Ground Robot Navigation using Uncalibrated Cameras

Olivier Koch, Matthew R. Walter, Albert S. Huang, Seth Teller
MIT Computer Science and Artificial Intelligence Laboratory
Cambridge, MA
Email: {koch, mwalter, ashuang, teller}@mit.edu

***Abstract*— Precise calibration of camera intrinsic and extrinsic parameters, while often useful, is difficult to obtain during field operation and presents scaling issues for multi-robot systems. We demonstrate a vision-based approach to navigation that does not depend on traditional camera calibration, and present an algorithm for guiding a robot through a previously traversed environment using a set of uncalibrated cameras mounted on the robot.**

On the first excursion through an environment, the system builds a topological representation of the robot's exploration path, encoded as a place graph. On subsequent navigation missions, the method localizes the robot within the graph and provides robust guidance to a specified destination. We combine this method with reactive collision avoidance to obtain a system able to navigate the robot safely and reliably through the environment. We validate our approach with ground-truth experiments and demonstrate the method on a small ground rover navigating through several dynamic environments.

I. Introduction

Vision-based robot navigation algorithms typically assume that the camera intrinsic parameters (focal length, optical center, and distortion) and extrinsic parameters (robot-relative pose) have been determined in advance and do not change over time unless specifically controlled by the robot. Knowledge of these parameters enables the projection of points on the image plane into rays in the robot body frame, allowing the robot to use its cameras to reason geometrically about the world with well-studied algorithms such as stereo, structure-from-motion, visual SLAM, and visual odometry.

Obtaining the calibration parameters typically requires a special calibration process involving visual patterns or environments that are specifically designed and constructed to aid parameter recovery. While the calibration procedure may be convenient and expeditious in a laboratory or controlled environment, it may not be feasible to execute in the field or in any situation in which the calibration tools and equipment are not available. Especially in the case of deployments of large numbers of robots, executing the calibration procedure may be challenging. Robots operating in real-world conditions may often be bumped, damaged, repaired, or otherwise altered in such a way that would invalidate previously acquired calibration data. Robots may often be disassembled for storage or transport and parts may shift slightly during reassembly. A natural question to ask is then: What can be accomplished without traditional camera calibration? We investigate this question in the context of mobile robot navigation, and propose a method for vision-based navigation using uncalibrated cameras, suitable for mobile ground robots.

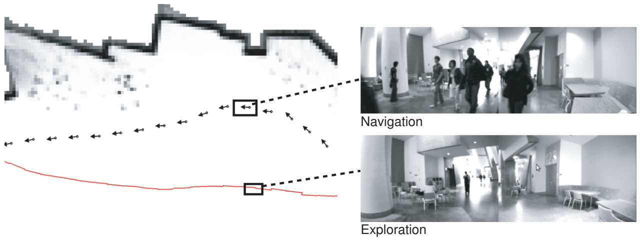

Fig. 1. We present a vision-based method for ground robot navigation. Assuming that the robot has previously explored the environment (*red path*), the method provides guidance in the robot's body frame during revisitation (*black arrows*). we demonstrate the robustness of our method on real-world experiments in highly dynamic scenes.

In particular, we consider the problem of robot navigation within a previously visited environment, a commonly desired ability for a mobile robot that operates exclusively within a target region. Robots serving as delivery agents, passenger transports, building guides, patrols, etc. all require this functionality, which may be considered as a baseline capability for a large class of mobile robots. We impose no constraint on the environment except that the traversable region can be modeled as a 2D manifold, and that it contains descriptive visual features.

The primary contribution of this paper is a method that extends our previous work in human-directed navigation [1] to provide coarse waypoint navigation for mobile robots. In contrast to our earlier work, this paper presents a method for autonomous robotic navigation with uncalibrated cameras, a problem that is very different from that of guiding a human. We describe and demonstrate an algorithm that uses multiple cameras to yield directional commands for a robot, without having or estimating the camera intrinsic parameters or the camera-to-robot rigid body transformations. We assume that the robot can accept directional commands, and has a basic reactive obstacle avoidance capability.

II. Related Work

Robotic navigation has been well-studied in the context of range-based sensors such as sonar and LIDAR. Numerous metric mapping techniques exist for robots operating in a 2D environment using bearing-only sensors [2], [3], and extensions to full 3D environments have been demonstrated. Many of these techniques have been successfully applied to visual navigation, where precise camera calibration is assumed. Visual odometry [4] and visual SLAM [5]–[7] in particular have seen significant progress in recent years. To alleviate some of the scaling issues present in metric mapping systems, researchers have also studied topological mapping techniques that use connectivity graphs to model traversable regions [8], [9].

Whether range- or vision-based, each of these approaches require precise calibration of the sensor intrinsic and extrinsic parameters. In the absence of executing a specialized camera calibration procedure, one approach is to attempt automatic calibration during field operation [10], [11]. Such self-calibration methods are typically able to recover the focal length, principal point, and image skew from camera rotation. Still an active area of research, existing methods are non-trivial to implement, and usually ignore nonlinear effects such as radial distortion. Meanwhile, recovering extrinsic parameters, specifically the camera-to-robot transformation, is not feasible using these techniques.

A key challenge for a navigation algorithm is the ability to recognize when the robot has arrived at a previously visited place, usually referred to as the "loop-closing" problem. Vision-based algorithms commonly accomplish this by comparing newly observed visual features with previously observed features. Recent bag-of-words approaches, which quantize features into visual "words" and use a set of observed words to represent a place, have improved the speed and robustness of visual loop closing [12], [13].

Our approach is motivated by simplicity in practice. It makes no assumptions about the environment except that it is traversable by a wheeled robot, and that it contains descriptive visual features. The latter is a typical requirement of any vision system, as a sparse visual field often results in degenerate solutions for many vision algorithms. Our approach does involve a "training" phase to extract some basic camera parameters. However, the training procedure is simple and can be executed quickly and with minimal effort.

III. Method Overview

Our approach assumes that the robot first explores the environment. During this *exploration phase*, the system builds a topological representation of the robot's path that we call the *place graph*. The place graph records visual information only. In our experiments, this step is performed manually. However, methods exist to automate this phase [14], [15].

In a second phase, the robot navigates autonomously through the explored environment. We split the navigation problem into two complementary processes. First, a high-level vision-based method provides global localization in the place graph as well as a coarse directional cue in the robot's body frame. The second consists of a low-level obstacle avoidance algorithm that steers the robot's motion in the desired direction while reactively avoiding local obstacles. Figure 2 illustrates the method.

A. Place Graph Generation

The place graph, shown in Figure 3, is a topological representation of the robot's exploration path as an undirected graph $G = (V, E)$. Each node in the graph $v \in V$ corresponds to a physical location in the environment, while an edge $e \in E$ corresponds to a known physical path between the corresponding pair of nodes. Associated with each node x_k is the set of visual features observed at that location on the way to each of its neighbors. No observations are associated with graph edges. Hence, the graph G can be thought of as a sparse visual representation of the robot's exploration path.

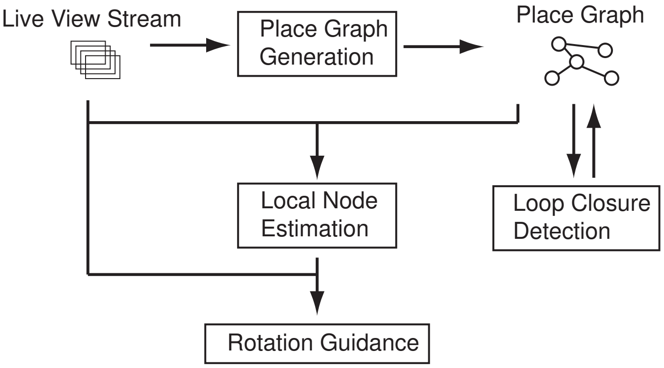

Fig. 2. Method overview. During exploration, the system builds a place graph representing the robot's exploration path. During revisitation, the method uses the graph to localize the robot and provide guidance in the robot's body frame.

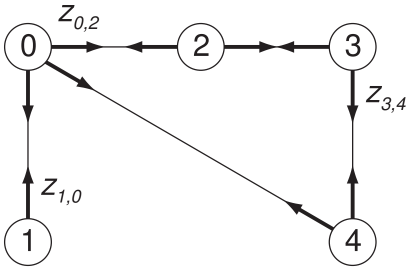

Fig. 3. We represent the robot's exploration path as an undirected graph where nodes represent locations in the world and edges represent physical paths between nodes. Visual observations (e.g. SIFT features [16]) are associated with each node.

At the start of the exploration phase, the graph G is empty. As the robot explores the environment, new nodes are added to the graph. Nodes are instantiated when the visual appearance of the local environment differs sufficiently from that of the previous node, as measured by the variation in visual features. More specifically, we measure variation in appearance based upon a distance function Ψ that matches sets of features between two images, and returns the normalized mean L_2 distance between matches in the feature descriptor space. Feature matching utilizes a function Φ that we describe in § III-B. When the distance Ψ between the current observations and those of the last node exceed a threshold, we instantiate a new node in the graph, joined with an edge to the previous node. We use a distance threshold of 0.85, although we found the method to yield similar results for a range of values (0.7 to 0.9).

B. Feature Matching

Our localization and guidance algorithms make use of a method for matching features between *views*, where we define a view to be a set of images captured by all cameras at the same time. The features within each set are often sparse and, as a result, it is difficult to find optimal matches within a high-dimensional feature space. Our approach is to use a brute-force feature matching algorithm, denoted as Φ throughout the remainder of the paper, which includes a mutual consistency check (i.e., no two features in one set may match the same feature in the other). The algorithm takes as input two sets of observations $\{z_i, z_j\}$ (respectively, two image views $\{f_i, f_j\}$), and outputs the matches between them, $\Phi(z_i, z_j)$ (respectively, $\Phi(f_i, f_j)$). For large feature sets, an optimized vocabulary tree provides fast matching (see § III-F).

C. Global localization

The global localization algorithm estimates the position of the robot in the graph $G = (V, E)$. We propose a general probabilistic approach to the problem that accounts for uncertainty in the robot's motion and in its visual observations. Specifically, the algorithm maintains a distribution over the robot's position in the graph, which we treat as the hidden state. The formulation models the motion of the robot within the graph as a first-order Markov process. The visual measurements z provide observations of the state. The localization algorithm maintains the position's distribution over time using a recursive Bayesian filter.

We model the robot's transitions between nodes in the graph $p(x_{k+1} \mid x_k)$ as a discrete Gauss window function $w(n) = \exp\left(-\frac{1}{2}(\frac{n}{N\cdot\sigma})^2\right)$ where n is the distance in graph space and N the width of the window. We define the observation model $p(z_k \mid x_k)$ using the Ψ distance introduced in § III-A as:

$$p(z_k \mid x_k) = 1/\Psi(z_k, z_{x_k}) \tag{1}$$

where z_{x_k} denotes the visual observations associated with node x_k. The intuition underlying this representation is that the probability for the robot to be at some graph location is directly related to the visual similarity of the current observations with the observations made at that node during the first visit.

Given the discrete distribution $p(x_k \mid z_k)$ at time k, the localization algorithm recursively estimates the new distribution at time $k+1$. In the prediction step, the filter first incorporates the effect of robot motion on the distribution $p(x_{k+1} \mid z_k)$ based upon a first-order Markov motion model $p(x_{k+1} \mid x_k)$. The update step then incorporates the observation model $p(z_{k+1} \mid x_{k+1})$ to obtain $p(x_{k+1} \mid z_{k+1})$ as:

$$p(x_{k+1} \mid z_k) = \sum p(x_{k+1} \mid x_k) \cdot p(x_k \mid z_k) \tag{2a}$$

$$p(x_{k+1} \mid z_{k+1}) = \lambda \cdot p(z_{k+1} \mid x_{k+1}) \cdot p(x_{k+1} \mid z_k) \tag{2b}$$

where λ is a normalization factor. The distribution is updated only on a local neighborhood around the current robot's position in the graph (i.e., the probability is zero elsewhere), which yields a constant time complexity for the algorithm.

D. Body-centered rotation guidance

We now present an algorithm that provides rotation guidance to the robot in its own body frame using only data from uncalibrated cameras. We show that using the *presence* of a feature in a camera as a measurement, rather than its precise position in the image, provides nearly the same navigation-relevant information when the number of observations is large.

The intuition underlying our method is as follows: if a feature corresponding to a static world point is observed on a camera at time t and re-observed on another camera at time t', the correspondence carries information about the relative orientation of the robot between time t and time t' (Figure 4). If the extrinsic and intrinsic calibration parameters are known, this is a trivial statement since the relative orientation may be deduced from basic geometric reasoning. However, we demonstrate a means of estimating the relative orientation with no prior intrinsic or extrinsic camera calibration.

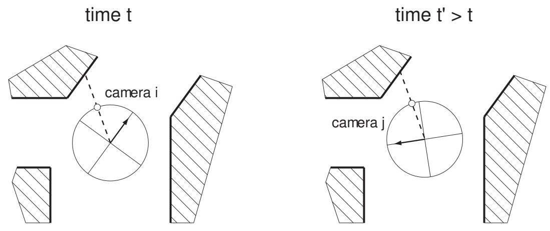

Fig. 4. We estimate the relative orientation of the robot between the first visit at time t and another visit at time t' using feature matches between the two views with no prior camera calibration.

Let us model the environment as a horizontal 2D plane (Z-axis up). We consider a set of n cameras rigidly mounted together observing the environment, each having the same field of view f. For each camera pair (i, j), we consider $p_{ij}(\alpha)$, the probability of re-observing a feature on camera j that was observed on camera i after a rotation of the cameras by an angle α. For instance, $p_{ii}(0) = 1$ for any i if we assume perfect feature matching. Similarly, $p_{ij}(0) = 0$ if the fields of view of camera i and camera j do not overlap and $i \neq j$. Assuming isotropic distribution of the features, each probability follows a triangular distribution $p_{ij} = \tau(\alpha; h_{ij}, \sigma_{ij})$ as illustrated in Figure 5. The angle h_{ij} is a function of the geometric configuration of the cameras with $h_{ii} = 0$ for any i. In addition, the limits of the triangular distribution are defined by the field of view common to all cameras $\sigma_{ij}^2 = \sigma^2 = f^2/6$ for any pair (i, j). We represent the set of distributions p_{ij} as an $n \times n$ matrix $H = [h_{ij}]_{i,j=1,\ldots,n}$ that we term the match matrix. We now demonstrate the process of determining the robot's relative orientation between time t and time t' using the match matrix and a set of feature correspondences. Given a match between a feature on camera i at time t and a feature on camera j at time t', we know that the relative orientation of the robot α follows the distribution p_{ij}, $\alpha \sim \tau(h_{ij}, \sigma)$. If we estimate α to be $\hat{\alpha} = h_{ij}$, the associated error is $\epsilon = \hat{\alpha} - \alpha \sim \tau(0, \sigma)$.

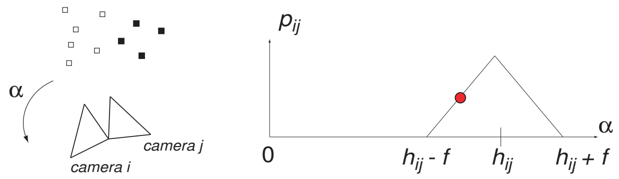

Fig. 5. We consider the probability $p_{ij}(\alpha)$ of re-observing a feature on camera j that was observed on camera i, after a rotation of the cameras by α. Assuming isotropic feature distribution, p_{ij} follows a triangular distribution whose limits are defined by the field of view of a single camera f.

Given N correspondences, the estimate becomes

$$\hat{\alpha} = \frac{1}{N} \sum_{1}^{N} h_{ij} \tag{3}$$

As per the Central Limit Theorem, under the assumption that the errors are independent and identically distributed, the error ϵ tends to a normal distribution as N increases,

$$\epsilon \sim \mathcal{N}(0, \sigma_N), \quad \sigma_N^2 = \frac{\sigma^2}{N}. \tag{4}$$

Hence, the error in the estimate of α is zero-mean and exhibits a variance that decreases inversely with the number of observations when N is large. For instance, a field of view $f = 90°$ yields $\sigma = 36.7°$ (for $N = 1$) and $\sigma_N = 2.6°$ (for $N = 200$).

Our method yields the following rotation guidance algorithm. Given two sets of image features z^t and $z^{t'}$ captured at the same location, we compute the feature matches $\Phi(z^t, z^{t'})$. For each match between a feature on camera i and a feature on camera j, we query the match matrix and obtain an estimate h_{ij}. We then estimate the relative orientation of the robot using Equation 3.

E. Learning the match matrix from training

The algorithm presented in the previous section relies on knowledge of the match matrix. Here, we describe a method that learns the match matrix from training. The training algorithm takes as input a video sequence captured while the robot rotates in place clockwise in an arbitrary environment and outputs the $h_{i,j}$ elements of the match matrix H. Outlined in Algorithm 1, the method proceeds as follows. Given any two views $\{F_p, F_q \mid p < q\}$ captured at times p and q and separated by a rotation of the robot by α_{pq}, the algorithm first computes the set of feature matches $\Phi(F_p, F_q)$, noting the start and end camera, $s_{k,p}$ and $s_{k,q}$, for each correspondence k. The algorithm then incorporates the estimate rotation α_{pq} to update the average estimate for each pair's element in the match matrix $H(s_{k,p}, s_{k,q})$. In order to take periodicity into account, the average $\hat{\eta}$ of a set of angles $\{\eta_i\}$ is derived from averaging rotation angles using the transformation from polar to Euclidean coordinates, i.e., $\bar{\eta} = \arctan(\sum \sin(\eta_i) / \sum \cos(\eta_i))$. We emphasize that the training method is fully automatic, is performed only once for a given camera configuration, and is independent of the training environment. The matrix H is therefore significantly easier to compute and more compact than the full set of intrinsic and extrinsic parameters would be.

We estimate the rotation angle based upon the assumption that the robot rotates in place at constant speed and performs r turns, which, over m views, yields the estimate $\alpha_{pq} = 2\pi r(q - p)/m$. While this assumption is not perfectly met in practice, we note that an error of δ degrees spread over the training sequence generates an error on α_{pq} that is linear in δ. Simulations reveal an error of $4.7°$ for $\delta = 20°$ and $m = 300$, which is acceptable for our application.

Algorithm 1 Match matrix training algorithm

```
Input: a training video sequence of $m$ views
Output: the $n \times n$ match matrix $H$ (for $n$ cameras)
Initialize $H(i,j) \leftarrow 0,\ 0 \le i,j < n$
Initialize $H_s(i,j) \leftarrow 0,\ 0 \le i,j < n$
Initialize $H_c(i,j) \leftarrow 0,\ 0 \le i,j < n$
for each pair of views $(F_p, F_q)$ in the sequence do
    Estimate the robot rotation angle $\alpha_{pq}$ linearly
    for each match $m_k = (f_{k,p}, f_{k,q}) \in \Phi(f_p, f_q)$ do
        Let $s_{k,p}$ $(s_{k,q})$ be the camera ID for $f_{k,p}$ $(f_{k,q})$
        $H_s(s_{k,p}, s_{k,q}) \leftarrow H_s(s_{k,p}, s_{k,q}) + \sin(\alpha_{pq})$
        $H_c(s_{k,p}, s_{k,q}) \leftarrow H_c(s_{k,p}, s_{k,q}) + \cos(\alpha_{pq})$
$H(i,j) \leftarrow \arctan(H_s(i,j)/H_c(i,j)),\ 0 \le i,j < n$
```

F. Loop closure using an online visual vocabulary

The ability to detect when the robot is returning to a place that it has previously visited (i.e., loop closure) is fundamental for autonomous navigation. We propose a method that detects loop closures automatically using only the visual appearance of the environment. Our method builds on the standard "bag-of-words" approach, in which features are represented by "words" in a visual dictionary [17]. Our method uses this vocabulary to efficiently compute the similarity between each pair of nodes in the graph. The data is stored in a *similarity matrix*. The algorithm then detects continuous sequences of highly similar nodes in the matrix. Our method requires no batch processing, no initial vocabulary, and takes as input only a stream of images.

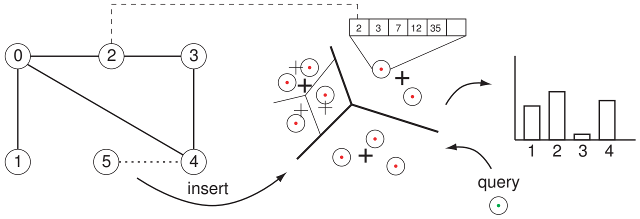

Fig. 6. The visual features of each new node in the place graph are stored in a vocabulary (*middle*). Each word stores an inverted index pointing to the nodes where it was observed. Search and query are optimized using a tree-based data structure.

We represent each word in the vocabulary as a sphere in the feature descriptor space, centered on the corresponding feature with a fixed radius. At the start of exploration, the vocabulary is empty. As nodes are inserted in the place graph, the corresponding visual features are inserted in the vocabulary. We declare a new word to match an existing word in the vocabulary when the L_2 distance between their centers does not exceed the fixed radius r_w. If no match is found, we instantiate a new word in the vocabulary. Each word stores an inverted index of the nodes in the place graph where it has been observed (Figure 6). The radius r_w is a parameter of the algorithm and influences the performance of the vocabulary.

A standard approach to optimize search and query in the vocabulary is to maintain a tree-based data structure [17]–[19]. A search within this structure is faster than a naive search as long as the number of examined nodes is bounded using a fast approximate search procedure. Alternatively, we also propose a method that optimizes the naive search when the feature descriptors are normalized. In this case, minimizing the L_2 distance between two features u and v is equivalent to maximizing their dot products, since $\|u - v\|^2 = \|u\|^2 + \|v\|^2 - 2 \cdot u \cdot v = 2 - 2 \cdot u \cdot v$. This approach is particularly powerful when multiple vocabulary queries are done at the same time, which is the case when a node is inserted in the graph.

We represent a vocabulary of n words as an $m \times n$ matrix W, where m is the dimension of the feature descriptor space. We represent a set of p input features as a $p \times m$ matrix F. The matrix $D = F \cdot W$ therefore contains the inner product of each input feature and each vocabulary word. A straightforward search through the matrix determines the closest word to each feature. We also apply this optimization to the exhaustive search used in the leaves in the tree-based method. We find that optimized linear algebra libraries enable the naive approach to outperform the tree-based method up to a certain vocabulary size, beyond which the tree-based method is faster (see § V-A).

The loop closure algorithm maintains a *similarity matrix* that contains the co-similarity between nodes. When a node is inserted in the graph, its features are searched in the vocabulary. A voting scheme then returns a similarity with all nodes currently in the graph (Figure 6). The method is causal and runs online during exploration.

Given a similarity matrix S, we identify sequences of visually similar graph nodes. We use a modified form of the Smith and Waterman algorithm [20], [21], which computes an *alignment matrix* A accumulating the score of diagonal moves through S. That is, $A(i, j)$ is the maximum similarity of two matching sequences ending at node i and node j. The algorithm then finds local maxima in the matrix and traces the corresponding sequence through A until the similarity falls below a given threshold. The algorithm is repeated on the matrix S with rows in the reverse order to detect alignments when the robot moves in the opposite direction.

A correspondence between a sequence of p nodes $\{v_{1,1}, \cdots, v_{1,p}\}$ and another sequence $\{v_{2,1}, \cdots, v_{2,p}\}$ means that nodes $v_{1,k}$ and $v_{2,k}$ ($1 \leq k \leq p$) correspond to the same physical location. We update the graph G accordingly. For each $k \in \{1, \cdots, p\}$, we connect all neighbors of $v_{1,k}$ to $v_{2,k}$ and remove the node $v_{1,k}$ from the graph. Additionally, $v_{2,k}$ replaces any reference to $v_{1,k}$ in the other node sequences. The number p is a parameter of the algorithm (we use $p = 5$).

IV. System Description

The vehicle used in this work is a small rover equipped with wheel encoders, a low-cost inertial measurement unit, an omnidirectional camera rig, and a planar laser range scanner. The rig is composed of four Point Grey Firefly MV cameras equipped with 2.8 mm Tamron lenses. The overall field of view of the rig is 360° horizontally and 80° vertically. The Hokuyo UTM LIDAR is used for obstacle avoidance, and to build maps for a metric validation of our approach. All algorithms run on an Intel quad-core laptop (2.5 GHz, 4 GB RAM) mounted on the robot.

A. Obstacle avoidance

Our algorithm provides high-level navigation instructions, and assumes that the robot can accept basic directional commands and avoid obstacles. Using the planar LIDAR for this task, the avoidance strategy assigns a cost metric to body-relative angles, penalizing angles that are far from the guidance direction and those in the direction of nearby obstacles. The robot then greedily drives in the direction of lowest cost. This simple reactive strategy has proven effective in practice, yet our navigation strategy is amenable to advanced methods based on sonar, infrared range sensors, touch sensors, and possibly even the uncalibrated cameras themselves [22], [23]. More accurate obstacle sensing will result in smoother robot trajectories, though the navigation algorithm is unaffected by the choice of obstacle avoidance method.

V. Validation and Results

A. Vocabulary Tree Evaluation

We compare the performance of the visual vocabulary for the naive method and the tree-based method [17] described in § III-F (Figure 7). We use the Intel Math Kernel Library for fast matrix multiplication in the naive approach. For the tree-based method, we use a tree branching factor $K = 10$. The method considers only $K_S \leq K$ children at every node. When $K_S = K$, the search is exhaustive and is less effective than the naive search due to the overhead cost of parsing the tree. However, for $K_S \leq 3$, the tree-based method outperforms the naive search beyond a vocabulary size of about 10^5 words (equivalent to roughly one hour of exploration with our system).

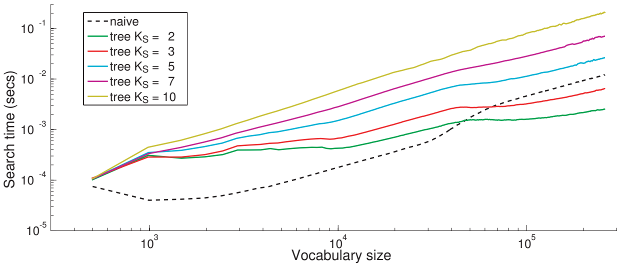

Fig. 7. Performance of the visual vocabulary for the naive method (dashed line) and the tree-based method (solid lines). The parameter K_S refers to the maximum number of children explored at every tree node (maximum is 10).

Dataset	Mission	Duration	Distance Traveled	Average Speed	# nodes	μ_K	μ_D	μ_G	μ_N	μ_R
LAB	Exploration	24 min	289 m	0.20 m/s	242					
	Mission A	18 min	99 m	0.09 m/s		3/3	0.16 m	0.25	0.43 m	13.1°
	Mission B	21 min	119 m	0.10 m/s		4/4	0.30 m	0.65	0.82 m	10.9°
	Mission C	35 min	160 m	0.08 m/s		3/3	0.31 m	0.61	0.78 m	11.7°
GALLERIA	Exploration	26 min	402 m	0.26 m/s	255					
	Mission D	29 min	171 m	0.09 m/s		3/3	1.19 m	1.02	2.20 m	17.5°

Fig. 8. Datasets.

B. Real-World Explorations

We demonstrate our algorithms on two datasets (Figure 8). The LAB dataset consists of a 24-minute exploration through various office spaces, followed by three missions (A, B and C) totaling 74 minutes. The GALLERIA dataset consists of a 26-minute exploration within a crowded mall-like environment followed by a 29-minute mission. In each mission, the robot is tasked with reaching a series of checkpoints within the graph. The missions often require the robot to traverse locations in the opposite direction of the first visit. Figure 9 shows the Ψ distance for a section of the GALLERIA dataset. Whenever the (smoothed) function exceeds a threshold, a node is added to the graph.

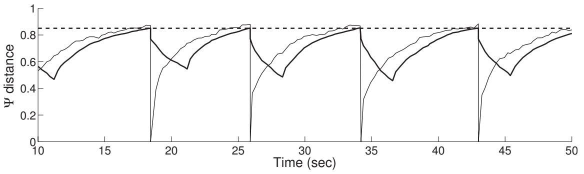

Fig. 9. The distance function Ψ during exploration for the GALLERIA dataset (smoothed function shown in bold). A node is added to the graph whenever the smoothed distance exceeds a threshold (we use 0.85).

C. Loop Closure Detection

Figures 10 and 11 illustrate the loop closure algorithm on the LAB dataset. Dark values in the similarity matrix indicate high similarity between nodes. The numbers associate the detected segments with their locations in the place graph. The method detects the three loop closure events effectively. We emphasize that our method runs online and does not build or require a metric map of the environment. We use metrical information only for ground-truth validation.

D. Body-Relative Rotation Guidance

We first evaluate the accuracy of the match matrix H obtained for the system described in § IV. Based upon the construction of the camera rig, which provides a ground truth separation of 90° between cameras, the error in the match matrix exhibits a standard deviation of 2.9°.

$$H = \begin{pmatrix} 0 & 88.8 & -175.8 & -84.5 \\ -88.8 & 0 & 93.3 & -175.4 \\ 175.8 & -93.3 & 0 & 89.0 \\ 84.5 & 175.4 & -89.0 & 0 \end{pmatrix}$$

We validate the rotation guidance algorithm using a sequence captured as the robot rotates in place for 30 seconds. For each pair of views in the sequence, we run the rotation guidance algorithm and compare its output with that of an on-board IMU exhibiting a drift rate of less than one degree per minute. Figure 12 shows the average rotation error with respect to the number of features per view (50,000 data points per trial). For large numbers of features, the standard deviation decreases roughly with the square root of the number of observations, as expected from Equation 4. We emphasize that the sequence was captured in an arbitrary environment that is different from the one used for training.

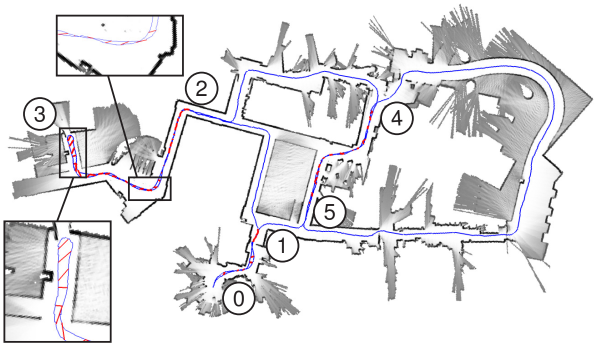

Fig. 10. Loop closure on a 24 minute exploration path across an office environment (LAB dataset). Loop closure detections are shown in red. Numbers refer to decision points in the place graph (Figure 11). We use the metric map and egomotion estimates for validation only.

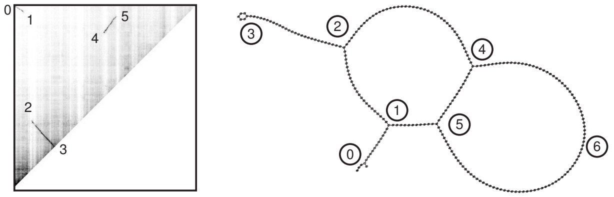

Fig. 11. Similarity matrix and place graph rendered using a spring-mass model (LAB dataset). Loop closure detections correspond to segments in the similarity matrix. The path of the robot was 0, 1, 2, 3, 2, 4, 5, 6, 4, 5, 1, 0.

Number of features	1408	1166	905	676	450	265	148	80	28	8
Rotation error (deg.)	2.0	2.3	2.2	2.4	2.8	3.2	3.9	6.4	20.0	53.7

Fig. 12. Rotation guidance error versus number of features.

We analyze the performance of the vision-guided navigation method using the ground truth vehicle trajectories, estimated with the publicly available GMapping [24] localization and mapping tool. The GMapping application provides a SLAM solution based upon a Rao-Blackwellized particle filter algorithm. The result is a maximum-likelihood estimate of the vehicle trajectory along with an occupancy grid map of the environment, as in Figure 10. For each of the two datasets, we first process the odometry and LIDAR data from the exploration phase to generate a reference map and an estimate of the ground truth exploration trajectory. We do the same for each of the navigation missions to resolve the robot's true trajectory within the corresponding map. We then align each navigation map with the mission's reference map in order to transform the ground truth navigation and corresponding exploration trajectories into a common reference frame.

The ground-truth localization of the robot provides several metrics to evaluate the navigation performance (Figure 13). First, we consider μ_K, the number of times the robot successfully reaches the target destination. In addition, we define μ_D as the distance between each point on the navigation path and the closest point on the exploration path. This metric measures the *reproducibility* of the navigation. We evaluate the precision of the local node estimation algorithm by defining μ_G as the distance in graph space between the location estimated by the robot and the true location. Similarly, we consider μ_N the metric distance between the current location of the robot and the physical location of the estimated node. Finally, we define the rotation guidance error μ_R as the difference between the body-centered rotation guidance and the direction from the current position of the robot to the next node in the path. Figure 8 summarizes the results.

μ_K	Successful arrival at destination	unitless
μ_D	Distance between exploration and navigation path	meters
μ_G	Place graph error	unitless
μ_N	Distance to estimated node	meters
μ_R	Rotation guidance error	degrees

Fig. 13. Evaluation metrics.

Figure 14 shows the evolution of μ_D over time for mission C. The average distance to the original path is 0.30 m and reaches about one meter at several locations along the mission. These variations can be explained by the low reaction time of the controller, yielding slightly off-path trajectories in tight turns. Figure 15 illustrates the error (in graph space) of the node estimation algorithm. Overall, the algorithm performs well and localizes the robot with a worst-case accuracy of three nodes. Similarly, the distance to the estimated node is bounded over time and exhibits an average of 0.42 m (Figure 16). Finally, the rotation guidance error average is nearly zero with a typical standard deviation of $12°$ (Figure 17). At time $t = 450$ sec, we observe a peak in the rotation error due to motion blur in the images which resulted in poor feature detection.

The GALLERIA dataset (Figure 19) is of particular interest. For this dataset, we purposely explored the environment during a very low-activity period of the week, while executing the navigation mission at rush hour. Consequently, many passers-by interfered with the robot's path during the mission. Despite the significant variation and busy conditions, the robot was able to recover from severe off-path trajectories and successfully reached its destination (Figure 1).

E. System Performance

We run all algorithms on an Intel quad-core computer (2.5 GHz, 4 GB RAM). Each camera produces 376×240 grayscale 8-bit images at 30 Hz. The computation of the SIFT features runs at 6 Hz for an average of 150 features per camera (each of the four cameras maps to one processor). A loop of the place graph generation runs in 100 msecs with 600 features per view. Node creation takes negligible time, so the place graph generation algorithms runs overall at 10 Hz. During navigation, the global localization algorithm runs at 1.5 Hz using a graph search radius of 5. The rotation guidance algorithm runs at 5 Hz. All algorithms run in parallel, so the system provides directional guidance at 5 Hz while localization updates occur at 1.5 Hz.

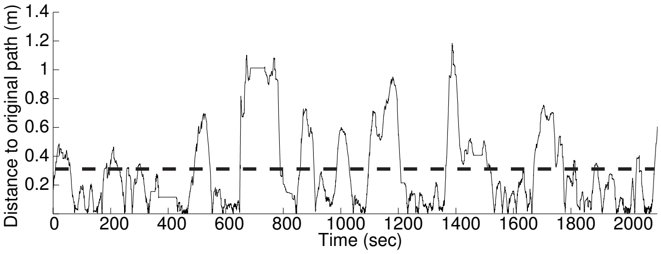

Fig. 14. Distance to original path (LAB dataset, Mission C). Mean value shown as a dotted line.

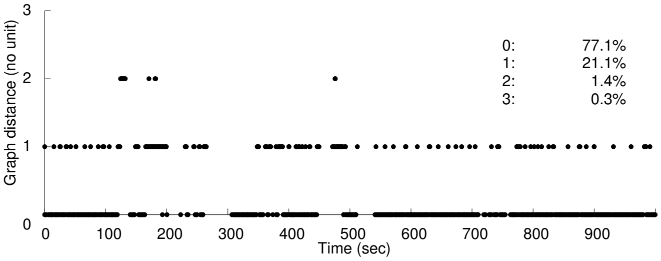

Fig. 15. Distance in graph space between the estimated node and the correct node (LAB dataset, Mission A).

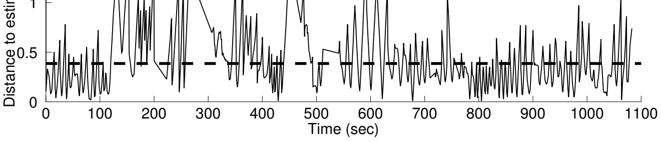

Fig. 16. Distance to the estimated node (LAB dataset, Mission A).

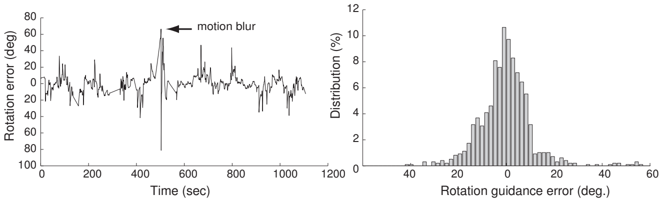

Fig. 17. Rotation guidance error with respect to the direction to the next node (LAB dataset, Mission A). Time lapse (*left*) and histogram (*right*).

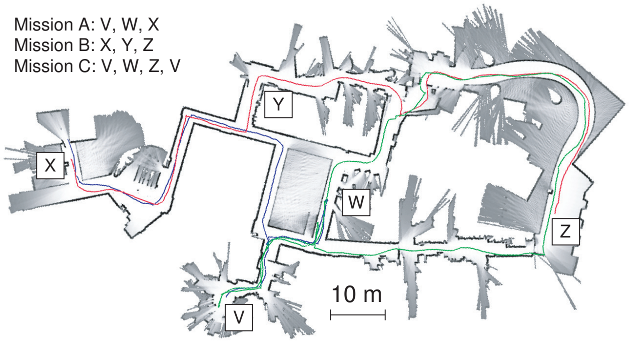

Fig. 18. LAB dataset: ground-truth paths followed by the robot during mission A (*blue*), B (*red*) and C (*green*).

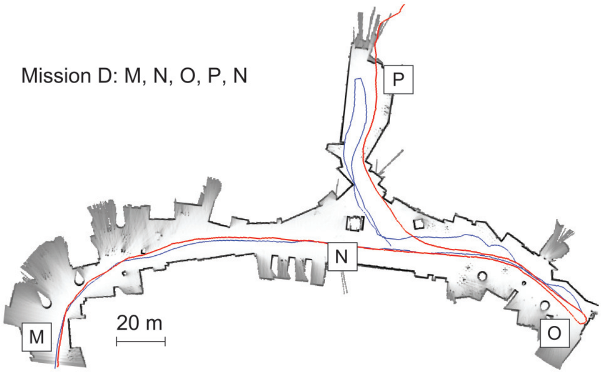

Fig. 19. GALLERIA dataset: exploration path (*red*) and revisit path (*blue*). During the second half of the mission, many passers-by interfere with the robot's path. Yet, the robot reaches its destination successfully.

VI. Conclusion

We described a vision-based navigation method for a ground robot in unknown environments. On the first excursion through the environment, the method builds a topological representation of the robot's path. Upon revisitation, the method localizes the robot in the graph and provides coarse, yet robust navigation guidance in the robot's body frame. Our method assumes no prior intrinsic or extrinsic camera calibration. We demonstrated our system on 2.5 hours and 1,200 m of exploration through real, dynamic environments.

Acknowledgments

We are grateful to Abe Bachrach for his assistance with ground-truth validation.

This work was sponsored by the Draper Laboratory University Research and Development program.

References

[1] O. Koch and S. Teller, "Body-relative navigation guidance using uncalibrated cameras," in *Proc. IEEE Int. Conf. on Computer Vision (ICCV)*, Kyoto, Japan, September 2009.

[2] J. Sola, A. Monin, M. Devy, and T. Lemaire, "Undelayed initialization in bearing only SLAM," in *Proc. IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, Edmonton, Alberta, August 2005, pp. 2499–2504.

[3] T. Lemaire, S. Lacroix, and J. Sola, "A practical 3D bearing-only SLAM algorithm," in *Proc. IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, Edmonton, Alberta, August 2005, pp. 2449–2454.

[4] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proc. IEEE Comp. Soc. Conf. Computer Vision and Pattern Recognition (CVPR)*, vol. 1, Washington, DC, June-July 2004, pp. 652–659.

[5] L. M. Paz, P. Piniés, J. Tardós, and J. Neira, "Large scale 6-DOF SLAM with stereo-in-hand," *IEEE Trans. on Robotics*, vol. 24, no. 5, pp. 946–957, October 2008.

[6] A. Davison, "Real-time simultaneous localisation and mapping with a single camera," *Proc. IEEE Int. Conf. on Computer Vision (ICCV)*, vol. 2, pp. 1403–1410, 2003.

[7] G. Klein and D. W. Murray, "Improving the agility of keyframe-based SLAM," in *Proc. IEEE Eur. Conf. on Computer Vision (ECCV)*, Marseille, October 2008, pp. 802–815.

[8] J. Modayil, P. Beeson, and B. Kuipers, "Using the topological skeleton for scalable global metrical map-building," *Proc. IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, vol. 2, no. 28, pp. 1530–1536, September 2004.

[9] S. Thrun, "Learning metric-topological maps maps for indoor mobile robot navigation," *Artificial Intelligence*, vol. 99, no. 1, pp. 21–71, 1998.

[10] S. J. Maybank and O. D. Faugeras, "A theory of self-calibration of a moving camera," *Int. J. Computer Vision*, vol. 8, no. 2, pp. 123–151, August 1992.

[11] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

[12] M. Cummins and P. Newman, "FAB-MAP: Probabilistic localization and mapping in the space of appearance," *Int. J. Robotics Research*, vol. 27, no. 6, pp. 647–665, 2008.

[13] A. Angeli, S. Doncieux, J.-A. Meyer, and D. Filliat, "Real-time visual loop-closure detection," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, Pasadena, CA, May 2008, pp. 1842–1847.

[14] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proc. IEEE Int. Symp. on Computational Intelligence in Robotics and Automation (CIRA)*, Monterey, CA, July 1997, pp. 146–151.

[15] P. Whaite and F. Ferrie, "Autonomous exploration: Driven by uncertainty," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 19, no. 3, pp. 193–205, 1997.

[16] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[17] A. Angeli, D. Filliat, S. Doncieux, and J.-A. Meyer, "A fast and incremental method for loop-closure detection using bags of visual words," *IEEE Trans. On Robotics*, vol. 24, no. 5, pp. 1027–1037, 2008.

[18] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *Proc. IEEE Comp. Soc. Conf. Computer Vision and Pattern Recognition (CVPR)*, Washington, DC, 2006, pp. 2161–2168.

[19] T. Yeh, J. Lee, and T. Darrell, "Adaptive vocabulary forests br dynamic indexing and category learning," in *Proc. IEEE Int. Conf. on Computer Vision (ICCV)*, Rio de Janeiro, October 2007, pp. 1–8.

[20] T. Smith and M. Waterman, "Identification of common molecular sequences," *J. of Molecular Biology*, vol. 147, pp. 195–197, 1981.

[21] K. Ho and P. Newman, "Detecting loop closure with scene sequences," *Int. J. Computer Vision*, vol. 74, no. 3, pp. 261–286, September 2007.

[22] J. Santos-Victor and G. Sandini, "Uncalibrated obstacle detection using normal flow," *Machine Vision and Applications*, vol. 9, no. 3, pp. 130–137, 1996.

[23] B. Horn, Y. Fang, and I. Masaki, "Time to contact relative to a planar surface," in *IEEE Intelligent Vehicles Symposium*, Istanbul, June 2007, pp. 68–74.

[24] C. Stachniss and G. Grisetti, "GMapping project at OpenSLAM.org." [Online]. Available: http://www.openslam.org/gmapping.html