Body-relative Navigation Guidance using Uncalibrated Cameras

by

Olivier Koch

Submitted to the Department of
Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2010

Author .....
Department of
Electrical Engineering and Computer Science
February 8, 2010

Certified by .....
Seth Teller
Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by .....
Terry P. Orlando
Chairman, Department Committee on Graduate Theses

Body-relative Navigation Guidance using Uncalibrated Cameras

by

Olivier Koch

Submitted to the Department of
Electrical Engineering and Computer Science
on February 8, 2010, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

The ability to navigate through the world is an essential capability to humans. In a variety of situations, people do not have the time, the opportunity or the capability to learn the layout of the environment before visiting an area. Examples include soldiers in the field entering an unknown building, firefighters responding to an emergency, or a visually impaired person walking through the city. In absence of external source of localization (such as GPS), the system must rely on internal sensing to provide navigation guidance to the user. In order to address real-world situations, the method must provide spatially extended, temporally consistent navigation guidance, through cluttered and dynamic environments.

While recent research has largely focused on metric methods based on calibrated cameras, the work presented in this thesis demonstrates a novel approach to navigation using uncalibrated cameras. During the first visit of the environment, the method builds a topological representation of the user's exploration path, which we refer to as the place graph. The method then provides navigation guidance from any place to any other in the explored environment. On one hand, a localization algorithm determines the location of the user in the graph. On the other hand, a rotation guidance algorithm provides a directional cue towards the next graph node in the user's body frame.

Our method makes little assumption about the environment except that it contains descriptive visual features. It requires no intrinsic or extrinsic camera calibration, and relies instead on a method that learns the correlation between user rotation and feature correspondence across cameras. We validate our approach using several ground truth datasets. In addition, we show that our approach is capable of guiding a robot equipped with a local obstacle avoidance capability through real, cluttered environments. Finally, we validate our system with nine untrained users through several kilometers of indoor environments.

Thesis Supervisor: Seth Teller
Title: Professor of Computer Science and Engineering

Contents

List of Figures

List of Tables

Acknowledgments

None of the work presented in this thesis would have been possible without the constant support of my advisor Prof. Seth Teller. With him, I shared a passion for computational geometry and had my first research experience on the City Scanning Project. Little did I know that his inexhaustible energy and enthusiasm would lead me to so many exciting challenges in robotics and machine vision. My committee members, Prof. Bill Freeman and Prof. Rob Miller seemed to always be available when I needed help and provided an invaluable new perspective on my work. Much of the work I did at MIT was inspired by Prof. Berthold Horn and his seminal class on machine vision.

I would like to thank the Draper Laboratory for their generous financial support and the trust it implies through all these years. Their feedback during our review meetings was always positive and constructive and helped me setup milestones throughout this work.

The neighborhood of 32-33x was like a second home to me. I immediately felt welcome there, far from home. Some of the members became very close friends. I shared an uncountable number of unique moments with Matthew Walter and Albert Huang, talking about research and many other things at the coffee machine. I owe to Matt how to make a great cappuccino, and to both of them some of the ideas and results that appear in Chapter 5. I would like to thank the other members of 32-33x for their friendship: Patrycja Missiuro, Alexander Bahr, Tom Kollar, Yoni Battat, Gleb Chuvpilo, Sam Prentice, Abe Bachrach and so many others.

During my years at MIT, I had the fortune to be part of a historic event in robotics, namely the participation to the DARPA Urban Challenge 2007. Spending a year and a half building an autonomous car and flying through the country to compete with the best teams in the field is an excitement that words can hardly express. It has been my privilege to contribute in a very modest way to this adventure. I am particularly grateful to Prof. John Leonard, Ed Olson, and David Moore. I would like to also thank Matt Antone in particular, for his countless ideas and constructive comments during both my Master's and my PhD.

Two people have been of incredible help to this project. As a lab assistant, Bryt Bradley did much more than she should have done for me, ordering hundreds of parts for the project and checking everything for my conference trips. Ron Wiken, through his infinite patience and support, provided crucial support at the machine shop and the laser cutter.

One of the great aspects of MIT is that it hosts not only great researchers, but also great people. I will be forever grateful for the wonderful minds that I met here, and who will remain friends for life: Tilke Judd, Tom Kollar, Hubert Pham, Dan Roy. I would like to thank Anya Obizhaeva, with whom I learned so much. Far from France, some people have reminded me of my dear Europe: Cathy Bolliet, Philippe Cudre-Mauroux, Blaise and

Valérie Gassend, Yann Le Tallec, Renaud Rinaldi, Léonide Saad, Fabien Sorin, Nicolas Stransky, Laure-Anne Ventouras, Jérome Waldispühl. I would like to also acknowledge the contribution of those who are not cited here and who will yet remain in my memories forever.

Next to the Institute is another institution. By hosting so many passionate discussions and providing heat in the winter, the Miracle of Science is also a contributor to this work. Thank you, MoS, for helping me find my way through the fog of my computer bugs with your fine brewed spirits.

Finally, I am grateful to Aurélia and to my family: to my parents, to my siblings Anne-Christine, Arnaud, Aurélie and Florence for their love and encouragement.

Some of the material from Chapters 4 and 5 appeared in previous publications [48,49].

Chapter 1

Introduction

The ability to navigate through the world is an essential capability to humans. Since the early days of humankind, people have explored the world in a neverending frontier on the ground, on the sea and in the air (the word "navigation" is derived from the Latin "navigare", meaning "to sail"). Recent work [51] gives a thorough description of the progress of navigation technology through the ages, from the mental maps of Polynesians navigators $4,000$ years ago to the advanced satellites of today.

Despite the progress of technology, the science of navigation has become more and more challenging as the environments built by humans have become larger and more complex. One of the latest advance in navigation technology, the Global Positioning System (GPS) is now embedded in many cell phones and portable devices. It provides localization on the entire Earth with an accuracy of a few meters and enables advanced software applications for navigation [2, 97]. However, GPS signals are either poor or absent in a variety of places such as buildings, caves, forests and canyons, thus motivating the need for stand-alone navigation systems.

Figure 1-1: *Left*: A blind person walk through the city. *Middle*: Firefighters participate in a hazardous materials exercise near California State University. *Right*: Marines enter a building during urban terrain training. All these situations involve human users who lack the time, capability or opportunity to learn the layout of the environment before traversing it.

This thesis addresses the problem of navigation in unknown environments where GPS (or any external source of localization) is unavailable. This includes indoor environments and outdoor places with limited sky visibility. In a typical scenario, a user first explores an

unknown environment. Given one or more explorations of the space, the user then revisits the same place and wishes to get assistance in navigating from a location to another. Typical applications include assisting the blind, rescue teams or soldiers in the field (Figure 1-1). More generally, our work addresses situations where the user does not have the time, capability or opportunity to learn the layout of the environment.

Human-oriented navigation in GPS-denied environments present several key challenges. First, the solution must rely on a spatially extended representation of the world in order to successfully assist a user at the scale of the building or the city. Second, in absence of global source of localization, the method needs to be temporally persistent, and provide a consistent output as the user revisits the environments over periods of hours, days or months. Third, because our environment is in constant change, a reliable navigation system must be able to cope with the clutter and the dynamic scenes that surround us. This include short term changes such as passers-by, but also long term modifications of the environment such as new constructions. And last but not least, the presence of a human at the center of the system poses interesting challenges in terms of user interface and choices in the representation of the world ("map"). On the latter aspect, we find some inspiration in studies showing that humans need only loose metric information to navigate through the world successfully [58, 85, 100]. Our approach circumvents the complexity of metric reconstruction by relying on a topological representation of the user's exploration path (*the place graph*). The system builds the place graph as the user moves through the unknown environment, then uses the graph to localize the user and provide node-to-node navigation in the body frame of the user.

1.1 Vision-based Navigation

A variety of sensors may be used for stand-alone navigation. Common approaches are based on ultrasound [26] or inertial sensing [3, 13, 102]. Inertial sensors provide an accurate estimate of the device's rotation speed and linear acceleration. Unfortunately, they drift over time. Standard portable inertial units drift by approximately one degree per minute [10], which makes them insufficient for long excursions. On the other hand, laser ranger finders provide precise metric information about the environment. Unfortunately, 2D laser ranger finders are strongly subject to *scene aliasing* (i.e. many scenes look the same) while 3D laser ranger finders suffer from weight and cost constraints [42].

Computer vision provides an attractive alternative to other modalities. Cameras are compact, lightweight and relatively inexpensive. Unlike inertial sensors, they are not subject to temporal drift. They also provide more information about the surrounding scene than 2D laser range finders. These advantages come at the cost of other drawbacks. Computer vision is sensitive to lighting changes, smoke and dust. In addition, cameras belong to the category of bearing-only sensors and, unlike laser range finders, provide only a 2D representation of a 3D world that is not directly interpretable.

In the last decade, advances in mobile computing have enabled real-time portable applications based on computer vision. Cameras are now commonly found on a large number of mobile platforms such as planes, boats and ground vehicles. Cameras have also become smaller and fit in cell phones and embedded devices. These advances have spurred active research in the field of vision-based navigation [52, 66, 86, 87, 90, 95, 105].

1.2 The Challenges of Camera Calibration

Much like other sensors, a camera requires calibration. When using a pinhole camera model [23], the *intrinsic* parameters refer to the focal length and the center of projection of the camera. In addition, a model for the optical distortion induced by the lens is often incorporated into the set of intrinsic parameters. On the other hand, the *extrinsic* camera parameters refer to the 3D position and orientation of the camera with respect to some reference coordinate frame, typically the body frame of the user. Most methods for vision-based localization and navigation assume that camera calibration is given. Indeed, camera calibration is a well-studied problem with known solutions [39, 94, 99, 109]. The camera calibration provides the ability to seamlessly convert observations in image space from and to features in the world space, and therefore to reason geometrically about the environment. This capability is critical to many vision-based methods.

However, camera calibration remains a challenge for real-world applications today. First, the intrinsic calibration of a camera changes under environmental conditions (humidity, temperature). Second, state-of-the-art calibration methods often require that the camera be aimed at a calibration pattern, which may pose practical issues. Extrinsic calibration, on the other hand, involves optimization methods over the space of possible camera configurations. The dimension of the search space grows quickly for multi-camera systems since there are six degrees of freedom for each camera in the general case. Therefore, determining the extrinsic calibration of a multi-camera system is difficult. Finally, some applications require the cameras to be loosely mounted on their support, as is the case in one of the applications presented in Chapter 4. In this case, the extrinsic calibration of the camera changes during operation and cannot be relied upon. As a conclusion, in addition to the challenging scientific endeavour, developing vision-based methods for uncalibrated cameras provides solution to problems in the real world.

1.3 Method Overview

This thesis presents a vision-based navigation method based solely on a set of uncalibrated cameras. When exploring new places, people tend to develop a "topological" representation of the environment based on landmark locations (the kitchen, the conference room) and salient objects (the mailbox, the street light). We build on this paradigm and devise a topological strategy by which we represent the user's exploration path as an undirected graph (*place graph*).

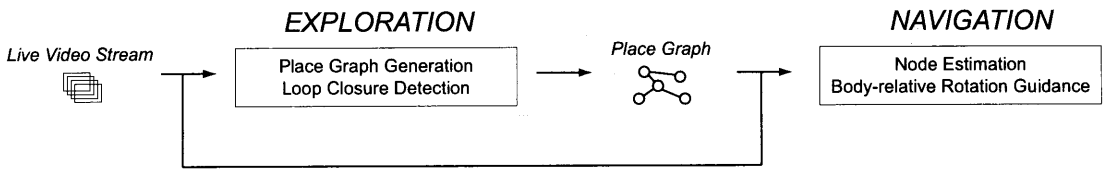

Figure 1-2: Method Overview

A node in the graph represents a location in the world, while an edge represents a physical path between two nodes. Given a place graph, the navigation problem consists in two sub-problems: first, localizing the user in the graph (*node estimation*); second, providing directional guidance to the user given a target destination (*rotation guidance*). We approach navigation as the general problem of moving from one place to another in the place graph. Our method therefore supports scenarios such as *retracing* (returning to the start location), *replay* (reaching the current location from the start location) and *point-to-point navigation* (moving from any given place to any other).

In addition, our method detects automatically when the user revisits a place that has been seen before during exploration (*loop closure*). Finally, the method presented in this thesis extends trivially to multi-user scenarios, where a user explores an environment while another user receives guidance in the same environment later on, without having ever seen it before. Our method presents several particular aspects. First, it does not attempt to build a map in a global coordinate frame. Instead, it reasons *locally* by providing directional guidance at every node in the body-frame of the user. Second, it requires no camera calibration. Therefore, our algorithms reason only in image space and do not rely on geometric primitives in the world.

1.3.1 Image Representation and Feature Point Descriptors

Feature point descriptors such as the Scale Invariant Feature Transform (SIFT, [57]) provide a compact and efficient representation of images. Since they rely on low-level image primitives such as the Difference of Gaussian images (DOGs), they are relatively agnostic on the type of environment being observed and therefore apply to a large class of places, whether indoor or outdoor, natural or human-made. We briefly describe the state-of-the art of feature point descriptors and feature matching in Chapter 3.

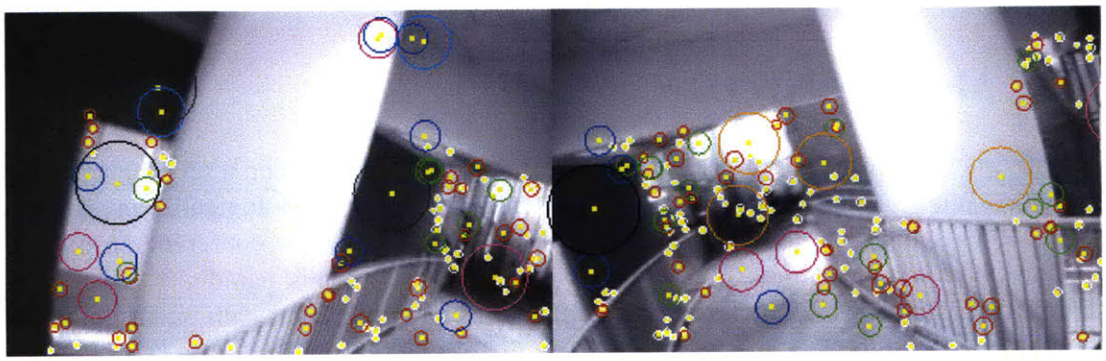

Figure 1-3: Scale Invariant Feature Transform (SIFT) detection on a two-image frame. Each point is assigned a scale represented as a circle and a feature descriptor.

1.3.2 Place Graph Generation and Loop Closure

During exploration, the method builds a place graph by generating nodes while the user moves through the environment. The decision to create a node is fully automatic and relies

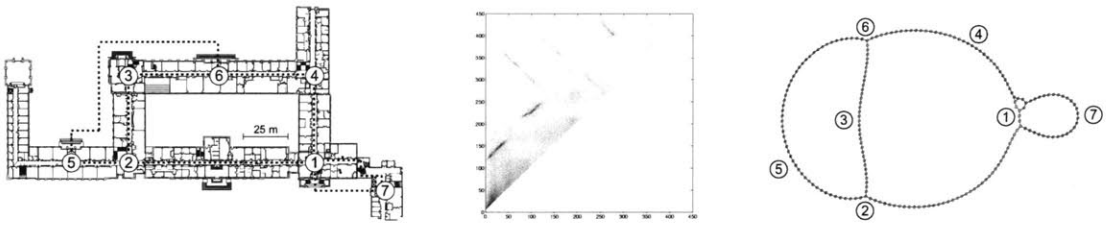

Figure 1-4: *Left*: Notional path followed by the user during a 30 minute-long exploration across MIT corridors. *Middle*: Similarity matrix. Dark regions correspond to high similarity. *Right*: Corresponding place graph displayed using a spring mass model.

on the variability of the appearance of the environment. This approach is generic and sets no constraint on the motion of the user. We represent a node by the set of visual features observed at the corresponding location. In essence, the place graph therefore corresponds to a sparse representation of the set of visual features observed during exploration.

The capability to automatically detect when the user comes back to a previously seen place is commonly referred to as loop closure and is of critical importance to any navigation system. Detecting loop closures allows the system to determine the correct physical connectivity of the place graph and to determine the shortest path between two locations. State-of-the art methods [19,40] build on the standard "bag-of-words" technique [67]. The intuition behind these approaches is to cluster visual features into "visual words" that constitute a vocabulary. Search methods then allow the system to efficiently determine the closest word to any given observation and to compute a measure of similarity between nodes in the graph. These methods typically assume that a visual vocabulary is given ahead of time, which raises the issue of using such a system in a completely new environment. We propose an alternative method that builds a vocabulary from the void, and demonstrate its effectiveness on a large class of environments.

1.3.3 Node Estimation

The problem of node estimation is to determine the location of the user in the place graph during a revisit. We approach the problem by assuming that the global localization of the user in the graph is known at the beginning of the revisit. Although this may seem like a strong assumption at first, most scenarios provide this information at the beginning of the revisit. We then frame the node estimation problem as a recursive Bayesian estimation problem, in which the location of the user is maintained as a probability distribution over the graph. We incorporate the motion continuity assumption by modeling the transition step using a normal distribution with no bias. On the other hand, the observation model relies on a generic similarity measure based on visual feature matching.

1.3.4 Body-relative Navigation Guidance

Given the position of the user in the graph, the goal of the rotation guidance algorithm is to provide the direction to the next node to the user in their own body frame. We state this

problem as the *relative orientation problem*, in which the goal is to determine the orientation of the user relative to its orientation when the node was first visited. Our approach brings a fundamental contribution to the field of vision-based navigation by demonstrating that camera calibration is not required to get a coarse solution to the relative orientation problem. Our method relies on two assumptions: first, that features are uniformly distributed in image space and second, that the number of features is "large". Under these assumptions, we show a method for solving the relative orientation problem and use the Central Limit Theorem to prove that the error decreases with the square root of the number of observations.

The intuition underlying our method is to work around camera calibration using learning. In a brief training stage, the system learns the average relative orientation corresponding to a match between any two cameras as the user rotates in place in an arbitrary environment. The training algorithm computes feature matches across cameras for each pair of frames and relates them to the estimated rotation of the user. The output is an $n \times n$ *match matrix* H (for n cameras), where $H(i, j)$ represents the relative rotation associated to a match between camera i and camera j. By definition, H is anti-symmetric. Given the match matrix, the relative orientation problem can then be solved by matching the current observations with the observations associated to the current node, where each match votes for a relative orientation angle.

The advantage of this method is that it provides a reasonably precise estimate of the relative orientation of the user, while avoiding a tedious extrinsic camera calibration. In addition, it is versatile and can be applied to a variety of camera configuration. In particular, it sets little constraint on the number of cameras or their position on the user. Finally, the method is robust to outliers as well as slight camera motions on the wearer's body.

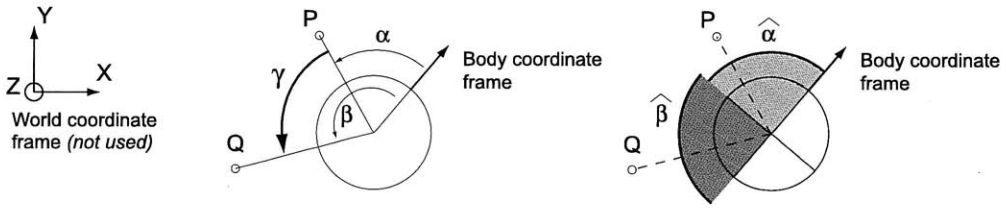

Figure 1-5: Using calibrated cameras (*left*): two world features P and Q yield two observable bearing measurements α and β. Their relative orientation is $\gamma = \beta - \alpha$. Uncalibrated case: (*right*): α and β are not observable but can be replaced by estimates $\hat{\alpha}$ and $\hat{\beta}$. When the number n of observations is large, the error on the estimate of γ decreases with $\sqrt{n}$.

1.4 Evaluation and User Study

A contribution of this thesis is the development of a prototype system composed of four cameras and a laptop computer mounted on a backpack (Figure 1-6). The system includes a tablet PC user interface and is capable of several hours of untethered operation. We demonstrate the effectiveness of our method on this system through a user study with 9 users spanning 2.5 hours and 6 km of exploration of indoor environments. We evaluate the efficiency and effectiveness of the system using various qualitative and quantitative metrics.

Figure 1-6: Our body-worn sensor suite includes four IEEE 1394 cameras mounted on the shoulders of the user, with a 360° field of view.

1.5 An Application to Ground Robot Navigation

We also demonstrate the effectiveness of our method in a robotic setting (Figure 1-7). Our goal is two-fold. First, we wish to advance the state of the art in robotic navigation by showing that our method is able to autonomously guide a robot through real environments, assuming that the robot is capable of processing high-level navigation commands. Second, we use a state-of-the-art laser-based SLAM method [91] to compute the ground-truth localization of the robot on large datasets. We then use these datasets to validate our method, in particular the node estimation algorithm and the rotation guidance algorithm.

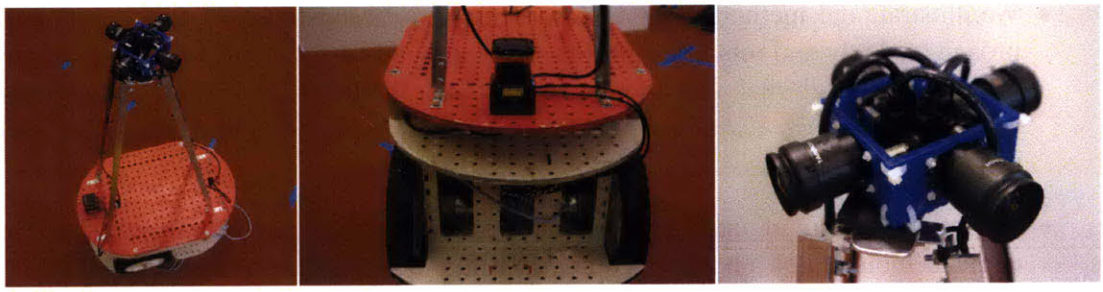

Figure 1-7: Our robot is composed of a two-motor wheeled base, a four-camera omnidirectional rig (*right*) and a laser ranger finder (*middle*). We combine a high-level vision-based navigation algorithm with a low-level obstacle avoidance algorithm to obtain robust autonomous navigation in unknown environments.

1.6 Contributions

Despite promising results in the past decade, vision-based navigation remains an open problem today, in particular in GPS-denied environments. Recent work on vision-based SLAM demonstrates promising results but suffers from several inherent limitations. In real environments, these problems are made harder by noisy camera sensors, cluttered scenes, and dynamic environments. Cameras operate with limited dynamic range and suffer from saturation and motion blur in general settings. In addition, despite its high information rate, computer vision is sensitive to aliasing, particularly in uniform or featureless environments.

While recent research has largely focused on metric methods based on calibrated cameras, the work presented in this thesis demonstrates a novel approach to navigation using uncalibrated cameras. In particular, our method circumvents the limitations of SLAM by operating in a topological representation of space. This approach in itself is not new [25, 31, 62]. The main contribution of this work lies in the following aspects.

- We present a novel approach to generating a topological map of the environment using only the visual appearance of the environment only. This is in contrast with more *ad hoc* methods [19, 31] or methods relying on metric mapping [5, 56, 62].

- We propose a method for rotation guidance that relies only on a set of uncalibrated cameras. The approach consists of learning the correspondence between feature matches across cameras and user rotation in the body frame. We demonstrate the validity of our approach in a probabilistic framework and show that the error in the estimate decreases with the square root of the number of observations. In addition, the method makes a single assumption about the environment, *i.e.* the isotropic distribution of features in image space.

- We take inspiration from state-of-the-art approaches to loop closure detection [1, 19] to achieve robust, large-scale detection of loop closure events from a single stream of images.

- We illustrate our method in a robotics setting and demonstrate the ability of our approach to safely and robustly guide a robot equipped with a local obstacle avoidance capability through real, extended environments. We include a ground-truth dataset spanning more than $1,200$ meters and 2.5 hours of exploration through indoor environments.

- We implement our algorithms on an operational prototype system composed of four cameras mounted on a backpack and a user interface implemented on a tablet PC. To our knowledge, our design is the first to provide a full omnidirectional field-of-view for a human-wearable system without relying on a head-mounted omnidirectional camera.

Chapter 2

Related Work

The past decade has seen tremendous progress in the field of vision-based navigation. In this chapter, we present a review of the state-of-the-art on this topic. We focus particularly on several directions of research: Simultaneous Localization and Mapping (SLAM), visual homing and various methods based on uncalibrated cameras.

2.1 Simultaneous Localization and Mapping

Perhaps the most natural approach to the navigation problem consists of building a metric map of the environment as the agent explores it, while maintaining the localization of the agent within the map. This approach is commonly referred to as Simultaneous Localization and Mapping (SLAM). Due to its wide range of applications, SLAM has attracted significant interest over the past few decades. SLAM is often framed as a data association problem, in which a series of noisy observations need to be associated to features in the map. Standard filtering methods such as the Extended Kalman Filter [45] allow implementers to manage the noise inherent to any real sensor.

A number of SLAM algorithms rely on laser range finders [62, 72, 95]. Indeed, laser data has the advantage of providing both range and bearing information about the surrounding world, which is then directly usable to build local metric maps. However, laser-based methods suffer from *visual aliasing*, *i.e.* that many places tend to yield similar measurements, which is a severe limitation for large-scale applications. Vision, on the other hand, yields bearing-only measurements, but provides richer and more distinguishable information about the surrounding environment. Recent progress in computing has unleashed the power of real-time computer vision for SLAM.

In the monocular case, the depth of a feature cannot be recovered from a single observation. Therefore, multiple observations must be combined as the camera moves through the world. The standard EKF formulation of SLAM applies well when the parallax effect is large [20], *i.e.* in small environments such as a desk or a room, but fails with features at infinity. Direct parameterization of the inverse depth of the features removes this limitation [65]. In combination with a hierarchical map approach [14], the method provides large-scale SLAM across hundreds of meters of exploration. Recent work demonstrates an efficient solution to the relative pose problem using five point correspondences [68] and its

applications to ground vehicle navigation [70]. Combining visual odometry, online path learning and multi-scale global mapping provides a full end-to-end navigation solution for an outdoor robot [50]. In addition, the use of stereo reduces the unobservability problem inherent to monocular cameras [75]. Detecting loop closures, *i.e.* when the user revisits a place previously seen before, is a fundamental capability for SLAM algorithms. Indeed, closing loops both reduces the size of the map and helps decrease the overall error on the map and the localization estimate by adding more constraints. Recent work based on the "bag-of-words" model [67] demonstrates successful loop closure using vision [1, 19, 40].

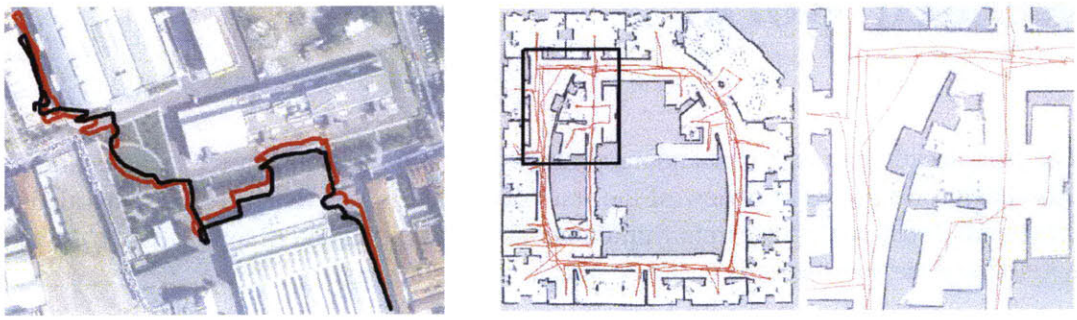

Figure 2-1: *Left*: Vision-based SLAM over a 650 m trajectory. Red: GPS ground-truth. Black: visual odometry (courtesy of [43]). *Right*: Laser-based SLAM in a 30×30 m building, Intel Research Lab dataset. (courtesy of [32]).

In its naive form, the complexity of SLAM grows quadratically with the number of landmarks in the map, which clearly does not scale to large environments. Much of the recent research effort in SLAM has therefore focused on reducing the complexity of SLAM algorithms. One approach consists of observing that only a subset of the map needs to be updated at sensor rate, and updating the global map at a lower frequency [34, 47]. Also, an alternative representation of the state estimate uses the inverse of the covariance matrix (known as the information matrix) which turns out to be sparse for large-scale maps [21, 22]. Finally, submap methods define a local coordinate frame and arrange submap structures into a hierarchical framework [1, 5, 35, 54, 56, 62, 108]. In general however, the majority of SLAM algorithms rely on some form of metric mapping.

Despite its well-deserved success, metric SLAM suffers from several inherent limitations. First, SLAM algorithms are often sensitive to degenerate user motions. For example, any rotation along an axis passing through the center of projection of a monocular camera system prevents from recovering the depth of the scene. Yet, this kind of motion is quite natural from a human perspective. Second, SLAM is sensitive to degenerate world configurations. A long, straight, uniform corridor is a typical configuration that breaks any laser-based SLAM algorithm. Third, most current SLAM approaches assume a static environment. Although recent research tackles the challenging problem of dynamic SLAM [7], much work remains to be done in this direction. Finally, the majority of SLAM algorithms assume full sensor calibration. Although camera calibration and laser calibration have been extensively studied, the calibration of a multi-sensor rig remains a challenging problem today. The work presented in this thesis aims at addressing these issues.

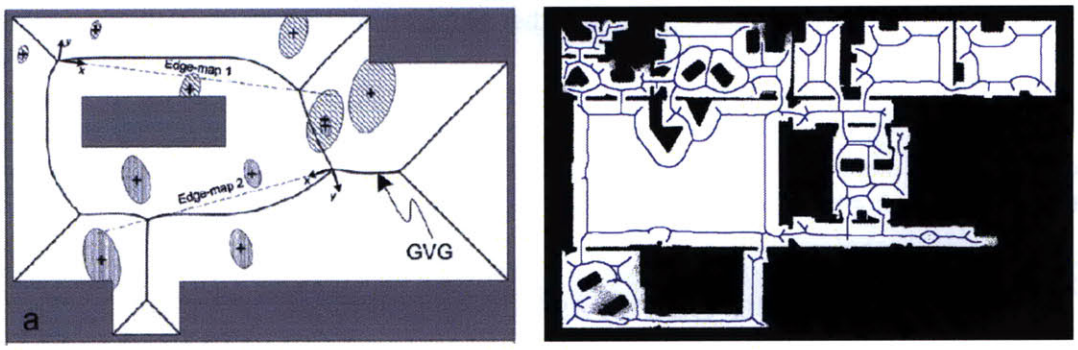

Figure 2-2: *Left*: Low-level *edge maps* (sets of features) are organized into a high-level topological structure based on the generalized Voronoi diagram (GVG). Courtesy of [56]. *Right*: Skeleton of the free space in the map using an extended Voronoi diagram (EVG). Courtesy of [5].

2.2 Visual Homing

The problem of *visual homing* consists of guiding an autonomous agent from an arbitrary start pose towards a goal pose defined by an image taken there [29]. The term "homing" is borrowed from the biology literature, where it usually describes the ability of living organisms such as insects to return to their home location [15]. Visual homing methods fall into two categories: those using depth information (SLAM) and those using intensity information. In this section, we focus on the latter.

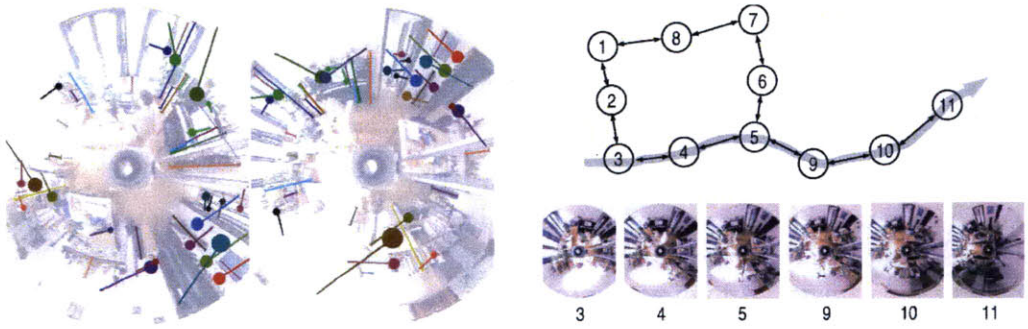

Figure 2-3: *Left*: A pair of omnidirectional images. Radial lines show line features. Circles with tails show SIFT features. Courtesy of [29]. *Right*: In combination with topological mapping, visual homing provides an end-to-end solution to vision-based navigation. Courtesy of [30].

Intensity-based visual homing is intuitively simpler than SLAM in that it does not require recovering a map of the environment *a priori*, yet performs successfully in a wide range of situations. The first type of approach is holistic, in a sense that it treats the image as a whole. Typical approaches include image warping [27], parametric methods such as contours [63] and Descent in Image Distances (DID) methods [107]. The latter approach relies on the observation that the image distance between two snapshots typically increases

smoothly with spatial distance. On the other hand, matching-based methods explicitly solve the correspondence problem. The first class of methods uses dense information, *i.e.* optical flow [101]. The second class of methods uses sparse features, such as SIFT or snapshot models [64]. In combination with topological mapping, visual homing can provide an end-to-end solution to vision-based navigation [28, 30, 31, 93]. It is also worth mentioning that visual homing has also inspired SLAM algorithms such as RatSLAM [61].

2.3 Uncalibrated Cameras

A variety of research directions have been explored based on uncalibrated cameras. Early work was concerned with approaching geometric problems related to multiple views. Seminal works show that projective reconstruction can be performed from line or point correspondences in three or more images [23,38,74]. In addition, a full Euclidean reconstruction may be obtained by adding metrical constraints [36, 38]. These methods may present convergent issues but have inspired a wide number of algorithms for structure-from-motion based on calibrated and uncalibrated cameras [68]. Methods have also been presented for reconstructing surfaces from uncalibrated views of contours [84] as well 3D curve reconstruction [103].

Obstacle detection is an another research area where uncalibrated cameras may be used successfully. A general approach relies on the idea that a static object observed by a moving camera generates a Field of Expansion (FOE) in image space. If the FOE does not move in image space, this means that the camera is moving along a line passing through the object (collision course). Other methods rely on computing the dense disparity map using the fundamental matrix [98] or using the normal optical flow [81].

Uncalibrated cameras have also been used to tackle the navigation problem. A method is presented for 2D robot navigation in man-made environments based on uncalibrated monocular images [33]. The method extract lines and estimates the rotation of the robot using the homography of lines at infinity. The algorithm then estimates free space ahead of the robot using line matches between pairs of images. A navigation method is also derived using a general camera model to represent an omnidirectional camera [11]. The algorithm takes as input two images of a ground plane and estimate the rotation center and rotation angle of the camera, as long as the rotation center is visible in both images. Finally, a method defines a set of image measurements that are invariant to intrinsic parameters of a camera and use them to solve the relative orientation problem from a set of correspondences [104]. To the best of our knowledge however, there exists no method that addresses both the localization problem and the navigation problem in arbitrary environments using fully uncalibrated cameras.

Chapter 3

Image Features

Camera images are rich in information. A typical image of 640×480 8-bit pixels represents 300 KB of information. At a standard frame rate of 30 Hz, this corresponds to a data flow of 8 MB/s. In comparison, a state-of-the art 3D laser range finder such as the Velodyne HDL-64E has an output rate of 4 MB/s. Until recently, such a high data rate has been a barrier against advanced real-time algorithms based on the raw data. The strategy to cope with the density of vision information consists of replacing the camera image by a sparse representation often referred to as *image features*. Image features come in a variety of forms such as points, lines, contours or blobs of pixels. In this chapter, we present an overview of the modern image features.

3.1 Detectors and Descriptors

The task of computing image features is often broken into two complementary sub-tasks. On one hand, a *feature detector* determines the location of features in the image. On the other hand, a *feature descriptor* amounts to a dense representation of the feature. The evaluation of a given detector or descriptor depends on the application. However, there exist several characteristics that are generally desirable for a feature detector, in particular the robustness to changes in viewpoint and to changes in lighting conditions. We use these two evaluation metrics to compare various feature descriptors in this section.

3.2 Point Features

The first point features to appear in the literature were corner features [37,88]. They present the advantage of being fast to compute but are not very descriptive. However, they are still used in many computer vision applications today. Since then, a number of more elaborate point features have been discovered. FAST features (Features from Accelerated Segment Test) consider a circle of pixels around a candidate point [78, 79]. If more than a fixed number of contiguous pixels are brighter by some margin than the nucleus, then the point is considered a feature. Based on the scale-space theory of images, the SIFT features [57] find the local extrema in the pyramid of difference of Gaussian images, computed at various

scales (Figure 3-1). These features demonstrate robustness to scaling, rotation, illumination changes and local geometric distortion. Their main drawback is their relative high computational cost. Several parameters allow to control the number of keypoints found in an image (Gaussian kernel size, number of octaves, number of levels per octave). More recently, SURF features (Speeded Up Robust Features, [4]) have demonstrated similar performance to SIFT at a much lower computational cost, thanks to the use of integral images, also known as summed area tables. The feature descriptor also benefits from integral images and can be computed quickly. The SURF features descriptor is a 64-byte vector and may be extended to a more descriptive, 128-byte version. Finally, DAISY features overcome the high computational cost of SIFT and SURF when the descriptor is computed at every pixel [96].

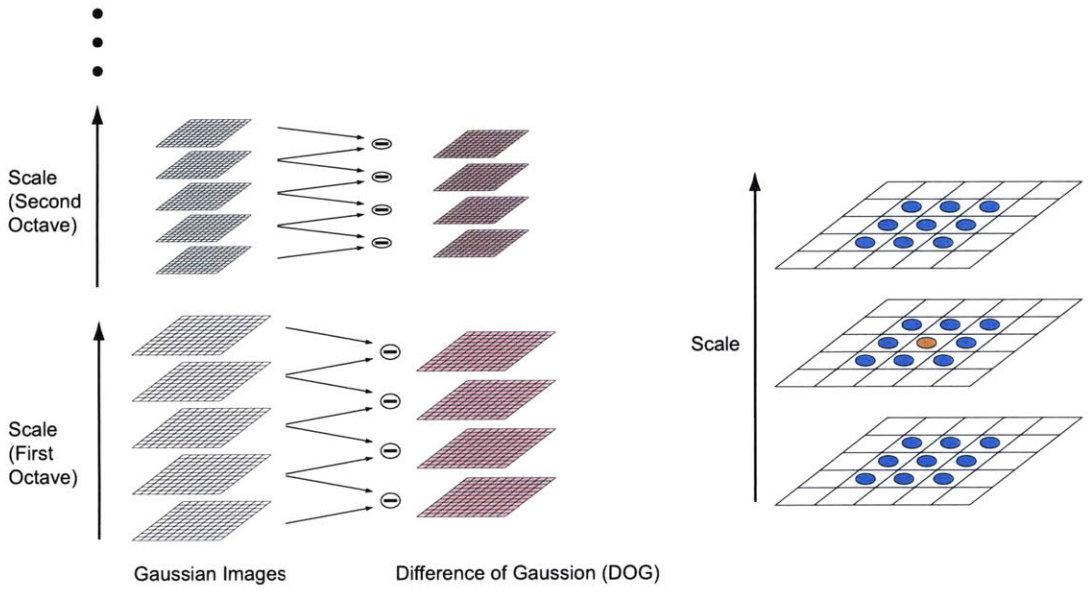

Figure 3-1: Scale Invariant Feature Transform (SIFT). *Left*: for each scale, the image is repeatedly convolved with Gaussians. Adjacent Gaussian images are subtracted to produce the Difference of Gaussian (DOG) pyramid. The image is downsampled by a factor of 2 between each scale. *Right*: features correspond to local extrema in the scale space pyramid. Each pixel value is compared to its 26 neighbors.

3.3 Line Features

When the environment is man-made and structured, it is reasonable to assume the presence of straight lines in the world. Assuming a projective camera model, 3D lines in the world project to 2D lines in the camera image. Hence, feature lines have been extensively studied in the literature [24, 76]. Line features are, in nature, more robust than point features to occlusion. In addition, when used for localization or structure-from-motion, they tend to provide a higher accuracy than points, since they make use of more pixels in the source image.

3.4 Blobs of Pixels

Point features may not be available in textureless regions of an image. Blob descriptors therefore come as a good complement to point features. The intuition is to consider not a single point location in the image, but a group of pixels or *blob*. The most common blob detector is based on the Laplacian of Gaussian. Given an input image, the detector first convolves the image with a Gaussian kernel, then computes the Laplacian operator on the convolved image. Bright and dark blobs correspond to local extrema in the image. The SIFT detector builds on a multi-scale extension of the Laplacian of Gaussian detector. Recently, the Maximally Stable Extremal Regions (MSER) detector has been developed based on the theory of level sets in the intensity map [59]. The detector is invariant to affine transformations of image intensity, multi-scale and highly repeatable but is also sensitive to light change. The comparison of affine region detectors is the subject of a recent study [60].

3.5 Point Feature Matching

Feature matching consists of determining matches between two sets of features A and B. The most common approach is based on the nearest neighbor algorithm. It consists of finding, for each feature in A, the closest feature in B given a distance measure in the descriptor space. It is common practice to use the L_2 norm as the distance function. Symmetry is a highly desirable property of feature matching. Hence, matching is often combined with mutual consistency check, by which a feature in A is matched to a feature in B if and only if that feature in B would be matched to the feature in A. A similar yet looser constraint is monogamy. In this case, no two features in A may match the same feature in B. Mutual consistency check does not imply monogamy (Figure 3-2). Finally, a standard approach consists in rejecting weak matches using the second neighbor ratio method. Given a ratio $0 < \rho < 1$, a match between a feature in A and a feature in B is accepted if the ratio of distances with the second neighbor is smaller than ρ, i.e. if :

$$d(f_i^A, f_j^B) < \rho \cdot \min_{k \neq j} d(f_i^A, f_k^B), \qquad 0 \leq i <\mid A \mid, 0 \leq j, k <\mid B \mid \tag{3.1}$$

where f_i^A is the i-th feature in A, f_j^B is the j-th feature in B and $d()$ is the distance function in the descriptor space.

3.6 Optimized Feature Matching

The matching algorithm described in § 3.5 requires computation of the distance between every pair of features and therefore has a complexity of $\mathcal{O}(n^2)$. Real-time applications often require more efficient matching methods. There exists an extensive literature on this topic. One strategy consists in reducing the size of the descriptor using such methods as Principal Component Analysis [46]. Another idea is to use intrinsic parameters of the features to speed up matching. In the case of SURF, for example, the sign of the Laplacian may be used to discard mismatches, hence allowing a best-case speed up of two. In a more general

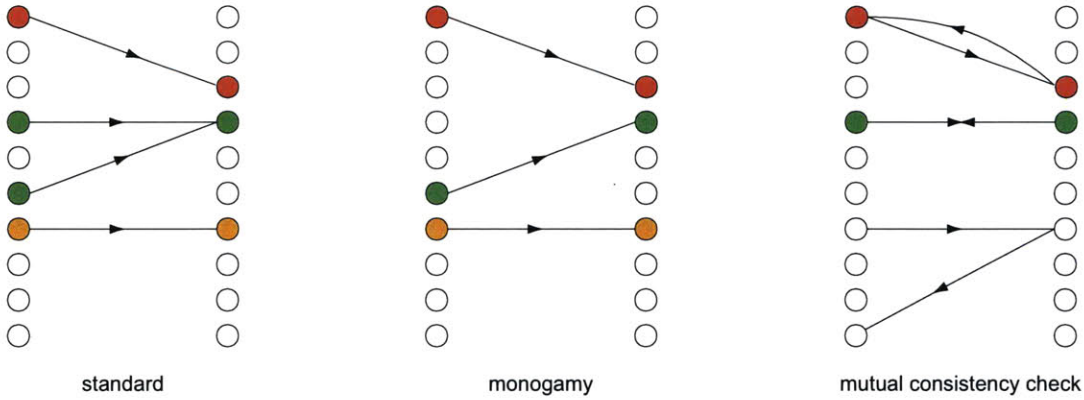

Figure 3-2: Feature matching between two sets. Arrows represent links between a feature in one set and the most similar feature in the other set. Colors represent accepted matches. *Left*: standard matching. *Middle*: monogamous matching. *Right*: mutual consistency check.

way, it is possible to optimize feature matching by using a Best-Bin-First approach [6] or a K-means based tree [44]. However, these methods are efficient only for large feature sets.

Yet, another way of optimizing feature matching is to take advantage of recent progress in the field of linear algebra computation. Indeed, if feature descriptors are normalized, minimizing the L_2 distance is equivalent to maximizing the dot product, since for two unit vectors u and v, $\mid u - v \mid^2 \;=\; \mid u \mid^2 + \mid v \mid^2 - 2 \cdot u \cdot v \;=\; 2 - 2 \cdot u \cdot v$. Given two sets of features A and B, the method consists of stacking the feature descriptors of the first set in a matrix A (one feature per row) and the feature descriptors of the second set in a matrix B (one feature per column). Then, the matrix $C = A \times B$ contains the dot product between each pair of feature. We found this method to outperform the state-of-the-art tree-based method [71] for set sizes as large as 10^5 (§ 4.5.2). However, for larger vocabularies, the tree-based approach is more efficient.

3.7 Feature Matching Performance Evaluation

We evaluate the feature matching algorithm using a set of images collected in the first floor of the Stata Center at MIT. We compare the performance of the method for various settings of the matching threshold ρ, various feature types (SIFT, SURF-64, SURF-128, FAST) and various viewpoint baselines (one meter, three meters). In each case, ground-truth is obtained by manually specifying correct matches using a user interface designed for this purpose.

To measure the quality of matching, we consider two quantities: the *precision* and the *recall*. The precision is the true positive rate, *i.e.* the ratio of elements in one set that also belong to the reference set (*ground-truth*). On the other hand, the recall is the fraction of elements in the reference set that appear in the set. Precision and recall are independent

terms. Formally, they can be defined as:

$$\text{Precision} = \frac{tp}{tp + fp} \tag{3.2}$$

$$\text{Recall} = \frac{tp}{tp + fn} \tag{3.3}$$

where tp is the number of true positives, fp is the number of false positives and fn is the number of false negatives. In addition, we consider the F-measure, defined as the harmonic mean of precision and recall.

$$\text{F-Measure} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.4}$$

A high F-measure represents a good trade-off between precision and recall. Ideally, a perfect feature matching algorithm provides a recall of 1 for any precision. In practice, the recall increases as the feature matching algorithm is more and more compliant, while the precision decreases.

Figure 3-3 compares the precision-recall performance of the standard matching algorithm (STANDARD), with the monogamous constraint (MONOGAMY) and the mutual consistency check (MCC) on a three-meter baseline dataset. As we can see, monogamy and mutual consistency check each bring significant improvements in matching. Since these constraints can be implemented at very little cost, we only consider the mutual consistency check algorithm in the next experiments. As the matching threshold increases, the precision of the algorithm decreases, but the recall increases. This experiment sheds light on which matching threshold is optimal. The maximum F-measure is obtained for $\rho = 0.8$. We use this value in the rest of this work. Figure 3-6 shows the frames used in the experiment and the feature matches for each algorithm ($\rho = 0.8$).

Figure 3-4 shows the relative matching performance using SIFT, SURF-64, SURF-128 and FAST features. Here again, the baseline between the two frames is three meters. The FAST features perform significantly worse than the other feature types. This is expected, since the matching algorithm we use is based on normalized cross-correlation and does not handle geometric distortions very well. We also observe that SURF-128 performs slightly better than SURF-64, which is in accordance with the fact that the latter provides less descriptive features than the former. Finally, SIFT performs better than all other feature descriptors. This finding is in disagreement with the original claim of the SURF authors [4] but in agreement with further experiments [80]. Finally, Figure 3-5 shows the performance of the two best feature types (SIFT and SURF-128) on a one-meter baseline and three-meter baseline. As expected, SIFT outperforms SURF-128 in each case. Also, each feature type performs significantly better on a one-meter baseline.

3.8 Discussion

In this chapter, we presented some of the most common point features found in the computer vision literature. We demonstrated a fast and robust feature matching algorithm based

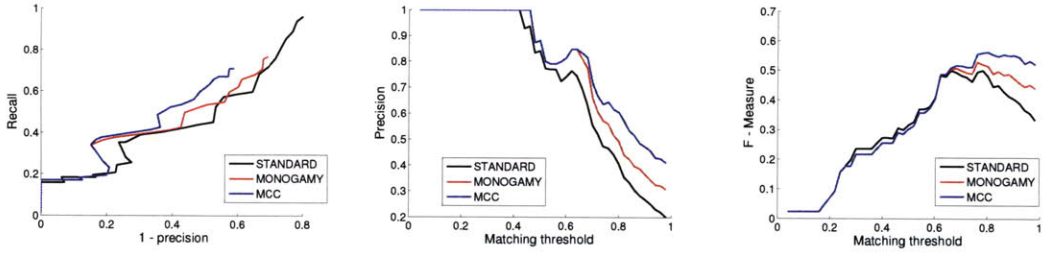

Figure 3-3: Precision-recall curves for SIFT feature matching for the standard algorithm, with monogamy and mutual consistency check. The baseline between the two frames is three meters.

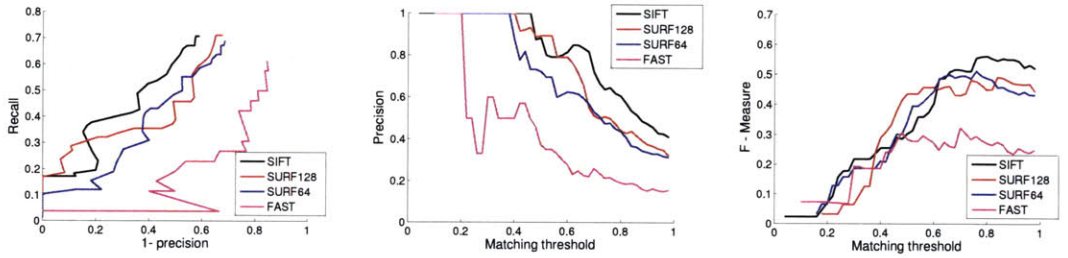

Figure 3-4: Precision-recall curves for various feature types.

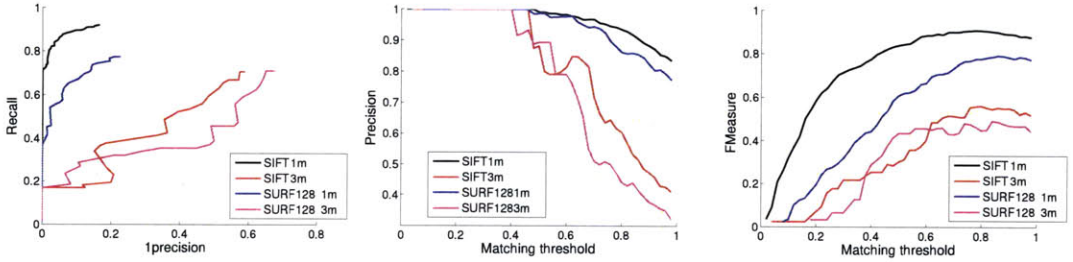

Figure 3-5: Precision-recall curves for SIFT and SURF-128 for a one-meter baseline and a three-meter baseline.

on a nearest-neighbor search. We found the SIFT features to outperform other feature detectors, but only by a relatively small margin in the case of SURF. Despite its high computational cost, SIFT remains a viable choice in our work since we are working with low resolution images (320×240 or smaller) hence allowing real-time SIFT performance on a portable computer.

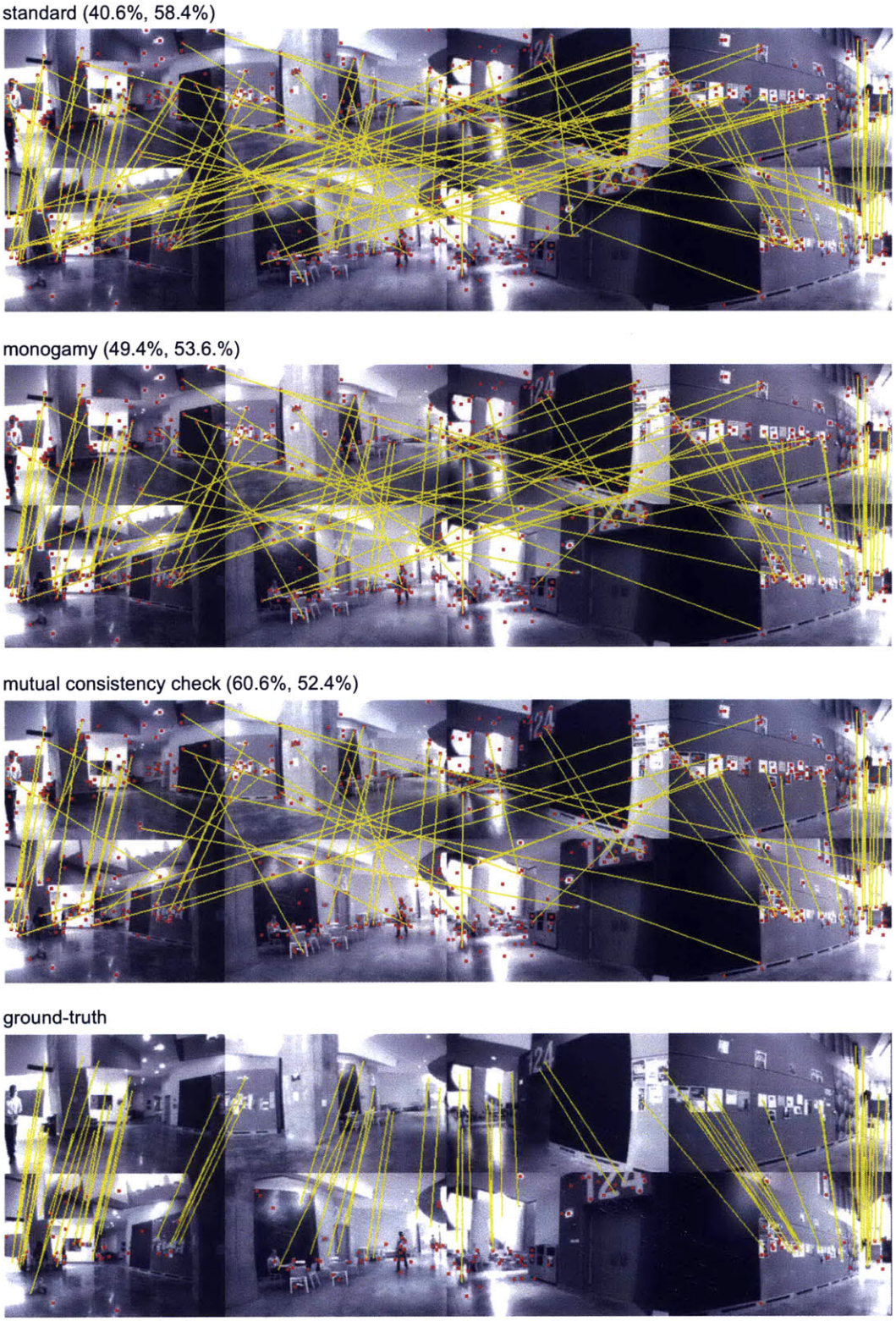

Figure 3-6: SIFT feature matching using the standard algorithm with monogamy and mutual consistency check constraints. The numbers indicate the precision and recall values. The baseline between the two frames is three meters.

Chapter 4

Body-relative Navigation Guidance using Uncalibrated Cameras

In this chapter, we present a vision-based navigation method that relies solely on a set of uncalibrated cameras mounted arbitrarily on the user. 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). 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.*

4.1 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 nodes. We refer to a topological map as a *place graph*, described in detail in § 4.1.1. Given the place graph representation, we cast the navigation problem as a "node-to-node hopping" problem (Figure 4-1). One sub-method, *local node estimation* determines the location of the user within the graph (§ 4.1.2). Another sub-method, *rotation guidance*, provides directional indication to the user at each node (§ 4.1.4). Our approach provides no guidance along edges and assumes that there is no ambiguity about the direction to follow between nodes. This assumption is reasonable in real-world environments as long as the node density is large enough. We discuss the limitations of this assumption in Chapter 6. Finally, a *loop closure* sub-method detects return visits to previously traversed places, and updates the graph accordingly (§ 4.1.5).

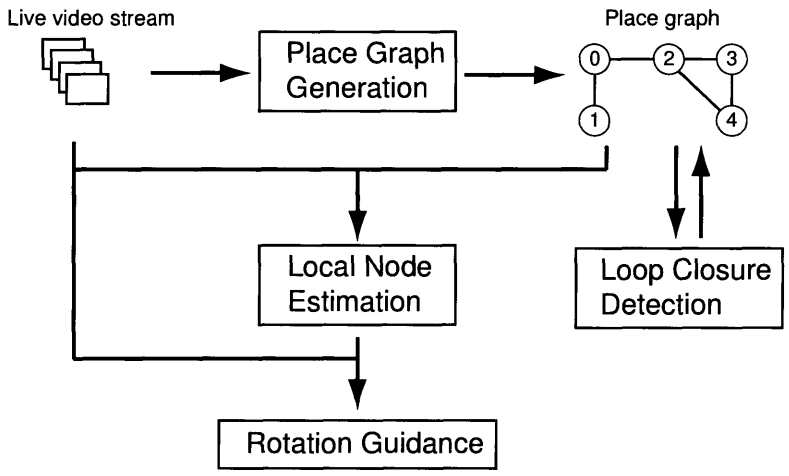

Figure 4-1: Method Overview. Our method takes as input a live video stream captured from a set of body-worn, uncalibrated cameras. It generates a topological representation of the explored environment (*place graph*) and estimates the location of the user in the graph (*local node estimation*). A *rotation guidance* module 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.

4.1.1 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 z_v. We use SIFT features [57] with 128-byte descriptors. An edge $e \in E$ represents a direct 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 § 4.1.5, we describe how to update the graph due to loop closure events. Since a node v may have several neighbors $\{v_k\}$, the set z_v is in fact a set of observations $z_v = \{z_{v,v_k} \mid (v, v_k) \in E\}$, where z_{v,v_k} represents the observations made at node v on the way to v_k. Figure 4-2 illustrates the place graph data structure.

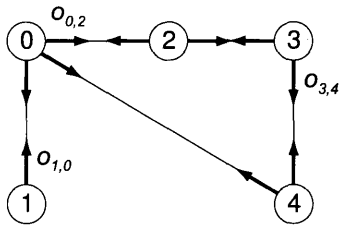

Figure 4-2: We represent the world as an undirected graph where nodes represent physical locations and edges represent physical paths between locations. Each node is associated with the observations made en route to and from each of its neighbors.

To determine when a node should be created, we define an energy function Ψ for two

sets of input observations. This function matches features between the two sets and returns the normalized averaged sum-of-square distance between matches using the L1 distance in the feature descriptor space. We use the Φ matching function described in § 4.1.3. A new node is created when the energy function computed between the current observations and the observations of the latest node exceeds an experimentally-determined threshold (we use 0.85).

$$\Psi(F_1, F_2) = \frac{1}{|F_1| + |F_2|} \cdot \left(|F_1| + |F_2| - 2 \cdot |\Phi(F_1, F_2)| + \sum_{(f_1^i, f_2^j) \in \Phi(F_1, F_2)} \left\| f_1^i - f_2^j \right\|_2 \right) \quad (4.1)$$

4.1.2 Local Node Estimation

The goal of local node estimation is to maintain an estimate of the user's position within the graph. 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 $p(x_{k+1} \mid x_k)$:

$$p(x_{k+1} \mid z_k) = \sum_k p(x_{k+1} \mid x_k) p(x_k \mid z_k). \quad (4.2)$$

Then, the update step incorporates the observation model $p(z_{k+1} \mid x_{k+1})$ to obtain $p(x_{k+1} \mid z_{k+1})$:

$$p(x_{k+1} \mid z_{k+1}) = \lambda p(z_{k+1} \mid x_{k+1}) p(x_{k+1} \mid z_k), \quad (4.3)$$

where λ is a normalization factor. In practice the distribution is updated only on a local neighborhood around the current user'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 Gauss window function (Figure 4-3).

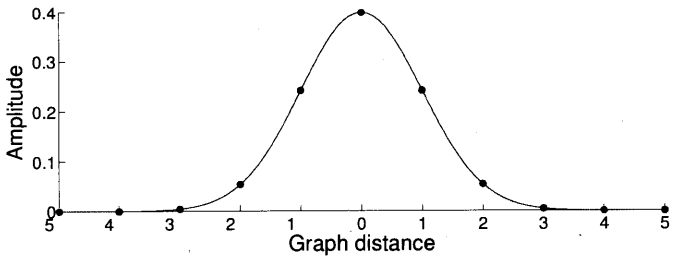

Figure 4-3: Gauss window function representing the motion model during localization. The standard deviation is a parameter of the method (we use $\sigma = 1.0$).

In addition, we define the observation model $p(z_k \mid x_k)$ using the Ψ distance introduced in 4.1.1:

$$p(z_k \mid x_k) = 1/(1 + \Psi(z_k, z_{x_k})) \quad (4.4)$$

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

4.1.3 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 chapter, 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 a single time. The algorithm Φ takes as input two sets of observations $\{z_i, z_j\}$ (or alternatively, two 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 [25] provides fast matching (see § 4.1.5).

4.1.4 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, preserves navigation-relevant information when the number of observations is large.

More specifically, we define the *relative orientation problem* as follows. Assuming that the user revisits at time t' a location visited at time $t < t'$, the problem is to determine the orientation of the user at time t' *relative* to her orientation at time t (Figure 4-4). In this work, we restrict ourselves to the 2D case, in which the orientation is a single scalar value. However, our method extends naturally to the 3D case.

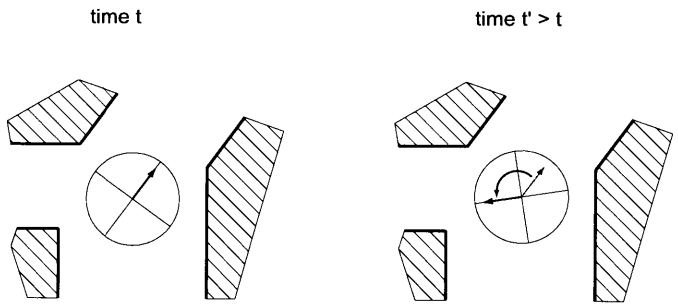

Figure 4-4: The relative orientation problem consists of determining the orientation of the user at a given location *relative* to her orientation at the same location at some previous time.

The intuition underlying our method is that if an image feature corresponding to a static 3D point is observed on a camera at time t and re-observed on a different camera at time t', the correspondence carries information about the relative orientation of the user. If extrinsic and intrinsic calibration parameters are known for all cameras, this is a trivial

statement since the relative orientation may be computed directly from the two feature bearings. However, we demonstrate that it is possible to estimate the relative orientation with no prior intrinsic or extrinsic calibration.

Let us consider a single camera observing 3D world features (Figure 4-5). If the camera does not move, and assuming perfect feature matching and a static world, the probability to re-observe a feature at a later time is 1. However, as the camera rotates, the average probability to re-observe any given feature decreases linearly with the angle of rotation. If the camera rotates by more than f, where f is the horizontal field of view of the camera, the probability becomes zero. Consequently, the probability to re-observe a feature follows a triangular distribution, as shown on Figure 4-5 (again, we assume that the distribution of features is uniform in image space).

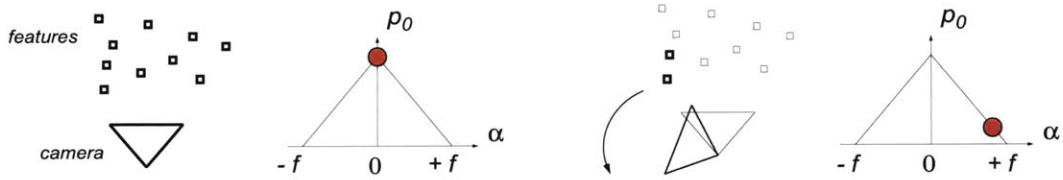

Figure 4-5: Assuming a static world and perfect feature matching, the probability of re-observing a 3D world feature on a rotating camera follows a triangular distribution centered on zero, with limits $-f$ and f, where f is the camera's horizontal field of view.

Let us now extend this reasoning to two rigidly attached cameras (Figure 4-6). Each camera makes observation at the first time. As the camera rig rotates, the probability of observing with one camera a feature that was previously observed by the other camera (*the re-observation probability*) also follows a triangular distribution. The distribution is no longer centered on zero. However, the limits of the distribution are still defined by f. We finally extend the method to an arbitrary number of n cameras. For each pair of cameras (C_i, C_j), there exists a re-observation probability p_{ij} that follows a triangular distribution $p_{ij} = \tau(h_{ij}, \sigma_{ij}^2)$. The mean h_{ij} is a function of the geometric configuration of the cameras on the rig (in particular, $h_{ij} = 0$ if $i = j$). The variance σ_{ij}^2 is a function of the field of view of the cameras. If all cameras have the same field of view, $\sigma_{ij}^2 = f^2/6$. We represent the set of distributions $\{p_{ij}\}$ as an $n \times n$ matrix, which we call the *match matrix* (Figure 4-6).

Learning the Match Matrix

We propose an algorithm for learning the match matrix from training data. During training, the camera rotates in place in an arbitrary environment at a known constant speed ω. Let us assume that a static 3D world feature is observed at time t on camera i and at time $t' > t$ on camera j. This occurrence is an instance of p_{ij} for an angle $\alpha = \omega \cdot \Delta t$, where $\Delta t = t' - t$. By matching features between each pair of frames, we obtain a large number of occurrences that allow to learn each distribution p_{ij}. Specifically, from a series of p occurences $\{\alpha_{ij}^0, \cdots, \alpha_{ij}^{p-1}\}$ for each pair (i, j) we extract an estimate of h_{ij} and σ_{ij}^2. As p becomes larger, the averaged estimate gets closer to the true values. In practice, the training sequence contains roughly 300 frames. Each frame contains 600 features,

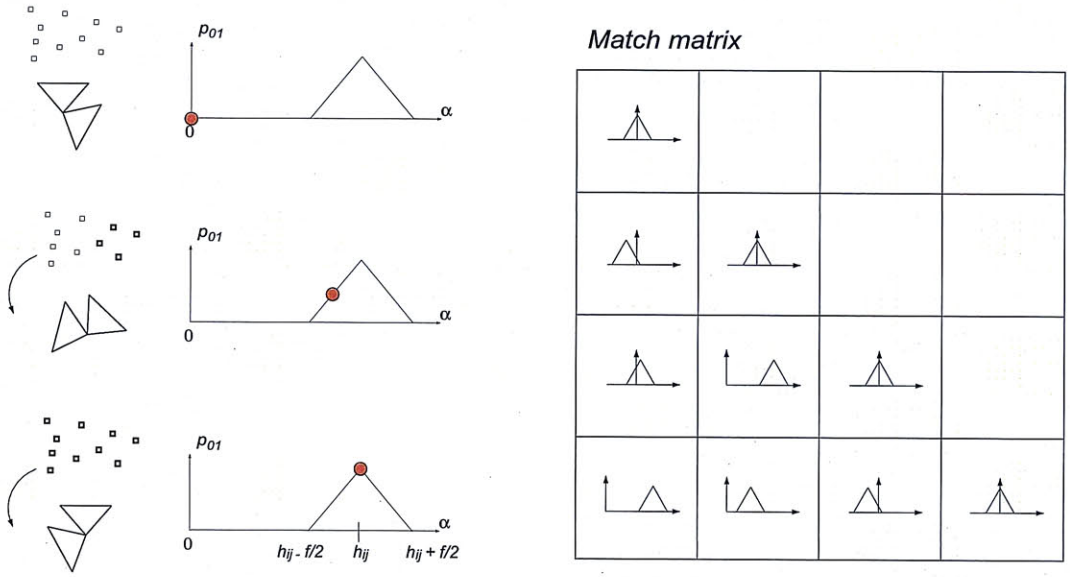

Figure 4-6: For two rigidly mounted cameras, the probability to re-observe with one camera a feature previously observed by the other camera p_{01} follows a triangular distribution with respect to the rotation angle. By extension, for n cameras, there exists $n \times n$ distributions p_{ij} which we represent as the *match matrix*.

which yields about three million instances per camera pair. The assumption of constant rotation speed need not be satisfied exactly. An error of δ degrees spread over the training sequence generates an error on ϵ that is linear in δ. Simulations show an error $\epsilon = 4.7°$ for $\delta = 20°$, which is acceptable for our application. In addition, our method includes an automatic estimation of the rotation speed ω by detecting a complete turn when the average image-space distance between feature matches reaches a local minimum.

Table 4.1 summarizes the training algorithm. The matrix H contains the mean values h_{ij}. 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.* $\bar{\eta} = \arctan(\sum \sin(\eta_i) / \sum \cos(\eta_i))$. Note that by symmetry, the matrix H is anti-symmetric, *i.e.* $H(i,j) = -H(j,i)$ for any pair (i,j) and in particular, $H(i,i) = 0$ for any i.

We emphasize that the training method is fully automated, need be performed only once for any given camera configuration, and is independent of the training environment. The matrix H is therefore significantly easier to compute and more compact than the full set of intrinsic and extrinsic parameters would be.

The Training Algorithm

Input: a training video sequence of f frames
Output: the $n \times n$ match matrix H (for n cameras)

1: Initialize $H(i,j) \leftarrow 0, 0 \leq i,j < n$
2: Initialize $H_s(i,j) \leftarrow 0, 0 \leq i,j < n$
3: Initialize $H_c(i,j) \leftarrow 0, 0 \leq i,j < n$
4: **for** each pair of frames (f_p, f_q) in the sequence **do**
5: Estimate the user rotation angle α_{pq} linearly
6: **for** each match $m_k = (f_{k,p}, f_{k,q}) \in \Phi(f_p, f_q)$ **do**
7: Let $s_{k,p}$ ($s_{k,q}$) be the camera ID for $f_{k,p}$ ($f_{k,q}$)
8: $H_s(s_{k,p}, s_{k,q}) \leftarrow H_s(s_{k,p}, s_{k,q}) + \sin(\alpha_{pq})$
9: $H_c(s_{k,p}, s_{k,q}) \leftarrow H_c(s_{k,p}, s_{k,q}) + \cos(\alpha_{pq})$
10: $H(i,j) \leftarrow \arctan(H_s(i,j)/H_c(i,j)), 0 \leq i,j < n$

Table 4.1: The Training Algorithm

Solving the Relative Orientation Problem

Given the match matrix, we now present a method to solve the relative orientation problem. Let us assume that the frame observes a set of features S^t at time t and the same set of features $S^{t'}$ at time $t' > t$. We wish to determine α, the relative orientation of the user at time t' with respect to its orientation at time t. Given a match between a feature $f_i^t \in S^t$ on camera i and a feature $f_j^{t'} \in S^{t'}$ on camera j, an estimate of α is $\hat{\alpha} = h_{ij}$. For a single match, this is a gross estimate of α. Indeed, the associated error is $\epsilon = \alpha - \hat{\alpha} = \tau(0, \sigma_{ij})$ where $\sigma_{ij} = \sigma = f^2/6$. For instance, $\sigma = 36°$ for $f = 90°$. However, for a large number of matches N, the estimate of α becomes:

$$\bar{\alpha} = \frac{1}{N} \sum^{N} h_{ij} \tag{4.5}$$

The associated error is:

$$\epsilon = \bar{\alpha} - \alpha = \frac{1}{N} \sum^{N} \tau(0, \sigma) \tag{4.6}$$

According to the Central Limit Theorem, the average of a set of independent and identically distributed random variables approaches the normal distribution with a mean μ_N and a variance σ_N^2 as the sample size N increases.

$$\epsilon \approx \mathcal{N}(\mu_N, \sigma_N^2) = \mathcal{N}(0, \frac{\sigma^2}{N}) \tag{4.7}$$

As a conclusion, the error on the relative orientation of the user follows a normal distribution for a large number of observations. For any desired standard deviation and camera field of view, there exists a mimimum number of observations that can achieve it. For instance, to achieve $\sigma_N = 2.6°$ using cameras with $f = 90°$, we need at least $N = 200$ observations.

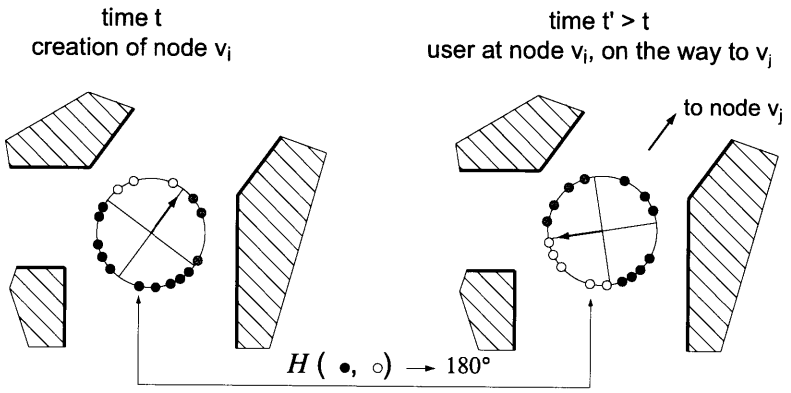

Figure 4-7: The rotation guidance algorithm matches observations between an earlier visit to some node (*left*) and a subsequent visit (*right*). Observations are shaded by camera ID. Each match votes for a rotation that will bring the user into alignment with his or her orientation during the first visit. The arrow represents the forward direction in the user's body frame.

Discussion

We discuss here the various assumptions that our method relies upon and analyze the impact they may have on a real system. First, our method assumes a uniform distribution of features in image space, both during training and during rotation guidance. To address this constraint, it is possible to synthetically transform a non-uniform distribution into a uniform one by sampling features uniformly in image space. This would come at the price of decreasing the number of observations, which would consequently increase the standard deviation of the guidance error. A second solution would be to use visual features that tend to naturally spread uniformly in image space, such as SIFT features, as opposed to other types of features such as FAST that tend to aggregate around corners and edges. Third, combining complementary visual features (SIFT points and MSER regions, for example) can increase the chances of achieving a uniform distribution of features. In practice, however, we found the uniformity assumption to be satisfied in a large class of environments using only SIFT features.

The second major assumption of our method is that features are re-observed from the same location in the world. In other words, the method does not account for situations where the user revisits the same location with an offset lateral to the motion path (baseline). The problem is limited during training since we can easily constrain the system to rotate in place. However, it may become significant during guidance. As a consequence, the lateral offset may prevent a number of features from matching, in particular in tight environments with a significant variability in scene depth. The use of features that are robust to affine transformations (such as SIFT) helps tackle such situations. Second, using a denser place graph reduces the maximum distance between two nodes, which in turns reduces the chances of a lateral offset between the user and the closest node. In essence, the place graph generation algorithm that we propose naturally adapts to the variability in the visual appearance of the scene, thus creating more nodes where the environment changes faster.

On the other hand, lateral motion also creates a geometric problem, since features may not appear on the camera they would have appeared on without baseline. Here again, the problem is even more prominent in tight environments. However, using large field-of-view cameras addresses this problem to some extent. In practice, we observe that our method provides robust guidance for baseline distances up to 10 meters in typical environments.

Third, our method assumes that all cameras have the same field of view. When this constraint is not satisfied, the probabilities p_{ij} are not identically distributed and the Central Limit Theorem is no longer applicable. From a system perspective, this constraint is quite reasonable since it does not set a constraint on the value of the field of view itself. On the other hand, it would certainly be interesting to analyze the effect of variable field of view for the cameras, which we leave for future work.

Finally, our method does not directly account for mismatches and dynamic scenes. One solution could involve actively searching for outliers during feature matching. Using a mutual consistency check or enforcing monogamy, as described in § 4.1.3, greatly improves the quality of matching. However, more sophisticated methods could be used. At the higher level, it is also reasonable to assume that erroneous matches spread evenly across the field of view of all cameras, hence yielding additional Gaussian noise to the rotation guidance output. In practice, we found our method to work well in a variety of environments, even those including large numbers of passers-by (§ 4.5.6).

4.1.5 Loop Closure Detection

Loop closure detection, *i.e.* recognizing that the user has returned to a previously visited 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 [18]. In a second stage, sequences of visually similar nodes are extracted from the similarity matrix.

Scene Similarity using a Dynamic Vocabulary Tree

Our method builds on the standard "bag-of-words" approach [67], 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 similar nodes in the matrix and updates the place graph accordingly. Our method requires no batch processing, no initial vocabulary and takes as input only a stream of image features.

A word in the vocabulary is represented by a sphere in the feature descriptor space. At the beginning of exploration, the vocabulary is empty. As nodes are inserted into the graph, the corresponding visual features are inserted into the vocabulary data structure. Given a fixed word radius r_w, we declare a *word match* as a word whose center lies within r_w of the input feature descriptor using the L_2 distance. The radius r_w is a parameter of the algorithm and influences the performance of the vocabulary ; Figure 4-8 (we use $r_w = 0.8$). 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.

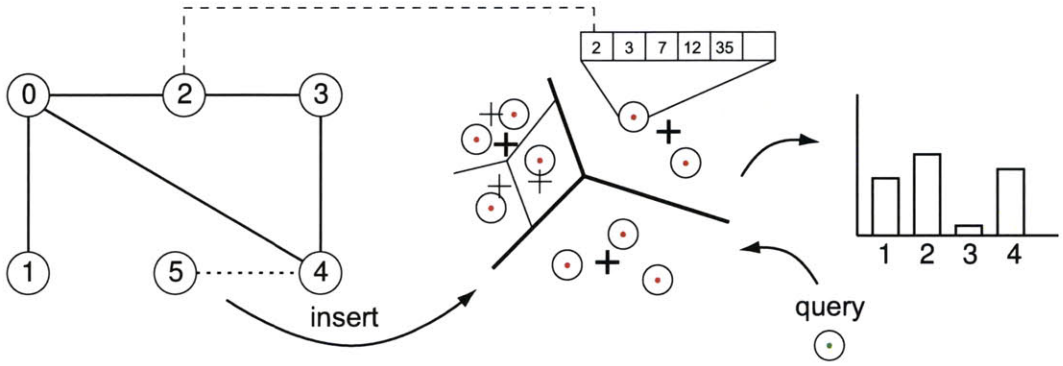

Figure 4-8: The visual features of each new node in the place graph are stored in a vocabulary data structure (*middle*). Each word stores an inverted index pointing to the nodes where it was observed. Search and query are optimized using a tree-based data structure.

A standard approach to optimize search and query in the vocabulary is to maintain a tree-based data structure [1, 71, 106]. Searching for words in these structures is faster than a naive search as long as the number of examined nodes is bounded using a fast approximate search procedure. However, we also propose a method for optimizing the naive search when feature descriptors are normalized. In this case indeed, minimizing the L_2 distance between two features u and v is equivalent to maximizing their inner products, since $\|u - v\|^2 = \|u\|^2 + \|v\|^2 - 2 \cdot u \cdot v = 2 - 2 \cdot u \cdot v$. This approach is particularly powerful when multiple 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 a $p \times m$ 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 also apply this optimization to the exhaustive search through the leaves used in the tree-based method. We found that optimized linear algebra libraries enabled the naive approach to outperform the tree-based method up to a certain vocabulary size, beyond which the tree-based method is faster (see § 4.5.2).

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 4-8). We emphasize that our method is causal and runs online during exploration. Table 4.2 summarizes the algorithm, which runs whenever a node is added to the graph (k represents the run count). At the beginning of the exploration, $k = 0$ and the vocabulary $\mathcal{V}$ is empty.

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 [89] described in [40], which computes an *alignment matrix* A accumulating the score of diagonal moves through S (Equation 4.8). That is, $A(i,j)$ is the maximum similarity of two matching sequences starting 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 history of the moves through A are stored in an index matrix M (Equation 4.9). 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.

$$A(i,j) = \begin{cases} H(i-1,j-1) + S(i,j) & \text{if } H(i-1,j-1) \text{ is maximal} \\ H(i,j-1) + S(i,j) - \delta & \text{if } H(i,j-1) \text{ is maximal} \\ H(i-1,j) + S(i,j) - \delta & \text{if } H(i-1,j) \text{ is maximal} \\ 0 & \text{if } S(i,j) < 0 \end{cases} \tag{4.8}$$

$$M(i,j) = \begin{cases} 1 & \text{if } H(i-1,j-1) \text{ is maximal} \\ 2 & \text{if } H(i,j-1) \text{ is maximal} \\ 3 & \text{if } H(i-1,j) \text{ is maximal} \\ 0 & \text{if } S(i,j) < 0 \end{cases} \tag{4.9}$$

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 $p = 5$). Table 4.3 summarizes the algorithm.

Our loop closure method requires retraversal of short path segments in order to detect loop closures, and is not able to detect loop closures at a single node. This is an important limitation of our method, which could be overcome by detecting single-node loop closures ($p = 1$) and filtering false positives using an *a posteriori* validation method based on structure from motion.

4.2 Motion Classification from Optical Flow

The behavior of a walking person typically falls within a set of distinctive categories: walking forward, turning, climbing stairs, and so on. The ability to determine the state of the user in real-time allows to extract important information that can then be used to augment the situational awareness of the system.

Real-time human motion classification using vision has been extensively studied. A number of methods exist that deal with the classification of human motion from the perspective of an external camera looking at one or more people. Although these methods address a slightly different problem, we mention a few of them here. Studying feature

The loop closure algorithm - Similarity Matrix Update

Input: a set of observations z_{k+1}
Input: the vocabulary $\mathcal{V}_k$
Input: the similarity matrix S_k
Output: the new vocabulary $\mathcal{V}_{k+1}$
Output: the new similarity matrix S_{k+1}

Initialize $\mathcal{V}_{k+1} = \mathcal{V}_k$
Initialize a similarity vector $s = 0_{1\times k}$
Match the observations z_{k+1} to the vocabulary $\mathcal{V}_k$
for each unmatched feature $v \in z_k$ **do**
    create a new word in the vocabulary: $\mathcal{V}_{k+1} = \mathcal{V}_{k+1} \cup \rho(v, r_w)$
    Insert k in the index list of $\rho(v, r_w)$
for each match between $v \in z_k$ and a word ρ_j, $\quad 0 < j < |\mathcal{V}_{k+1}|$ **do**
    for each index k' in the index list of ρ_j **do**
        Increment $s(k') = s(k') + 1$
Normalize $\|s\|_1$ to 1
Update the similarity matrix $S_{k+1} = \begin{pmatrix} \mathbf{s} & 0 \\ S_k & \mathbf{0_{k\times 1}} \end{pmatrix}$

Table 4.2: The Loop Closure algorithm (similarity matrix update).

The loop closure algorithm - Place Graph Update

Input: the similarity matrix S_k
Input: the place graph G
Output: the updated place graph $\bar{G}$

Convolve S_k with a diagonal Sobel operator
Compute the alignment matrix A_k and index matrix M_k using Equations 4.8 and 4.9
for each local maximum in A_k **do**
    Determine the corresponding sequence of moves through A_k using the matrix M_k
    for each match (i_p, j_p) in the sequence **do**
        Merge nodes i_p and j_p in the graph G

Table 4.3: The Loop Closure algorithm (place graph update)

point trajectories allows to discern walking from running motions [83]. A classifier built on support vector machines can learn to classify human motions given a set of example video clips [12]. Finally, using the Lie Algebraic approach [55], standard statistical learning techniques allow to infer common motion patterns from video.

We approach the motion classification problem using a learning technique based on uncalibrated cameras. We assume that a sequence is provided for a set of motion templates. Each motion template consists of a sequence of a few tens of seconds captured while the user performs the corresponding motion. In this work, we use five templates: going for-

ward; turning to the left and to the right; ascent and descent. Our method assumes that an optical flow algorithm is given. We use the iterative Lucas-Kanade tracker [8] although any other optical flow algorithm could be used instead. For each training sequence, the algorithm first computes the instantaneous optical flow at every frame and averages it over the entire sequence. This step is performed off-line, once for each training sequence. The method is agnostic about the number of cameras being used. Indeed, we process and store the average flow field for each camera independently.

Classification does not require storage of the dense flow field. Instead, we sample the image space of each camera using an $n \times n$ grid. The method averages the flow field over each grid cell and stores the average field as a single 2D vector in each cell. Figure 4-9 shows the average flow field for the five motion templates mentioned above. We use $n = 4$. The four cameras are aligned on a single row, resulting in a 4×16 vector matrix for each template. More formally, we represent a flow field $\mathbf{F}$ by a $2 \times p$ vector, where $p = q \times n \times n$ and q is the number of cameras. The training was performed in the first floor of the MIT Stata Center. Figure 4-10 shows one frame of the "ascent" template sequence.

Given a set of p motion templates, we devise a probabilistic classification strategy that takes as input the instantaneous flow field and maintains a probability distribution of the motion class over all possible motion categories. We represent the probability estimate as a $p \times 1$-dimensional normalized vector $\mathbf{p}$, where $p(i)$ represents the probability that the user is executing a motion of class i. The vector is initialized to a uniform distribution at the beginning of the exploration and follows a recursive Bayesian update as new observations are made. We define a similarity measure γ between two input fields $\mathbf{F}$ and $\mathbf{G}$ as the inner product of their vector representations.

During exploration, the dominant motion category is associated to each edge in the place graph and stored with the graph. The guidance algorithm then uses this information during revisitation to help the user anticipate the next moves by providing "turn-by-turn" directions. More specifically, given the position of the user in the place graph and a path to a destination, the algorithm searches for the next motion class in the path and displays the information on the user interface (§ 4.4).

4.3 System Description

We implemented the algorithms described in this thesis on a wearable set of four Point Grey Firefly MV cameras loosely mounted on the shoulder straps of a backpack (Figure 4-11). The total field of view of the system is 360° horizontally and about 90° vertically. For the sole purpose of establishing ground truth (see § 4.5.6), 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.

The data coming from the camera travels through an IEEE 1394 ("Firewire") bus to the computer. The Firewire bus was developed in order to provide interconnection between a computer and external devices at high speed and in real time [92]. One of the major advantages of Firewire over USB is its ability to operate in synchronous mode, which

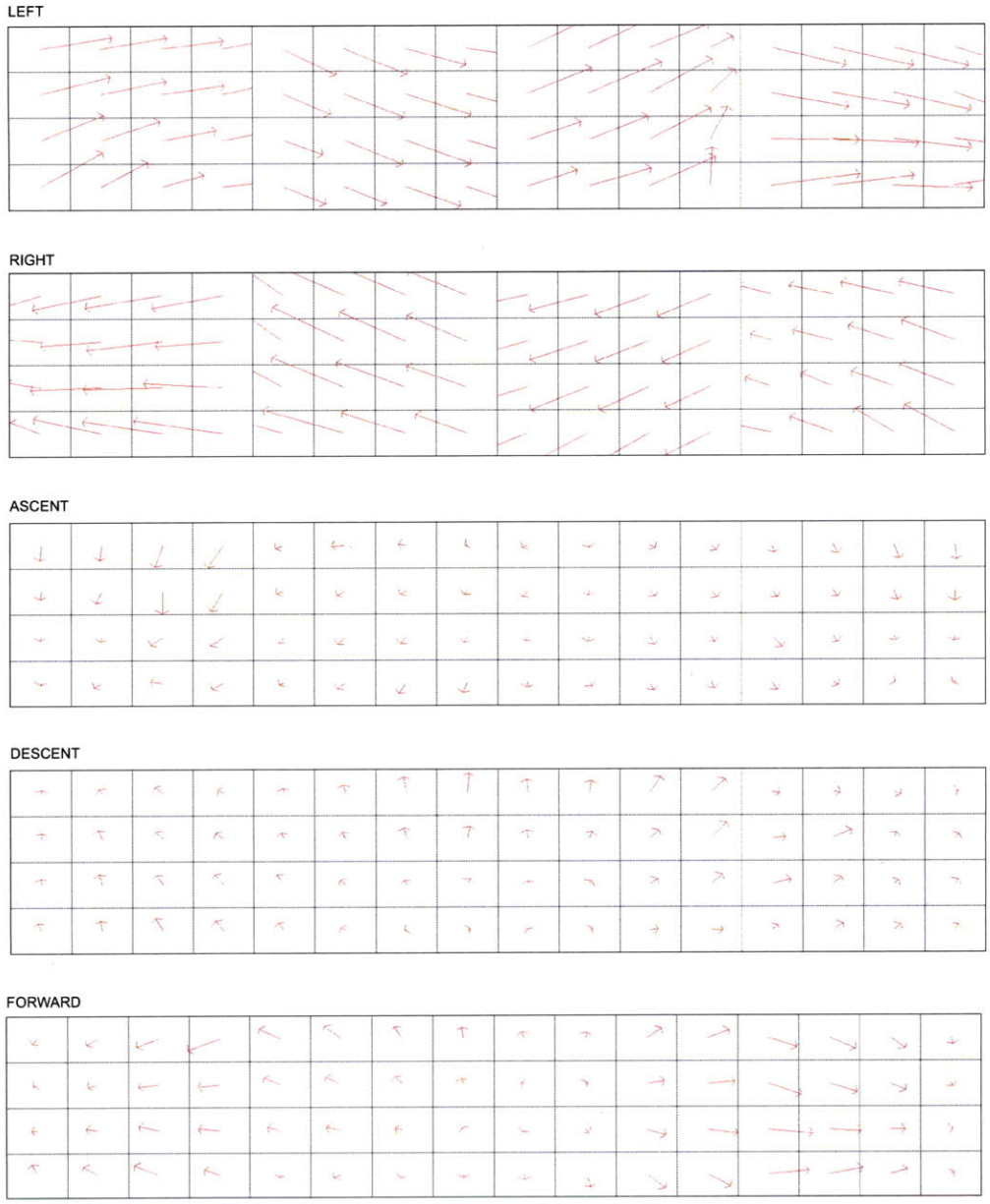

Figure 4-9: Average optical flow field for five template motions. The flow fields exhibit significant differences which allow for accurate motion-type classification.

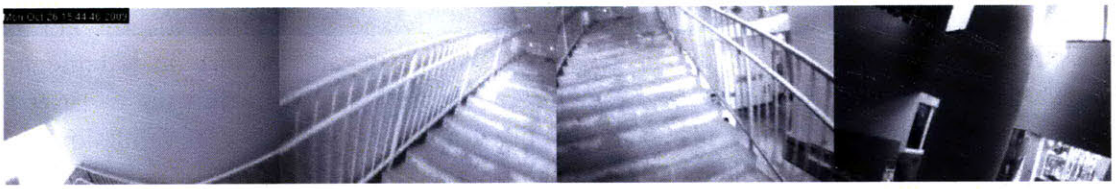

Figure 4-10: A frame of the "ascent" template sequence (MIT Stata Center, 1st floor).

enables sending packets of specific sizes at regular time intervals. The drawback of using Firewire is that the Firewire bus commonly found on laptop computers does not meet the power requirement of a camera. Therefore, an external power supply is required. We use a 12V 4500mAh NiMH battery, and a 800 MB/s Firewire bus capturing four streams of 372×240 8-bit grayscale images at 60 Hz. Due to the computational needs of our algorithms, we reduce the frame rate to 5 Hz. Figure 4-12 shows a sample image frame.

Figure 4-11: System overview including four cameras loosely mounted on the shoulder straps of a backpack. The horizontal field of view is $360°$. The system also incorporates a laptop, a 12V 4500mAh battery and a Firewire hub and can operate for three hours on a battery charge.

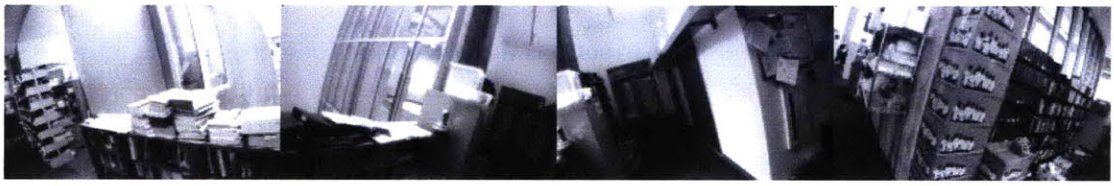

Figure 4-12: Sample frame captured by the system. The second and third (resp. first and fourth) images correspond to the forward (resp. backward) facing cameras.

4.4 User Interface

The system includes a user interface implemented on a Nokia N810 Tablet PC. The interface enables the system to provide guidance to the user in a human-understandable way (Figure 4-13). The main element of the interface is a red "compass" that displays the output of the rotation guidance algorithm in the user's body frame. The bottom right section of the interface shows a high-level command that helps the user anticipate the next move (in this case, a left turn 20 meters ahead). We use a constant walking speed of $1\,m/s$ to convert durations into distances. The top left section displays the confidence of the system. Finally, the top left area shows two images captured by the system at the current node during the first visit. The system automatically determines whether the forward- or backward-facing images should be displayed, depending on the output of the rotation guidance.

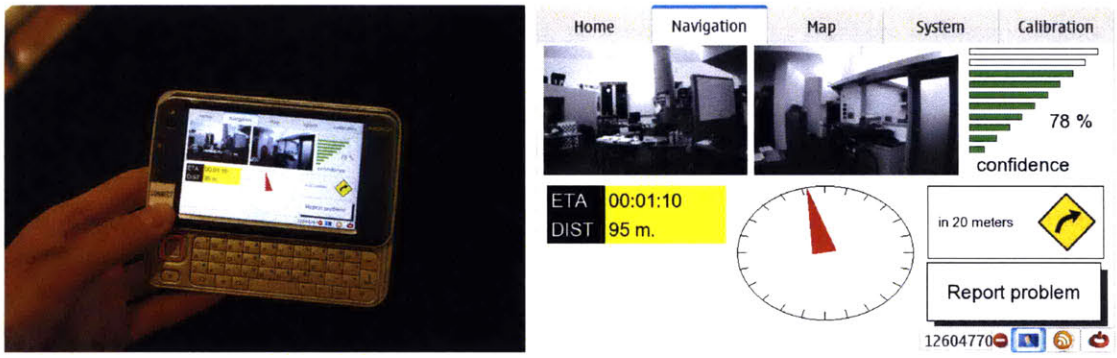

Figure 4-13: The user interface is programmed on a tablet personal computer and provides real-time guidance to the user.

4.5 Method Evaluation

This section presents a quantitative evaluation of our method using real data. In particular, we use an inertial measurement unit (IMU) to provide ground truth for the user's orientation over short sequences. In addition, we devise a vision-based algorithm to provide ground truth over extended periods of time and use floor landmarks for localization ground truth.

4.5.1 Match Matrix

Figure 4-14 shows the probability distributions that we obtain for a one-minute training sequence captured in an arbitrary indoor environment. Each curve represents the probability distribution of co-observation for one of the 16 camera pairs. Based on our method, we expect each probability to fit a triangular distribution whose limits are defined by $-f$ and f, where f is the common field-of-view of all the cameras. Using the MATLAB camera calibration toolbox, we find an actual horizontal field-of-view of $85°$ at the center line and $77°$ at the top and bottom edges, which yields an average field-of-view of $80°$. On the other hand, by fitting a triangular distribution to the curves, we find an estimated field-of-view of $76°$, which is close to the actual field-of-view of the cameras.

We show below the corresponding match matrix H. Each cell in the matrix corresponds to the center of the corresponding probability distribution. 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 § 4.1.4 provide error metrics for the matrix. We attribute this error to the fact that the training 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.

$$H = \begin{pmatrix} 0.0 & 91.7 & 153.8 & -101.6 \\ -91.7 & 0.0 & 64.0 & 165.9 \\ -153.8 & -64.0 & 0.0 & 102.6 \\ 101.6 & -165.9 & -102.6 & 0.0 \end{pmatrix}$$

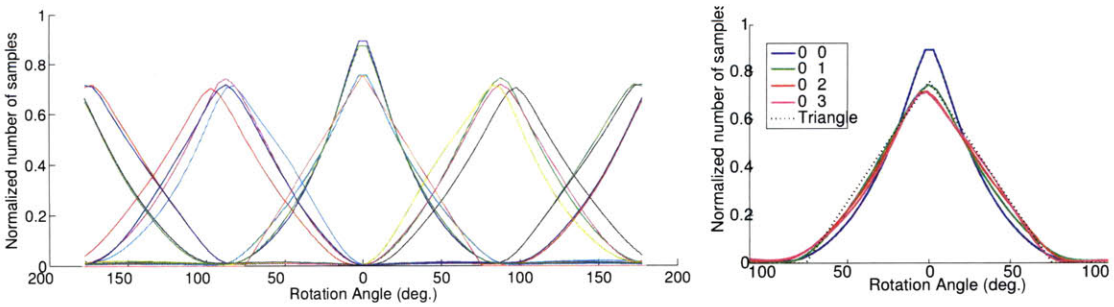

Figure 4-14: Probability distributions for each camera pair after one minute of training in an arbitrary environment. *Left*: Each curve corresponds to one of the 16 camera pairs. *Right*: Closeup view for only four camera pairs. The dotted line corresponds to the triangular fit.

4.5.2 Vocabulary Tree Evaluation

We study the performance of the visual vocabulary for the naive method and the tree-based method [1] described in § 4.1.5. We use the Intel Math Kernel Library for fast matrix multiplication in the naive approach. For the tree-based method, we use a tree branching factor $K = 10$. The method consists of 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. However, for $K_S \leq 3$, the tree-based method outperforms the naive search beyond a vocabulary size of 10^5 words (Figure 4-15). Meanwhile, $K_S = 3$ yields an error rate of around 0.6% compared to the naive search [1].

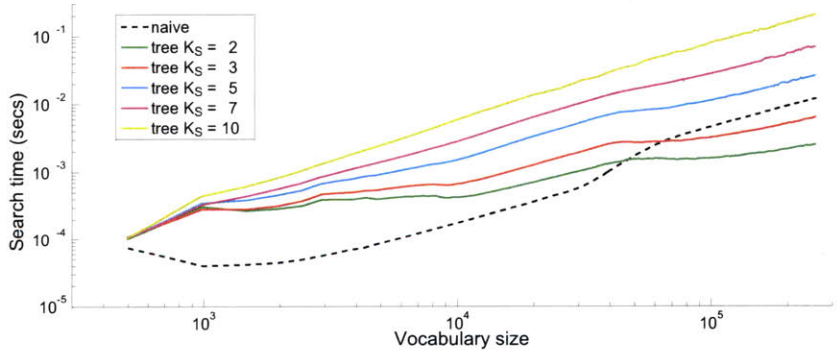

Figure 4-15: Performance of the visual vocabulary for the naive method (dashed line) and the tree-based method (solid lines). The parameter K_S refers to the maximum number of children explored at every node of the tree (maximum is 10).

4.5.3 Rotation Guidance Validation

We validated the rotation guidance algorithm by rigidly mounting an XSens MTi Inertial Measurement Unit (IMU) on the system, with a drift rate of less than one degree per minute.

We captured a 30-second sequence while the user rotates in place in an arbitrary environment (different from that used for training). We then run the rotation guidance algorithm between each pair of frames in the sequence, and compare 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). One purpose of the "turn-by-turn" directions (§ 4-6) is to address the issue of ambiguous rotation guidance.

4.5.4 Large-scale Rotation Evaluation

In order to obtain ground truth for rotation guidance all along the exploration path, we equip 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 contains a primary orientation (Figure 4-16). 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 validate the rotation baseline algorithm using a sequence containing 30 images captured by the ground truth camera as the user rotates in place. We run the rotation baseline algorithm for each pair of frames in the sequence and compared its output to that of the IMU. We obtain a standard deviation of $1.9°$, and conclude that the rotation baseline algorithm provides robust ground truth to the rotation guidance algorithm. Figure 4-17 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 $10.5°$.

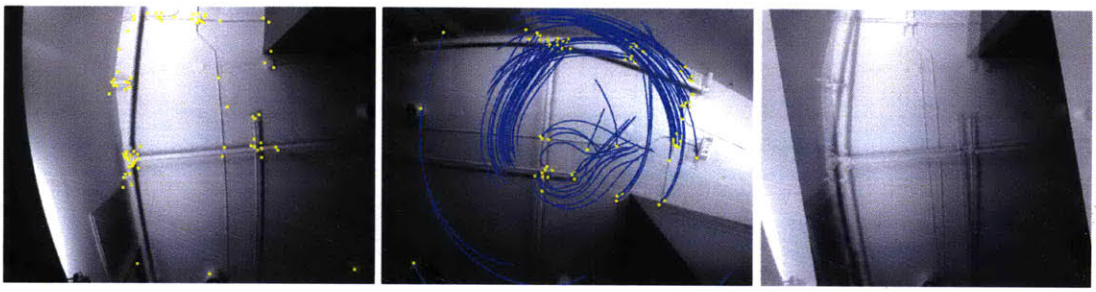

Figure 4-16: Rotation baseline (LAB DATASET). *Left*: first image I_p. *Middle*: second image I_q. *Right*: alignment of I_p onto I_q. SIFT features are shown in yellow. Blue lines represent feature matches. The algorithm estimates the rotation between the two images to within $2°$.

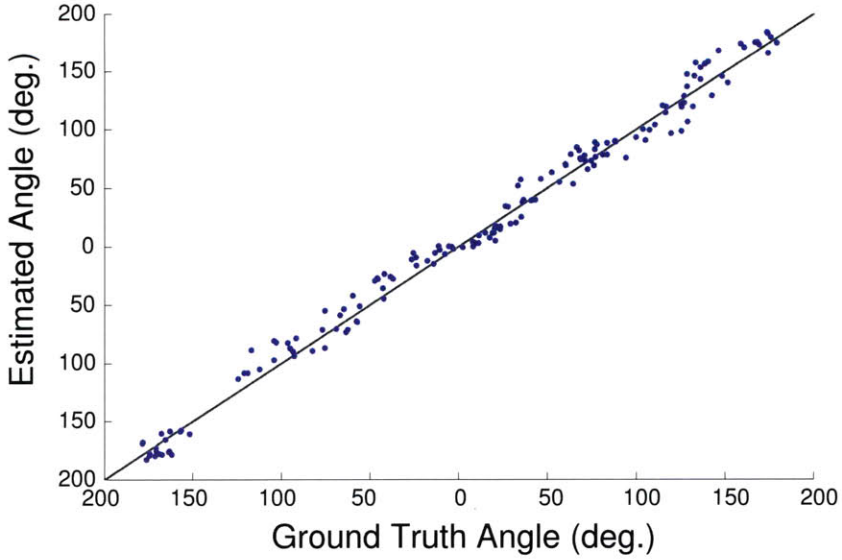

Figure 4-17: Comparison of the rotation guidance output against vision-based ground-truth. The deviation to identity is $10.5°$.

4.5.5 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 passes by each checkpoint multiple times, and uses the user interface to insert a timestamp into the captured video sequence. We then compare the output of the local node estimation with ground truth (Figure 4-18). We observe that node estimation is correct at all checkpoints along the path, *i.e.* that it robustly estimates the user's position within the graph.

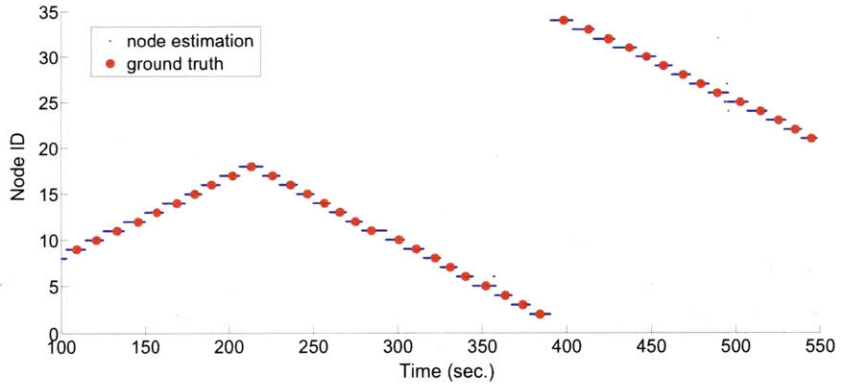

Figure 4-18: Validation of node estimation (*blue line*) against ground-truth (*red dots*). The localization is correct except in a few instances ($t = 360$ and $t = 500$).

4.5.6 Real-world Explorations

We demonstrate our method on three datasets described in Table 4.4. 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 composed 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 many passers-by.

Name	Scenario	Duration	Distance traveled	# frames	# nodes
MEZZANINE	replay	10 min	400 m	6000	91
GALLERIA	homing	15 min	700 m	9000	154
CORRIDORS	point-to-point	30 min	1500 m	18000	197

Table 4.4: Pedestrian 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 4-20 (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.

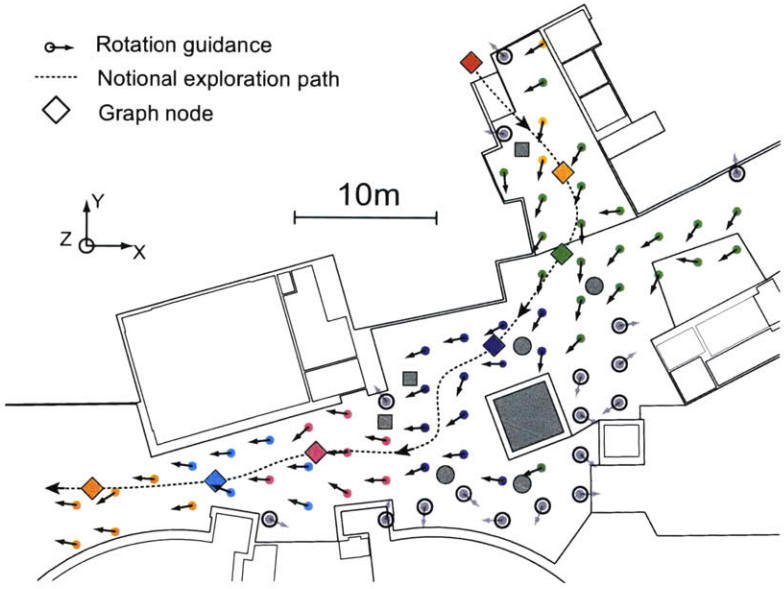

Figure 4-19: Excerpt GALLERIA dataset. The dotted line represents the notional path followed by the user during the exploration. Colored squares represent nodes in the topological map. Arrows represent the actual(not notional) guidance provided to the user upon revisit in a *replay* scenario. Black circles denote failures due to occlusion by building structure.

Figure 4-19 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 distance between two neighbor dots 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 provides robust guidance even when the user is more than ten meters from the original path. Locations marked by an open 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 4-20: *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.

Figure 4-27(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 4-21(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.

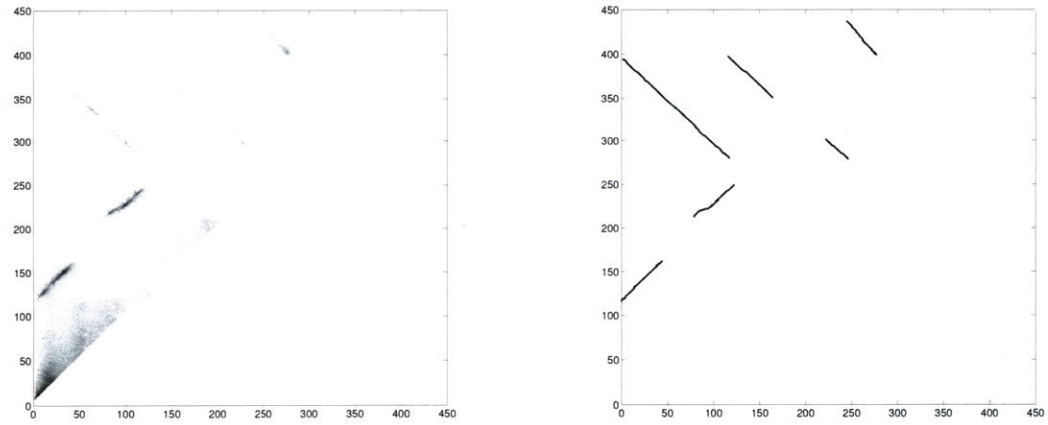

Figure 4-21: *Left*: Upper triangular part of the similarity matrix for the CORRIDORS dataset. Dark regions correspond to high similarity (loop closure events). *Right*: Output of the Smith-Waterman algorithm. Dark lines correspond to sequence alignments.

4.5.7 Motion Classification Evaluation

We validate the motion classification algorithm on two datasets. The STATA-33X dataset consists of a three-minute long exploration inside the offices of the MIT Stata Center (Figure 4-25). The exploration includes going up and down a staircase. The STATA-OUTDOOR dataset consists of a six-minute long exploration inside and outside the MIT Stata Center. The exploration includes an indoor and an outdoor ramp and stairwells (Figure 4-26).

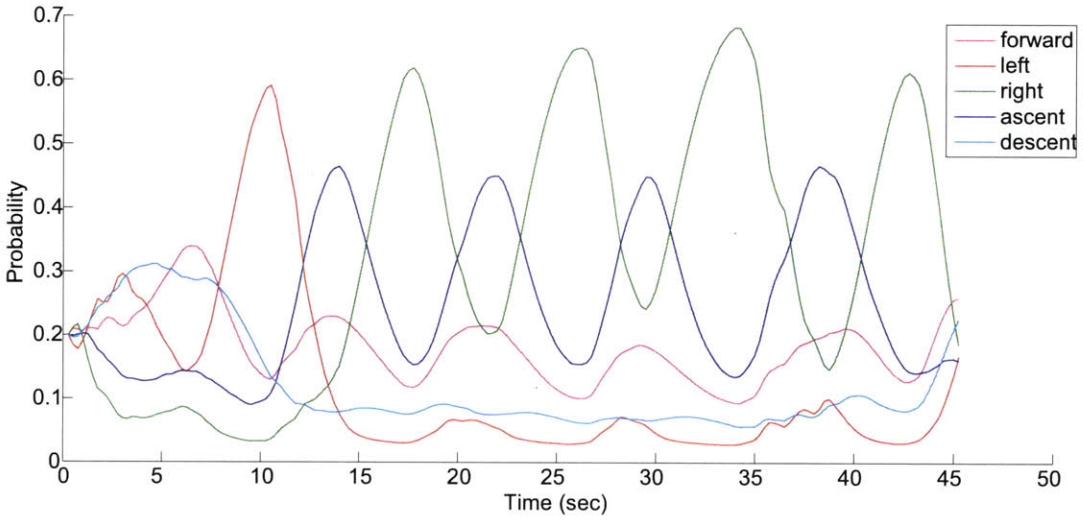

Figure 4-22: Probability distribution for five motion categories (STATA-33X dataset excerpt). The user is climbing a staircase, producing a series of "ascent" and "right" motions.

Figure 4-22 shows the evolution of the probability distribution of the motion category over time for the STATA-33X dataset. At this point in the exploration, the user is walking up a clockwise-oriented staircase, which results in a succession of "ascent" and "right" motions. For each dataset, ground-truth is obtained by manually labeling the video using a user interface designed specifically for this purpose. Given ground-truth data, we compute the precision, i.e. the rate of output motion types that are actually correct, and the *recall*, i.e. the rate of true motion types correctly identified. An ideal classifier would have a precision of 1 for any recall.

Table 4.5 shows precision and recall values for various image resolutions (full resolution corresponds to an image size of 376×240). As we can see, the performance of the algorithm remains surprinsingly good as the resolution decreases. The loss in performance is negligeable up to a scale factor of 18%, which corresponds to an image size of 68×43 (Figure 4-23(b)). Figure 4-24 shows the timeline of the motion classification for the STATA-OUTDOOR dataset. From $t = 80$ to $t = 130$, the user is walking down a spiral-shape staircase, which results in a hybrid "right"/"descent" motion.

Image Scale	100 %	75%	50%	35%	25%	18%	12%	10%	7%	5%
Image Size	376 × 240	282 × 180	188 × 120	132 × 84	94 × 60	68 × 43	45 × 29	38 × 24	26 × 17	19 × 12
STATA-33X dataset										
Precision (%)	80.9	82.4	82.5	81.8	83.0	81.1	78.5	76.8	69.5	58.6
Recall (%)	75.0	75.7	74.9	75.0	75.8	74.0	72.7	69.1	60.0	29.0
STATA-OUTDOOR dataset										
Precision (%)	77.5	78.6	79.3	79.6	79.6	80.1	77.7	76.9	65.9	48.7
Recall (%)	68.5	68.4	69.0	69.6	69.4	69.2	65.8	65.4	51.8	21.3

Table 4.5: Motion classification precision-recall for various image resolutions (STATA-33X and STATA-OUTDOOR datasets).

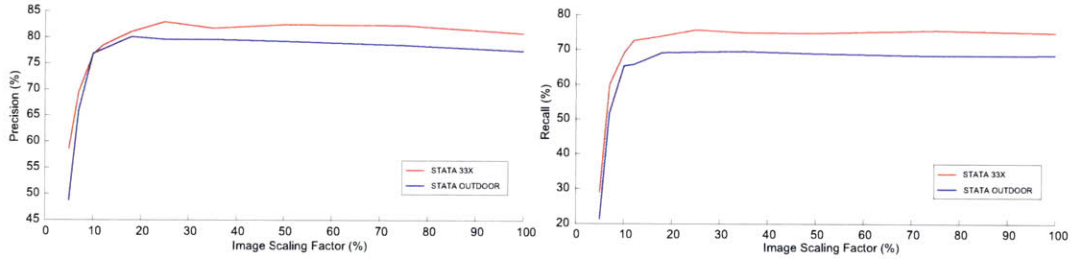

Figure 4-23: Precision and recall of the motion classification with respect to image resolution.

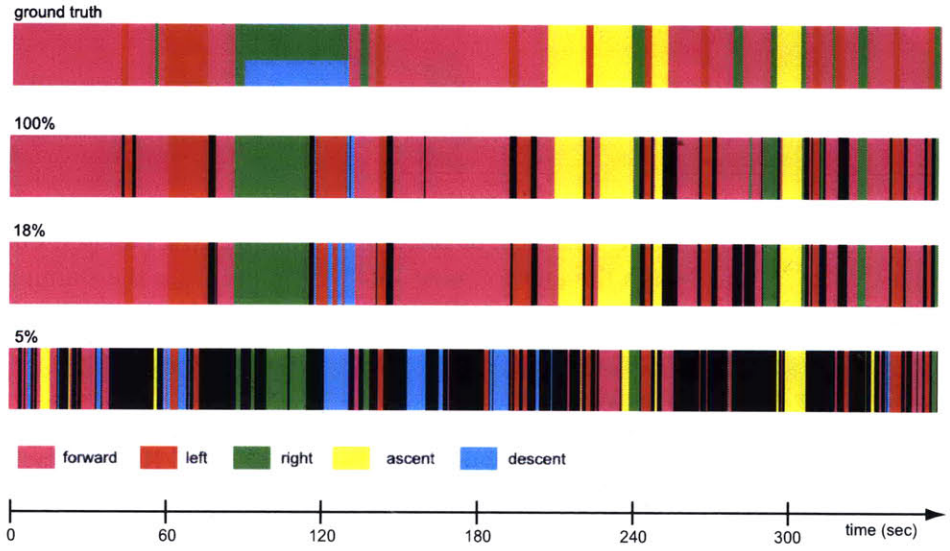

Figure 4-24: Timeline of the motion classification for the STATA-OUTDOOR dataset for various image scale factors (5%, 18% and 100%). Manually-labeled ground-truth at top. From $t = 80$ to $t = 130$, the user is walking down a spiral-shape staircase, which results in a hybrid "right"/"descent" motion.

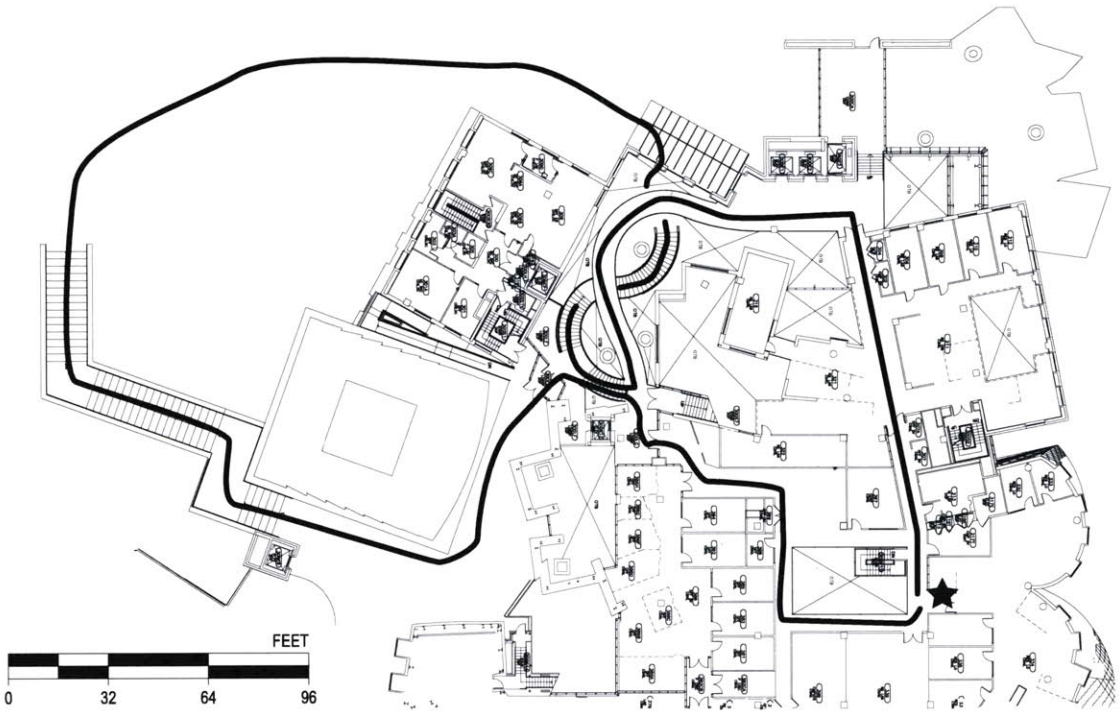

Figure 4-25: Notional path followed by the user (STATA-OUTDOOR dataset). The sequence is six minutes long and includes indoor and outdoor ramps. The exploration started and ended at the location marked with a star.

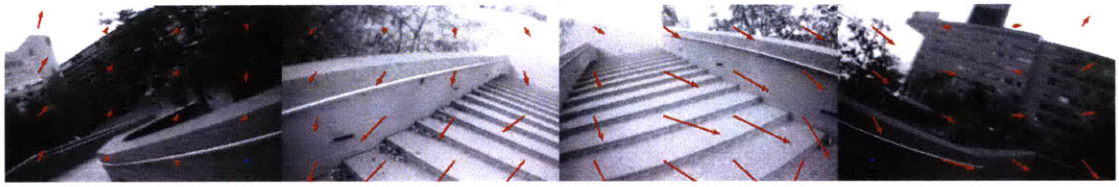

Figure 4-26: A frame classified as "ascent" as the user is walking up a stairwell outside the MIT Stata Center. Even though the environment looks different from the training setting (Figure 4-10), the classifier determines the motion category correctly.

4.5.8 System Performance

We run all algorithms on an Intel quad-core computer (2.5 GHz, 4GB RAM). Each camera produces 376×240 grayscale 8-bit images. We report the following performance when all algorithms run in parallel on one quad-core laptop. Computation of SIFT features runs at 6 Hz for an average of 150 features per camera (each of the four cameras maps to one processor). A loop of the place graph generation runs in 100 msec with 600 features per frame. The creation of a node is instantaneous, so the place graph generation algorithms runs overall at 10 Hz. During navigation, the global localization algorithm runs in 700 msec using a radius of 5 in the place graph. The rotation guidance algorithm runs in 200 msec. All algorithms run in parallel, so that the system provides directional guidance at 5 Hz while the localization update occurs at 1.4 Hz.

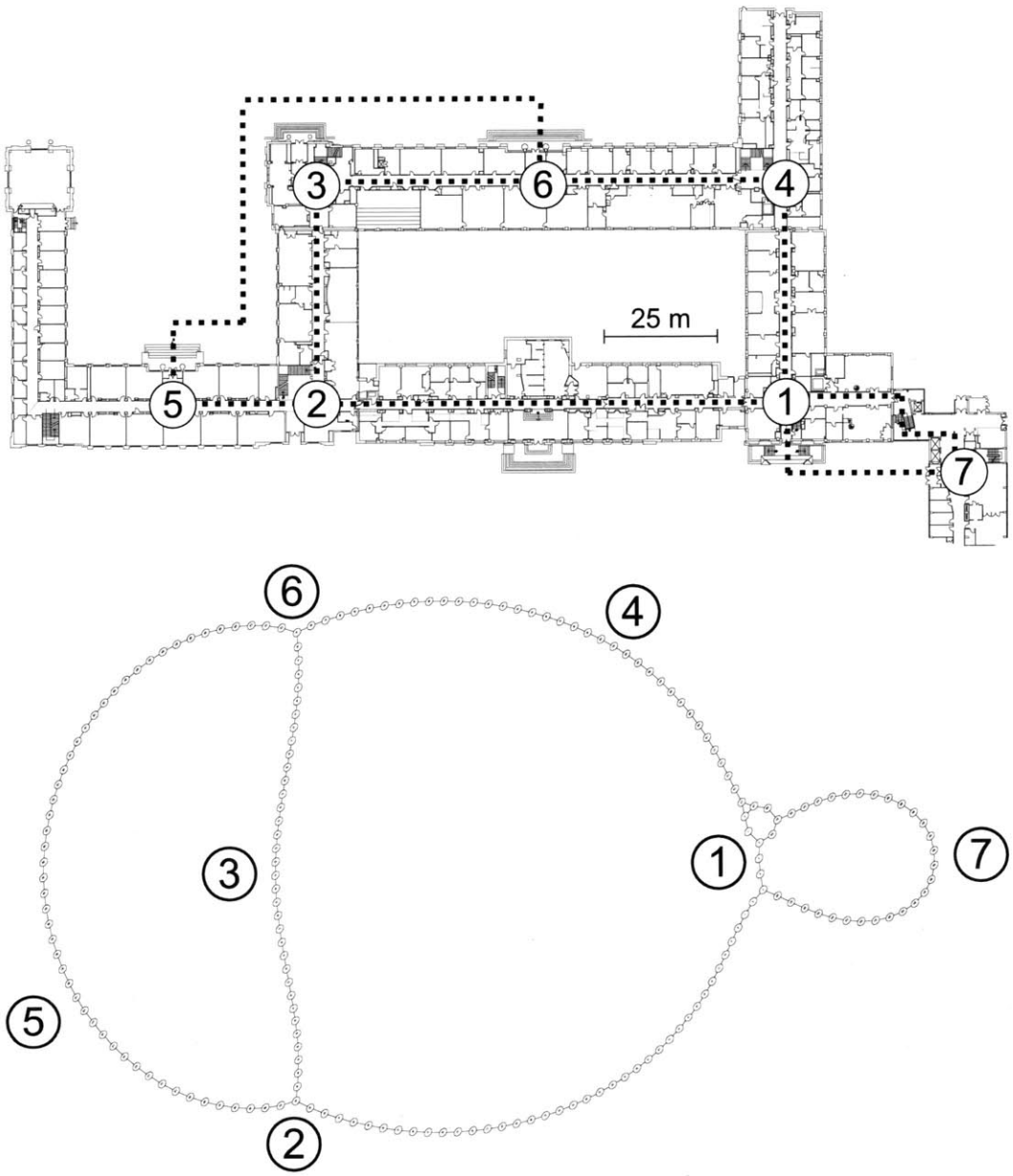

Figure 4-27: Floor plan of the explored environment for the CORRIDORS dataset (*left*) and corresponding topological map displayed with a spring-mass model (*right*). The exploration path (shown as a dotted line) was: 1, 2, 3, 4, 1, 2, 5, 6, 4, 1, 7, 1, 4, 3, 2, 1, 7, 1. Exploration includes both indoor and outdoor environments over a course of $1,500$ meters (30 minutes).

Chapter 5

Ground Robot Navigation using Uncalibrated Cameras

5.1 Robotic Navigation

The last decade has seen promising progress in the field of vision-based navigation [20, 53, 69, 75, 77]. However, vision-based 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.

In this chapter, we extend the work described in Chapter 4 to provide coarse waypoint navigation for a mobile robot operating within such an environment. Specifically, we describe and demonstrate how the rotation guidance algorithm can be used 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.

5.2 Method Overview

We devise a strategy that relies on the methodology described in Chapter 4. On one hand, the vision-based navigation method provides localization and high-level navigation commands. On the other hand, a low-level, laser-based obstacle avoidance algorithm takes the navigation commands as input and drives the robot in the commanded direction while avoiding obstacles (Figure 5-1).

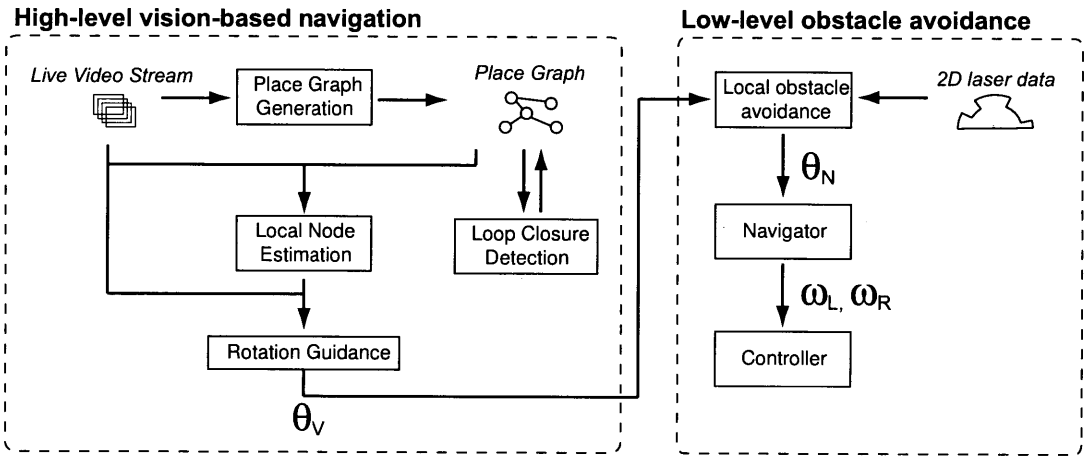

Figure 5-1: The vision-based navigation method described in Chapter 4 acts as a high-level module feeding a coarse directional cue (θ_V) to a local obstacle avoidance algorithm which determines a locally optimal direction to follow (θ_N).

5.3 Obstacle Avoidance and Control

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; the strategy consists of assigning a cost metric to body-relative angles, where angles far from the target direction 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 [41, 82]. More accurate obstacle sensing will result in smoother robot trajectories, though the navigation algorithm itself is unaffected by the choice of obstacle avoidance method.

The obstacle avoidance algorithm takes as input a desired orientation angle θ_v expressed in the robot's body frame, and outputs motion commands to the controller. The algorithm proceeds in two steps. First, it analyzes the surrounding environment using planar laser data and determines an optimal direction θ_s. Second, it converts θ_s into motion commands, namely a rotation speed ω_r and a translation speed v_r. The algorithm takes as input a set of laser returns associated with their orientation in the robot's body frame $S = \{(r_i, \theta_i)\}$. Given the input angle θ_v, the algorithm finds the optimal angle θ_s in the sense of the following energy function:

$$\theta_s = \underset{S}{\text{argmax}} \frac{r_i}{1 + \sin^2(\theta_i - \theta_v)} \tag{5.1}$$

This function penalizes rays that are far from the target direction θ_v or close to an obstacle (Figure 5-2). The algorithm uses only instantaneous laser returns and does not attempt to make a geometric interpretation of the environment. It provides the advantages of being robust to a large class of configurations and easy to implement. In practice, we found this algorithm to work very robustly across our datasets.

Given the angle θ_s, the algorithm then determines the control commands ω_r and v_r as follows:

$$\omega_r = \lambda_\omega \cdot \theta_s \tag{5.2}$$

$$v_r = \max(0, \lambda_v \cdot (1 - \frac{|\ \theta_s\ |}{\theta_{min}})) \tag{5.3}$$

where λ_ω and λ_v are hardware-dependent constant parameters and θ_{min} is a threshold angle beyond which the robot turns in place (we use $\theta_{min} = 25°$). The local obstacle avoidance algorithm is summarized in Table 5.1.

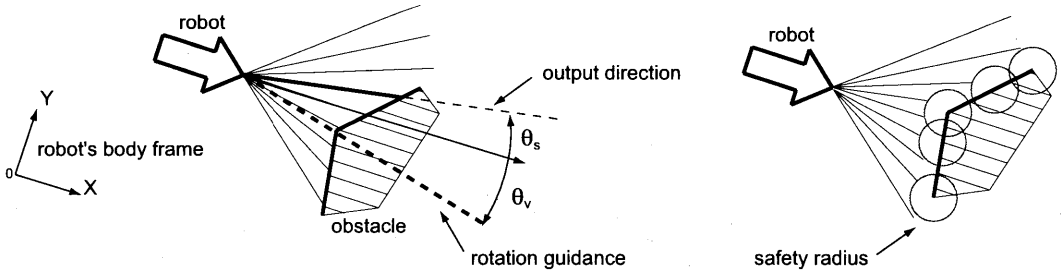

Figure 5-2: *Left:* Given a goal direction θ_v, the obstacle avoidance algorithm computes θ_s, the optimal direction based on an energy function that penalizes rays that are short due to an obstacle or different from the desired direction. *Right:* observations are inflated using a safety radius to account for the robot's footprint.

The Local Obstacle Avoidance Algorithm

Input: a direction θ_V
Input: a set of laser returns $S = \{(r_i, \theta_i)\}$
Output: the instantaneous controller commands (ω_r, v_r)

1: Convert the returns S into a set of 2D points P in the local frame of the robot
2: Decimate the returns S into a subset $\bar{S} \in S$ (we use a resolution of $2.5°$)
3: **for** each ray $\bar{s}_k \in \bar{S}$ **do**
4: Intersect $\bar{s}_k$ with a circle centered at each point $p \in P$
5: Determine $\bar{r}_k$ the smallest distance to all circles
6: Compute the score g_k using Equation 5.1
7: Find $\bar{s}_r \in S$ the ray with highest score
8: Compute (ω_r, v_r) using Equations 5.2 and 5.3

Table 5.1: The Local Obstacle Avoidance Algorithm

5.4 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 (Figure 5-3). 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 (2.4 GHz, 4GB RAM) mounted on the robot.

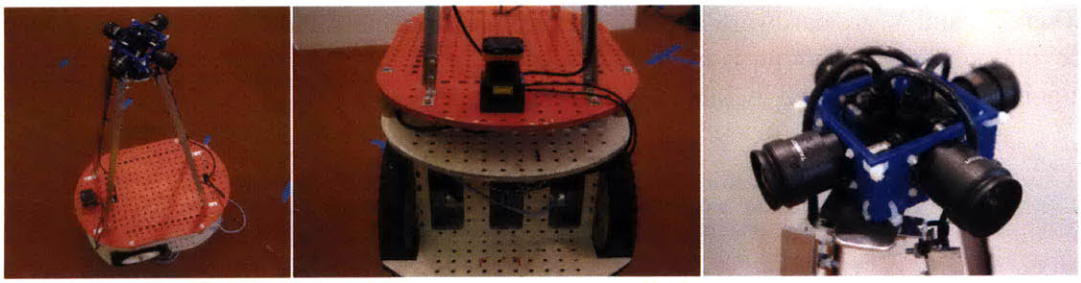

Figure 5-3: Our robot is composed of a two-motor wheeled base, a four-camera omnidirectional rig (*right*) and a laser range finder (*middle*). We combine a high-level vision-based navigation algorithm with a low-level obstacle avoidance algorithm to obtain robust autonomous navigation in unknown environments.

5.5 Method Evaluation

We demonstrate our algorithms on two datasets (Table 5.2). The STATA 3RD FLOOR dataset consists of a 24-minute exploration through several office areas, followed by three missions

(A, B and C) totaling 80 minutes. The STUDENT STREET dataset consists of a 26-minute exploration within a crowded mall-like environment followed by a 24-minute mission. The overall distance traveled by the robot was 1,200 m. In each mission, the robot was tasked to reach a series of checkpoints within the graph and navigated fully autonomously. The missions often required the robot to traverse locations in the direction opposite of its first visit (Figures 5-9 and 5-10).

Dataset	Mission	Duration	Distance	Mean Speed	# nodes	μ_K	μ_D	μ_G	μ_N	μ_R
STATA 3RD FLOOR	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°
STUDENT STREET	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°

Table 5.2: Robotics datasets.

5.5.1 Loop Closure Detection

Figures 5-4 and 5-5 illustrate the loop closure algorithm on the STATA 3RD FLOOR 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.

Figures 5-6 and 5-7 show the output of the loop closure algorithm on the STUDENT STREET dataset. This dataset involves both indoor and outdoor environments. As the robot exits the building (point 4), the terrain becomes uneven and the laser-based mapping algorithm fails. Therefore, there is no ground-truth for the localization of the robot after this point in the dataset (green trajectory on Figure 5-7). However, the vision-based loop closure algorithm successfully detects both loop closure events.

5.5.2 Body-Centric Rotation Guidance

We show below the match matrix H obtained for the system described in § 5.4 (angles are in degrees). 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°.

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

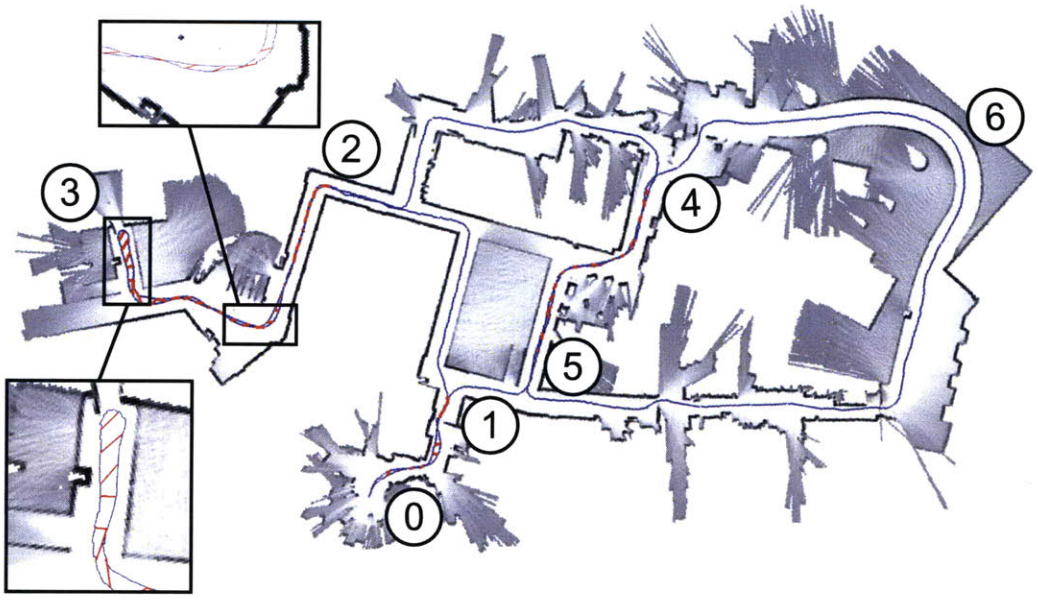

Figure 5-4: Loop closure on a 30 minute exploration path across an office environment (STATA 3RD FLOOR dataset). Loop closure detections are shown in red. Numbers refer to decision points in the place graph (Figure 5-5). We use the metric map for validation only.

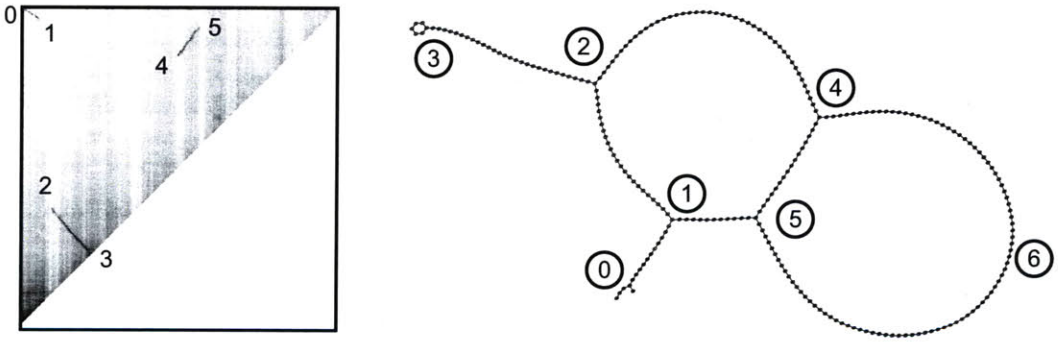

Figure 5-5: Similarity matrix and place graph rendered using a spring-mass model (STATA 3RD FLOOR dataset). Loop closure detections correspond to segments in the similarity matrix. The path of the robot was 0, 1, 2, 3, 2, 4, 5, 6, 4, 5, 1, 0.

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 the 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 $8°$ (Figure 5-8). We emphasize that the sequence was captured in an arbitrary environment that is different from the one used for training.

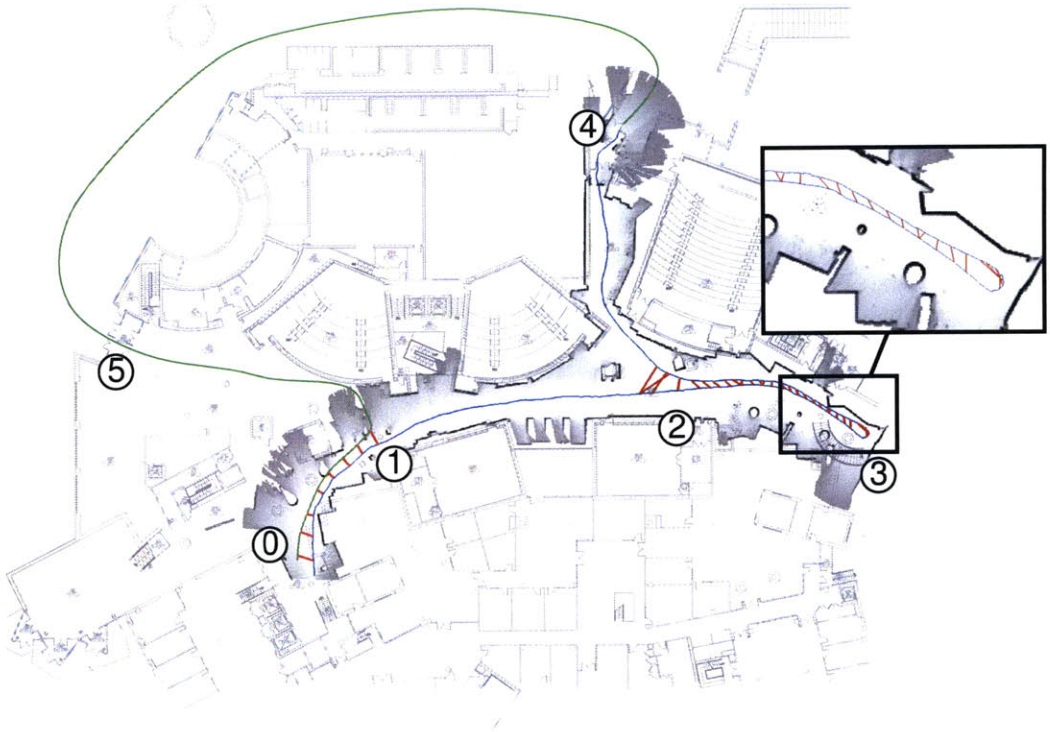

Figure 5-6: Loop closure on a 30-minute exploration within a mall-like environment (STUDENT STREET dataset). Ground-truth exploration path shown in blue. The laser-based mapping algorithm failed on the outdoor section of the exploration (notional path shown in green). Yet the algorithm was able to detect a loop closure on section 1 - 0.

5.5.3 Ground-truth Validation

We analyze the performance of the vision-guided navigation based upon the ground truth vehicle trajectories that we estimate using the publicly available GMapping [91] 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 5-4. 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 STATA 3RD FLOOR and STUDENT STREET 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. We aligned the maps manually using a tool designed specifically for this purpose. We estimate an alignment accuracy of a few centimeters. An automated method based on Iterative Closest Point could be used instead.

The ground-truth localization of the robot provides several metrics to evaluate navigation performance (Table 5.3). First, we consider μ_K, the number of times the robot

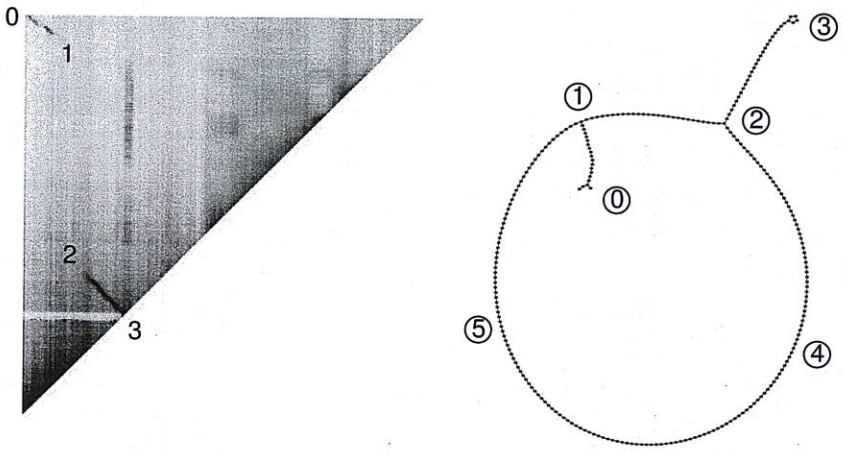

Figure 5-7: Similarity matrix and place graph rendered using a spring mass model (STUDENT STREET dataset). Loop closure events correspond to segments in the similarity matrix. The path of the robot was 0, 1, 2, 3, 2, 4, 5, 1, 0.

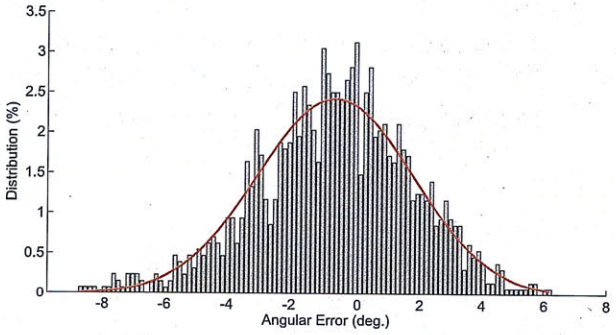

Figure 5-8: Distribution of the angular error of the rotation guidance algorithm compared against an inertial measurement unit for a sequence of 30 seconds (1,300 datapoints, $\mu = -0.72°, \sigma = 2.50°$).

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. In addition, we evaluate the precision of the local node estimation algorithm by defining μ_G as the distance in graph space between the location estimated by the robot and the true location. Similarly, we consider μ_N the metric distance between the current location of the robot and the physical location of the estimated node. Finally, we define 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 5.2 summarizes the results. Figure 5-11(a) shows the evolution of μ_D

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

Table 5.3: Evaluation metrics.

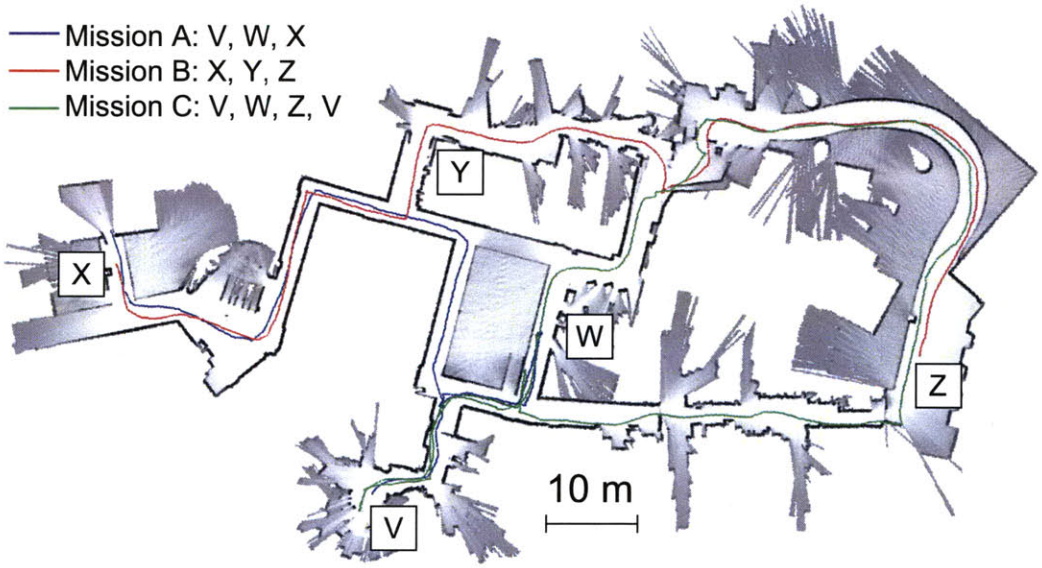

Figure 5-9: STATA 3RD FLOOR dataset: ground-truth paths followed by the robot during missions A (*blue*), B (*red*) and C (*green*).

over time for mission C. The distance to the original path is on 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 5-12 illustrates the error (in graph space) of the node estimation algorithm. Overall, the algorithm performs well and localizes the robot with a worst-case accuracy of two nodes. Similarly, the distance to the estimated node is also bounded with an average of 0.42 m (Figure 5-11(b)). Finally, the rotation guidance error averages around zero with a typical standard deviation of 12° (Figure 5-13). At time t=450 s, 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 temporarily large error.

The STUDENT STREET dataset (Figure 5-10) 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, many 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 5-14).

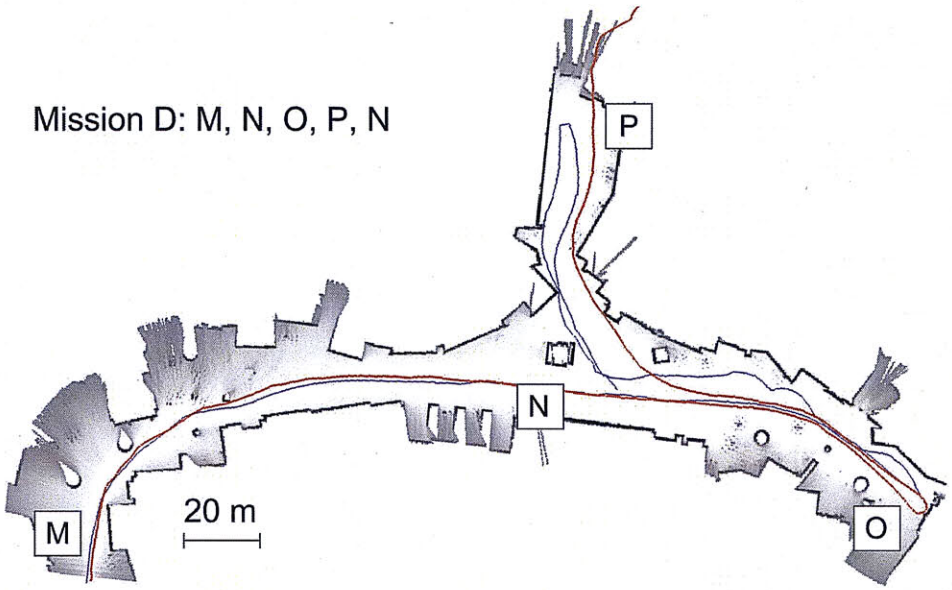

Figure 5-10: STUDENT STREET dataset: exploration path (*red*) and revisit path (*blue*). During the second half of the mission, a high number of passers-by interfere with the robot's path. Yet, the robot reaches its destination successfully.

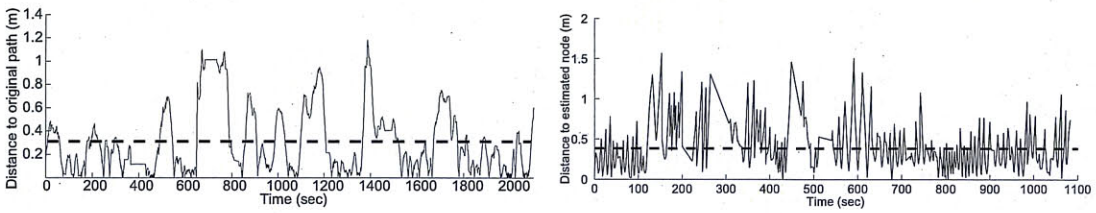

Figure 5-11: *Left*: Distance to original path (STATA 3RD FLOOR dataset, Mission C). *Right*: Distance to the estimated node (STATA 3RD FLOOR dataset, Mission A). Mean values shown in dotted line.

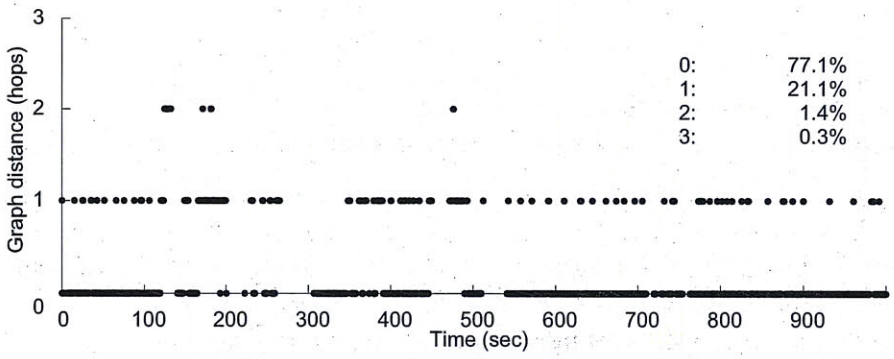

Figure 5-12: Distance in graph space between the estimated node and the correct node (STATA 3RD FLOOR dataset, Mission A).

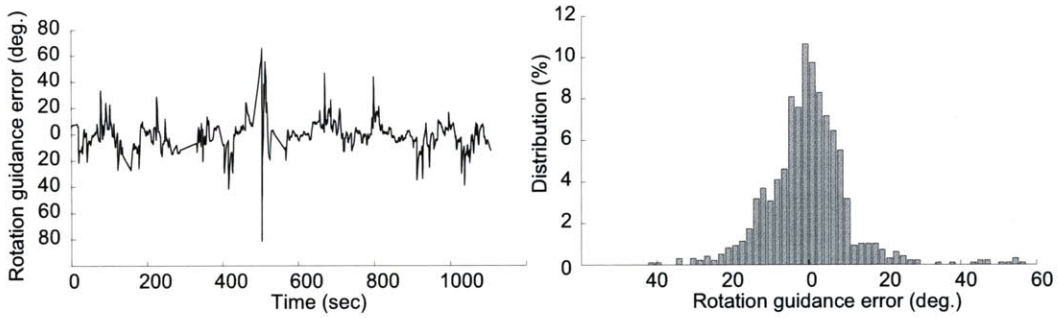

Figure 5-13: Rotation guidance error with respect to the direction pointing to the next node (STATA 3RD FLOOR dataset, Mission A) as time series (*left*) and histogram (*right*).

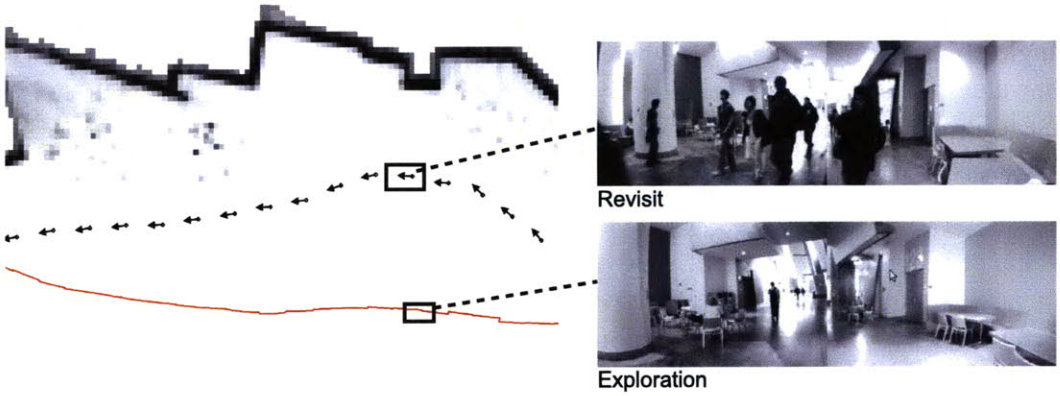

Figure 5-14: STUDENT STREET dataset. Assuming that the robot has previously explored the environment (*red path*), the method provides guidance in the body frame of the robot during revisit (*black arrows*). we demonstrate the robustness of our method on real-world experiments that involve an outstanding level of dynamic scenes.

5.6 Method Performance and Image Resolution

This section analyzes how the performance of the algorithms varies as the image resolution decreases. Specifically, we compare the values for the localization metrics (μ_G and μ_N) and the rotation guidance metrics for various image resolutions on the STATA 3RD FLOOR dataset. The full image resolution is 320×240.

Table 5.4 shows the localization performance for various image resolutions. As expected, the performance decreases with resolution. However, the decrease is negligeable up to a scale factor of 18%, which corresponds to an image size of 67×43. For lower resolutions, the performance drops dramatically. The dramatic decrease in performance can be explained by the fact that the node estimation algorithm relies on a first-order Markov model. If the algorithm fails to track the position of the robot in the graph at some point, it has little chance of recovering the correct position at a later time. The robustness of the localization algorithm relies on the average number of features observed by the cameras over time. From Table 5.4, we conclude that more than 100 features are required for robust localization, but that more features bring little more precision to the localization.

To evaluate the impact of image resolution on rotation guidance, we compare the output of the rotation guidance algorithm with ground truth obtained with an IMU, as in § 5.5.2. We perform the experiment for various image sizes. Table 5.5 shows the evolution of the rotation guidance error. As expected, the performance of the rotation guidance algorithm decreases with the image size. Since the decrease in image resolution generates a decrease in number of observed features, we can also analyze the evolution of the rotation guidance performance with the number of features. Figure 5-15 shows the rotation guidance error with respect to the image scale and the number of features, on a log scale. From Equation 4.7, we know that :

$$\log \sigma_N = \log \sigma - \frac{1}{2} \cdot \log N \tag{5.4}$$

where N is the number of observations and σ is the standard deviation associated with a single observation: $\sigma^2 = f^2/6$ where f is the common field of view of all cameras. Using the values for our system, we find $f = 80°$, $\sigma = 32.6°$ and $\log \sigma = 3.48$. We perform a linear fit on the data (Figure 5-15(b)) and find:

$$\log \sigma = 3.29 - 0.36 \cdot \log N \tag{5.5}$$

which corresponds to a field of view of 65°. We emphasize that Equation 4.7 applies only for a large number of observations. Therefore, we fit data only for a large number of features ($N > 50$) to obtain Equation 5.4. As we can see in Figure 5-15, the fit improves as the number of feature increases.

We conclude from these experiments that the performance of the node estimation algorithm and the rotation guidance algorithm decreases with image resolution, but remains acceptable for an image resolution as low as 94×60. We also demonstrate that the rotation guidance error follows the Central Limit Theorem for large numbers of features, as stated in § 4.1.4.

Image size	372 × 240		188 × 120		94 × 60		67 × 43		45 × 28	
Image scale factor	100%		50%		25%		18%		12.5%	
Number of features	553	±189	206	±62	64	±20	36	±11	13	±5
μ_G (unitless)	0.51	±0.96	±0.51	0.91	±0.46	±0.74	0.65	±1.16	31.9	±17.2
μ_N (meters)	0.77	±0.44	0.87	±0.91	0.86	±0.63	0.94	±0.92	28.3	±15.3

Table 5.4: Localization performance for various image resolutions on the STATA-33X dataset. *First column*: mean value. *Second column*: standard deviation.

Image size	376 × 240	329 × 210	282 × 180	235 × 150	188 × 120	131 × 84	94 × 60	67 × 43	47 × 30	26 × 16
Image Scale factor	100%	87.5%	75%	62.5%	50%	35%	25%	18%	12.5%	7%
Rotation error (deg.)	2.0	2.3	2.2	2.4	2.8	3.2	3.9	6.4	20.0	53.7
Number of features	1408	1166	905	676	450	265	148	80	28	8

Table 5.5: Rotation guidance performance for various image resolutions (STATA-33X dataset).

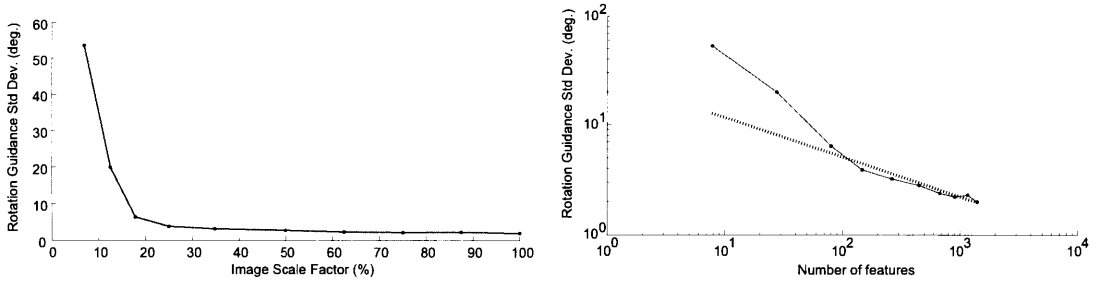

Figure 5-15: *Left*: Rotation guidance error with respect to image resolution. *Right*: Rotation guidance error with respect to number of features, log scale. The dotted line represents the linear fit.

5.7 Node Density and Visual Vocabulary

The method presented in § 4.1.1 relies on an experimentally-selected threshold for the energy function Ψ. The value of the threshold has an impact on the density of the place graph. For higher values of the threshold, nodes are created less often and the density of the place graph decreases. We analyze the impact of the energy threshold on the physical distance between two consecutive nodes using the ground-truth STUDENT STREET dataset collected with the robotic platform (26 minutes). Figure 5-16 shows the distribution of inter-node distances for threshold values varying between 0.5 and 0.90. The inter-node distance increases smoothly as the threshold increases and varies between one meter and two meters for a threshold range of 0.6 - 0.85. We can therefore conclude that the impact of the energy threshold on the node density in the place graph is smooth and well-behaved.

We also analyze the sensitivity of the visual vocabulary presented in § 4.1.5 with respect to the node density. In particular, we are interested in studying how the vocabulary grows as the node density increases. The intuition here is that the distribution of features in descriptor space is not random between two consecutive nodes due to the continuum in visual appearance of the environment. Therefore, we expect the vocabulary to grow significantly less than linearly as the node density increases.

Figure 5-17 shows the vocabulary size for various node densities (blue dots). The red curve represents a power fit $f(x) = a \cdot x^b$. We obtain various node densities by varying the energy threshold, as descripted previously in this section. We observe that the vocabulary grows roughly as the square root of the node density. The values for the fit are $a = 1.13 \cdot 10^5$ and $b = 0.52$. We did not find an explanation for the square root evolution of the vocabulary. However, the observations are in agreement with the expected results, as the vocabulary grows less than linearly with the node density.

5.8 Discussion

Despite extensive research in the field of vision-based navigation for robots, relatively little attention has been paid to methods based on uncalibrated cameras. In this work, we demonstrate the algorithms developed in Chapter 4 are able to provide robust guidance to

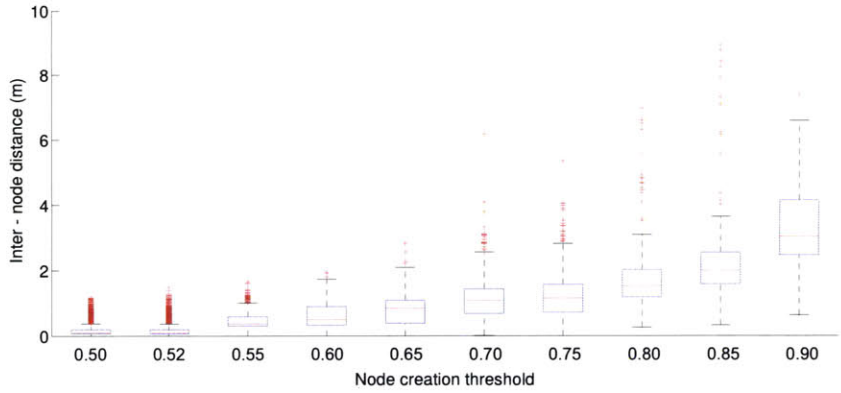

Figure 5-16: The physical distance between consecutive nodes increases smoothly as the energy threshold increases.

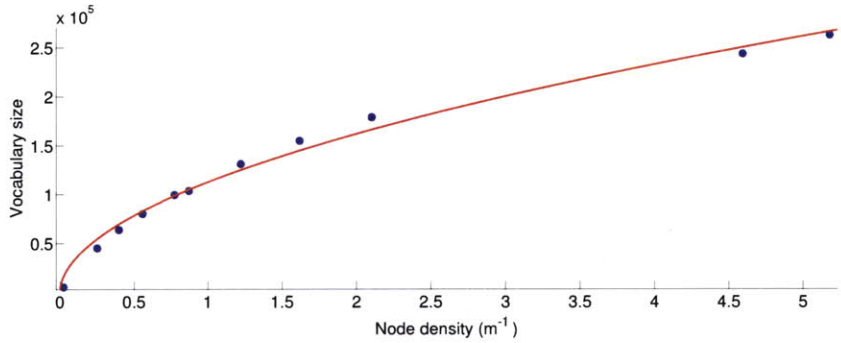

Figure 5-17: The vocabulary grows as the node density increases. Blue dots represent actual data. The red curve represents the power fit.

a robot equipped with a local obstacle avoidance capability. Unlike many approaches, our method does not build a metrical representation of the environment, even locally.

Figure 5-18 illustrates one limitation of the method. Here the algorithm fails to entirely detect the loop closure sequence due to a lower visual similarity of the nodes in this region. As a consequence, the place graph does not fully reflect the local topology of the environment. Upon revisit, the robot follows the correct path in the place graph. However, the path is not optimal (nor natural) in the metric world. A solution could be to lower the threshold in the loop closure algorithm. However, this would increase the probability of false positives as well.

Our approach enables successful navigation in environments that would be challenging for state-of-the-art metric mapping algorithms. However, by reducing the representation of the exploration path to a topological map, our method does not allow geometric reasoning and determination of the relative position of the nodes in the graph. Consequently, our method cannot use localization cues to generate loop closure hypotheses. In addition, it does not allow the robot to follow a path that has never been explored before. In other words, our method requires the agent to explore the environment actively and extensively ahead of time, which may be a limitation depending on the application.

However, our approach sets no constraint on the environment other than that it contains visually descriptive features. In particular, our method does not rely on assumptions typically made by structure-from-motion algorithms, such as the presence of large planar surfaces or that of a flat terrain. Our experiments show that our method performs well in environments where a state-of-the-art laser-based mapping algorithm failed.

In addition, our method requires no camera calibration, which makes it well-suited to fields robotics. The setup is trivial and consists of mounting the camera rig at any place on the robot. The training step may even be skipped if we consider that the match matrix is given ahead of time. All algorithms run online and require no batch processing. Our method then provides robust navigation guidance, assuming that the robot has of a low-level local obstacle avoidance capability.

In addition to advancing the state of the art in robotic navigation, the work described in this chapter also provides a ground-truth dataset for the evaluation of our method. Using this data, we study the relation between the number of feature points and the reliability of the method. Our experiments show that the performance of the method decreases with the number of feature points, but remains acceptable for images as small as 94×60. From a system perspective, the goal is to find a trade-off between reliability and computational requirements. Given the hardware described in § 4.3, we found that an image resolution of 188×120 provided robust navigation, with all algorithms running in parallel at 5 Hz.

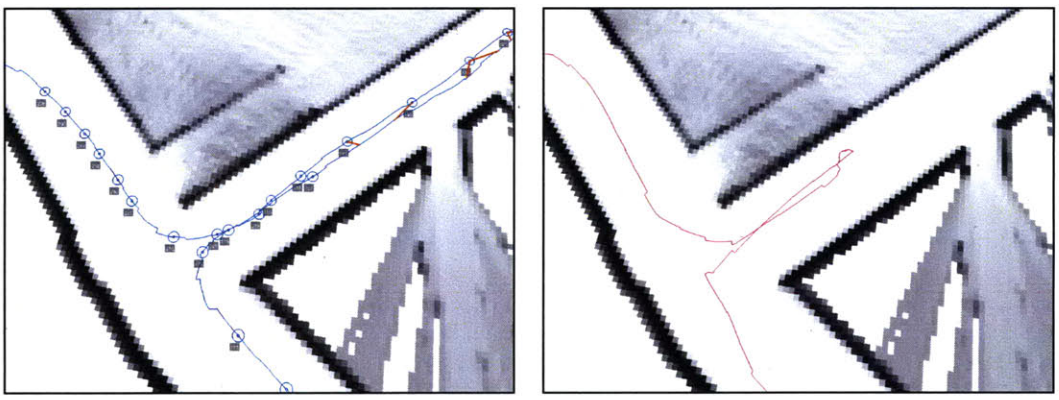

Figure 5-18: During exploration (*left*), the loop closure algorithm misses a few matches (shown in red). The consequence is directly visible during revisit (*right*). The path of the robot is not faithful to the original exploration but it does reach its destination successfully.

Chapter 6

User Study

We provide an evaluation of the system described in chapter 4 by running several experiments with untrained users through extended environments. We evaluate both the efficiency and the effectiveness of the system. The effectiveness is a qualitative metrics, that estimates whether the system is doing the right task, while the efficiency is a quantitative metrics, that measures how well the system is executing its task. We evaluate the efficiency of the system by defining several quantitative metrics based on data logged during the experiments and the effectiveness of the system using the experience of the user collected using a survey.

6.1 Experiment A

In this experiment, a subject first explored an environment while wearing our system. The environment consisted of open spaces and corridors on the first floor and third floor of the MIT Stata Center and included a staircase. The exploration spanned approximately 10 minutes and 400 meters. The experiment consisted of handling the system to a series of untrained users (the "walkers"). The procedure involved specifying a target destination to the system, unknown to the walker. The goal for the walker was to follow the directions provided by the system in order to reach that destination. The experimenter (the "follower") was following the walker at all times while maintaining some distance to avoid interference with the walker. If the walker failed to take the correct direction along the way, the follower would notify the walker about it and use the interface to log a message in the system (USER LOST). The follower would then show the correct direction to the walker and the experiment would resume. The user interface also included a button that would allow the walker to log a message when she/he felt that the navigation guidance provided by the system was unclear (UNCLEAR GUIDANCE). Each experiment was approximately 10 to 20 minutes long.

We use the data logged during exploration to provide a quantitative evaluation of the efficiency of the system. Specifically, we declare as successful a mission for which the walker reached the destination with no intervention of the follower (*i.e.* no USER LOST messages). We define γ_T as the rate of UNCLEAR GUIDANCE messages per minute and γ_D as the rate of UNCLEAR GUIDANCE messages per kilometer. Similarly, we define λ_T as the rate of USER LOST messages per minute and λ_D the rate of USER LOST messages per

kilometer. A good measure of efficiency is also the *speed ratio*, that is the ratio between the time required by the walker to finish a mission and the time it took the exploration user to go from the corresponding starting place to the corresponding ending place. Table 6.1 summarizes the values of these metrics for each user. We convert durations to distances using an average walking speed of $0.7\,m/s$ (or $2.5\,km/h$). In order to evaluate the utility of the display of images on the user interface, we turned off this option for the last three users (G to I).

User	# missions	Duration (min)	Distance (m)	Success rate (%)	Speed ratio	λ_T (min^{-1})	λ_D (km^{-1})	γ_T (min^{-1})	γ_D (km^{-1})
A	3	9.4	400	66.7	0.38	0.1	2.5	-	-
B	5	19.1	800	80.0	0.37	0.1	2.5	-	-
C	6	18.2	750	50.0	0.46	0.2	3.9	0.6	14.3
D	3	5.9	250	66.7	0.61	0.2	4.0	0.0	0.0
E	7	16.3	700	71.4	0.51	0.3	5.8	0.3	5.8
F	10	21.3	900	100.0	0.50	0.0	0.0	0.3	7.8
G	8	18.4	750	50.0	0.58	0.1	2.6	0.7	15.5
H	6	13.5	550	62.5	0.58	0.3	7.1	0.2	3.5
I	11	19.3	800	72.7	0.60	0.2	3.7	0.3	7.4
Average	7	15.7	650	70.2	0.51	0.2	3.6	0.3	7.8

Table 6.1: Efficiency evaluation (Experiment A)

In average, each user performed 7 missions over a course of 16 minutes and 650 meters. The average success rate is 70%, with a lower bound of 50% (users C and G) and an upper bound of 100% (user F). The speed ratio is similar for all users, with an average of 0.5. This means that during navigation, the user walks approximately half of the usual speed. The fact that the user must frequently look at the user interface and interpret the navigation commands explains this relatively low speed ratio. On average, users reported an unclear guidance every three minutes (or 140 m) and was lost every five minutes (or 200 m).

We notice that removing the image display on the user interface did not alter the success rate of the users significantly (last three users). In fact, the speed ratio is slightly higher for these users in average, which is consistent with the fact that the user tends to be overwhelmed by the amount of information displayed on the user interface when images are enabled (Table 6.5). Indeed, users G, H and I seemed to process the command given by the interface faster and also to move faster through the environment.

We evaluated the effectiveness of the system by asking users to answer a survey shortly after each experiment. The survey contained questions related to the user's experience of the various components of the user interface, namely the compass, the confidence level, the high-level guidance and the images. We use a 5-point Likert scale in order to evaluate the agreement of the user with various statements. We report the content of the survey in Table 6.2 and the answers of each user in Table 6.3. The survey also includes two open questions that allow users to provide feedback in a textual form on the positive aspects of the system and the improvements they would suggest. Tables 6.4 and 6.5 summarize their answers.

Compass effectiveness	How effective was the compass at telling you which way to go? (1=excellent, 5=poor)
High-level guidance relevance	How relevant was the high-level guidance? (1=very relevant, 5=not relevant at all)
False positive rate	How many false positives did you observe in the High-level guidance? (1= 0%, 5=100%)
Confidence usefulness	How useful do you think the Confidence information was? (1=very useful, 5=not useful at all)
Image usefulness	How much did you rely on the images to navigate? (1=a lot, 5=not at all)
Contradictory guidance	How often were the compass and high-level guidance contradictory? (1=never, 5=always)
Overall trust	How much do you trust this system overall? (1=a lot, 5=none)
Look & feel	What is your overall satisfaction with the look and feel of the user interface? (1=excellent, 5=poor)

Table 6.2: Effectiveness evaluation. Survey questions (Experiment A).

Question	Scores (on a scale from 1 to 5)						Average	Std Dev
Compass effectiveness	4	2	3	1	3	4	2.8	1.1
High-level guidance relevance	2	2	3	2	3	2	2.3	0.5
False positive rate	3	2	2	1	2	4	2.3	1.0
Confidence usefulness	1	4	3	4	1	3	2.6	1.3
Images usefulness	5	3	3	5	2	2	3.3	1.3
Contradictory guidance	3	2	2	1	3	4	2.5	1.0
Overall trust	2	1	3	2	3	3	2.3	0.8
Look & feel	2	2	3	1	2	4	2.3	1.0

Table 6.3: Effectiveness evaluation. Survey answers (Experiment A).

The usual statistical tools for reliability analysis (*e.g.* Cronbach's alpha [17], Kuder-Richardson Formula 20 [16]) do not apply due to the low number of answers (the Cronbach's alpha value for the data presented on Table 6.3 is -0.47). In addition, the answers from the participants exhibit a large variability across users. However, one item presents a significantly lower standard deviation than the others, namely the relevance of the high-level guidance. The data shows that all users judged this aspect of the interface relevant. This information is further confirmed by the textual feedback of the users in Table 6.4, as well as the verbal feedback collected by the experimenter during the study runs. It appears indeed that the participants relied for an important part on non-metric information for navigation (*i.e.* images, confidence level and turn-by-turn directions). The users preferred this medium over the compass, and had the tendency to give priority to non-metric information over the compass in case of disagreement. This effect was particularly striking at a specific location of the environment where the compass would provide the correct direction while the turn-by-turn directions did not, due to a missed true positive in the motion classification

described in § 4.2. The first six users (A to F) systematically went in the wrong direction at that location, while the three users who did not have access to the images (G to I) chose the correct direction. The recent explosion of navigation tools providing turn-by-turn directions (*e.g.* Google Maps, GPS in cars) may be a strong contributor to the preference of the users for this mode of guidance.

Tables 6.4 and 6.5 show the feedback provided by the users shortly after each study run. Most of the positive feedback relates to non-metrical aspects of the user interface such as the high-level guidance and the confidence level. Overall, most users enjoyed the study and felt that the guidance provided by the system was appropriate and understandable. Most suggestions for improvements related to a more intuitive, more situationally-aware device. For example, several users suggested displaying information on the images, or providing more elaborate turn-by-turn directions. The users reacted differently to situations where the compass and the turn-by-turn directions where inconsistent. Some users stood still until the system would provide more consistent guidance, while others would try to guess the right direction and kept walking.

High-level guidance
"I thought the system did a good job indicating where I should go! I thought having the compass front and center was a good idea as that was what I looked at most of the time. Overall the directions it gave were quite clear."
"The photos / street-view like feature are very nice and would make a world of difference for most users. If speed was not a issue, this component would be the basis for an optimal user interface that would superimpose the other instructions on top of the images ... since I think most people find rich visual information much more useful."

Compass
"The compass was accurate most of the time and was easy to use. Image display is also a helpful addition."

Other aspects
"Overall it was quite intuitive. Little explanation needed to get going."
"The text was clear and not ambiguous."
"The weight is fine for a typical person."
"The Confidence information was probably the best performing tool."
"High-level guidance and the images were very nice complements to the arrows. This made it more intuitive, and served as another way to check the output."
"Good cool factor. Definitely not something I've played with before."

Table 6.4: User feedback: positive aspects (Experiment A).

High-level guidance
"More graduations in the left/right turns so as to distinguish hard turns from gentle turns."
"Perhaps you could give information about where *not* to turn, so as to clear up the ambiguity."
"The high level guidance was good, but it was sometimes hard to tell when it was confused and when it was just moving on to the next bit of the tour. I think that having the previous instruction, current instruction, and next instruction, (think directions while driving like in Google Maps) would go a long way towards fixing this issue."
"In areas where there's only one real path, it would be nice to condense the output to something suggesting that the user walks until some waypoint. In places where there are many possible directions, it would be nice Efor the system to be more explicit, or somehow adapt to the context. Ideally it could recognize that a particular area is potentially ambiguous, and present the directions differently."
"I didn't make too much use of the high-level guidance. Maybe something as simple as icons that look more different from each other (e.g., a green icon for upstairs/downstairs)."
"It would be nice if there were a way to confirm I was going the correct way (my test was without pictures, so these may fill that role)."
"The lookahead turn/stairs information was useful, but not entirely accurate. This needs to be improved to have less false positives, otherwise it can be the leading source of confusion by giving the user a completely wrong instruction. This was the leading source of confusion for me and I began to discard it." *(two users)*

Compass
"Could the confidence and compass be merged somehow?"
"The compass performance is pretty poor at times."

Other aspects
"Nice to have: drawing the route on the images." *(two users)*
"The system could definitely be faster – enough so that a person can walk at a normal pace and still get good guidance."
"There is a bit too much information to take in at once, and especially when different pieces of information are contradictory and you have to learn which to trust."
"My main comment is that for general users the interface should be simplified. The interface has the feel of an engineer-designed interface, one that gives the engineer all the information he would want or need, but for the general user overwhelms them." *(two users)*

Table 6.5: User feedback: suggestions for improvement (Experiment A).

6.2 Experiment B

The goal of Experiment B was to test the system on the spatially and temporally extended dataset. Experiment B involved two users. In the first phase of the experiment, one user (the "explorer") performed an extended exploration across the MIT campus while wearing

the backpack. The exploration spanned two floors across nine buildings at a busy time of a week day, for a 22-minute walk (approximately 1.1 km) and included an outdoor section. Figure 6-1(a) shows the notional path followed by the explorer. In the second phase of the experiment, another user (the "retracer") retraced the path followed by the explorer using the navigation guidance provided by the system. Experiment B spans a significantly larger area than Experiment A and thus provides an insight on the scalability of the method.

The explorer left from the MIT Stata Center and performed a loop through MIT campus, passing through the Infinite Corridor and Lobby 7. The exploration last 22 minutes. The retracer took 32 minutes to follow the same route. The speed ratio was 0.70. The user got lost three times and reported an unclear guidance eight times. Figure 6-1(b) shows the location of these events on the map. We analyzed the root cause of the failures and determine two general failure modes (Figure 6-2). In the first case, a false positive in the turn-by-turn direction leads the user in the wrong direction. This happens for instance if a "turn right" command is given and the user has the possibility to turn right. In the second case, the user faces an ambiguous world configuration, while the guidance provided by the system is too coarse to proceed. This happens for instance when the user must turn left but is left to choose between two adjacent corridor entries (or two doors). This failure mode could be addressed by incorporating metrical information in the place graph during exploration.

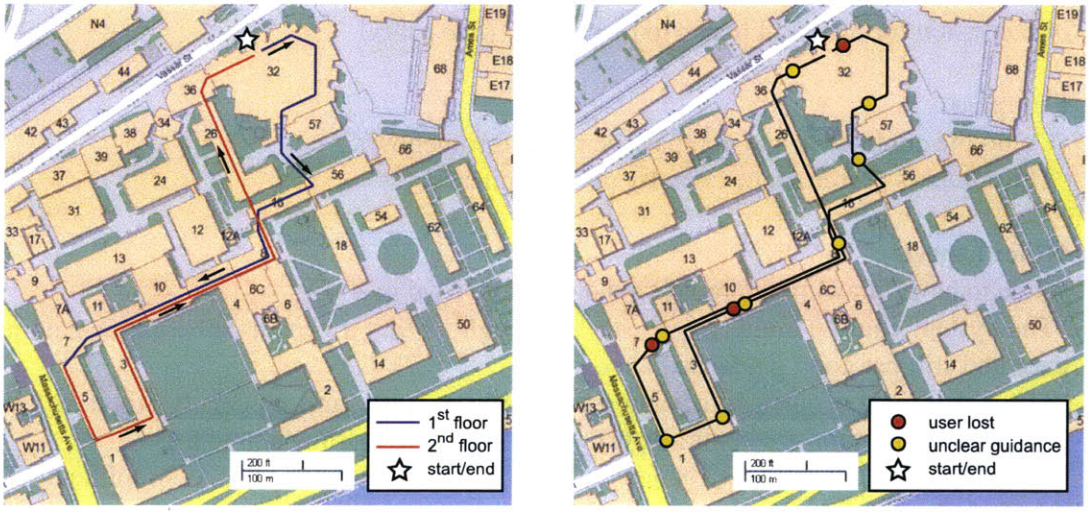

Figure 6-1: *Left:* Notional path followed by the explorer in Experiment B. The exploration path spans two floors of the MIT campus over a 22-minute walk (approx. 1.1 km). *Right:* Notional path followed by the retracer (Experiment B). The user got lost three times and reported unclear guidance at eight locations.

6.3 An Application for the Blind

The visually impaired population is one of the motivating applications of this work. Indeed, 314 million people in the world are visually impaired, out of whom 45 million are

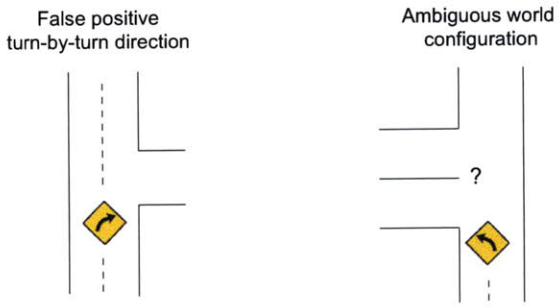

Figure 6-2: Failure modes of the method. *Left*: a false positive in the turn-by-turn direction may lead the user to the wrong direction. *Right*: ambiguous world configurations may make it hard for the user to know which way to choose.

blind [73]. A number of GPS-based solutions have been developed to assist the blind in recent years [9]. However, they are limited to outdoor locations with full sky visibility and suffer from GPS multi-path in urban areas. Today, many blind people still rely on a trained dog or a human assistant to go anywhere, which severely limits their freedom to move in the world and interact with it.

The work presented in this thesis is an attractive solution to navigation for the blind. It relies only on a set of cameras and therefore provides useful navigation guidance in a large class of indoor and outdoor environments. In addition, it circumvents the tedious constraint of camera calibration and presents a natural form factor to a human user by posting the cameras on the shoulders, leaving the user's hands and head free.

Navigation solutions for the blind poses obvious safety and liability issues. In that sense, our work provides only an insight on what an end-to-end solution could be. In particular, the question of the means by which the guidance should be brought to the user's attention is of crucial importance. Clearly, the user interface presented in this work does not provide a full solution to this problem. Most of the participants in our study, who were non-impaired and had a strong engineering background reported being overwhelmed by the amount of information provided by the interface.

The author of this thesis was fortunate to meet with IBM Research Fellow Chieko Asakawa during her visit to the MIT Computer Science Artificial Intelligence Laboratory (CSAIL) in October 2009. Ms. Asakawa spent more than 25 years doing reseach on accessibility at IBM and engaged in a discussion about navigation for the blind during her visit. From this discussion, we learned that audio-based solutions may not be appropriate for the blind, as they use sound for crucial situational awareness. However, vibrating devices such as belts or rings could be a powerful medium to provide directional cues to a blind person in an intuitive and discret fashion.

Chapter 7

Conclusion

In the past decade, vision-based navigation has shown promising results, in particular in the field of Simultaneous Localization and Mapping (SLAM). Recent methods demonstrate robust localization across hundreds of meters through indoor and outdoor environments [53, 69,77]. In order to transform observations from sensor space to world space, these methods rely on camera calibration, which can be achieved in numerous ways [39,94,99,109].

However, camera calibration suffers from several limitations. First, intrinsic calibration is a well-known problem but remains a tedious operation, in particular for large numbers of units. Second, extrinsic camera calibration aims at recovering the camera-to-body transformation and requires optimizing a function in a high-dimension space (6 DOF per camera). This operation becomes quickly untractable as the number of sensor grows. Finally, both intrinsic and extrinsic camera calibration may change under challenging field conditions. Changes in temperature, shocks, humidity are some of the parameters that may affect the calibration of a camera.

This thesis presented a set of algorithms for vision-based navigation that rely only on a set of uncalibrated cameras. The intuition behind our approach is to let the system learn the correspondence between feature matches across cameras and rotation in the user's body frame. We demonstrate the theoretical soundness of our approach using a probabilistic framework and show that our method provides rotation guidance with an accuracy that increases with the square root of the number of observations. Our experiments show an accuracy of 8° with a maximum error of 16° in typical indoor environments.

We also present a method for topological mapping based solely on the variability of visual appearance of the environment. Our method represents the user's exploration path as an undirected graph (*place graph*). Using a recursive Bayesian algorithm, we then demonstrate a method for robust real-time localization of the user within the explored environment.

Loop closure detection, *i.e.* recognizing that the user has returned to a previously seen location, is a fundamental capability for navigation. We build on state-of-the-art "bag-of-words" methods [1,19] and present a fully incremental algorithm for loop closure detection that takes as single input a continuous stream of images. We illustrate our algorithm on several hours of exploration through indoor environments.

We also demonstrate a method for user motion classification using template sequences. Given a 30-second template sequence for each motion category, the method computes the

average optical flow sampled over a coarse grid in image space. During exploration, the algorithm compares the instantaneous optical flow with the database of flow templates to determine the most likely user motion. We use the output of the classification to "augment" the place graph and provide "turn-by-turn" guidance to the user during revisit.

Demonstrating the end-to-end system on real, untrained users is also an important goal of our work. We perform a user study with 9 users spanning 2.5 hours and approximately 6 km of exploration. Each user was asked to follow the guidance provided by the system in order to reach a series of unknown locations in a building. Each mission lasted approximately 20 minutes. We evaluated the efficiency of the system using quantitative metrics such as the ratio of successful missions and the speed of the user. Furthermore, we assessed the effectiveness of the system using a qualitative and quantitative user survey.

Robotics navigation is another field of application of our work. We demonstrated the ability of our method to provide navigation to a wheeled robot equipped with a local obstacle avoidance capability. In a first step, the robot is manually driven through the environment. Using our navigation framework, the robot could then navigate safely and robustly from places to places in the environment. Our approach is particularly appealing from a robotics perspective since it requires no *a priori* camera calibration, which makes it suitable to field robotics applications. We demonstrated our method on several hours of exploration through cluttered and dynamic environments.

Bibliography

[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] Rashmi Bajaj, Samantha Lalinda Ranaweera, and Dharma P. Agrawal. GPS: Location-tracking technology. *Computer*, 35(4):92–94, 2002.

[3] Billur Barshan and Hugh F. Durrant-Whyte. Inertial navigation systems for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 11, pages 328–342, June 1995.

[4] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speedep up robust features. In *Computer Vision and Image Understanding (CVIU)*, volume 110, pages 346 – 359, 2008.

[5] Patrick Beeson, Nicholas K. Jong, and Benjamin Kuipers. Towards autonomous topological place detection using the extended Voronoi graph. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4373–4379, Barcelona, Spain, April 2005.

[6] J. Beis and D. G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Proceedings of the IEEE Internatioal Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1000–1006, Puerto Rico, 1997.

[7] C. Bibby and I. Reid. Simultaneous localisation and mapping in dynamic environments (SLAMIDE) with reversible data association. In *Proceedings of Robotics: Science and Systems (RSS)*, 2007.

[8] Jean-Yves Bouguet. Pyramidal implementation of the Lucas Kanade feature tracker. In *OpenCV Distribution*, 2000.

[9] N. A. Bradley and M. D. Dunlop. An experimental investigation into wayfinding directions for visually impaired people. *Personal Ubiquitous Computing*, 9(6):395–403, 2005.

[10] M A Brodie, A Walmsley, and W Page. The static accuracy and calibration of inertial measurement units for 3D orientation. *Comput Methods Biomech Biomed Engin*, 11(6):641–8, 2008.

[11] Vincenzo Caglioti and Pierluigi Taddei. Planar motion estimation using an uncalibrated general camera. In *Workshop on Omnidirectional Vision, Camera Networks and Non-classical cameras (OMNIVIS)*, October 2008.

[12] Dongwei Cao, O.T. Masoud, D. Boley, and N. Papanikolopoulos. Online motion classification using support vector machines. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pages 2291–2296, 2004.

[13] A. Chatfield. *Fundamentals of High Accuracy Inertial Navigation.* American Institute of Aeronautics and Astronautics, Inc. (AIAA), 1997.

[14] L.A. Clemente, A. J. Davison, I.D Reid, J. Neira, and J.D.Tardos. Mapping large loops with a single hand-held camera. In *Proceedings of Robotics: Science and Systems (RSS)*, Atlanta, GA, USA, June 2007.

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

[16] Jose M. Cortina. What is coefficient alpha? an examination of theory and applications. *Journal of Applied Psychology*, 78(1):98–104, 1993.

[17] Lee Cronbach. Coefficient alpha and the internal structure of tests. *Psychometrika*, 16(3):297–334, 1951.

[18] Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision at the European Conference on Computer Vision*, pages 1–22, 2004.

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

[20] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. MonoSLAM: Real-time single-camera SLAM. *Pattern Analysis and Machine Intelligence (PAMI)*, 26(6):1052–1067, 2007.

[21] Frank Dellaert and Michael Kaess. Square root SAM: Simultaneous localization and mapping via square root information smoothing. *International Journal of Robotics Research*, 25(12):1181–1203, 2006.

[22] R.M. Eustice, H. Singh, and J.J. Leonard. Exactly sparse delayed-state filters. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2417–2424, April 2005.

[23] Olivier Faugeras, Quang-Tuan, and T. Papadopoulou. *The Geometry of Multiple Images.* MIT Press, Cambridge, MA, USA, 2001.

[24] Leandro A. F. Fernandes and Manuel M. Oliveira. Real-time line detection through an improved hough transform voting scheme. *Pattern Recognition*, 41(1):299–314, 2008.

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

[26] Carl Fischer, Kavitha Muthukrishnan, Mike Hazas, and Hans Gellersen. Ultrasound-aided pedestrian dead reckoning for indoor navigation. In *MELT '08: Proceedings of the first ACM international workshop on Mobile Entity Localization and Tracking in GPS-less environments*, pages 31–36, New York, NY, USA, 2008. ACM.

[27] Matthias O. Franz, Bernhard Schlkopf, Hanspeter A. Mallot, and Heinrich H. Blthoff. Where did I take that snapshot? Scene-based homing by image matching. *Biological Cybernetics*, 79:191–202, 1998.

[28] Joshua Gluckman and Shree K. Nayar. Ego-motion and omnidirectional cameras. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, page 999, Washington, DC, USA, 1998. IEEE Computer Society.

[29] T. Goedemé, T. Tuytelaars, L. Van Gool, D. Vanhooydonck, E. Demeester, and M. Nuttin. Is structure needed for omnidirectional visual homing? *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 303–308, June 2005.

[30] T. Goedemé, T. Tuytelaars, G. Vanacker, M. Nuttin, and L. Van Gool. Omnidirectional sparse visual path following with occlusion-robust feature tracking. In *Proceedings of the sixth workshop on omnidirectional vision camera networks and non-classical cameras (OMNIVIS)*, 2005.

[31] Toon Goedemé, Marnix Nuttin, Tinne Tuytelaars, and Luc Van Gool. Omnidirectional vision based topological navigation. *International Journal of Computer Vision*, 74(3):219–236, 2007.

[32] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with Rao-Blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46, 2007.

[33] J. J. Guerrero and C. Sagüés. Uncalibrated vision based on lines for robot navigation. *Mechatronics*, 11(6):759 – 777, 2001.

[34] J. Guivant and E. Nebot. Optimization of the simultaneous localization and mapping and map building algorithm for real time implementation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 17, pages 242–257, 2001.

[35] J. Guivant and E. Nebot. Improving computational and memory requirements of simultaneous localization and map building algorithms. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2731–2736, 2002.

[36] Mei Han and T. Kanade. Creating 3D models with uncalibrated cameras. In *Fifth IEEE Workshop on Applications of Computer Vision*, pages 178–185, 2000.

[37] C. Harris and M. Stephens. A combined corner and edge detection. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151, 1988.

[38] R.I. Hartley. Projective reconstruction from line correspondences. In *Proceedings of the IEEE Internatioal Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 903–907, Jun 1994.

[39] Janne Heikkila and Olli Silven. A four-step camera calibration procedure with implicit image correction. In *Proceedings of the IEEE Internatioal Conference on Computer Vision and Pattern Recognition (CVPR)*, page 1106, Washington, DC, USA, 1997. IEEE Computer Society.

[40] Kin Leong Ho and Paul Newman. Detecting loop closure with scene sequences. *International Journal of Computer Vision*, 74(3):261–286, September 2007.

[41] B.K.P. Horn, Yajun Fang, and I. Masaki. Time to contact relative to a planar surface. In *IEEE Intelligent Vehicles Symposium*, pages 68–74, June 2007.

[42] Velodyne Lidar Inc. High definition LIDAR HDL-64E S2 specifications, 2008.

[43] Andrew J. Davison Javier Civera, O. Garcia and J. M. M. Montiel. 1-point RANSAC for EKF-based structure from motion. In *Proceedings of the IEEE International Conference on Intelligent Robots and System (IROS)*, October 2009.

[44] P. Jost, P. Vandergheynst, and P. Frossard. Tree-based pursuit: Algorithm and properties. In *IEEE Transactions on Signal Processing*, volume 54, pages 4685–4697, 2006.

[45] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[46] Yan Ke and R. Sukthankar. PCA-SIFT: a more distinctive representation for local image descriptors. In *Proceedings of the IEEE Internatioal Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 506–513, 2004.

[47] J. Knight, A. Davison, and I. Reid. Towards constant time SLAM using postponement. In *Proceedings of the IEEE International Conference on Intelligent Robots and System (IROS)*, pages 405–413, 2001.

[48] Olivier Koch and Seth 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.

[49] Olivier Koch, Matthew Walter, Albert Huang, and Seth Teller. Ground robot navigation using uncalibrated cameras. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Anchorage, Alaska, May 2010. To appear.

[50] Kurt Konolige, Motilal Agrawal, Morten Rufus Blas, Robert C. Bolles, Brian Gerkey, Joan Solà, and Aravind Sundaresan. Mapping, navigation, and learning for off-road traversal. *IEEE Journal of Field Robotics*, 26(1):88–113, 2009.

[51] D. Launer. *Navigation through the ages*. Sheridan House, first edition, April 2009.

[52] T. Lemaire, S. Lacroix, and J. Sola. A practical 3D bearing-only SLAM algorithm. In *Proceedings of the IEEE International Conference on Intelligent Robots and System (IROS)*, pages 2449–2454, August 2005.

[53] Thomas Lemaire, Cyrille Berger, Il-Kyun Jung, and Simon Lacroix. Vision-based SLAM: Stereo and monocular approaches. *International Journal of Computer Vision*, 74(3):343–364, 2007.

[54] J. Leonard and P. Newman. Consistent, convergent, and constant-time slam. In *Proceedings of the 18th international joint conference on Artificial intelligence (IJCAI)*, pages 1143–1150, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc.

[55] Dahua Lin, John Fisher, and Eric Grimson. Learning visual flows: A Lie algebraic approach. In *Proceedings of the IEEE Internatioal Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 747–754, 2009.

[56] Brad Lisien, Deryck Morales, David Silver, George A. Kantor, Ioannis Rekleitis, and Howie Choset. *The hierarchical atlas*, 21(1):473 – 481, June 2005.

[57] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[58] K. Lynch. *The image of the city*. Cambridge, MA: MIT Press, 1960.

[59] J. Matas, O. Chum, U. Martin, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proceedings of British Machine Vision Conference*, volume 1, pages 384–393, London, 2002.

[60] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1-2):43–72, 2005.

[61] 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.

[62] J. Modayil, P. Beeson, and B B. Kuipers. Using the topological skeleton for scalable global metrical map-building. *Proceedings of the IEEE International Conference on Intelligent Robots and System (IROS)*, 2(28):1530 – 1536, September 2004.

[63] R. Möller. Do insects use templates or parameters for landmark navigation? *Journal of Theoretical Biology*, 210(1):33–45, 2001.

[64] R. Möller, D. Lambrinos, T. Roggendorf, R. Pfeifer, and R. Whener. Insect strategies of visual homing in mobile robots. *Biorobotics - Methods and Applications*, pages 37–66, 2001.

[65] J. Montiel, J. Civera, and A. Davison. Unified inverse depth parametrization for monocular SLAM. In *Proceedings of Robotics: Science and Systems (RSS)*, Philadelphia, USA, August 2006.

[66] Don Murray and James J. Little. Using real-time stereo vision for mobile robot navigation. *Autonomous Robots*, 8(2):161–171, 2000.

[67] M. E. Nilsback and A. Zisserman. A visual vocabulary for flower classification. In *Proceedings of the IEEE Internatioal Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 1447–1454, 2006.

[68] David Nistér. An efficient solution to the five-point relative pose problem. *Pattern Analysis and Machine Intelligence (PAMI)*, 26(6):756–777, 2004.

[69] David Nister, Oleg Naroditsky, and James Bergen. Visual odometry. *Proceedings of the IEEE Internatioal Conference on Computer Vision and Pattern Recognition (CVPR)*, 1:652–659, 2004.

[70] David Nister, Oleg Naroditsky, and James Bergen. Visual odometry for ground vehicle applications. *IEEE Journal of Field Robotics*, 23(1):3–20, 2006.

[71] David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *Proceedings of the IEEE Internatioal Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2161–2168, Washington, DC, USA, 2006. IEEE Computer Society.

[72] Edwin Olson. *Robust and Efficient Robotic Mapping*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, June 2008.

[73] World Health Organization. Visual impairment and blindness, May 2009. http://www.who.int/mediacentre/factsheets/fs282/en.

[74] A Zisserman P A Beardsley and D W Murray. Navigation using affine structure from motion. In *Proceedings of the IEEE European Conference on Computer Vision (ECCV)*, Lecture Notes in Computer Science, pages 85–96. Springer, 1994.

[75] Lina María Paz, Pedro 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.

[76] S. Perantonis, N. Vassilas, Th. Tsenoglou, and K. Seretis. Robust line detection using weighted region based hough transform. *Electronics Letters*, 34(7):648–650, April 1998.

[77] Ingmar Posner, Mark Cummins, and Paul Newman. A generative framework for fast urban labeling using spatial and temporal context. *Autonomous Robots*, 26(2-3):153–170, April 2009.

[78] Edward Rosten and Tom Drummond. Fusing points and lines for high performance tracking. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1508–1511, October 2005.

[79] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *Proceedings of the IEEE European Conference on Computer Vision (ECCV)*, volume 1, pages 430–443, May 2006.

[80] Boris Ruf, Effrosyni Kokiopoulou, and M. Detyniecki. Mobile museum guide based on fast SIFT recognition. In *Sixth International Workshop on Adaptive Multimedia Retrieval*, Berlin, Germany, June 2008.

[81] Jose Santos-Victor and Giulio Sandini. Uncalibrated obstacle detection using normal flow. In *Machine Vision and Applications*, volume 9, pages 130–137, May 1996.

[82] José Santos-Victor and Giulio Sandini. Uncalibrated obstacle detection using normal flow. *Machine Vision and Applications*, 9(3):130–137, 1996.

[83] Angel D. Sappa, Niki Aifanti, Sotiris Malassiotis, and Michael G. Strintzis. Unsupervised motion classification by means of efficient feature selection and tracking. *International Symposium on 3D Data Processing Visualization and Transmission*, 0:912–917, 2004.

[84] J. Sato and R. Cipolla. Affine reconstruction of curved surfaces from uncalibrated views of apparent contours. *Pattern Analysis and Machine Intelligence (PAMI)*, 21(11):1188–1198, Nov 1999.

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

[86] Derik Schroeter and Paul Newman. On the robustness of visual homing under landmark uncertainty. In *In Proc. of 10th International Conference on Autonomous Systems (IAS)*, Baden Baden, Germany, July 2008.

[87] Bruno Sinopoli, Mario Micheli, Gianluca Donato, and John Koo. Vision based navigation for an unmanned aerial vehicle. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1757–1765, 2001.

[88] S.M. Smith and J.M. Brady. SUSAN - a new approach to low level image processing. In *International Journal of Computer Vision*, volume 23, pages 45–78, May 1997.

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

[90] J. Sola, A. Monin, M. Devy, and T. Lemaire. Undelayed initialization in bearing only SLAM. In *Proceedings of the IEEE International Conference on Intelligent Robots and System (IROS)*, pages 2499–2504, August 2005.

[91] C. Stachniss and G. Grisetti. GMapping project at OpenSLAM.org. http://www.openslam.org/gmapping.html.

[92] D. Steinberg and Y. Birk. An empirical analysis of the IEEE-1394 serial bus protocol. *IEEE Micro*, 20:58–65, 2000.

[93] Irem Stratmann and Erik Solda. Omnidirectional vision and inertial clues for robot navigation. *Journal of Robotics Systems*, 21(1):33–39, 2004.

[94] Peter Sturm and Steve Maybank. On plane-based camera calibration: A general algorithm, singularities, applications. In *Proceedings of the IEEE Internatioal Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 432–437, Juin 1999.

[95] Sebastian Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

[96] Engin Tola, Vincent Lepetit, and Pascal Fua. A fast local descriptor for dense matching. In *Proceedings of the IEEE Internatioal Conference on Computer Vision and Pattern Recognition (CVPR)*, Alaska, USA, 2008.

[97] Doug Tougaw. Finding your way with the Garmin GPS V. *Computing in Science and Engineering*, 4(3):10–13, 2002.

[98] N.B. Touzene and S. Larabi. Obstacle detection from uncalibrated cameras. In *Panhellenic Conference on Informatics*, pages 152–156, Aug. 2008.

[99] Roger Y. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. pages 221–244, 1992.

[100] Hendrik A. H. C. van Veen, Hartwig K. Distler, Stephan J. Braum, and Heinrich H. Bülthoff. Navigating through a virtual city: using virtual reality technology to study human action and perception. *Future Generation Computer Systems. Special Issue on Virtual Reality in industry and research.*, 14(3-4):231–242, 1998.

[101] Andrew Vardy and Ralf Moeller. Biologically plausible visual homing methods based on optical flow techniques. *Connection Science, Special Issue: Navigation*, 17:47–90, 2005.

[102] David Vissière. *Guidance, navigation and control solutions for unmanned heterogeneous vehicles in a collaborative mission.* PhD thesis, Ecole des Mines de Paris (ParisTech), June 2008.

[103] I. Weiss. 3D curve reconstruction from uncalibrated cameras. In *Proceedings of the IEEE Internatioal Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 323–327, Aug 1996.

[104] M. Werman, S. Banerjee, S.D. Roy, and M.L. Qiu. Robot localization using uncalibrated camera invariants. In *Proceedings of the IEEE Internatioal Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 353–359, 1999.

[105] Brian Williams, Mark Cummins, Jos Neira, Paul Newman, Ian Reid, and Juan D. Tardos. An image-to-map loop closing method for monocular SLAM. In *Proceedings of the IEEE International Conference on Intelligent Robots and System (IROS)*, Nice, France, 2008.

[106] Tom Yeh, John Lee, and Trevor Darrell. Adaptive vocabulary forests by dynamic indexing and category learning. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1–8, October 2007.

[107] J. Zeil, H.I. Hoffmann, and J.S. Chal. Catchment areas of panoramic images in outdoor scenes. *Journal of the Optical Society of America A*, 20(3):450–469, 2003.

[108] Wei Zhang and Jana Košecká. Hierarchical building recognition. *Image and Vision Computing*, 25(5):704–716, 2007.

[109] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. *International Journal of Computer Vision*, 1:666, 1999.