Body-Relative Navigation Guidance Using Uncalibrated Cameras

Olivier Koch
koch@csail.mit.edu

Seth Teller
teller@csail.mit.edu

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology

Abstract

We present a vision-based method that assists human navigation within unfamiliar environments. Our main contribution is a novel algorithm that learns the correlation between user egomotion and feature matches on a wearable set of uncalibrated cameras. The primary advantage of this method is that it provides robust guidance cues in the user's body frame, and is tolerant to small changes in the camera configuration. We couple this method with a topological mapping algorithm that provides global localization within the traversed environment. We validate our approach with ground-truth experiments and demonstrate the method on several real-world datasets spanning two kilometers of indoor and outdoor walking excursions.

1. Introduction and background

Navigation is a fundamental task for humans that can be difficult when the environment is unfamiliar or visually complex. We consider the problem of human navigation in unknown environments when deprived of external sources of location, such as GPS, the sun or a compass. We propose a vision-based solution that relies only upon a wearable set of uncalibrated cameras. Our approach applies, in particular, to situations in which the user does not have the time or the ability to learn the topological or metrical structure of the environment.

Simultaneous Localization and Mapping (SLAM) is a natural approach to navigation. Recent research in this field has proved successful for robotic navigation in unknown environments [7, 17]. However, human navigation possesses distinctive characteristics. First, humans do not require precise geometric directions to reach their destination. Second, humans tend to maintain topological, rather than metrical, representations of their environment [19] – this aspect has inspired recent research in robotics [15, 23]. Finally, general human motion can be challenging for current structure-from-motion algorithms.

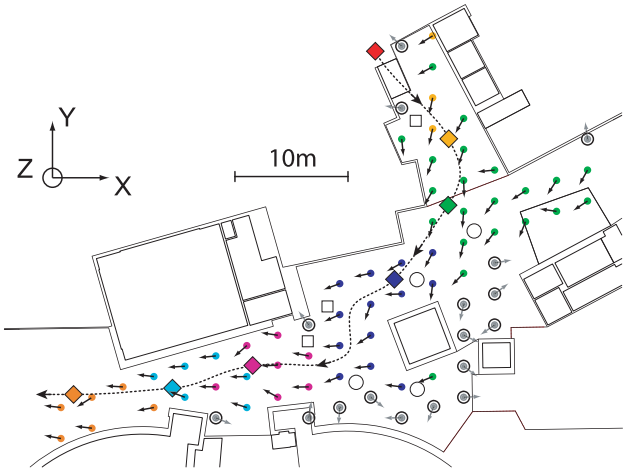

Figure 1. Excerpt of the GALLERIA dataset. The dotted line represents the notional path followed by the user during the first visit. Colored squares represent nodes in the topological map. Arrows represent the exact (not notional) guidance provided to the user upon revisit in a *replay* scenario. Black circles denote failures due to occlusion by building structure.

We propose a novel way to assist human navigation. The method takes as input a live video stream captured while the user moves through an unknown environment. During exploration, the system provides the user with several capabilities that are critical to navigation: *homing* (going from the current location to the starting point of the exploration), *replay* (retracing the exploration path from the starting point) and *point-to-point navigation* (going from any previously visited place to any other). The method may be useful for those with visual or cognitive impairments, or to police or military personnel moving within GPS-denied environments.

We approach the problem from a topological, non-metrical perspective. Our method builds a topological map of the environment online during exploration, and uses it to localize and guide the user. Specifically, we introduce an algorithm that learns the correlation between user egomotion and feature correspondence across cameras to provide rotation guidance in the *body frame* of the user.

2. Related work and contributions

Recent work [7, 12, 14, 16, 17] in visual SLAM demonstrates robust and accurate localization and mapping in unknown environments. However, the majority of these methods are targeted at robotics applications. Structure-from-motion algorithms are often sensitive to degenerate camera motions (*e.g.* rotation about the center of projection) and may be subject to uncontrolled map growth. As a consequence, methods that scale to large environments have been proposed based on topological maps [1, 15, 23]. Purely appearance-based localization has also been investigated [6].

Other methods, known as *teach and replay*, consist of recording the motion of image features during a training phase. During replay, an optimal path is followed that reproduces image-space feature motion. Early work in this area [4] has inspired recent research [3, 22]. Omnidirectional sensors have been widely used [10, 11, 21] for such approaches, as they provide a large field of view and fewer constraints on camera heading. Vision has also been used to build feature-based representations of the environment for augmented reality applications [8].

Our work is distinct from the methods described above in several respects. First, it does not compute a metrical reconstruction of the environment, but rather a topological representation of the user's path. Second, it is applicable to an arbitrary number of uncalibrated cameras, with little constraint on their relative positions on the user. Finally, it provides coarse (rather than precise) navigation guidance in the user's body frame, rather than in a global or externally-referenced frame.

Our method does not provide the accuracy of metrical SLAM. However, it has the advantage of providing robust navigation from uncalibrated cameras across large and dynamic environments, which makes it ideal for body-worn applications. We demonstrate our method on extended real-world datasets spanning more than 2 km of indoor and outdoor walking excursions (§ 4.6).

3. Method overview

Our approach relies on a topological representation of the user's path through the world. An undirected graph $G = (V, E)$ represents the explored environment, where nodes N represent places and edges E represent physical paths between places. We refer to a topological map as a *place graph*, described in detail in § 3.1. Given the place graph representation, we cast the navigation problem as a "node-to-node hopping" problem (Figure 2). One sub-method, *local node estimation* determines the location of the user within the graph (§ 3.2). Another sub-method, *rotation guidance*, provides a directional indication to the user at each node (§ 3.4). Our approach provides no guidance along edges and assumes that there is no ambiguity on the direction to follow between nodes. Finally, a *loop closure* sub-method detects return visits to previously traversed places, and updates the graph accordingly (§ 3.5).

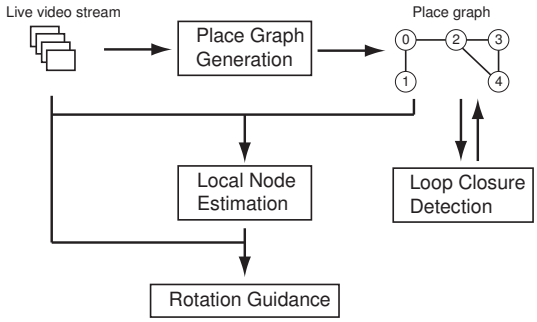

Figure 2. Method Overview. Our method takes as input a live video stream captured from a set of body-worn, uncalibrated cameras. It generates a topological representation of the explored environment (*place graph*) and estimates the location of the user in the graph (*local node estimation*). *Rotation guidance* provides body-relative navigation guidance at each node. *Loop closure detection* updates the graph when the user revisits a place. All modules run online and in parallel during the user excursion.

3.1. The place graph

The place graph represents the user's excursion as an undirected graph $G = (V, E)$. A node $v \in V$ consists of a set of visual observations $\mathcal{O}_v$. We use SIFT [13] features with 128-byte descriptors. An edge $e \in E$ represents a path followed by the user between two adjacent nodes. No observations are associated with graph edges.

At the start of exploration, the graph G is empty. Whenever a node is created, it is added to the graph and connected to the most recent existing node. In the absence of loop closure detection, the graph is a simple chain. In § 3.5, we describe how to update the graph due to loop closure events. Because a node may have several neighbors, the set $\mathcal{O}_v$ is in fact a set of observations $\mathcal{O}_v = \{\mathcal{O}_{v,v_k} \mid (v, v_k) \in E\}$, where $\mathcal{O}_{v,v_k}$ represents the observations made at node v on the way to v_k. Figure 3 illustrates the place graph data structure.

To determine when a node should be created, we define a distance function Ψ for two sets of input observations. This function matches features between the two sets and returns the averaged sum-of-square distances between matches using the L1 distance in the feature descriptor space. We use the Φ matching function described in § 3.3. A new node is created when the Ψ-distance between the current observations and the observations of the latest node exceeds an experimentally-determined threshold (we use 0.3).

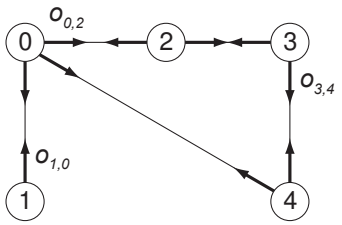

Figure 3. We represent the world as an undirected graph where nodes represent physical locations and edges represent physical paths between locations. Each node is associated with the observations made en route to and from each of its neighbors.

3.2. Local node estimation

The goal of local node estimation is to maintain an estimate of the user's position within the graph. Given $v_i^t \in V$, the position of the user in G at time t, we wish to determine v_j^{t+1}, *i.e.* the position of the user at time $t+1$. Assuming that user motion is continuous, we employ a discrete local search that takes as input the current observations $\mathcal{O}^{t+1}$ and searches for the node v that minimizes $\Psi(\mathcal{O}^{t+1}, \mathcal{O}_v)$ for all nodes v in the neighborhood of v_i^t. This search spans only a local neighborhood of v_i^t and is therefore independent of the graph size.

3.3. Feature Matching

Feature matching is difficult to optimize for small sets in a high-dimensional space. We use a brute-force feature matching algorithm, denoted Φ throughout this paper, combined with a mutual consistency check (*i.e.* no two features in one set may match the same feature in the other). We define a frame as a set of images captured by all cameras at the same time. The algorithm takes as input two sets of observations $\{\mathcal{O}_i, \mathcal{O}_j\}$ (or alternatively, two frames $\{F_i, F_j\}$) and outputs the matches between them $\Phi(\mathcal{O}_i, \mathcal{O}_j)$ (respectively $\Phi(F_i, F_j)$). For large feature sets, an optimized vocabulary tree provides fast matching (see § 3.5.1). No algorithm in our method involves continuous frame-to-frame matching.

3.4. Body-relative rotation guidance

The purpose of body-relative rotation guidance is to guide the user's direction of travel. We develop our method in this section and state the fundamental assumptions it relies upon. We show that using the *presence* of a feature in a camera as a measurement, rather than the feature's precise image-space location, removes little navigation-relevant information when the number of observations is large.

We model the world as a horizontal 2D plane (Z-axis up), and each camera as a bearing-only sensor that measures the azimuth of each observation in the user's body frame. We represent a set of world feature observations as a set of q bearing angles $\{\alpha_i\}_{0 \le i < q}$. We assume that, when revisiting the same location at a later time, a subset of the same features is observed with bearing angles $\{\beta_i\}_{0 \le i < p \le q}$ and that the matching function Φ provides data association between the two sets. We use these two sets of observations to determine the relative user's orientation γ of the user with respect to the orientation during the first visit (Figure 4). If

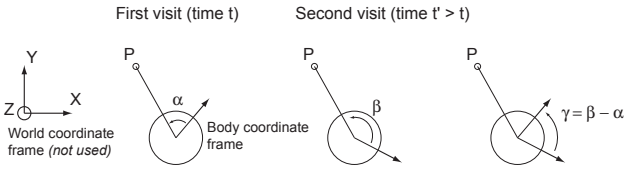

Figure 4. At time t, a feature in the world (P) is observed by the system with bearing angle α. At time $t' > t$, the feature is re-observed with bearing angle β. The angle $\gamma = \beta - \alpha$ defines the relative orientation of the user between time t and time t'.

the intrinsic and extrinsic camera parameters are known, the bearing measurements are directly available and the relative orientation is defined as $\gamma = \frac{1}{n} \sum_{0 \le i < p} \beta_j - \alpha_i$, where β_j matches α_i. This corresponds to minimizing the average bearing error with respect to the original user orientation (using L1 distance). Let us now assume that the intrinsic camera parameters are not known. The measurements $\{\alpha_i\}_{0 \le i < p}$ are no longer observable. However, we can define a set of coarser measurements $\{\hat{\alpha}_i\}_{0 \le i < p}$ 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. Note that, in contrast with α which is a continuous variable, $\hat{\alpha}$ is a discrete variable with n possible values (for n cameras). The intuition behind our method is

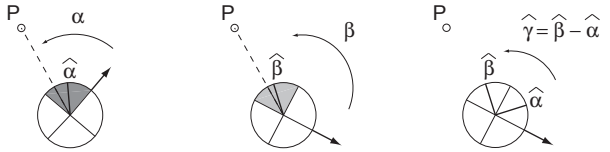

Figure 5. With uncalibrated cameras (represented here as the four quadrants of a circle), the bearing measurements (α, β) are not observable. They become discrete measurements ($\hat{\alpha}$, $\hat{\beta}$) which are constant within each camera. Even though the variances of $\hat{\alpha} - \alpha$ and $\hat{\beta} - \beta$ are high, with a sufficient number of observations the variance of $\hat{\gamma} - \gamma$ will tend to zero.

that using the coarser measurement $\hat{\alpha}$, instead of the measurement α itself, yields a statistically valid estimate of γ while not requiring intrinsic calibration of the cameras (Figure 5). Assuming that the set of possible observations is uniformly distributed in image space $\alpha = U(0, 2\pi)$, the error between $\hat{\alpha}$ and α is a uniformly distributed variable $\delta = \hat{\alpha} - \alpha \sim U(a, b)$. In the configuration shown on Figure 5, $a = -\pi/4$ and $b = \pi/4$. The associated vari-

ance is $\sigma_\delta^2 = (b-a)^2/12$, *i.e.* $\sigma_\delta \sim 26°$. Given a sufficiently large number p of bearing estimates $\{\hat{\alpha}_i\}_{0 \leq i < p}$, the central limit theorem (CLT) states that the averaged sum of $\{\delta_i\}_{0 \leq i < p}$ is normally distributed, with a standard deviation $\sigma = \sigma_\delta / \sqrt{p}$ (*i.e.* $\sigma = 2.6°$ for $p = 100$). By additivity, the error $\epsilon = \hat{\gamma} - \gamma$ is normally distributed as well, with a standard deviation $\sigma_\epsilon = 2\sigma$. Here, our approach assumes that several hundred visual features are observable at all times during exploration, which is reasonable for an omnidirectional camera system in many environments.

Given two discrete observations $\hat{\alpha}$ and $\hat{\beta}$, the variable $\hat{\gamma} = \hat{\beta} - \hat{\alpha}$ is discrete as well. We choose to represent $\hat{\gamma}$ as a matrix H of size $n \times n$ (for n cameras) which we call the *match matrix*. The value $H(i,j)$ represents the rotation angle associated to a match between a feature in camera i and a feature in camera j. The matrix H_0 corresponding to the ideal system shown in Figure 5 is given below (in degrees). In this example, a match between camera 0 and camera 1 yields a rotation angle of $H(1,0) = -\pi/2$.

$$H = \begin{pmatrix} 0 & - & - & - \\ -90 & 0 & - & - \\ 180 & -90 & 0 & - \\ 90 & 180 & -90 & 0 \end{pmatrix}$$

The matrix H is related to the extrinsic parameters of the cameras but is more compact. It has two interesting properties. First, it is antisymmetric by definition. Second, it satisfies the circular equality of Equation (1) below. We use these properties to validate the actual match matrix obtained with our system (§ 4). Also, our approach assumes a common vantage point between observations of the same location. However, by considering only the presence of a feature in an image, our method introduces strong robustness to parallax effects (§ 4.6). Our method is by nature robust to slight changes in the camera positions or to minor variations in the intrinsic calibration of the cameras.

$$\sum_{0 \leq i < n} H(i, (i+1) \mod n) = 0 \mod 2\pi \qquad (1)$$

3.4.1 Learning the match matrix from training

The match matrix must be determined when the extrinsic parameters of the cameras are unknown. We use a method that learns H from training. The training algorithm takes as input a video sequence captured while the user rotates in place clockwise n_{rot} times in an arbitrary environment (we use $n_{rot} = 2$). 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 user 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 n
Output: the $n \times n$ match matrix H

Initialize $H(i,j) \leftarrow 0, 0 \leq i,j < n$
Initialize $H_s(i,j) \leftarrow 0, 0 \leq i,j < n$
Initialize $H_c(i,j) \leftarrow 0, 0 \leq i,j < n$
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 $f_{k,p}$ ($f_{k,q}$)
    $H_s(s_{k,p}, s_{k,q}) \leftarrow H_s(s_{k,p}, s_{k,q}) + \sin(\alpha_{pq})$
    $H_c(s_{k,p}, s_{k,q}) \leftarrow H_c(s_{k,p}, s_{k,q}) + \cos(\alpha_{pq})$
$H(i,j) \leftarrow \arctan(H_s(i,j)/H_c(i,j)), 0 \leq i,j < n$

3.4.2 Rotation guidance using the match matrix

Let us consider the problem of guiding the user from node v_i to one of its neighbors v_j (Figure 6). The rotation guidance algorithm takes as input $\mathcal{O}_{v_i v_j}$, the observation made at node v_i on the way to v_j during the first visit and $\mathcal{O}^t$, the current set of observations. The algorithm computes the matches $\Phi(\mathcal{O}^t, \mathcal{O}_{v_i v_j})$. Each feature match m_k between a feature on camera $s_{k,p}$ and a feature on camera $s_{k,q}$ votes for a rotation angle $H(s_{k,p}, s_{k,q})$. The average $\bar{\alpha}$ of all votes represents the amount of rotation that the user should execute in the body coordinate frame in order to face in the direction of v_j.

We now consider the problem of guiding the user from a current location $v_i \in V$ to a any desired location $v_k \in V$. The algorithm uses Dijkstra's algorithm to determine the shortest topological path $\{v_i, \cdots, v_k\}$ between v_i and v_k and guides the user from node to node along the path. The penalty along an edge is a linear function of the elapsed time required for the first traversal of each edge.

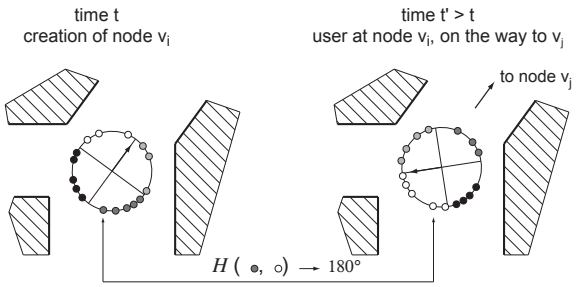

Figure 6. The rotation guidance algorithm matches observations between the first visit of a node (*left*) and the current visit (*right*). Observations are shaded by camera ID. Each match votes for a rotation that contributes to bringing the user in alignment with the orientation of the first visit. The arrow represents the forward direction in the user's body frame.

3.5. Loop closure detection

Loop closure detection, *i.e*. recognizing that the user has returned to a previously seen location, is a fundamental capability for navigation. The loop closure detection method discussed in this section operates online and works by detecting sequences of nodes with similar appearances. In a first stage, the method builds a *similarity matrix* between nodes in the graph using an incremental "bag of words" algorithm [5]. In a second stage, sequences of visually similar nodes are extracted from the similarity matrix.

3.5.1 Scene similarity using a dynamic vocabulary tree

Our approach to scene similarity relies on the popular "bag of words" approach [5], in which each image is considered to be a document composed of "visual words". A word $w = \{c_w, r_w\}$ in the vocabulary is a sphere in the feature space (in our case, 128-dimensional SIFT descriptors) centered on c_w with radius r_w. The method maintains an incremental vocabulary tree [9]. At the beginning of the exploration, the vocabulary is empty. Each time a node is added to the graph, its associated features are added to the vocabulary. Each time a new feature is added, either it belongs to an existing word, or a new word is created. The radius of a word r_w is constant and influences the vocabulary size and the performance of the recognition. We implement the vocabulary as a tree, where each leaf stores words along with the list of node indices where the word was observed. Searching the tree is optimized using a fast approximate search procedure in which only the closest tree nodes to the input feature are considered.

For each node v_i added to the graph, we compute the similarity between v_i and all nodes $\{v_j\}_{j<i}$. Each word found in v_i votes for node indices using the normalized inverse document frequency [18]. The output of the algorithm is a similarity matrix S which elements $S_{i,j}$ correspond to the similarity between node v_i and node v_j (Figure 7).

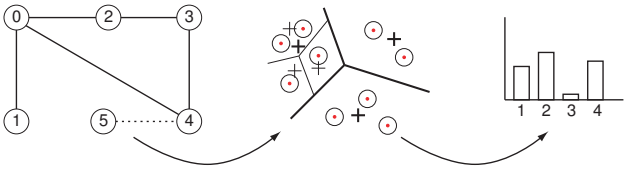

Figure 7. Each time a node is added to the graph (*left*), its associated features are compared to the vocabulary (*middle*) to compute the similarity with all nodes seen so far. The features are then inserted into the vocabulary. Circles with a red dot represent words. Black crosses represent the vocabulary tree nodes.

3.5.2 Extracting visually similar sequences

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 [20] described in [12], 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 $p = 5$). Figures 12(b) and 14(b) show a typical output of the algorithm.

4. Validation and Results

4.1. System description

We tested the algorithms described in this paper on a wearable set of four Point Grey Firefly MV cameras loosely mounted on the shoulder straps of a backpack (Figure 8). The cameras' aggregate field of view is 360° horizontally and about 90° vertically. For the sole purpose of establishing ground truth (see § 4.6), a fifth camera was mounted oriented upward on the backpack. The system includes a tablet PC interface allowing the user to interact with the system, mark places in the pose graph for the purpose of validation, and request guidance to a target node in the graph.

4.2. Match matrix

We show below the match matrix H for our implementation, obtained using a one-minute training sequence cap-

Figure 8. Our system includes four uncalibrated cameras loosely mounted on the shoulder straps of a backpack. The horizontal field of view is 360°. The system includes a laptop and can operate for three hours on a battery charge.

tured in an arbitrary indoor environment. Note that we do not expect H to match H_0 since our system has a slightly different camera configuration. However, the properties of the match matrix described in § 3.4.1 provide error metrics for the matrix. The anti-symmetry property, measured on all entries of the matrix, yields an error of 14.5°, while the circular equality (Equation (1)) yields an error of 1.5°. We attribute this error to the fact that the user may not rotate at exactly constant speed during training and to parallax between cameras. However, some level of error is inherent to the system since the cameras are free to move slightly during use.

$$H = \begin{pmatrix} -19.9 & 91.3 & -164.7 & -66.9 \\ -101.8 & -11.9 & 101.4 & -151.5 \\ 155.1 & -95.9 & -16.2 & 105.9 \\ 59.9 & 164.1 & -93.4 & -6.7 \end{pmatrix}$$

4.3. Rotation guidance validation

We validated the rotation guidance algorithm by rigidly mounting an Inertial Measurement Unit (IMU) on the system, with a drift rate of less than one degree per minute. We captured a sequence for 30 seconds while the user rotates in place in an arbitrary environment (different from that used for training). We then ran the rotation guidance algorithm between each pair of frames in the sequence, and compared the output of the algorithm with the output of the IMU. The standard deviation of error in the rotation guidance algorithm is 8.0°, with a worst-case error of 16°. This level of error would be acceptable to a human in most cases. However, situations exist where the user could be misled (*e.g.* at the start of two nearly-parallel corridors).

4.4. Large-scale rotation ground truth

In order to obtain ground truth for rotation guidance all along the exploration path, we equipped the backpack with a high-resolution camera looking upward. We propose a rotation baseline algorithm that takes as input two images captured by this camera (one at the first visit of the node, one upon revisit) and outputs the rotation angle that brings them into alignment. This angle defines "ground truth" for the rotation guidance algorithm. It is important to note that not all places along the exploration path provide distinctive visual features on the ceiling. Therefore, the rotation baseline algorithm alone would not be a viable option for navigation.

Given two input images I_p and I_q, the algorithm computes SIFT feature matches between them $\Phi(I_p, I_q)$. Each feature descriptor implies a primary orientation (Figure 9). For each match $m = (f_p, f_q)$, the algorithm computes the difference between the orientation of f_p and f_q and averages this value over all feature matches (after outlier rejection). The output is the angle of the rotation that best aligns I_p onto I_q. We found this method to perform robustly across our datasets.

We validated the rotation baseline algorithm using a sequence containing 30 images captured by the ground truth camera as the user rotates in place. We ran the rotation baseline algorithm for each pair of frames in the sequence and compared its output to that of the IMU. We obtained a standard deviation of 1.9°. We conclude that the rotation baseline algorithm provides robust ground truth to the rotation guidance algorithm. Figure 10(a) compares the output of the rotation guidance algorithm against ground truth obtained from the rotation baseline algorithm for 200 data points. The standard deviation is 10.5°.

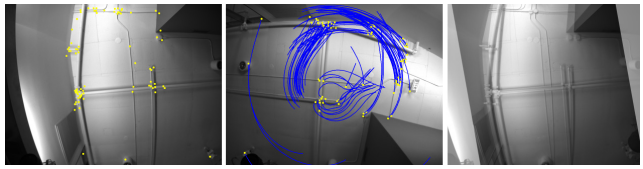

Figure 9. Rotation baseline (LAB DATASET). *Left*: first image I_p. *Middle*: second image I_q. *Right*: alignment of I_p onto I_q. SIFT features are shown in yellow. Blue lines represent feature matches. The algorithm estimates the rotation between the two images to within 2°.

4.5. Local node estimation

To obtain "ground truth" for local node estimation, we selected checkpoints along the exploration path and marked them with fiducials on the floor. During exploration, the user purposely passed by each checkpoint multiple times, and used the user interface to insert a timestamp in the captured video sequence. We then compared the output of the local node estimation with ground truth (Figure 10(b)). We observed that the node estimation was correct at all checkpoints along the path, *i.e.* that it robustly estimated the user's position within the graph.

4.6. Real-world explorations

We demonstrate our method on three datasets described in Table 11. The MEZZANINE dataset consists of a 10-minute exploration in a typical laboratory environment (replay scenario). The GALLERIA dataset consists of a 15-minute long exploration in a mall-type environment com-

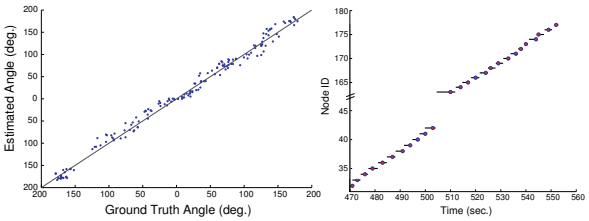

Figure 10. *Left*: Rotation guidance against rotation baseline for 200 checkpoints (GALLERIA and LAB datasets). The solid line represents the identity. The standard deviation is $10.5°$. *Right*: Output of the local node estimation (CORRIDORS dataset). Horizontal lines represent the node estimate. Red dots represent ground truth. At time $t = 505$, the user enters a new branch of the place graph, hence the discontinuity in node ID (point #2 on Figure 14).

posed of large open-spaces (homing scenario). The CORRIDORS dataset consists of a 30 minute-long exploration in a network of narrow corridors (point-to-point scenario). All datasets involve cluttered and dynamic scenes (including passers-by).

Name	Scenario	Duration	Length	# frames	# nodes	# checkpoints
MEZZANINE	replay	10 min.	400m	6,000	91	36
GALLERIA	homing	15 min.	700m	9,000	154	150
CORRIDORS	point-to-point	30 min.	1,500m	18,000	197	0

Figure 11. Exploration datasets.

In each case, the user first explores the (unknown) environment, then requests guidance for one of the scenarios. The system outputs visual guidance as shown on Figure 13 (forward-facing cameras on the top row, backward-facing cameras on the bottom row). The red arrow shows the actual (not notional) guidance provided to the user. In addition, rotation guidance is converted into audio cues (*e.g.* "turn left", "go straight") uttered to the user. We emphasize that except for the training algorithm, which is run only once, all algorithms take as input a live video stream and require no batch processing.

Figure 1 illustrates the robustness of the rotation guidance algorithm to off-path trajectories in the GALLERIA dataset. Colored squares represent nodes in the place graph. Colored dots represent test positions at which the user was standing facing always the same direction (Y-axis). The resolution of the grid is three meters. Black arrows indicate the actual (not notional) direction guidance given to the user. Colors indicate the identification of the node in the map output by the local node estimation. As can be seen, the algorithm usually provides robust guidance even when the user is more than three meters away from the original path. Locations marked by a black circle indicate places where either the node identification or the rotation guidance failed. We explain these failures by the strong occlusion due to building structure at these locations. Figure 14(a) shows the exploration path for the CORRIDORS dataset overlaid on a 2D map as well as the corresponding topological map output by our method. This dataset resulted in relatively few nodes, due to numerous loop closure events. Figure 12(a) shows intermediate results (*i.e.* the similarity and alignment matrices). Dark regions correspond to loop closure detection. The method is able to detect all loop closures robustly. All algorithms run on a 2.4-GHz Intel processor with 2GB of RAM. SIFT feature detection runs at 4 Hz. Using SURF [2] instead of SIFT provides a detection rate of 8 Hz with minor loss in matching performance. The local node estimation runs in two seconds (500 features per node). The rotation guidance runs at 4 Hz. The loop closure detection processes each node in 750 ms.

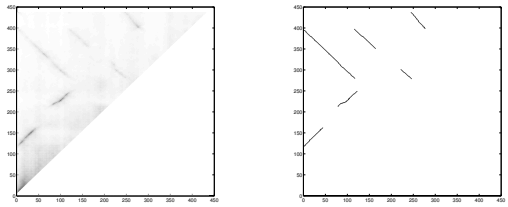

Figure 12. *Left*: Upper triangular part of the similarity matrix for the CORRIDORS dataset. Dark regions correspond to high similarity (loop closure events). *Right*: Output of the Smith-Waterman algorithm. Dark lines correspond to sequence alignments.

Figure 13. *Left*: first visit of a node (CORRIDORS dataset). Feature points are shown in yellow. *Right*: coming back to the same location during a homing scenario. The red compass shows the direction given to the user.

5. Conclusion

We described a novel approach to body-centered navigation using uncalibrated cameras. The method relies on a topological representation of the world, and uses a learning algorithm to correlate the user's rotation with feature correspondence across cameras. The method shows robust navigation guidance with an accuracy better than $15°$ on extended real-world datasets. In future work, we hope to extend our approach to 3D navigation (e.g., stairwell ascents and descents) and explore ways to achieve higher rotation accuracy.

Acknowledgments

We thank Matthew Antone, Albert Huang and Matthew Walter for their useful ideas and comments. We gratefully acknowledge the support of Draper Laboratory's University Research and Development Program.

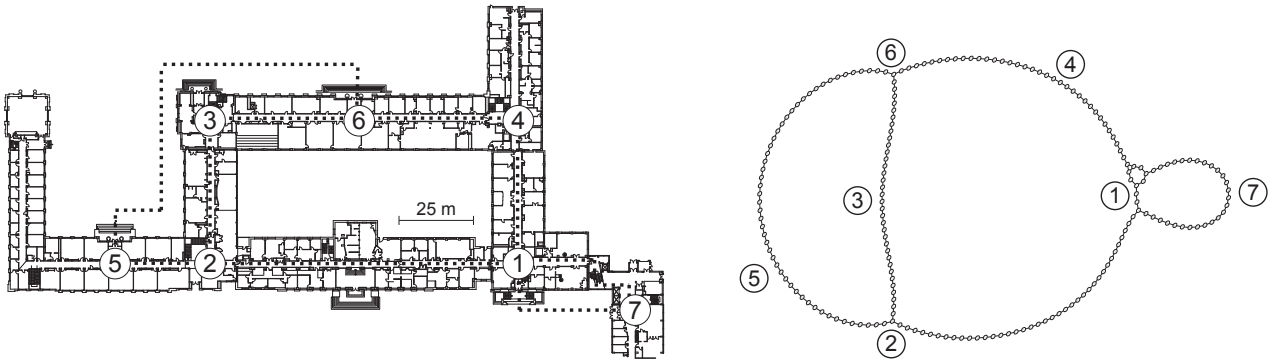

Figure 14. Floor plan of the explored environment for the CORRIDORS dataset (*left*) and corresponding topological map displayed with a spring-mass model (*right*). The exploration path (shown as a dotted line) was: 1, 2, 3, 4, 1, 2, 5, 6, 4, 1, 7, 1, 4, 3, 2, 1, 7, 1. Exploration includes both indoor and outdoor environments over a course of $1,500$ meters (30 minutes).

References

[1] 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*, pages 1027–1037, 2008.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool. SURF: Speeded up robust features. *Computer Vision and Image Understanding (CVIU)*, 10(3):346–359, 2008.

[3] Z. Chen and S. T. Birchfield. Qualitative vision-based mobile robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2686–2692, 2006.

[4] T. Collet. Landmark learning and guidance in insects. *Philosophical Transactions of the Royal Society of London: Biological Sciences*, 337:295–303, 1992.

[5] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22, 2004.

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

[7] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: Real-time single-camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):1052–1067, 2007.

[8] S. DiVerdi, J. Wither, and T. Hollerei. Envisor: Online environment map construction for mixed reality. In *IEEE Virtual Reality Conference*, pages 19–26, Reno, Nevada USA, March 2008.

[9] D. Filliat. Interactive learning of visual topological navigation. In *Proceedings of the 2008 IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 248–254, September 2008.

[10] J. Gluckman and S. K. Nayar. Ego-motion and omnidirectional cameras. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*, page 999, Washington, DC, USA, 1998. IEEE Computer Society.

[11] T. Goedemé, M. Nuttin, T. Tuytelaars, and L. V. Gool. Omnidirectional vision based topological navigation. *Int. J. Comput. Vision*, 74(3):219–236, 2007.

[12] K. L. Ho and P. Newman. Detecting loop closure with scene sequences. *International Journal of Computer Vision*, 74(3):261–286, September 2007.

[13] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[14] M. Milford, G. Wyeth, and D. Prasser. Efficient goal directed navigation using RatSLAM. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1097–1102, April 2005.

[15] 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)*, 2(28):1530 – 1536, September 2004.

[16] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1):3–20, 2006.

[17] L. M. Paz, P. Piniés, J. D. Tardós, and J. Neira. Large scale 6-DOF SLAM with stereo-in-hand. *IEEE Transactions on Robotics*, 24(5):946–957, October 2008.

[18] G. Salton. *Introduction to Modern Information Retrieval (McGraw-Hill Computer Science Series)*. McGraw-Hill Companies, September 1983.

[19] B. Schnapp and W. Warren. Wormholes in Virtual Reality: What spatial knowledge is learned for navigation? *Journal of Vision*, 7(9):758–758, 6 2007.

[20] T. Smith and M. Waterman. Identification of common molecular sequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[21] I. Stratmann and E. Solda. Omnidirectional vision and inertial clues for robot navigation. *J. Robot. Syst.*, 21(1):33–39, 2004.

[22] A. M. Zhang and L. Kleeman. Robust appearance-based visual route following for navigation in large-scale outdoor environments. *The International Journal of Robotics Research*, 28(3):331–356, 2009.

[23] W. Zhang and J. Košecká. Hierarchical building recognition. *Image and Vision Computing*, 25(5):704–716, 2007.