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 that a traditional camera calibration is not required for the vision-based navigation problem, and present an algorithm for guiding a robot through a previously visited environment using a set of uncalibrated cameras mounted on the robot.**

On the first visit to 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 destination point. We combine this with a reactive avoidance algorithm to obtain a system able to navigate the robot safely and reliable through the environment. We validate our approach with ground-truth experiments and demonstrate the method on a small ground rover navigating through several dynamic scenes.

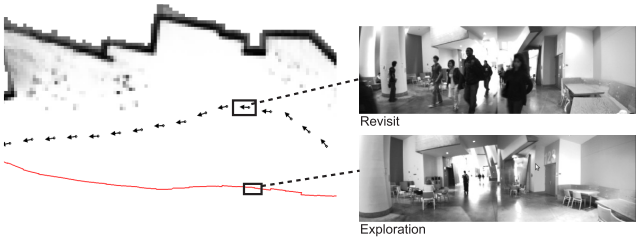

Fig. 1. We present a vision-based method for the navigation of a ground robot. Assuming that the robot has previously explored the environment (*red path*), the method provides guidance in the body frame of the robot during revisit (*black arrows*). we demonstrate the robustness of our method on real-world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

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. By enabling the projection of points on the image plane into rays in the robot body frame, these parameters allow 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.

Unfortunately, obtaining these 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 where the calibration tools and equipment are not available.

Robots operating in real-world conditions may often be bumped, damaged, repaired, or otherwise altered in such a way that would invalidate a previously acquired calibration. Robots may often be disassembled for storage or transport, and parts may shift, however slightly, during the reassembly process. Especially in the case of deployments of large numbers of robots, executing the calibration procedure may quickly become a challenging and difficult endeavor. A natural question to ask is then: what can be accomplished without a 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.

The topic considered is 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 regions 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 operating within such an environment. Specifically, we describe and demonstrate an algorithm that uses multiple cameras to give directional commands to 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 basic reactive obstacle avoidance capabilities.

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 [2], and extensions to a full 3D environment have been demonstrated. Many of these techniques have been successfully applied to visual navigation, where precise camera calibration is assumed. Visual odometry [3] and visual

SLAM [4]–[6] 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 space [7], [8].

All of this work, range- or vision-based, requires precise calibration of sensor intrinsic and extrinsic parameters. In the absence of executing a specialized camera calibration procedure, one approach is to attempt an automatic calibration during field operation [9], [10]. Self-calibration methods, as they are called, 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 radial distortion lens effects. Recovering extrinsic parameters, specifically the camera-to-robot transformation, is not feasible using these techniques.

A key challenge for a navigation algorithm is being able 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 then use a set of observed words to represent a place, have improved the speed and robustness of visual loop closing [11], [12].

Our approach is motivated by simplicity in practice. It makes no assumption on 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 trivially simple, makes no additional assumptions on the environment, 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 which we call the *place graph*. The place graph records visual information only. In our experiments, this step is performed manually. However, methods exist that could automate this phase [13], [14].

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. Next, a low-level obstacle avoidance algorithm controls the robot to move in the desired direction while reactively avoiding local obstacles. Figure 2 illustrates our method.

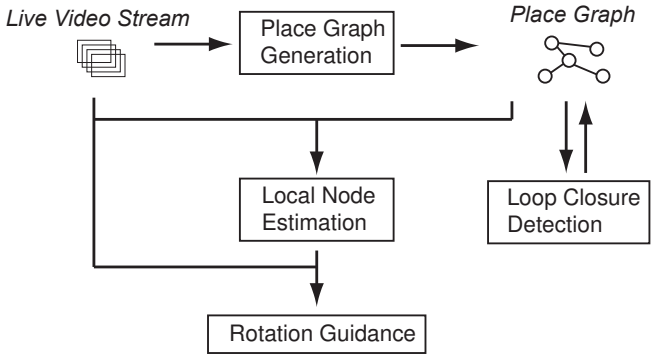

Fig. 2. Method overview. During exploration, the system builds a place graph representing the robot's exploration path. During revisit, the method uses the graph to localize the robot and provide guidance in the body frame of the robot.

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 during exploration $\{z_{k,j} \mid (x_k, x_j) \in E\}$. 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.

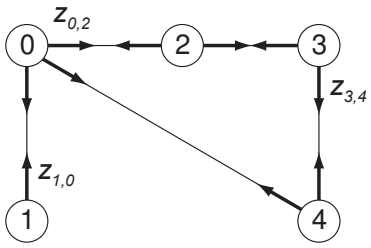

Fig. 3. We represent the robot's exploration path as an undirected graph where nodes represent locations in the world and edges represent physical paths between nodes. Visual observations (e.g. SIFT features) are associated with each node.

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. When the visual appearance of the environment differs sufficiently from the previous node, as indicated by the variation in visual features, a new node is instantiated. More specifically, we measure variation in appearance based upon a distance function Ψ that matches sets of features between two images, and returns the mean L_2 distance between matches in the feature descriptor space. The feature matching utilizes a function Φ that we describe in Section III-B. When the distance Ψ between the current observations and those of the last node exceed a threshold (we have empirically found a threshold of 0.8 to be a robust

measure of variability), we instantiate a new node in the graph.

B. Feature Matching

Features are matched between *frames*, where we define a *frame* 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, combined with 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 $\{z_i, z_j\}$ (or, alternatively, two image frames $\{f_i, f_j\}$), 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). No algorithm in our method involves continuous frame-to-frame matching.

C. Global localization

Given a graph $G = (V, E)$, the global localization algorithm determines the position of the robot in the graph. We propose a general probabilistic approach to the problem and model the motion of the robot in the graph as an unobserved Markov process. The observed states of the model are the visual observations z. The hidden state is the discrete distribution function of the robot over the graph. A recursive Bayesian algorithm determines the distribution function recursively over time.

Given the distribution $p(x_k \mid z_k)$ at time k, the recursive loop estimates the new distribution at time $k+1$. First, the prediction step estimates the distribution $p(x_{k+1} \mid z_k)$ given a motion model for the robot (we use a Gaussian model). Then, the update step incorporates the observation model $p(z_{k+1}|x_{k+1})$ to obtain $p(x_{k+1} \mid z_{k+1})$.

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

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

where λ is a normalization factor. In practice 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. We incorporate the motion continuity assumption by defining the motion model $p(x_{k+1} \mid x_k)$ as a Gaussian distribution $N(0, \sigma)$. In addition, define the observation model $p(z_k \mid x_k)$ using the Ψ distance introduced in III-A:

$$p(z_k \mid x_k) = 1/\Psi(z_k, z_{x_k}) \tag{3}$$

where z_{x_k} represents the visual observations associated to node x_k. The intuition underlying this choice is that the probability for the robot to be at a location in the graph is directly related to the visual similarity with the observations made at that node during the first visit.

D. Body-centered rotation guidance

We now present an algorithm that provides rotation guidance to the robot in its own body frame using only a set of uncalibrated cameras. We show that using the presence of a feature in a camera as a measurement rather than the precision position of the feature on the image removes little navigation-relevant information when the number of observations is large.

The algorithm takes as input the current location of the robot in the graph x_k and the current observations z_k. The output of the algorithm is the angle γ representing the orientation of the robot *relatively to* its orientation when the node x_k was created (Figure 4). Let us model

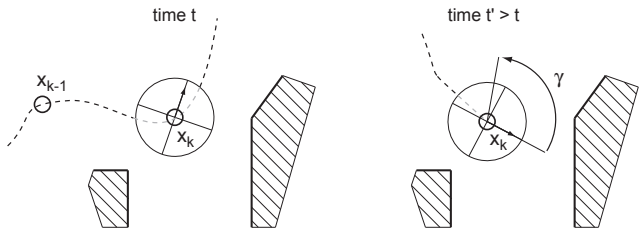

Fig. 4. When revisiting a node x_k in the graph at time t', the rotation guidance algorithm estimates the orientation of the robot *relative to* its orientation when the node was created at time $t < t'$.

the environment as a horizontal 2D plane (Z-axis up). We consider each camera as a bearing-only sensor that measures the azimuth of each observation in the body frame of the robot. Thus, the observations z_k are represented by q bearing angles $z_k = \{\beta_1, \cdots, \beta_q\}$ and the observations z_{x_k} by p bearing angles $z_{x_k} = \{\beta_1, \cdots, \beta_p\}$. If the extrinsic and intrinsic camera parameters are known, the bearing angles are directly observable and the angle γ is estimated by $\gamma = \frac{1}{n} \sum_{0 \leq i < p} \beta_j - \alpha_i$, where β_j matches α_i. Let us now assume that the intrinsic parameters are not known. The bearing angles $\{\alpha_i\}$ and $\{\beta_i\}$ are not observable anymore. However, we can define a set of coarser measurements $\{\hat{\alpha}_i\}_{0 \leq i < p}$ as follows: for any observation α by a given camera, let $\hat{\alpha}$ be a constant defined as the *average* of all possible observations made by this camera. In other words, $\hat{\alpha}$ completely disregards the *location* of the detection on the image and depends only on the presence of the feature and the camera's extrinsic parameters (Figure 5). By analogy, we

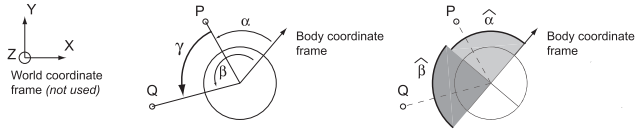

Fig. 5. Calibrated case (*left*): two world features P and Q yield two observable bearing measurements α and β. Their relative orientation is $\gamma = \beta - \alpha$. Uncalibrated case: (*right*): α and β are not observable but can be replaced by coarse estimates $\hat{\alpha}$ and $\hat{\beta}$. When the number n of observations is large, the error on the estimate of γ decreases with $\sqrt{n}$.

define $\hat{\gamma}$ the estimate of γ as $\hat{\gamma} = \hat{\beta} - \hat{\alpha}$. Let us assume that the observations are uniformly distributed in image space. The errors $\delta_\alpha = \hat{\alpha} - \alpha$ and $\delta_\beta = \hat{\beta} - \beta$ are therefore

uniformly distributed between $-f/2$ and $f/2$, where f is the horizontal field of view of each camera. That is $\delta_\alpha = U(-f/2, f/2)$ and $\delta_\beta = U(-f/2, f/2)$. The associated variance is $\sigma_f^2 = f^2/12$. When the number of observations is sufficiently large, the Central Limit Theorem states that the average of the measurements errors δ_α and δ_β is uniformly distributed, with a standard deviation $\sigma_{\alpha,\beta} = \sigma_f/\sqrt{n}$. As a consequence, the average of the standard errors δ_γ is also normally distributed with a standard deviation $\sigma_\gamma = 2\sigma_{\alpha,\beta}$. For instance, $f = 90°$ and $n = 200$ yields $\sigma_\gamma \sim 3.5°$. As a conclusion, we can obtain a reasonable estimate of the relative orientation of the robot while discarding the position of the features in the image.

We propose an algorithm that estimates the relative orientation of the robot using a simple voting scheme. Given two sets of images features z_k and z_m captured at the same location, we compute the feature matches $\Phi(z_k, z_m)$. Each feature match *votes* for a relative orientation of the robot. The value of the vote is the relative orientation of the two cameras observing the features. In particular, the vote is 0 if the two features originate from the same camera. This algorithm requires to know the relative orientation of each camera pair. We represent this information as an $m \times m$ matrix H (for m cameras) which we refer to as the *match matrix*. Since the extrinsic camera parameters are unknown, the matrix H is unknown. We propose a method that learns H from training.

E. Learning the match matrix from training

We present here 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 n_r times in an arbitrary environment (we use $n_r = 2$). The algorithm pseudo-code is given on below. Given the number of frames in the video sequence f and the rotation angle of the user α_{pq} between any two frames $\{f_p, f_q \mid p < q\}$, the algorithm computes the set of feature matches $\Phi(f_p, f_q)$. For each match m_k, we denote as $s_{k,p}$ and $s_{k,q}$ the start and end camera identifier of the match. The algorithm updates the average in cell $H(s_{k,p}, s_{k,q})$ with the angle α_{pq}. In order to take periodicity into account, the average $\bar{\eta}$ of m angles $\{\eta_1, \cdots, \eta_m\}$ 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 automated, 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 extrinsic parameters would be.

In practice, we assume that the robot rotates in place at constant speed and performs exactly n_{rot} turns, which yields the following linear expression: $\alpha_{pq} = 2\pi n_{rot}(q-p)/f$. This assumption is of course not perfectly realized in reality. However, an error of δ degrees spread over the training sequence generates an error on α_{pq} that is linear in δ. Simulations show an error of $4.7°$ for $\delta = 20°$ and $f = 300$, which is acceptable for our application.

The training algorithm

Input: a training video sequence of f frames
Input: the number of cameras m
Output: the $m \times m$ match matrix H

Initialize $H(i,j) \leftarrow 0$, $0 \le i,j < m$
Initialize $H_s(i,j) \leftarrow 0$, $0 \le i,j < m$
Initialize $H_c(i,j) \leftarrow 0$, $0 \le i,j < m$
for each pair of frames (F_p, F_q) in the sequence **do**
  Estimate the user rotation angle α_{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 $f_{k,p}$ ($f_{k,q}$)
    $H_s(s_{k,p}, s_{k,q}) \leftarrow H_s(s_{k,p}, s_{k,q}) + \sin(\alpha_{pq})$
    $H_c(s_{k,p}, s_{k,q}) \leftarrow H_c(s_{k,p}, s_{k,q}) + \cos(\alpha_{pq})$
$H(i,j) \leftarrow \arctan(H_s(i,j)/H_c(i,j))$, $0 \le i,j < m$

F. Loop closure using an online visual vocabulary

The ability to detect that the robot is visiting a previously seen place is fundamental for an autonomous navigation system. We propose a method that allows to detect loop closure automatically using the visual appearance of the environment only. Our method builds on the standard "bags-of-words" approach, in which features are represented by "words" in a visual dictionary. Our method uses this vocabulary to compute the similarity between each pair of nodes in the graph efficiently. The data is stored in a *similarity matrix*. The algorithm then detects continuous sequences of highly similar nodes in the matrix and updates the place graph accordingly. Our method requires no bach processing, no initial vocabulary and takes only as input a stream of images.

A word in the vocabulary is represented by a sphere in the feature descriptor space. At the beginning of an exploration, the vocabulary is empty. As nodes are inserted in the graph, the corresponding visual features are inserted in the vocabulary. Given a fixed word radius r_w, we declare a word match as a word which center lies within r_w of the input feature descriptor using the L_2 distance. If no match was found, a new word centered on the feature descriptor is created in the vocabulary. Each word stores an inverted index of the nodes in the place graph where it has been observed. The radius r_w is a parameter of the algorithm and influences the performance of the vocabulary (Figure 6). A standard

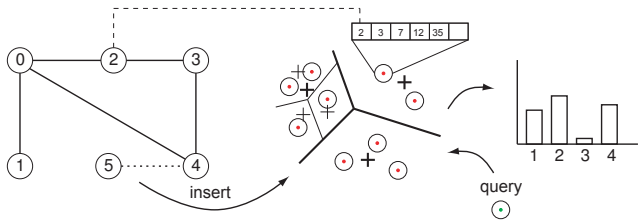

Fig. 6. The visual features of each new node in the place graph are stored in a vocabulary (*middle*). Each word stores an inverted index pointing to the nodes where it was observed. Search and query are optimized using a k-d tree data structure.

approach to optimize search and query in the vocabulary is to maintain a k-d tree data structure [15]–[17]. Searching for words in these structure is faster than a naive search as long as the number of examined nodes is bounded using a fast approximate search procedure. However, the naive search method can be optimized and yield a better performance than the tree-based method. Indeed, when the feature descriptors are normalized, minimizing the L_2 distance between two features u and v is equivalent to maximizing their dot products, since $\mid u - v \mid^2 = \mid u \mid^2 + \mid v \mid^2 -2 \cdot u \cdot v = 2 - 2 \cdot u \cdot v$. This approach is particularly powerful when multiple queries to the vocabulary 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 V, where m is the dimension of the feature descriptor space. We represent a set of p input features as an $p \times m$ matrix F. The matrix $D = F \cdot V$ therefore contains the dot product between each input feature and each word in the vocabulary. A straightforward search through the matrix determines the closest word to each feature. We found that optimized linear algebra libraries enabled the naive approach to outperform the k-d tree method independently of the size of the vocabulary (see § V-A). We note that the naive approach also has the advantage to provide exact results.

The loop closure algorithm maintains a *similarity matrix* that contains the co-similarity between nodes. To this end, each word in the vocabulary maintains a list of the nodes where it has been observed. 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). We emphasize that our method is causal and runs online during exploration.

Given a similarity matrix S, we wish to identify sequences of visually similar graph nodes. We use a modified form of the Smith and Waterman algorithm [18] described in [19], 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 user 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 $v_{2,k}$ correspond to the same physical location ($1 \leq k < p$). 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 $p = 5$).

To summarize, we combine a fast and fully incremental "bags-of-words" technique with detector of similar node sequences to provide an online loop closure detection capability.

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 $90°$ 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, 4GB 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 has some form of obstacle avoidance. We use a planar LIDAR for this task, and the strategy consists of assigning a cost metric to body-relative angles, where angles far from the target direction and angles leading to nearby obstacles are penalized. The robot greedily drives in the direction of lowest cost. This is a simple reactive strategy, and its functionality could be provided by a number of other solutions such as sonar, infrared range sensors, touch sensors, and possibly even the uncalibrated cameras themselves [20], [21]. More accurate obstacle sensing will result in smoother robot trajectories, though the navigation algorithm itself is unaffected by the choice of obstacle avoidance method.

V. Validation and Results

A. Vocabulary Tree Evaluation

We study the performance of the visual vocabulary for the naive method and the k-d tree method [15] described in § III-F. For the k-d tree method, we use a tree branching factor $K = 10$. The method consists in considering only K_S children at every node ($0 < K_S \leq K$). If $K_S = K$, the search is exhaustive and becomes less effective than the naive search due to the overhead cost of parsing the tree. In practice however, the naive method is significantly faster that the k-d tree method for any value of K_S (Figure 7). We attribute this result to the outstanding performance of the Intel Math Kernel Library for matrix multiplication.

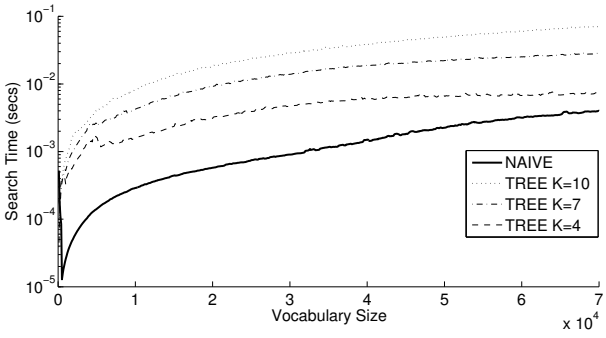

Fig. 7. Performance of the visual vocabulary for the naive method (solid line) and the k-d tree method (dotted lines). The parameter K_S refers to the maximum number of children explored at every node of the tree (maximum is 10). The naive method is significantly faster and provides exact results.

B. Loop Closure Detection

Figures 8 and 9 illustrate the loop closure algorithm on the LAB dataset. Dark values in the similarity matrix correspond to high similarity between nodes. Detected segments are marked by numbers which correspond to decision points in the place graph. The method detects the three loop closure events effectively. Our method runs fully online and only takes as input a live video stream coming from uncalibrated cameras. We emphasize that our method does not build or require a metric map of the environment. We only use this information for ground-truth validation.

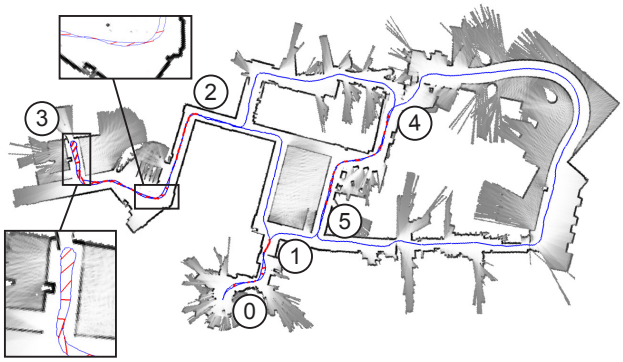

Fig. 8. Loop closure on a 30 minute exploration path across an office environment (LAB dataset). Loop closure detections are shown in red. Numbers refer to decision points in the place graph (Figure 9). We use the metric map for validation only.

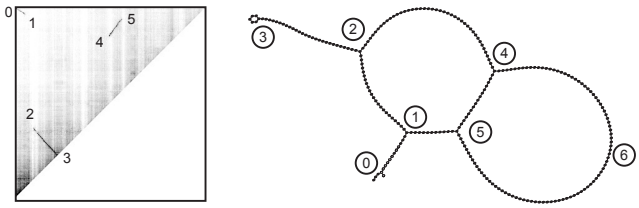

Fig. 9. Similarity matrix and place graph rendered using a spring-mass model (LAB dataset). Loop closure detections correspond to segments in the similarity matrix. The path of the robot was 0, 1, 2, 3, 2, 4, 5, 6, 4, 5, 1, 0.

C. Body-Relative Rotation Guidance

We show below the match matrix H obtained for the system described in § IV. Since we know by construction of the system that each camera makes an angle of 90° with its neighbors, we can compare H with a ground-truth match matrix and obtain 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 frames in the sequence, we run the rotation guidance algorithm and compare its output with that of an IMU mounted on the robot. The IMU has a drift rate of less than one degree per minute. We obtain a standard deviation of 2.5° and a worst-case error of 8°. We emphasize that the sequence was captured in an arbitrary environment that is different from the one used for training.

D. Real-World Explorations

We demonstrate our algorithms on two datasets (Figure 10). The LAB dataset consists of a 24-min exploration across office spaces, followed by three missions (A, B and C) totaling 80 minutes. The GALLERIA dataset consists of a 26-min exploration within a crowded mall-like environment followed by a 24-min mission. In each mission, the robot is tasked to reach a series of checkpoints within the graph. The missions often require the robot to traverse locations in the opposite direction of the first visit (Figures 16 and 17).

We analyze the performance of the vision-guided navigation based upon the ground truth vehicle trajectories that we estimate using the publicly available GMapping [22] localization and mapping tool. The GMapping application provides a a Simultaneous Localization and Mapping (SLAM) solution based upon a Rao-Blackwellized particle filter algorithm. The result is a maximum-likelihood estimate for the vehicle trajectory along with an occupancy grid map of the environment, such as that shown in Figure 8. For each of the two datasets, we first process the odometry and LIDAR data from the exploration phase to generate a reference map of the LAB and GALLERIA environments along with an estimate for the ground truth exploration trajectory. We do the same for each of the navigation missions to resolve the robot's true trajectory through the environment and 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 (Table 11). First, we consider μ_K, the number of times the robot successfully reaches the target destination. This metric provides a high-level evaluation of the method. 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. However, we also 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 μ_R the rotation guidance error 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 10 summarizes the results.

Figure 12 shows the evolution of μ_D over time for mission C. The distance to the original path is in average $0.30\,m$ and reaches about one meter at several locations along the mission. We explain these variations by the low reaction time of the controller, which may yield slightly off-path trajectories in tight turns. Figure 13 illustrates the error

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. 10. Datasets.

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. 11. Evaluation metrics.

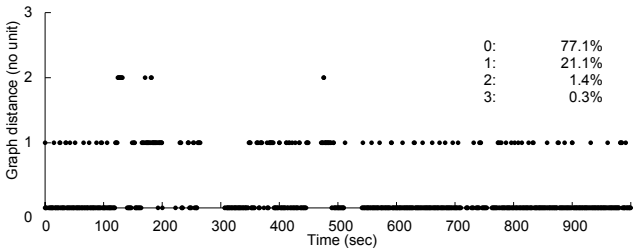

Fig. 13. Distance in graph space between the estimated node and the correct node (LAB dataset, Mission A).

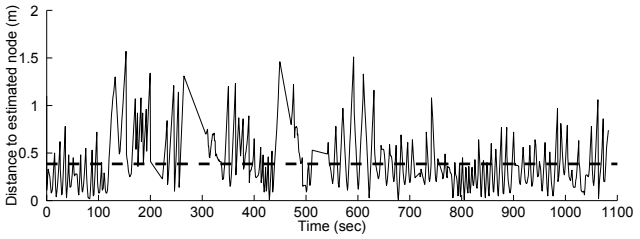

Fig. 14. Distance to the estimated node (LAB dataset, Mission A).

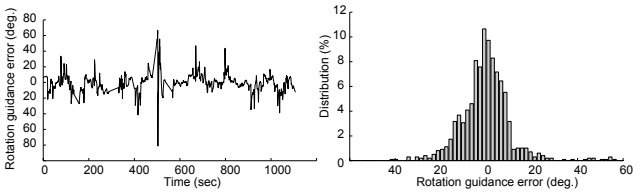

Fig. 15. Rotation guidance error with respect to the direction pointing to the next node (LAB dataset, Mission A). Time lapse (*left*) and histogram (*right*).

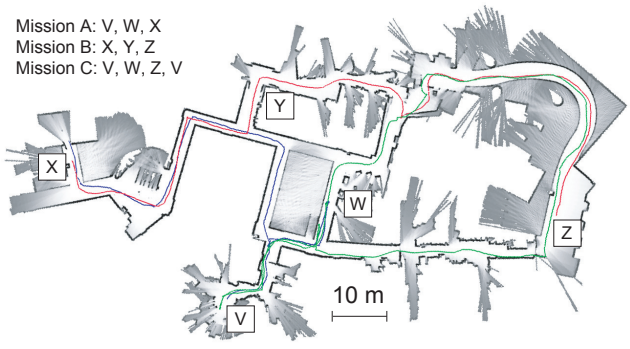

Fig. 16. LAB dataset: ground-truth paths followed by the robot during mission A (*blue*), B (*red*) and C (*green*).

(in graph space) of the node estimation algorithm. Overall, the algorithm performs well and localizes the robot with a worst-case accuracy of two nodes. Similarly, the distance to the estimated node is also bounded with an average of $0.42\,m$ (Figure 14). Finally, the rotation guidance error averages around zero with a typical standard deviation of 12° (Figure 15). At time $t = 450\,sec$, we observe a peak in the rotation error. This peak was due to an artifact in the controller which generated a very fast instantaneous rotation of the robot. The rotation speed was higher than the rotation guidance algorithm update speed, which generated a temporary outstanding error.

The GALLERIA dataset (Figure 17) 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. As a consequence, an outstanding number of passers-by interfered with the robot's path during the mission. Despite these drastic conditions, the robot was able to recover from severe off-path trajectories and reached its destination successfully (Figure 1).

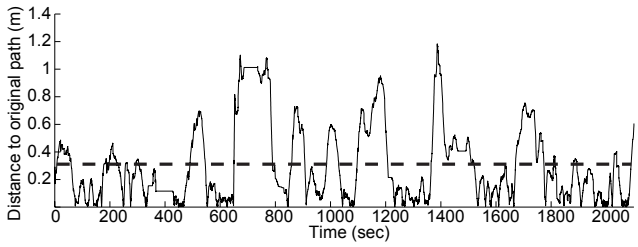

Fig. 12. Distance to original path (LAB dataset, Mission C). Mean value shown in dotted line.

VI. Conclusion

We presented a vision-based navigation method for a ground robot in unknown environments. Given a place graph representation of the robot's exploration path during the first visit, the method localizes the robot in the graph and provides coarse, yet robust navigation guidance in the body frame

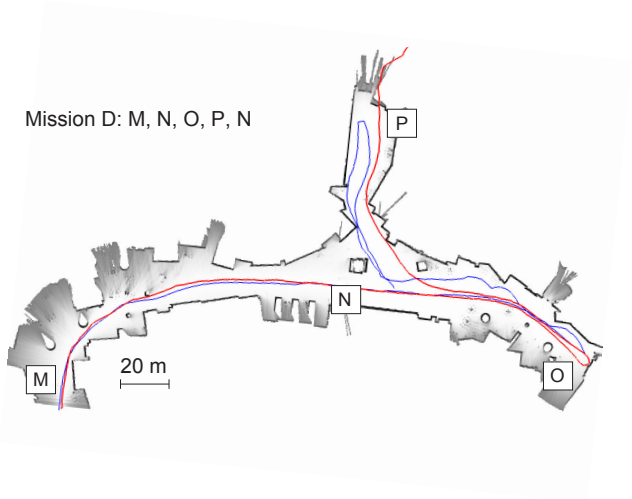

Fig. 17. GALLERIA dataset: exploration path (*red*) and revisit path (*blue*). During the second half of the mission, a high number of passers-by interfere with the robot's path. Yet, the robot reaches its destination successfully.

of the robot. Our method makes few assumptions about the environment except that it contains descriptive visual features. We presented a guidance algorithm based on a set of uncalibrated cameras and demonstrated our system on more than two hours of exploration through real, dynamic environments.

ACKNOWLEDGMENTS

We thank Abe Bachrach for his support on the ground-truth validation.

REFERENCES

[1] O. Koch and S. Teller, "Body-relative navigation guidance using uncalibrated cameras," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Kyoto, Japan, September 2009, to appear.

[2] E. Olson, J. Leonard, and S. Teller, "Fast iterative optimization of pose graphs with poor initial estimates," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2006, pp. 2262–2269.

[3] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," *Computer Vision and Pattern Recognition*, vol. 1, pp. 652–659, 2004.

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

[5] A. Davison, "Real-time simultaneous localisation and mapping with a single camera," vol. 2, pp. 1403–1410, 2003.

[6] G. Klein and D. W. Murray, "Improving the agility of keyframe-based slam," in *ECCV (2)*, 2008, pp. 802–815.

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

[8] S. Thrun, D. Fox, and W. Burgard, "A probabilistic approach to concurrent mapping and localization for mobile robots," *Machine Learning*, vol. 31, pp. 29–53, 1998.

[9] 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, Aug. 1992.

[10] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521623049, 2001.

[11] 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. [Online]. Available: http://ijr.sagepub.com/cgi/content/abstract/27/6/647

[12] A. Angeli, S. Doncieux, J.-A. Meyer, and D. Filliat, "Real-time visual loop-closure detection," in *Int. Conf. Robotics and Automation*, May 2008, pp. 1842–1847.

[13] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *CIRA '97: Proceedings of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation*. Washington, DC, USA: IEEE Computer Society, 1997, p. 146.

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

[15] 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 Transactions On Robotics, Special Issue on Visual SLAM*, pp. 1027–1037, 2008.

[16] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 2161–2168.

[17] T. Yeh, J. Lee, and T. Darrell, "Adaptive vocabulary forests br dynamic indexing and category learning," in *Proceedings of the IEEE 11th International Conference on Computer Vision (ICCV)*, Oct. 2007, pp. 1–8.

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

[19] K. L. Ho and P. Newman, "Detecting loop closure with scene sequences," *International Journal of Computer Vision*, vol. 74, no. 3, pp. 261–286, September 2007.

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

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

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