MIT-CSAIL-TR-2008-032 June 6, 2008

Non-Metrical Navigation Through Visual Path Control

Albert S. Huang and Seth Teller

Non-Metrical Navigation Through Visual Path Control

Albert Huang and Seth Teller

***Abstract*—**

We describe a new method for wide-area, non-metrical robot navigation which enables useful, purposeful motion indoors. Our method has two phases: a *training* phase, in which a human user directs a wheeled robot with an attached camera through an environment while occasionally supplying textual place names; and a *navigation* phase in which the user specifies goal place names (again as text), and the robot issues low-level motion control in order to move to the specified place. We show that differences in the visual-field locations and scales of features matched across training and navigation can be used to construct a simple and robust control rule that guides the robot onto and along the training motion path.

Our method uses an omnidirectional camera, requires approximate intrinsic and extrinsic camera calibration, and is capable of effective motion control within an extended, minimally-prepared building environment floorplan. We give results for deployment within a single building floor with 7 rooms, 6 corridor segments, and 15 distinct place names.

I. Introduction

Effective navigation in general environments presents a fundamental challenge for humans, attracting attention since antiquity. The classical formulation of robot motion planning [11] has focused on the construction and usage of a precise metrical map of the robot's workspace. In this setting, high-level planners have required that the robot be able to localize itself precisely within this map, and that the goal pose also be expressed in this same reference frame.

These requirements are reasonable for structured settings in which (for example): the robot is bolted to a factory floor; materials to be manipulated are delivered to the robot by a conveyor belt; humans are kept away for safety reasons; and the environment is otherwise unchanging. Motion planning methods for courier robots (e.g. in hospitals) intended to move along marked paths also make these assumptions. However both cases also require the environment to be prepared extensively (or "structured") beforehand by human engineers, and maintained in the structured site for the duration of robot operation.

In larger and more dynamic environments that are harder to control and observe, the construction of a precise metrical map is both difficult and, we argue, unnecessary. Both measurement uncertainties and the sheer scale of a large environment serve to confound and overwhelm the localization and map-building process. Humans commonly perform analogous complex navigation tasks in extended environments, without relying upon global sensing or externally-provided metrical coordinate systems. We believe humans do so using three mechanisms: (1) an ability to learn, from traversing an environment, a topological map (with weak metrical attributes) of the environment; (2) an ability to associate natural-language names with map elements (i.e. places); and (3) an ability to determine, using both vision and temporal continuity, one's proximity and orientation with respect to a previously traversed place or path.

The authors are with the EECS Department, CS & AI Laboratorys, Massachusetts Institute of Technology, Cambridge, MA 02139, {albert,teller}@csail.mit.edu

Our robot navigation method has two phases: a *training* phase, in which a human user directs a wheeled robot with an attached camera through an environment while occasionally supplying textual place names; and a *navigation* phase in which the user specifies goal place names (again as text), and the robot issues low-level motion control to move to the specified place.

During training, the robot constructs a graph of labeled nodes and edges. Each graph *node* corresponds to a place named by the user; each node label is supplied by the user as a place name. Each graph *edge* corresponds to a path traversed by the robot between two named places. Each edge is associated with a set of descriptors for visual features observed by the robot as it moved along the path.

Many features observed during training are re-observed during navigation. We show that differences in the visual-field locations of features matched across training and navigation, and particularly the scale at which they are detected, can be used to construct a simple and robust control rule that guides the robot onto and along the training motion path, reproducing the training path at any desired speed.

The remainder of this paper is structured as follows. Section II describes related work in robot localization and navigation. Section III gives an overview of our approach. Section IV describes our experimental framework, evaluation metrics, and experimental results. Section V discusses the method's behavior and failure modes, and Section VI concludes.

II. Related Work

This section describes a number of related efforts in robotics, machine vision, and assistive technology.

Robotics researchers have long sought to develop efficient SLAM (Simultaneous Localization and Mapping) algorithms capable of producing accurate maps of an arbitrary environment, and an accurate estimate of the robot's sensing path through the environment, with little or no prior knowledge. Researchers have proposed a variety of SLAM methods that address the uncertainty inherent in sensor data and robot motion, including topological [10], particle filter [21], [15], feature-based [19], and hybrid [9], [2] methods. Successful SLAM approaches have been developed based on the combination of laser scan-matching with Bayesian state estimation [9], [21].

Researchers have exhibited motion control using similarity measures on omnidirectional imagery, demonstrating motion control over short distances (a few meters) [24]. Others have used omnidirectional imagery for localization within a multi-room environment while requiring that a global metrical map be constructed prior to navigation [23]. Researchers have also explored the idea of binding human names to particular motion paths inferred from odometry, composing multiple paths into routes and performing motion correction using image cross-correlation [22]. A related approach took a pre-specified topological map and manually acquired references images at each link to demonstrate a topological navigation capability based on visual servoing [17].

Researchers developing autonomous or assistive wheelchairs have addressed issues of user goal specification, high-level motion planning, and low-level motion control. The Wheelsley wheelchair [25] provided indoor and outdoor navigation through user gaze control (limiting the specified goal poses to those visible to the user). The TAO project [7] and University of Texas Intelligent Wheelchair [8] used computer vision and infrared sensing to track landmarks and distinctive places in the environment. The VAHM project [3], [4] demonstrated path planning with collision avoidance. The navigation system developed by Yoder et al. [26] included a command interface similar to ours, but required that the environment be prepared with artificial visual fiducial markers at surveyed locations. The Arizona State University wheelchair [14] provided route-planning capability, but required a prior metrical map. The PSUBOT [16], [20] navigated from room to room using vision-based landmark tracking.

Recent advances in visual feature detection [12], feature matching, and image retrieval have made vision-based, non-metrical topological mapping and navigation attractive [6]. As precise metrical coordinates are not as useful in large-scale environments, researchers have also begun using graph-based methods for vision-based topological navigation [5], [1].

Our approach falls within this last category, and demonstrates that the scale of a visual feature, and not just its location, provides highly useful information for vision-based navigation. We use this information to construct a simple and robust control rule for visual path following. In some respects, it can be thought of as using both global and local statistics of optical flow for path control.

III. Visual Path Control

This section describes the visual path control method. After some preliminaries about the robot and sensors, we describe the training phase and navigation phase.

A. Preliminaries

This section describes the conventions we use for robot and camera coordinates, the camera's field of view, and the feature descriptor we use to characterize the robot's immediate visual environment.

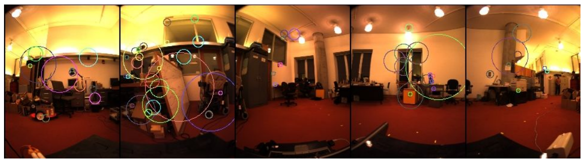

Fig. 1. An omnidirectional camera frame of five outward-looking images. Our camera also collects one upward-looking image (not shown), which is unused by our method.

1) Body Coordinates and Omnidirectional Images: We adopt a body coordinate system with its origin at the center of the robot's center axle. We define the azimuth angle or "bearing" θ to be zero directly ahead of the robot, and to increase counter-clockwise when viewed from above.

An omnidirectional camera is mounted with its center directly above the robot origin (Figure 4), and captures image "frames," each associated with a monotonically-increasing integer "frame index." We define an altitude angle ϕ to be zero at the image horizontal, and $\pi/2$ at the image vertical. Our omnidirectional camera observes the entire hemisphere above its center, along with an additional spherical sector about $\pi/3$ radians high below the camera's equatorial plane. That is, it observes a solid angle of viewing directions described by

$$d(\theta, \phi) = (\cos\theta\cos\phi, \sin\theta\cos\phi, \sin\phi)$$

where

$$-\pi \leq \theta < +\pi \text{ and } -\pi/3 \leq \phi \leq +\pi/2.$$

This paper addresses 3-DOF, i.e. $(x, y, \text{heading})$, motion planning in a planar world, using only visual features left, right, ahead of and behind (but not directly above or underneath) the robot.

2) SIFT Descriptors and Frame Signatures: To identify distinctive visual features, we use the "SIFT" (Scale-Invariant Feature Transform) operator [12]. This operator takes a raster image as input, and produces as output a programmer-specifiable number of "interest regions" in the image, and a multi-dimensional "SIFT descriptor" for each interest region. Each SIFT descriptor consists of a location and characteristic scale expressed in image coordinates, and a histogram of image gradient directions sampled at many locations within the (contrast-normalized) interest region (cf. Figure 1).

SIFT features were designed to provide a measure of invariance to changes in viewing direction and lighting conditions, and have proved highly successful in the vision literature for tasks such as object recognition [12] and vision-based SLAM [18]. Our preliminary experiments indicate that the SIFT operator reliably and consistently identifies certain surface fragments in the environment as the camera moves toward, abreast of, and beyond them, or while it rotates in place, under a variety of natural and artificial lighting conditions. It is thus a useful building block for the construction of robust vision-based control methods.

We assume that the camera is approximately calibrated, so that each image-space location corresponds to a known direction (θ, ϕ) in camera coordinates. We fix the extrinsic (camera-to-robot) calibration through the mounting procedure above, aligning the camera and robot body $(\theta = 0)$ axes manually. Approximate intrinsic calibration parameters for our camera were supplied by its manufacturer; these are sufficient for our purpose. We use the intrinsic calibration information to estimate the range of bearing angles subtended by each observed SIFT feature, i.e., the feature's apparent size.

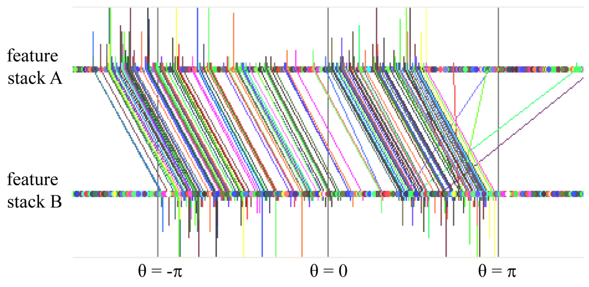

Fig. 2. A pair of matched feature stacks, where the robot has rotated by approximately $\frac{\pi}{2}$ radians. Matched features are drawn in the same color, and vertical bars denote the scale of each feature.

We define the "signature" of an omnidirectional video frame as the set of SIFT descriptors observed in that frame. We store the descriptors for each frame, and for each feature its bearing θ and *scale* (apparent size) s (both in radians) in each frame. Finally we define a "stack" of features in a particular frame to be feature bearings and scales plotted along a θ axis (Figure 2).

3) Place Graph: The training phase constructs, and the navigation phase later uses, an underlying environment representation that we call a "place graph." As its name suggests, this is a graph data structure, with additional information attached to its nodes and edges.

Place graphs are connected and undirected. Each place graph *node* corresponds to a distinct "place," or particular location in the environment, as indicated by the human user. Each node (place) has a unique associated *name* as specified by the user; names are stored as text strings in a flat (non-hierarchical) manner. Example names in our environment include "Kitchen," "Robotics lab foyer," "Robotics lab east bay," "Women's room," "Synthetic biology lab foyer," "Tom's office," etc.

Place names may refer to other names, as long as the resulting name is unique, e.g., "Storage room off of the Kitchen." The user may also specify a numerical (unique integer) name for any place which is functionally important – e.g. a corridor junction – but for which specifying a globally unique non-numeric name would be awkward or difficult. (Consider an example from our environment: "Junction of the corridor leading from the atrium to the printer station with the corridor leading from the third-floor lounge to the third-floor carrel area.")

Each place graph *edge* is an undirected edge corresponding to a motion path segment, traversed by the robot during training, which connects two places (the places associated with the nodes defining the edge). Each image captured along a path segment has an associated feature stack. Each path segment has an associated set of "feature traces" for the features observed during motion along that segment. For convenience, we define a parameter $t \in [0..1]$ for each segment as

$$t = (f - s)/(e - s)$$

where s and $e > s$ are respectively the starting and ending indices of frames captured along that segment, and f is any intermediate frame index $s \le f \le e$. We say that the "interior" of a place graph segment consists of all points on the segment except its two defining endpoints.

B. Training Phase

Our approach is inspired by Lynch's formulation of "landmarks" (visually or functionally noteworthy places) and "paths" (connections between landmarks) in his seminal studies of human navigation strategies in urban areas [13]. Rather than consider visual landmarks only at places, however, we consider them as distributed throughout the environment, and associate them with training motions from place to place.

In the training phase, the place graph is initialized to the null graph, i.e., a graph with no nodes and no edges. A human user then leads the robot throughout the environment of interest, occasionally specifying a place name that describes the robot's current location. Whenever the robot re-traverses a previously-named place, the user specifies its place name. The captured images and user-supplied names provide enough information for the training algorithm to infer the following elements of the place graph:

- Nodes (created whenever the user specifies a place name for the first time);
- Node labels (the user-specified place name);
- Edges (created whenever a path segment between two places is traversed for the first time);
- Edge distance labels (set to the acquisition time for the path segment corresponding to the edge); and
- Edge feature traces (extracted by storing SIFT features within the imagery captured along the path segment).

These elements amount to a SLAM-like "map" representation of the traversed environment, which, although not metrical, contains enough information to enable the robot to orient itself when at place graph nodes, localize itself when on place graph segments, and perform autonomous motion along such segments during the navigation phase.

Note that the guarantee of certain user input (in the current implementation) enables us to avoid the difficult SLAM sub-problem of "loop closing," i.e., determining when a place is revisited. The user *provides* loop closures simply by indicating each place's name whenever the robot arrives at that place. Thus the training algorithm can insert an edge $e(u, v)$, in so doing closing a loop, exactly when places u

and v are visited in succession (in either order), but $e(u, v)$ does not previously exist in the place graph.

In summary, the training phase produces a place graph with exactly one node (and associated name) for each named place, and with an edge $e(u, v)$ between two nodes u and v if and only if the robot traversed a path segment from u to v or from v to u or both. The next section describes how a place graph can be used to achieve autonomous motion planning and effective motion within the environment.

C. Common Ingredients

Both the training phase and the navigation phase use the same technical ingredients. However, they are used in a different order, and in slightly different ways. The following table summarizes their use.

Training Phase	**Navigation Phase**
Motion Control (Supplied by User)	Place Names (Supplied by User; Queried within Map)
SIFT Feature Detection (Inserted into Map)	SIFT Feature Detection (Queried within Map)
Localization within Map	Localization within Map
Place Names (Supplied by User; Inserted into Map)	Motion Control (Generated by Algorithm)

D. Navigation Phase

During navigation, the training process is reversed; the user provides a place name as part of a command (e.g., "Go to the Kitchen,") and the robot must identify the "goal node" corresponding the specified place name, or report failure if no such node exists. Otherwise, the robot must move so as to satisfy the user's command. Our principal contribution is the formulation of a robust control rule based on the relative locations and scales of matched SIFT features.

Achieving a satisfying motion involves four sub-computations: (1) localizing within the place graph; (2) determining the goal node, if any, corresponding to the user-specified place name; (3) formulating a high-level motion plan (i.e., a path within the place graph) that leads to the goal place; and (4) executing the motion plan as a series of low-level motions. We discuss each sub-computation below. We say that a camera frame or its features are "live" if they have been captured during navigation, rather than during trainng.

1) Localization: Robot localization, in our setting, requires that the robot determine its location and orientation with respect to the place graph. This amounts to localization on a 1-D manifold (along path segments) with occasional discrete bifurcations (i.e., at place nodes) – a much simpler problem than general 3-DOF (x, y, θ) localization in the plane.

At the start of the navigation phase, the user places the robot, with some arbitrary orientation, at a specified place graph node. This provides a useful initial condition to the navigation algorithm: the robot knows itself to be on a particular place graph node. Subsequently, the robot attempts to move so as to occupy the goal node while staying (very nearly) on the training path.

There are two sub-cases of interest: "at-node" orientation and "along-segment" localization. At-node orientation is invoked whenever the robot believes itself to be at a place graph node, and that node is not the goal node. In this case, the robot must orient itself so as to align with the desired outgoing segment it wishes to traverse. Along-segment localization is invoked whenever the robot believes itself to be on (or near) a place graph segment, but not at a node. Both cases are explained further in § III-D.5 and § III-D.6.

Location and orientation determination can be solved using temporal continuity, if the robot location was known at some earlier time and the robot has been at rest since that time. Otherwise, location determination involves a search for *visual features* within the place graph that match the set of visual features currently being observed. If this search fails, the robot enters a LOST state, reverting to manual control.

2) Goal Determination: Determining the goal node within the graph amounts to searching for the node, if any, which has as its associated name the place name specified by the user. We use linear search over all graph nodes, and text string comparison, to find the desired node.[1] If no goal node is found, the robot simply reports failure, and awaits another user command.

3) High-Level Motion Plan: Formulating a high-level motion plan amounts to searching the graph for a shortest path linking some source node in the graph (the robot's current location) with some goal node in the graph (corresponding to the user-specified place name). Given distance labels on each edge, single-source shortest path identification is a well-understood problem, efficiently solvable (for example) using Dijkstra's algorithm.

The resulting high-level motion plan consists of a sequence of places and segments in alternation, to be traversed by the robot as it moves from the current node to the goal node. The only task remaining for the robot is to *execute* the motion plan, i.e. traverse the derived sequence of places and segments, while avoiding collisions. We assume that the environment is static; the consequence of this assumption is that if a path segment was traversed during training, it will remain obstacle-free during the navigation phase. (Our current implementation exploits this assumption by generating precise motion control to stay on the training paths, at the cost of some fragility in overall system behavior. We plan to reduce the method's fragility by adding low-level obstacle avoidance in future.)

4) SIFT Feature Matching: Both at-node and along-segment motion control make use of a matching operation between two sets of SIFT features (one observed during training, one during navigation). To match a single SIFT feature $\mathcal{F}$ to a set of N candidate SIFT features, we compute the Euclidean distance [12] between the SIFT descriptor

[1]Much more efficient solutions to this problem are possible, for example through string hashing. However this step requires negligible time in our system so we have not sought to optimize it.

for $\mathcal{F}$ and each of the candidate descriptors, retaining the two best (i.e. closest) candidates. If the best candidate is much better than the second-best, we classify it as a match; otherwise we conclude that $\mathcal{F}$ has no match. To identify the best matches among two sets of M training features and N navigation features respectively, we invoke the above algorithm M times. To prevent many-to-one matching, we removing matching feature pairs from both sets whenever a match is found.

5) At-Node Motion Control: The robot believes itself to be at a known node when one of two conditions holds. First, at the start of the navigation phase, the user physically places the robot at a node, and informs the robot of the node name by specifying it as text. Second, whenever the robot completes traversal of a known path segment in a particular direction, it concludes that it has arrived at the place graph node terminating that segment's corresponding edge.

In either case, the currently occupied node is either the goal node with the place name specified by the user, or it is not. In the former case, the robot has satisfied the user's command and can stop moving and report success. In the latter case, the robot must prepare to follow a path segment originating at the current node which will bring it closer to the goal node. (Such a path segment must exist, since the place graph is connected, and both the current node and goal node are known to be in the place graph.)

6) Along-Segment Motion Control: The robot achieves travel along a segment simply by executing motions that cause the traces of currently observed features to replicate (approximately) the traces of matching features observed during training. We define a virtual lookahead point or "carrot" to lead the robot along the path segment. The carrot's position is initialized to match the robot's ($t = 0$), then t is gradually increased (causing the robot to follow the carrot) until $t = 1$. We use the features observed at the carrot position on the training path, and the features observed during navigation, to formulate a *robust control rule* that modifies the robot motion to correct any deviation from the navigation path segment. The control rule takes as input two feature sets C and N observed during training and during navigation, respectively, and computes two outputs:

- An angle r such that rotating the robot from its current position through this angle would align its features with those observed at the carrot position; and
- A vector $\mathbf{X}$ which determines the direction of translation that will cause the scales of observed features to better match the scales of any matching features observed at the carrot position.

Features are used to compute control outputs as follows. We use the simple observation that the scale of an observed feature decreases with increasing distance between the robot and the scene element giving rise to the feature. Thus if the scale of a live feature is less than the scale of its training match, then the robot should head towards that feature. Conversely, if the scale of the live feature is greater than the scale of its training match, then the robot should move away

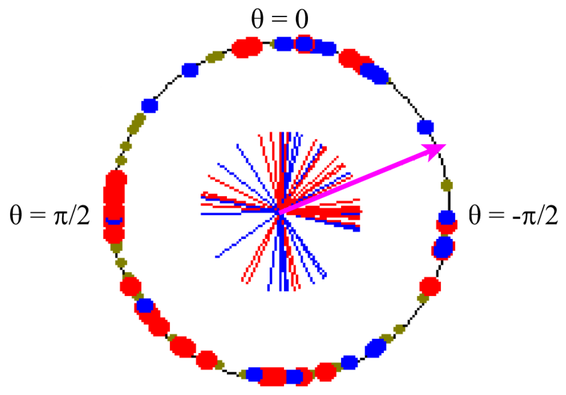

Fig. 3. Each matched feature votes on the direction the robot should travel, based on the scale difference from the matched training feature. Features are shown here on a unit circle, with votes drawn as lines emanating from the origin. The sum of every vote is drawn as a longer arrow. Red features indicate a scale increase, blue features indicate a scale decrease, and features with no significant scale change are shown in brown.

from that feature. In this way, each matched feature "votes" on the direction in which the robot should move (Figure 3).

For each feature L_i matched from live imagery to a feature T_i in the training image, we use the camera's intrinsic parameters to compute unit vector $\mathbf{R_i}$, in the robot's body frame, that points towards the feature. This vector is scaled by a value $repel_i$ defined as:

$$\begin{array}{lll} \text{if} & scale(L_i) * S < scale(T_i) & \text{then } repel_i = -1 \\ \text{if} & scale(L_i) > scale(T_i) * S & \text{then } repel_i = 1 \\ \text{else} & & repel_i = 0 \end{array}$$

where S is a threshold to suppress noise arising from the feature scale computation. We have found $S = 1.1$ to be a reasonable choice. The direction vector $\mathbf{X}$ can then be defined as:

$$\mathbf{X} = \sum_i \mathbf{R_i} * repel_i$$

When the robot is in exactly the same position as in the target training image, $\mathbf{X}$ should have zero magnitude. When the robot has a translation offset, $\mathbf{X}$ will point towards the training position. The magnitude of $\mathbf{X}$ gives information on how confident the robot is about its direction. If $||\mathbf{X}||$ is small relative to the number of votes, then the robot is either lost, or has reached its goal. The actual number of matched features can be used to determine which. In making this decision, we define the following variables:

$$\begin{aligned} nmatches &= \text{\# of successfully matched features} \\ nvotes &= \text{\# of features with a non-zero vote on } \mathbf{X} \\ confidence &= \frac{||\mathbf{X}||}{nvotes} \end{aligned}$$

We then use the following criteria to determine whether the robot is lost.

```
if confidence < K_conf and nmatches < K_lost
then
```

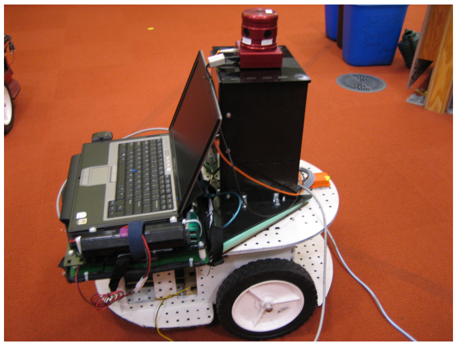

Fig. 4. The robot platform used for training and navigation experiments. The omnidirectional camera is mounted approximately 1 meter off the ground, centered above the vehicle's only axle.

```
    enter LOST mode
else if confidence < K_conf and nmatches > K_advance
then
    advance carrot
else
    move in direction X
```

Where K_{conf}, K_{lost}, and $K_{advance}$ are experimentally determined thresholds. In our experiments, we used $K_{conf} = 0.2$, $K_{lost} = 10$, and $K_{advance} = 35$.

IV. Experimental Results

We evaluated the proposed method on a complex, spatially extended floorplan with many offices, open spaces, and branching corridors (Figure 5). Our robot platform was custom-built with a two-wheel rear axle, front and rear single casters, and one on-board laptop providing about 2GHz of processing power (Figure 4). The omnidirectional camera was purchased from Point Grey Research, which also supplied rough intrinsic calibration parameters. We configured the camera to output 6 images per second at 1024x768, of which we discarded one (the top camera) and decimated the others to 128x96. The camera lenses imposed significant radial distortion, which we did not rectify; this affected the repeatability of SIFT features.

Before the training phase, we chose a set of locations from the environment to designate as named places. We chose places based on familiarity, and functional utility.

Our environment contained about 196 square meters (or approximately 2,110 square feet) of usable floor area, and about 55 linear meters (or approximately 185 feet) of hallway. The user trained the robot by moving it through the environment using a simple joy-stick controller (and a 10-meter long cable, so as not to subtend a large angle in the camera's visual field). Each time a node was visited during the training phase, the user would type in the name of the node on a portable device, which would then transmit the name to the robot.

Finally, the user can operate the robot in navigation mode, sending it on a mission by providing the names of a start node and goal node. An analysis is given for three missions representative of the robot's general performance (Figure 6).

In the first mission, the robot was place at node *lab* and commanded to travel to node 7. The robot completed this

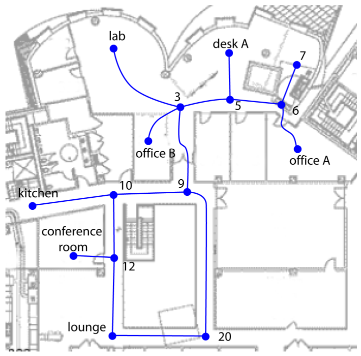

Fig. 5. Floorplan of the test environment, and the place graph defined during the training phase. Edges indicate the path approximately taken during training.

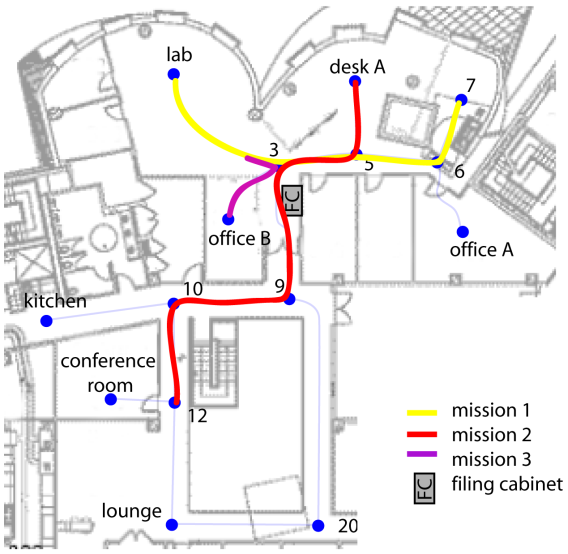

Fig. 6. A manually drawn diagram describing the paths taken in the three described missions.

mission in 10 minutes by traveling through nodes 3, 5, and 6. The maximum observed deviation from the training path was approximately 30 cm.

In the second mission, the robot was placed at node *desk A* and commanded to travel to node *lounge*. Nodes 5 and 3 were reached without issue. Shortly after passing node 3, the robot collided with a filing cabinet. This happened because the lookahead point selected in the trained path was past a turn the robot had not yet taken, causing the robot to turn sooner than it should have.

The robot eventually moved past the filing cabinet on its own and continued to nodes 9, 10, and 12. After reaching node 12, the robot was unable to localize itself on the training path from 12 to *lounge*. This was largely because its orientation and position upon reaching 12 was slightly different (offset by about 6 inches and rotated by 90 degrees) from the trained path from 12 to *lounge*, and the severe radial distortion of the camera negatively impacted the feature matching. The total time taken to reach node 12 from *desk A* was approximately 30 minutes.

Although SIFT is invariant to global lighting variations, nonuniform changes in lighting, such as direct sunlight and cast shadows, affect our method's ability to localize on a path. On a final mission from node 5 to *lab* during daylight (the training was done at night), the robot was able to travel from node 5 to node 3, but was not able to succesfully travel from node 3 to *lab*. While a number of features were succesfully matched, there were not enough consistent matches to exceed the robot's movement thresholds.

V. Discussion

Our method has a number of limitations. It assumes a static environment during both training and navigation (except for illumination variations). During training, it requires that the user name each place whenever it is traversed, not just the first time. To begin navigation, it assumes that the user has placed the robot at a place traversed during training, and has specified its name. It does not perform collision avoidance during navigation. We sketch briefly how each of these limitations might be addressed through future algorithmic or systems improvements.

Our current system assumes that if a path was traversed during training, it will remain passable, i.e., that the environment is static (except for changes in lighting). The static environment assumption is also used as the basis for an expectation that at all times during navigation, some features visible during training can be reacquired. If this is not the case, then the method must be able to tolerate brief periods of being lost, i.e., it must be able to localize or relocalize.

A more capable system would perform low-level obstacle detection and collision avoidance, executing local deviations from previously traversed segments, or replanning, as necessary due to obstacles encountered during navigation. A global index of all visual features observed during training would be useful for this purpose. Moreover, if faced with an entirely unfamiliar local scene, the robot could attempt to traverse it until it sees something it recognizes. This approach could also be used for automatic initialization, and for reacquisition of the training path after brief detours for collision avoidance.

The method uses user naming for two purposes. The first is to solve SLAM loop closing easily (at the cost of an added user burden). The second is to provide for a natural command interface during navigation. We hope to retain the second capability, while removing the burden attendant on the first capability, by incorporating SLAM loop-closure techniques from the vision and robotics literature into our method.

Many corridor junctions have no natural global names, but they do have local names. It would be useful to specify the globally non-unique name "corner" (or "T-junction" or "four-way junction") wherever two corridors meet, and have the training algorithm *infer* a globally unique name from the unique names of other nodes one or more hops away in the place graph, and some high-level logic. In the long run, we expect to be able to remove this limitation entirely by improving the training algorithm so that it can infer the existence of junctions (from segments that overlap partially, but diverge). This would obviate the need for users to specify place names for such "un-named" (but important!) locations.

The method can retrace only those motion paths that have been demonstrated to the robot during training. The addition of a collision-avoidance capability should enable it to traverse, during navigation, logical segments which were not traversed during training, for example, moving through the center of a room of which the robot was trained only around the perimeter.

Our current system assumes a single user training a single robot within a single environment. We hope to investigate multiple users training single or multiple robots within (overlapping) subsets of a single environment. In this way, the burden of training the robot on the entire environment can be distributed across several users, and each user will get the benefit of the group's training (provided that the method can match different users' names for places during or after training).

VI. Conclusion

This paper described a robot navigation method based on searching a place graph for a route to a goal node, then using the image feature descriptors stored with edges along the route for vision-based motion control. The paper also described how the robot can construct a place graph automatically, provided suitable training by a human user. During training, the user provides a kind of narrated tour to the robot, giving the robot distinct names for distinct places and exhibiting the existence of motion paths connecting certain pairs of places. In both training and navigation, the only human-robot interaction required for place description is the utterance (typing) of a name whenever the robot traverses a place.

Our method has a number of advantages. It does no metrical egomotion estimation or require odometry. It performs no feature tracking within any image sequence, but only feature matching across the training and navigation sequences. It does not require accurate intrinsic or extrinsic camera calibration. It does not retain raw image data, but only much more compact and sparse image descriptors. As a result, it is quite simple to describe and replicate, and it is capable of successful navigation within a complex environment encompassing an extended office floorplan with a mix of visually rich and visually spare areas.

The method also has a number of significant limitations. It assumes a static environment during both training and navigation (except for illumination variations). During training, it requires that the user name each place whenever it is traversed, not just the first time. To begin navigation, it assumes that the user has placed the robot at a place traversed during training, and has specified its name. During navigation, it can traverse only those paths that were exhibited by the user during training, and it does not perform collision avoidance. We sketched how each of these limitations might be addressed through future interface, algorithmic, or systems improvements.

References

[1] O. Booij, B. Terwijn, Z. Zivkovic, and B. Krose. Navigation using an appearance based topological map. In *Proc. IEEE Int. Conf. Robotics and Automation*, 2007.

[2] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller. An atlas framework for scalable mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003.

[3] G. Bourhis and Y. Agostini. Man-machine cooperation for the control of an intelligent powered wheelchair. *Journal of Intelligent and Robotic Systems*, 22:269–287, 1998.

[4] G. Bourhis and Y. Agostini. The vahm robotized wheelchair: system architecture and humanmachine interaction. *Journal of Intelligent and Robotic Systems*, 22(1):39–50, 1998.

[5] F. Fraundorfer, C. Engels, and D. Nister. Topological mapping, localization and navigation using image collections. In *Proc. IEEE Int. Workshop on Intelligent Robots and Systems*, 2007.

[6] T. Goedeme, T. Tuytelaars, and L. Van Gool. Visual topological map building in self-similar environments. In *Int. Conf. Informatics in Control, Automation and Robotics*, August 2006.

[7] T. Gomi and A. Griffith. Developing intelligent wheelchairs for the handicapped. In V. Mittal, H. A. Yanco, J. Aronis, and R. C. Simpson, editors, *Lecture Notes in Artificial Intelligence: Assistive Technology and Artificial Intelligence*, pages 150–178. Springer-Verlag, 1998.

[8] W. S. Gribble, R. L. Browning, M. Hewett, E. Remolina, and B. J. Kuipers. Integrating vision and spatial reasoning for assistive navigation. In V. Mittal, H. A. Yanco, J. Aronis, and R. C. Simpson, editors, *Lecture Notes in Artificial Intelligence: Assistive Technology and Artificial Intelligence*, pages 179–193. Springer-Verlag, 1998.

[9] J-S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *International Symposium on Computational Intelligence in Robotics and Automation*, 1999.

[10] B. J. Kuipers. The spatial semantic hierarchy. *Artificial Intelligence*, 2000.

[11] J-C. Latombe. *Robot Motion Planning*. Boston: Kluwer Academic Publishers, 1991.

[12] David G. Lowe. Object recognition from local scale-invariant features. In *Proc. of the International Conference on Computer Vision ICCV*, pages 1150–1157, 1999.

[13] Kevin Lynch. *Image of the City*. MIT Press, 1960.

[14] R. L. Madarasz, L. C. Heiny, R. F. Cromp, and N. M. Mazur. The design of an autonomous vehicle for the disabled. *Autonomous Mobile Robots: Control, Planning, and Architecture*, pages 351–359, 1991. Originally appeared in IEEE Journal of Robotics and Automation, Volume RA-2, Number 3, September 1986, pages 117-126.

[15] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada, 2002. AAAI.

[16] M. A. Perkowski and K. Stanton. Robotics for the handicapped. In *Northcon Conference Record*, pages 278–284, 1991.

[17] Jose Santos-Victor, Raquel Vassallo, and Hans Schneebeli. Topological maps for visual navigation. In *ICVS*, pages 21–36, 1999.

[18] Stephen Se, David Lowe, and Jim Little. Local and global localization for mobile robots using visual landmarks. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 414–420, Maui, Hawaii, October 2001.

[19] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In I. Cox and G. Wilfong, editors, *Autonomous Robot Vehicles*, pages 167–193. Springer-Verlag, 1990.

[20] K. B. Stanton, P. R. Sherman, M. L. Rohwedder, C. P. Fleskes, D. R. Gray, D. T. Minh, C. Espinoza, D. Mayui, M. Ishaque, and M. A. Perkowski. Psubot - a voice-controlled wheelchair for the handicapped. In *Proceedings of the 33rd Midwest Symposium on Circuits and Systems*, volume 2, pages 669–672, 1991.

[21] S. Thrun. A probabilistic online mapping algorithm for teams of mobile robots. *Int. J. Robotics Research*, 20(5):335–363, May 2001.

[22] Raquel Frizera Vassallo, Josi Santos-Victor, and Hans Jvrg Schneebeli. Using motor representations for topological mapping and navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 478–483, EPFL, Lausanne, Switzerland, October 2001.

[23] J. Wolf, W. Burgard, and H. Burkhardt. Using an image retrieval system for vision-based mobile robot localization. In *Proc. of the International Conference on Image and Video Retrieval (CIVR)*, 2002.

[24] Y. Yagi, K. Shouya, and M. Yachida. Environmental map generation and ego-motion estimation in a dynamic environment for an omnidirectional image sensor. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 3493–3498, May 2000.

[25] H. A. Yanco. *Shared User-Computer Control of a Robotic Wheelchair System*. PhD thesis, Massachusetts Institute of Technology, September 2000.

[26] J.-D. Yoder, E. T. Baumgartner, and S. B. Skaar. Initial results in the development of a guidance system for a powered wheelchair. *IEEE Transactions on Rehabilitation Engineering*, 4(3):143– 151, 1996.