained image similarity measure for visual mapping, localization and navigation, in: Proceedings of the 3rd European Conference on Mobile Robots, ECMR 07, 2007, pp. 168–174.
[84] O. Booij, B. Terwijn, Z. Zivkovic, B. Kröse, Navigation using an appearance based topological map, in: Proceedings of the IEEE Intnernational Conference on Robotics and Automation, ICRA 07, 2007, pp. 3927–3932.
[85] H. Sahin, L. Guvenc, Household robotics: autonomous devices for vacuuming and lawn mowing, IEEE Control Systems Magazine 27 (2) (2007) 20–96.
[86] E. Garcia, M.A. Jimenez, P.G. De Santos, M. Armada, The evolution of robotics research, IEEE Robotics & Automation Magazine 14 (1) (2007) 90–103.
[87] N. Elkmann, J. Hortig, M. Fritzsche, Cleaning automation, in: S.Y. Nof (Ed.), Handbook of Automation, Springer, 2009, pp. 1253–1264.
[88] S. Rhim, J. Ryu, K. Park, S. Lee, Performance evaluation criteria for autonomous cleaning robots, in: International Symposium on Computational Intelligence in Robotics and Automation, CIRA 07, 2007, pp. 167–172.
[89] T. Palleja, M. Tresanchez, M. Teixido, J. Palacín, Modeling floor-cleaning coverage performances of some domestic mobile robots in a reduced scenario, Robotics and Autonomous Systems 58 (1) (2010) 37–45.
[90] H. Choset, Coverage for robotics—a survey of recent results, Annals of Mathematics and Artificial Intelligence 31 (1–4) (2001) 113–126.
[91] Y.-H. Choi, T.-K. Lee, S. Baek, S.-Y. Oh, Online complete coverage path planning for mobile robots based on linked spiral paths using constrained inverse distance transform, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2009, pp. 5788–5793.
[92] Y. Liu, X. Lin, S. Zhu, Combined coverage path planning for autonomous cleaning robots in unstructured environments, in: Proceedings of the World Congress on Intelligent Control and Automation, WCICA 08, 2008, pp. 8271–8276.
[93] C. Luo, S.J. Yang, A bioinspired neural network for real-time concurrent map building and complete coverage robot navigation in unknown environments, IEEE Transactions on Neural Networks 19 (7) (2008) 1279–1298.
[94] T. Lee, S. Baek, S. Oh, Sector-based maximal online coverage of unknown environments for cleaning robots with limited sensing, Robotics and Autonomous Systems 59 (10) (2011) 698–710.
[95] B. Yamauchi, A frontier-based approach for autonomous exploration, in: Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation, CIRA 97, 1997, pp. 146–151.
[96] B. Yamauchi, A. Schultz, W. Adams, Mobile robot exploration and map-building with continuous localization, in: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 98, Vol. 4, 1998, pp. 3715–3720.
[97] L. Freda, G. Oriolo, Frontier-based probabilistic strategies for sensor-based exploration, in: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 05, 2005, pp. 3881–3887.
[98] W.Y. Jeong, K.M. Lee, CV-SLAM: A new ceiling vision-based SLAM technique, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2005, pp. 3195–3200.
[99] W.Y. Jeong, K.M. Lee, Visual SLAM with line and corner features, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2006, pp. 2570–2575.
[100] H.S. Lee, K.M. Lee, Multi-robot SLAM using ceiling vision, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2009, pp. 912–917.
[101] H.S. Lee, K.M. Lee, Multiswarm particle filter for vision-based SLAM, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2009, pp. 924–929.
[102] K. Choi, J. Park, Y.H. Kim, H.K. Lee, Monocular SLAM with undelayed initialization for an indoor robot, Robotics and Autonomous Systems 60 (6) (2012) 841–851.
[103] T.B. Kwon, J.B. Song, S.C. Kang, MCL-based global localization of cleaning robot using fast rotation-invariant corner matching method, in: Proceedings of the International Conference on Control, Automation, and Systems, ICROS 10, 2010, pp. 1988–1992.
[104] J.S. Gutmann, G. Brisson, E. Eade, P. Fong, M. Munich, Vector field SLAM, in: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 10, 2010, pp. 236–242.
[105] J.S. Gutmann, E. Eade, P. Fong, M. Munich, A constant-time algorithm for vector field SLAM using an exactly sparse extended information filter, in: Proceedings of Robotics: Science and Systems Conference, RSS 10, 2010, p. P25.
[106] Y. Yamamoto, P. Prijanian, J. Brown, M. Munich, E.D. Bernardo, L. Goncalves, J. Ostrowski, N. Karlsson, Optical sensing for robot perception and localization, in: Proceedings of the Workshop on Advanced Robotics and its Social Impacts, ARSO 05, 2005, pp. 14–17.
[107] I. Vallivaara, J. Haverinen, A. Kemppainen, J. Röning, Simultaneous localization and mapping using ambient magnetic field, in: IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems, MFI 10, 2010, pp. 14–19.
[108] I. Vallivaara, J. Haverinen, A. Kemppainen, J. Röning, Magnetic field-based SLAM method for solving the localization problem in mobile robot floor-cleaning task, in: Proceedings of the IEEE International Conference on Advanced Robotics, ICAR 11, 2011, pp. 198–203.
[109] R. Nelson, J. Aloimonos, Finding motion parameters from spherical motion fields (or the advantage of having eyes in the back of your head), Biological Cybernetics 58 (4) (1988) 261–273.
[110] W. Stürzl, D. Soccol, J. Zeil, N. Böddeker, M.V. Srinivasan, Rugged, obstruction-free, mirror-lens combination for panoramic imaging, Applied Optics 47 (32) (2008) 6070–6078.
[111] W. Stürzl, M. Suppa, D. Burschka, Leight-weight panoramic mirror design for visual navigation, in: Workshop Proceedings on the International Conference on Simulation, Modeling, and Programming for Autonomous Robots, SIMPAR 08, 2008, pp. 218–229.
[112] B. Cartwright, T. Collett, Landmark learning in bees, Journal of Comparative Physiology A 151 (4) (1983) 521–543.
[113] M.O. Franz, B. Schölkopf, H.A. Mallot, H.H. Bülthoff, Where did I take that snapshot? scene-based homing by image matching, Biological Cybernetics 79 (3) (1998) 191–202.
[114] R. Möller, Local visual homing by warping of two-dimensional images, Robotics and Autonomous Systems 57 (1) (2009) 87–101.
[115] J. Zeil, M.I. Hoffmann, J.S. Chahl, Catchment areas of panoramic images in outdoor scenes, Journal of the Optical Society of America A 20 (3) (2003) 450–469.
[116] E. Batschelet, Circular Statistics in Biology, Academic Press, 1981.
[117] R. Möller, Insect visual homing strategies in a robot with analog processing, Biological Cybernetics 83 (3) (2000) 231–243.
[118] P. Greguss, Panoramic imaging block for three-dimensional space, US Patent No. 4,566,763, 1986.
[119] D. Comaniciu, P. Meer, Mean shift: a robust approach toward feature space analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (5) (2002) 603–619.

Lorenz Gerstmayr-Hillen received a diploma in computer science with focus on neurobiology from the University of Tübingen (Germany). He is currently a doctoral student at the Computer Engineering Group and at the Center of Excellence "Cognitive Interaction Technology" at Bielefeld University (Germany). His research interests are local visual homing methods, in particular optical flow methods, navigation in topological maps, and the visual control of domestic floor-cleaning robots.

Frank Röben received a diploma in computer science and his Ph.D. from the Bielefeld Faculty of Technology. Before leaving the Computer Engineering Group, he was working as a postdoctoral researcher with focus on vision-based outdoor navigation. He is now working for a large German manufacturer of agricultural machinery.

Martin Krzykawski was a doctoral student at the Computer Engineering Group. His research interests were warping methods, FPGA implementations of visual homing methods, and visual navigation of domestic cleaning robots.

Sven Kreft was a former diploma student at the Computer Engineering Group and received a degree in computer science. He then joined the Product Engineering group of the Heinz Nixdorf Institute at Paderborn, Germany. There, his focus is on the generation of environment models for virtual reality driving simulators based on geographic information systems. He received a Ph.D. from the university of Paderborn.

Daniel Venjakob received his B.Sc. and M.Sc. degrees in computer science from the Bielefeld Faculty of Technology. His working areas were visual navigation methods of autonomous robot and visual tracking of mobile robots for data evaluation.

Ralf Möller heads the Computer Engineering Group at the Faculty of Technology of Bielefeld University, Germany. His research interests include behavior-based approaches to visual cognition, visual navigation, biorobotics, neuromorphic systems, and parallel computation.