Automatic Mapping of Dynamic Office Environments

Clayton Kunz(1)
Thomas Willeke(2)
Illah R. Nourbakhsh(3)

(1) Silicon Graphics, Inc.
(2)Mobot, Inc.
(3)The Robotics Institute, Carnegie Mellon University

1 INTRODUCTION

Research in automatic robot mapping has made significant strides in recent years. However, a critical requisite for the use of these technologies in real-world settings, the ability to function in unmodified and populated office buildings, has remained out of reach.

In this paper, we present a robot architecture that enables automatic mapping in unmodified office buildings during normal business hours. The architecture that we describe, which is an instance of the feature-extraction approach to mapping and navigation, is also unique in its ability to hypothesize the existence of unseen passages in its environment and then test and validate its hypotheses through exploration.

As will become clear throughout this paper, the most unusual aspect of this work is the architecture's heavy reliance on odometry information. The inherent noise of odometry is alleviated by depending upon symmetries and collinearity properties that are valid in the indoor office environment. Ranging sensors such as sonar are, of course, critical to the present architecture; however, the impact of ranging data is felt only indirectly by the mapping algorithm, which *infers* environmental structure based on the motion behavior of the robot.

In addition to describing this architecture in detail, we present initial empirical results based on our ongoing effort to test InductoBeast in various office buildings at Stanford University and elsewhere. An important aspect of this project is to demonstrate both reliable mapping and, thereafter, reliable navigation in a significant number of different venues and over a large set of trials. The empirical tests reported here demonstrate the ease with which InductoBeast can be introduced to a new environment as well as its long-term navigation reliability.

2 GOALS

The primary goal of this work is to develop a robot navigation system that functions well enough to control an unsupervised mobile robot functioning in a real office environment. Because this top-level goal is somewhat vague, we chose to identify the specific criteria to be addressed as well as a list of the precise simplifying assumptions that we would make. In sections 2.1 and 2.2, we describe these criteria and simplifying assumptions in detail.

2.1 Criteria for autonomous navigation

The criteria we impose on a robot navigator fall roughly into two categories: criteria concerning the environment of the robot (outside the robot's control), and criteria concerning the run-time behavior of the robot. First we discuss a description of the types of environments in which InductoBeast must operate.

• Dynamic Worlds
An autonomous navigator must function reliably in a dynamic office environment. By *dynamic* we mean that the environment will undergo unpredictable and transient structural changes. Doors close and open; unexpected obstacle sometimes block hallways partially or entirely; humans crowd the hallways at lunchtime, and leave the hallways empty at other times; et cetera. We therefore require the navigator to function well under these ordinary circumstances of structural change that are common in all office buildings.

We require the robot to perform mapping and navigation tasks during working hours in office environments. Thus,

all of the empirical results presented in Section 5 were recorded in dynamic environments.

• Almost Autonomous Mapping
We believe that the robot must explore and map an unfamiliar environment with no human input. We will, however, allow human *supervision* in order to avert disaster if the robot encounters stairs (the sonars will not see stairs so we stand there to protect the robot). Thus the term, "Almost Autonomous."

• Insistent Exploration
The present work purposefully blurs the distinction between exploration and navigation. If a human provides InductoBeast with a navigation goal, it will honor that goal before continuing with exploration. But even that navigation task will be transformed, whenever possible, into an episode of exploration. For instance, if the hallway geometry suggests a possible shortcut to the goal point, InductoBeast will try to find and use that shortcut before reverting to a known path to the goal (thus the name, InductoBeast, derived from *induction*).

The concept of continuous exploration is not only intellectually appealing, it is a requisite to being able to handle dynamic worlds. Since the world is ever-changing, it is up to the robot to continuously detect changes to its world model and thereby track the changing world. The robot is always an explorer.

• Reliable Navigation
InductoBeast must be able to travel to any point in its map consistently and with neither human help nor human supervision. The most important facet of this goal is that we require InductoBeast to demonstrate significant reliability *over the long term*, using large numbers of navigation runs in a wide variety of building layouts.

Controls in such real-world tests are extremely hard to institute. Our self-imposed rules included testing only during normal office hours, testing for a minimum of 3 hours, and performing no training of the personnel in the environment.

2.2 Simplifying assumptions

The following is an exhaustive discussion of simplifying assumptions we have made. The small number and easy satisfiability of these assumptions have been crucial in enabling us to introduce InductoBeast to various buildings without modifying the robot architecture.

Ongoing research involves exploration of techniques for relaxing some of these assumptions, most notably the Rectilinearity assumption. Our newest research results indicate that a complete shift from range-based navigation (e.g. sonar, infrared, etc.) to visual navigation is key to eliminating this assumption most effectively.

• Rectilinearity
The strongest assumption we make is that the office building in question is rectilinear; that is, all intersections join halls at angles of 90 or 180 degrees. This assumption rules out some office buildings. Note, however, that open areas such as foyers may be of any shape.

• Hallway Widths
Our second assumption is a relaxation of the ultimate autonomy goal: that the robot learn to navigate a new office building with zero a priori knowledge of the floorplan. We allow the robot one piece of information about a new office building: an approximate range of hallway widths. This simple piece of information greatly simplifies discrimination of hallways from open areas.

• Intersection Separation
InductoBeast's final assumption is that adjacent hallway intersections are always separated by a minimum distance of 6 feet. As with hallway width information, this assumption helps InductoBeast distinguish between changes in the topology of one hallway (a transient, wider portion) and a new hallway.

There is one type of intersection that this assumption rules out: the staggered intersection, depicted in Figure 1.

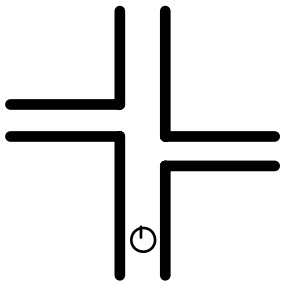

Figure 1: An example of a staggered hallway, a topology assumed out of existence by InductoBeast.

Together, the above set of assumptions and the goals in Section 2.1 define the problem InductoBeast is slated to solve. The most important elements of this problem, then, are that the robot navigator must map and navigate competently, taking advantage of potential shortcuts whenever possible. And, it must do so without human

intervention, in the dynamic real world of an office building during working hours.

In the following sections, we describe the basic architecture for mapping, induction and navigation. Then, we go on to describe the implemented robot system and associated empirical results.

3 ARCHITECTURE

The architecture of our system is best understood visually. The system is divided between the high level control and reasoning and lower level robot behaviors. High level control is further divided into six sections, shown in Figure 2 below. The ovals represent separate modules while the arcs represent the transfer of information between modules via shared data structures.

It is important to note that the control architecture we describe is single-threaded (see Section 4). The modules in Figure 2 are not loosely coupled, autonomous modules that execute in parallel, as one would expect from a behavior-based architecture. Rather, this modularization represents the classification of the control code into distinct competencies. A single thread of control calls upon functions that belong to each of these modules as needed during the robot's execution.

As implemented, the system is not as clean as the graph indicates; additional lines of communication have been required in order to capture side effects between the reasoning module and the planner, for instance. Sections 3.1 through 3.3 describe the six components in Figure 2, and expand upon the unique characteristics of this architecture and its learning and planning components.

3.1 High-level components of the architecture

• Explorer
The basic function of the robot is to expand the extent of its map. For this behavior to be deliberate rather than just coincidental, the robot must actively pursue the goal of moving to the frontier of its map, then stepping beyond it. Unless a human interacts with InductoBeast, it will function by calling the planner with a goal set composed of locations at the fringes of its map. The planner finds a path to the nearest fringe, and the robot executor proceeds to take the robot to the edge of its map, and then beyond.

• Planner
Navigation requires planning. This module is actually the simplest of the six, as planning is well-defined. Given the gross scale of the map, planning is also very fast because of the high degree of geometric abstraction. The planner is

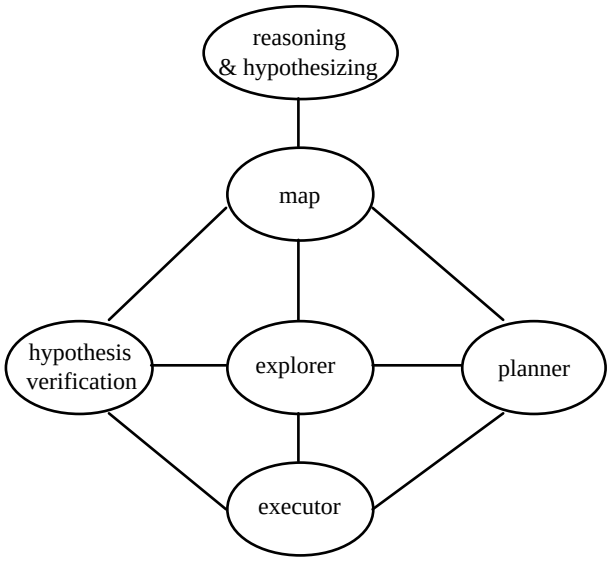

Figure 2: Abstract architecture of InductoBeast.

called either by the explorer module, or if a human indicates a user-specified goal to InductoBeast by clicking on a particular location of the map.

The planner conducts polynomial-time search via breadth-first search with marked nodes. The search algorithm is similar to standard cell decomposition and subsequent potential field cell growth.

An important aspect of the planner's search algorithm is that explicit, hypothesized (and as yet untested) hallways are used as well as validated routes. Therefore, the planner returns the shortest-length *hypothetical* solution for traveling from the current location to the goal location.

• Executor
In the simplest case, the executor's job is to control execution of the planner's output. If the planner's solution makes exclusive use of valid hallways, then this is merely a navigation exercise, and the executor's part is straightforward.

However, when the solution plan contains traversals of hypothetical hallways, then the executor is responsible for ensuring that the behavior of the robot is graceful if the hypothetical hallway is proven not to exist after all.

When traversal of a hallway is complete, the hypothesis verification module determines either success, when the encoder values validate the hypothesis, or failure, when the encoder values indicate that the robot is in an unexpected location. At this point, the executor is responsible for testing the validity of the remaining plan steps, re-invoking the planner as needed on the now updated world map. This process of plan re-invocation and map modification repeats until the robot achieves its goal location.

• Map
The map is a simple graph that represents the topology of the office building. Intersections and transitions between hallways and open areas are represented by nodes in the graph. Each node contains information about relative location, the behavior used to navigate between nodes, and whether or not adjacent nodes have been visited.

Behavior choices are left wall follow, right wall follow, and corridor follow. The latter is used in well-defined hallways while all other open areas are navigated with the first two behaviors.

An important aspect of the map is that it captures not only the experimentally demonstrated topology of the environment, but also contains all hypothetical connectivity (marked as such). Thus, the planner is able to find solutions in a search space that is really the union of the known and the hypothetical.

• Reasoning and hypothesizing
This module does not involve any robot control, but rather manipulates the map directly. Every time the map changes, an event is generated that triggers this module to reexamine the map and generate appropriate hypothetical hallways based on building and collinearity. Any collinear intersections are assumed to be connected by a hallway or set of hallways (until proven otherwise).

An interesting effect of this module involves retraction of hypotheses: when InductoBeast explores a hypothetical hallway, only to discover that its proposed topology was wrong, then the Hypothesis Verification module will immediately modify the map to make it consistent with the observed connectivity. But this map modification once again triggers the present module, causing new hypotheses to be generated in keeping with the most recently observed data. Thus, the interplay between this and other modules can lead to a very rapid "evolution" of the map as InductoBeast explores and modifies its map in an interleaved fashion.

• Hypothesis verification
This is a difficult component to implement because it combines competences from exploration and plan execution. On the one hand, the terrain being traversed is novel, so exploration is required. On the other hand, a path has been predicted through the terrain, and so this path must be checked for correctness. Hypothesis verification not only determines if the hypothetical hallway exists; it must also determine whether the robot reaches known intersections or is truly exploring a new area.

This module functions by comparing estimates of the robot's position in its map with the actual ground track observed during navigation. Recall that the robot "localizes" primarily using encoder data, depending on sonar range-finding data only insofar as that information impacts the robot's motion over time in behaviors such as *corridor-follow*. Given the rectilinearity assumption, rotational drift becomes a non-problem (since hallways are assumed straight), and linear distance traveled becomes the primary navigation information, based on encoder values.

3.2 Motivation of architecture

Our robot is unusual in that the high level controller takes nearly all of its information about the world from the robot's behavior over time rather than the robot's sensors. Though the robot behavior itself is dependant upon the sensors, there is a large difference in abstraction. The code which controls behavior is very simple and reactive; thus by using the behaviors for our map we avoid the need for a costly feature extractor and get high level information almost for free. For example, if the robots heading changes by 90 degrees, we know we have turned a corner without having to extract this information from the sonar data. This reliance upon behavior rather than sensor values makes the robot more resistant to sonar inaccuracies, glitches, and obstacles during mapping. Since none of this information is directly captured by the map, InductoBeast ignores transient sonar readings and temporary obstacles during subsequent navigation episodes.

An interesting result of this is that our map is not just a topological map, but also a behavioral map. It records which motion behaviors will be needed to get from one intersection to the next, on the simple theory that if those behaviors worked when InductoBeast first mapped the area, they should work on subsequent navigation tasks.

Figure 2 implies that hypothesis generation can modify the map that is used during navigation. The result of this coupling is a dynamic map. We believe that a dynamic map is more realistic than a static map, since the world is itself dynamic. Coupled with intelligent exploration, the robot will be able to recover from hallways that are completely blocked off during navigation, even if its map indicates that no detour exists (assuming such a detour *actually* exists).

3.3 Learning and planning

Reasoning and learning appear in several guises throughout this work. InductoBeast proposes hypothetical hallways based on office building symmetry and the collinearity of mapped intersections. Abduction, the standard mapping method, occurs whenever such hypothetical hallways are verified and during normal mapping (Shanahan 1995). The architecture depicted by Figure 2 is designed to facilitate both of these types of reasoning.

Planning occurs in two places: a human can specify any intersection as a goal, and the robot will plan and execute a path to it. Planning also takes place during intelligent exploration. The robot always seeks to increase its knowledge about world topology. Whenever intelligent exploration is called for, the robot plans a path to the nearest location where the map indicates that unexplored territory may lie. Replanning occurs in three cases: when the robot encounters an impassable obstacle; when the robot explores and rejects an inductively proposed shortcut to its goal position; or when the inductively proposed shortcut exists but does not go where expected.

4 IMPLEMENTATION

We implemented the above architecture using a Nomad 150 robot, controlled by a Macintosh Common Lisp 2.01 program running on a Macintosh Powerbook 170. The Nomad 150 is equipped with motor shaft encoders and with 16 sonars arranged radially.

4.1 High-level robot control

The map representation is simply an annotated graph in which nodes represent intersections and arcs represent methods for traveling between intersections. Each node may have up to four arcs pointing from it to other nodes (this limitation follows from the rectilinearity assumption). Each arc has a mode of travel (hall or wall follow), a distance (a scalar or vector, depending on travel mode), and an arc type (verified, hypothetical, blocked, wall, unexplored) associated with it.

The robot has two basic motion states: exploration and navigation. Whenever the robot has no human-given navigation goal, it will attempt to expand its map by traveling to unexplored places. InductoBeast will add new intersections during exploration in the following circumstances:

- a corner in the hallway is detected, by observing that the robot's direction of travel has changed
- an opening long enough to travel through is detected by examining a series of side sonar range values
- a hallway is detected after the robot has been in an open area

When an opening is detected during exploration of a new hallway, the robot assumes that it has entered an open area and switches from hallway follow mode to wall follow mode. If InductoBeast has never been to this location before it will attempt to follow the wall that has disappeared from sight (i.e. it will turn into the opening). If it has been through this intersection before it will explore new ground by following the wall opposite from the opening. Openings and hallways are detected using moving-window averages of recent sonar values.

The second motion state, navigation, is engaged when a human gives InductoBeast a goal. When this occurs, InductoBeast behaves like a traditional robot, planning and executing a path to the destination. But because the planner does not distinguish between valid hallways and hypothesized, unexplored hallways, InductoBeast's path may use a hypothetical connection. Therefore, during execution, the verification component monitors navigation and replans if the hypothetical connection turns out to be incorrect.

4.2 Navigation and localization

The planner's output is an ordered list of adjacent nodes. Based on this list and the map, InductoBeast determines the distance and direction to travel between each pair of nodes. The map annotations also indicate whether InductoBeast should use hall- or wall-follow behavior. To determine when it has reached the goal node, InductoBeast compares the distance it has travelled since the start node to the distance it should travel to reach the destination node using relative encoder values. If the mode of travel is wall-follow, InductoBeast also looks for the presence of a hallway as an exit condition.

InductoBeast also avoids obstacles as it travels. This is implemented in a low-level black box module utilizing case-based motion vectors to discretely approximate a potential field approach (Khatib 1986; Nourbakhsh et al. 1995). At the high level we need only to execute a function called *move* which controls all of the robots forward motion and obstacle avoidance. When it is in hall-follow mode, InductoBeast also attempts to remain centered in the hallway in the absence of obstacles.

To compensate for encoder drift, the robot's conception of straight is adjusted approximately every 12 feet during hallway travel to match the actual trajectory that the robot has taken during that period. Reasoning about obstacles is extremely simple; we use a case-based system that directs the robot toward the area with the greatest freespace.

Our localization system is unusual in that it ignores sonar data completely. We localize from node to node using relative encoder values, the compensated conception of straight ahead, and the rectilinearity assumption. The encoders are accurate to within a few linear inches after the robot has traveled the length of a hallway. Our strategy of localizing after each leg eliminates potentially cumulative encoder errors that can be caused if the encoder values are used over more than the length of a single hallway.

Conventional wisdom states that reliance on encoder values for localization is doomed because of cumulative drift. As our empirical results indicate, we have found this to be untrue in the case of rectilinear office buildings. Of course, this is due in part to our rectilinearity assumption since it is well known that the key source of encoder error in mobile robotics is rotational drift and not linear inaccuracy.

5 EMPIRICAL RESULTS

We have conducted empirical tests to demonstrate both accurate mapping and reliable navigation using InductoBeast's own maps. During the exploration phase, we did not allow dynamic obstacles to interfere with InductoBeast in open areas, although humans regularly passed InductoBeast in hallways. During the navigation phase, we were primarily interested in demonstrating that the maps InductoBeast generates are sufficiently accurate for reliable navigation. Again, we barred malicious humans from open areas although we allowed normal interaction in hallways.

We have tested InductoBeast in the Gates Computer Science building and the Psychology building on the Stanford campus, and in the robot contest arena at AAAI 1996. The development of the code took place entirely in the Gates building; the only parameters modified between subsequent locales were the hallway width ranges and minimum distance required between adjacent intersections. Under no circumstances were tape measures used nor were InductoBeast's maps manually modified.

The following subsections briefly describe these venues.

• Development in the Gates building

InductoBeast was developed entirely in the Gates building. InductoBeast (then known as InductoBeastie 3000) was successfully demonstrated at the AAAI Spring Symposium at Stanford in March 1996, during which the robot made 5 exploration runs and navigation runs through crowded hallways, with no collisions and without any failures.

InductoBeast underwent several changes after the symposium, the most major of which was the ability to navigate through open areas by wall following. During this phase of development, which spanned approximately 160 hours of testing, InductoBeast collided with walls fewer than five times.

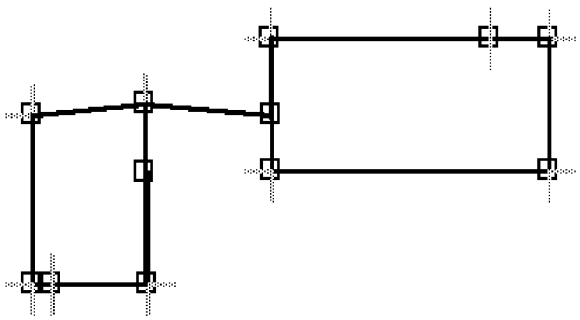

Figure 3: InductoBeast's map of two wings of the second floor of the Gates Computer Science building at Stanford.

• Foreign environment test in the Psychology building

We tested InductoBeast on the second floor of Stanford's psychology building during normal business hours with no code or parameter changes. InductoBeast performed extremely well on its first run, producing a map which was topologically correct but composed mostly of open areas due to the fact that the psychology building has much wider corridors than Gates. After we increased the approximate hallway width parameter by 3 feet, InductoBeast produced the map shown in Figure 4.

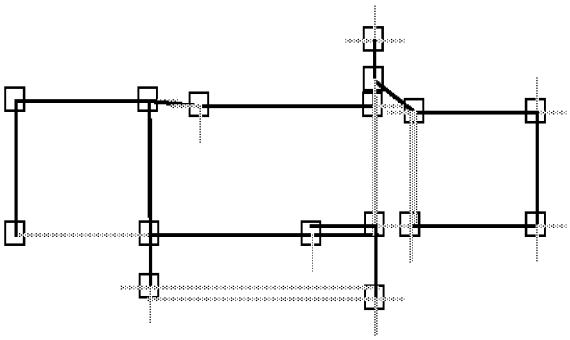

Figure 4: The map made by the third run that InductoBeast made in the 2nd floor of the Psychology building at Stanford, after the hallway width parameter was adjusted. The gray lines are hypothetical hallways.

• Artificial benchmark environment at AAAI

The AAAI '98 robot contest included a robot navigation and meeting coordination task. For this contest, an artificial office environment consisting of approximately 7 hallways, 1 foyer and 8 rooms was set up in a large convention center in Seattle, WA. The walls were built of partition material normally found in office buildings with cubicles.

This contest maze is smaller and less cluttered than most office building floors, which makes it at the same time unrealistic and extremely useful, because situations can be easily constructed and repeated. An exciting measure of unpredictability was due to the fact that this office environment was populated, not by humans alone, but by mobile robots and their programmers. Teams participating in the national robot contest were testing their robot navigation systems day and night in this environment, and it is in this crowded robot environment that InductoBeast was tested.

InductoBeast successfully constructed 5 maps of the arena in less than five hours running time, from various starting locations, over the course of two days. Figure 5 depicts one of the 5 maps.

We also tested these maps' accuracy by running long-term navigation tasks. We designed a random goal selector to demonstrate the robustness with which InductoBeast navigates; at AAAI, 58 plans were executed successfully out of 60 attempts. One of the two failures was due to a programming error that was corrected, and the second error was due to a Mac OS crash.

InductoBeast never collided with any walls, chairs or tables during any of these exploration and navigation tests. In fact, the only collisions that took place occurred when other robots hit InductoBeast while it was stationary.

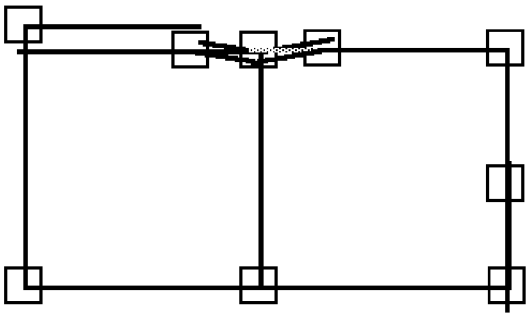

Figure 5: One of the maps generated in the AAAI '96 contest arena.

• Building and navigation statistic

Table 1 summarizes the results of experiments conducted with InductoBeast during normal business hours with human and robot traffic. During the navigation portion of testing, tasks were invented based on a randomly generated sequence of desired goal positions. These positions were given in turn to InductoBeast, which then had to complete the navigation tasks.

Location	Gates	AAAI
# intersections	12	9
# hallways	11	8
# open areas	2	3
total navigable length	387	257
total # navigation runs	50	60
total navigation hours	3.5	3
contacts / hr.	0	0
collisions / hr.	0	0
navigation success rate	96%	96.7%
navigation failure rate	4%	1.7%

Table 1: Empirical results for InductoBeast in two office environments.

Terms:

open area A path through a physical open area

navigable length The distance in feet between two nodes

contact Non-fatal touch between robot and environment

collision A robot-environment touch requiring restart

navigation success rate Percentage of random navigation runs in which the robot reached its destination

navigation failure rate Percentage of random navigation runs in which the robot got lost — this

plus the success rate may not sum to 100 if, for instance, batteries die, or the computer's operating system crashes.

6 RELATED WORK

Indoor automatic mapping research can be split broadly into two categories: geometric methods and feature-based, or topological methods. Geometric approaches to map-building have the end-goal of constructing a two-dimensional map of the environment, from which global geometric properties of the world may be deduced. Both continuous, solid geometry-based approaches and occupancy grid approaches (Moravec & Elfes 1985); (Elfes 1987) fall in this category.

In the geometric map-building approaches, the map is generally populated through the use of a series of discrete sensor measurements. In general, such sensor measurements are reduced to ranging values, both in the case of sonar-based map-building (Thrun et al. 1998); (Castellanos et al. 1997) and vision-based map-building (Murray & Jennings 1997). In the case of (Thrun et al. 1998), a statistical analysis of the sensor data can remove much of the sensor noise inherent in such discrete range samples. The general approach of using imperfect sensors combined with probabilistic models in a geometric framework has been extremely successful as a *navigation* architecture of late (Simmons & Koenig 1995), (Thrun, 1995).

An extremely important recent advancement is the use of *a priori*, generic knowledge regarding domain symmetry and collinearity to provide constraints on the map-building process, resulting in more accurate geometric representations of the environment (Thrun et al. 1998); (Castellanos et al. 1997). Several researchers have already demonstrated that this map post-processing step can dramatically improve the overall map quality.

There is, however, a fundamental disadvantage to geometric approaches that even this post-processing step cannot overcome. Since geometric methods rely on discrete ranging samples of the environment, any environment in which the underlying geometry is dynamic will proffer formidable challenges. Drastic changes in an office building topology are one such examples, but a much more subtle case would be a populated building, with peoples, boxes and chairs on the move.

Geometric automatic mapping systems are generally capable only of monotonically creating their maps of the world (Thrun et al. 1998); (Castellanos et al. 1997); (Delahoche et al. 1998); (Araujo et al. 1998); (Vlassis & Tsanakas 1998). Two notable exceptions are (Murray & Jennings 1997) and (Yamauchi et al. 1998).

(Murray & Jennings 1997) have developed a three-camera ranging system that serves as the sole ranging sensor of a robot that automatically maps an office environment. Their architecture has a reliance on a static world; however, the robot's mapping process is somewhat dynamic, as it actively plans paths to areas of the environment where its ranging information is insufficient for generating a complete map.

(Yamauchi et al. 1998) explicitly mixes, or interleaves, navigation with exploration at the planning level. Their mobile robot plans navigation tasks explicitly to travel to the edge of its known map. From such *frontier points*, mapping continues until that vein of exploration is exhausted. Thus, the map is dynamic. However, changes to the map are only allowed at locations beyond the frontier. Therefore, dynamics in areas of the world that have already been explored cannot be captured; the mapping process does not include any form of nonmonotonicity as does InductoBeast in its ability to produce and physically validate defeasible hypotheses.

In summary, geometric methods tend to suffer from their reliance on static environments, and an examination of the experimental setups of such research validates that tests are generally performed in controlled settings, frequently in small subregions of a much larger, navigable environment (Castellanos et al. 1997), (Murray & Jennings 1997); (Delahoche et al. 1998), (Cahut et al. 1998); (Castellanos 1998); (Araujo 1998).

In contrast to geometric map-building approaches, feature-based methods identify locally unique areas in the environment (features) and determine the relative positions of those features to one-another (Kuipers 1993), (Gat 199?), (Rencken 1994); (Castellanos et al. 1998); (Cahut et al. 1998).

Feature-based mapping is attractive because transient errors inherent in ranging sensors can have less impact on the resulting map. The philosophy behind this approach is that the robot does not require a detailed, geometric map of the environment to function effectively. Therefore, construction of a detailed map, along with the necessary error compensation computations, are a waste of computation. Instead, feature-based researchers

attempt to identify the minimum information required for robust navigation, then capture that information in a topological map that frequently includes a limited amount of geometric information as well (Gutierrez-Osuna & Luo, 1996).

The learning community has attempted to identify navigable features automatically; (Galles 1993) implemented a Nomad 100 robot control system that mapped part of Stanford's computer science department using TR-Tree representations to recognize "landmark" locations, such as intersections, then navigate between them. In this research, feature detection was not sufficiently reliable to enable robust navigation after completion of the training stage.

(Kuipers et al. 1993) also focused on identifying distinctive positions and noting control methods for navigating between them. In this case, however, the schema for unique locations, or features, were determined by the researchers and not by a classification system. But as with the geometric approaches, both Galles' and Kuipers' system are limited to static environments because their feature extraction strategies depend upon discrete ranging measurements of the environment. Indeed, feature extraction based on corner features and intersection features can be extremely susceptible to the transient "noise" introduced by human presence, since a flat wall may be mistaken for a corner if a person stands beside the wall.

Indeed, identification of features based on a ranging *snapshot* of the environment is the norm in this community; (Cahut et al. 1998) uses correlative techniques to cluster sonar strikes into line segments, constructing a navigation map based on the subsequent set of line segments. (Castellanos et al. 1998) use both a laser rangefinder and a CCD camera to identify and range vertical edges in the scene. In this work, line segments are only an intermediate representation in the effort to identify two high-level features: *corners* (formed by intersecting line segments) and *semiplanes* (derived from the free endpoints of line segments).

A more recent feature-based mapping system by (Chong & Kleeman 1997) improves upon earlier work in this area in two ways. (Chong & Kleeman 1997) make use of collinearity constraints that are viable in an indoor environment, thereby reducing their dependence on single sensor readings somewhat. They have also manufactured a custom feature sensor consisting of multiple sonar transducers and mounted on a pan axis, that detects corners with high accuracy.

However, the same fundamental limitations apply to this group of feature-based mapping research: these methods depend upon the static world assumption, and experimental evidence of success is limited to small-scale tests in artificial environments.

Given that the feature-based approach does not depend as critically on the detailed environmental geometry, it may come as a surprise that there is a lack of large-scale empirical results using this approach in real-world environments. The reason for this shortcoming has to do with the small set of features used (Kuipers et al. 1993), (Chong & Kleeman 1997). Real office buildings have geometries that are far more complex that the composition of simple corners, perfectly flat walls and intersections. As a result, the design of feature extractors that are robust with respect to these real-world vagaries (indentations; foyers; ledges, et cetera) has proved extremely challenging. This explains why much of the body of real-world mapping results has been achieved in artificial environments made from materials such as cardboard (Delahoche et. al 1998); (Cahut et al. 1998); (Kuipers et al. 1993).

InductoBeast is fundamentally a feature-based system. However, the basic goal of this project was to implement a mapping system that would succeed in a variety of *unmodified*, real-world, dynamic environments. Thus, in order to avoid the errors inherent in discrete sensors measurements, be they for geometric extrapolation or for feature extraction, we only use sensor measurements indirectly.

Sensor measurements have a direct impact on the motion characteristics of the InductoBeast robot. Feature extraction consists of nothing more than the introspective observation of motion patterns over time, primarily through the observation of encoder values. But this implies a significant reliance on encoders and is therefore a source of cumulative error. The solution, as with (Chong & Kleeman 1997) and (Thrun et. al 1998) is to make use of characteristics generally true of indoor environments. In our case, this meant specifically taking advantage of the fact that office buildings frequently have symmetry around an axis and can be tessellated into a partial grid pattern.

InductoBeast improves upon previous work in this direction by not stopping simply at the "map correction" post-processing phase. This system also make use of symmetry to hypothesize additional structure in parts of the world that are still unseen, *and then explore those*

structures. This is consonant with the approach taken in (Yamauchi et al. 1998), and stands in contrast to methods in which the mobile robot does not explicitly plan and act for the sake of information gain (Vlassis & Tsanakas 1998); (Araujo et al. 1998); (Delahoche et al. 1998); (Thrun et al. 1998).

This additional step is crucial in enabling InductoBeast to deal with dynamic worlds and with obstacles. Since the present system looks at the motion behavior of the robot over the long term, any transient sensor readings due to dynamic obstacles are completely ignored during the mapping process.

A final contribution of this present work is a set of experimental results that, we hope, raise the bar for the automatic mapping community. Real world tests have been performed in office buildings at Stanford University and also in the fabricated office environment of AAAI's National Robot Contest. In all cases, the environment was heavily occupied, under normal business hours, by untrained personnel. Indeed, in the case of AAAI, the environment was continuously occupied by robots and contestants of the robot contest. Under these conditions, both the explicit accuracy of repeated mapping iterations have been measured as well as the navigation reliability of the robot, using its own, self-generated map of the environment. Such long-term, real-world tests are crucial to ensuring advancement of the state of the art.

7 CONCLUSIONS

InductoBeast's strongest suit is that it works very well in a variety of real-world environments. Furthermore, the robot achieves reliability using a conceptually simple algorithm. This simplicity represents an important lesson we have learned through this work: always try the simplest approach first. We have a firm belief, based on the experiences of InductoBeast, that the indoor navigation problem can be fully solved using extremely simple methods.

Of course, malicious interference with the robot can easily confuse such simple techniques. For instance, physically picking up the robot and moving it to a new portion of its world will guaranteeably confuse InductoBeast, not least of all because it uses encoder values for localization. To deal with such catastrophic "failures," we believe one would need to revert to more complex methods for global localization. While probabilistic techniques for using range data show great promise in this respect (so long as the geometric map is sufficiently accurate), perhaps the most robust long-term solution would involve visual recognition of the robot's neighborhood.

In contrast to navigation under our simplifying assumptions, learning represents a much more challenging problem. InductoBeast's learning component depends heavily on a highly reliable and predictable low-level motion package. Because we know what environmental conditions will cause which behaviors to fire, we can use InductoBeast's behaviors as a set of mid-level control tools to inform the learning system about the type of environment the robot is exploring.

We have also found that the use of hypothetical hallways can make learning and navigation more effective. Mapping is simplified because not all hallways need to be explored. Navigation can be more efficient because the robot can use shortcuts that it has not even encountered yet. Currently, InductoBeast uses only relatively simple hypotheses. Future work will expand the manner in which InductoBeast reasons about the map and hypothesizes new hallways and intersections.

Finally, this project has made us aware of the significance of having navigation competence in open, unstructured areas. It is interesting to note that one of the major stumbling blocks of probabilistic navigation systems has also been the problem of dealing with open areas. In the both cases, open areas are challenging because ranging is a poor measurement technique in such spaces. Simply put, the proportion of useful data to noise drops precipitously.

When we first started this project, InductoBeast was designed only to map and navigate in hallways. The ability to traverse open areas by wall following was a later addition. This was a large source of difficulty and complication because the concepts and information needed for open area navigation did not fit smoothly into the map structure and general architecture designed for hall following. If InductoBeast were to be rewritten, we would consider open areas as the normal environmental condition and hallways as a special case.

ACKNOWLEDGMENTS

Professor Michael Genesereth (Stanford University) provided both intellectual motivation for this project to proceed as well as the Nomad 150 mobile robot and

Powerbook 170. Thanks also to the reviewers, whose thoughtful comments improved this article.

BIBLIOGRAPHY

Araujo, R., de Almeida, A.T. 1998. Map Building Using Fuzzy ART, and Learning to Navigate a Mobile Robot on an Unknown World. In *Proceedings of the 1998 IEEE International Conference on Robotics & Automation*. Leuven, Belgium. 1998.

Castellanos, J.A., Tardos, J.D., Schmidt, G. Building a global map of the environment of a mobile robot: The importance of correlations. *Proceedings of the 1997 IEEE Conference on Robotics and Automation*. 1997.

Chong, K.S., Kleeman, L. 1997. Sonar based map building for a mobile robot. *Proceedings of the 1997 IEEE Conference on Robotics and Automation*. 1997.

Delahoche, L., Pegard, C., Mouaddib, E.M., Vasseur, P. 1998. Incremental Map Building for Mobile Robot Navigation in an Indoor Environment. In *Proceedings of the 1998 IEEE International Conference on Robotics & Automation*. Leuven, Belgium. 1998.

Elfes, A. 1987. Sonar-based real world mapping and navigation. *IEEE Journal of robotics and automation*. Vol. RA-3, No. 3. 249-265, 1987.

Galles, D. 1993. Map Building and Following Using Teleo-Reactive Trees. Stanford Internal Report. Stanford University Robotics Laboratory.

Gutierrez-Osuna, R. and Luo, R. 1996. LOLA: Probabilistic Navigation for Topological Maps. *AI Magazine* 17(1).

Khatib, O. 1986. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *The International Journal of Robotics Research* 5(1):90-98.

Kuipers, B., Froom, R., Lee, W-K. and Pierce, D. 1993. The Semantic Hierarchy in Robot Learning. *Robot Learning*. Jonathan Connell & Sridhar Mahadevan (eds.), Kluwer Academic Publishers.

Murray, D., Jennings, C. 1997 Stereo vision based mapping and navigation for mobile robots. *Proceedings of the 1997 IEEE Conference on Robotics and Automation*. 1997.

Nilsson, Nils. 1996. Challenge Problems for Artificial Intelligence: Toward Flexible and Robust Robots, *Proceedings, Thirteenth National Conference on Artificial Intelligence*. pp. 1344-45, AAAI Press.

Nourbakhsh, I., Powers, R. and Birchfield, S. 1995. Dervish, An Office-Navigating Robot. *AI Magazine* 16(2).

Racz, J., Dubrawski, A. 1995. Artificial neural network for mobile robot topological localization. *Robotics and Autonomous Systems* 16(1995): 73-80.

Shanahan, M. 1995. Default Reasoning about Spatial Occupancy, *Artificial Intelligence*, vol. 74(1).

Simmons, R. and Koenig, S. 1995. Probabilistic robot navigation in partially observable environments. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, Montreal, Canada, Morgan Kaufmann.

Thrun, S. 1995. An approach to learning mobile robot navigation. *Robotics and Autonomous Systems* 15(1995): 301-19.

Thrun, S., Fox, D., Burgard, W. 1998. Probabilistic Mapping of an Environment by a Mobile Robot. In *Proceedings of the 1998 IEEE International Conference on Robotics & Automation*. Leuven, Belgium. 1998.

Vlassis, N.A., Tsanakas. 1998. A Sensory Uncertainty Field Model for Unknown and Non-stationary Mobile Robot Environments. In *Proceedings of the 1998 IEEE International Conference on Robotics & Automation*. Leuven, Belgium. 1998.

Yamauchi, B., Schultz, A., Adams, W. 1998. Mobile Robot Exploration and Map-Building with Continuous Localization. In *Proceedings of the 1998 IEEE International Conference on Robotics & Automation*. Leuven, Belgium. 1998.