Proceedings of the 1997 IEEE
International Conference on Robotics and Automation
Albuquerque, New Mexico - April 1997

Automatic Mapping of Dynamic Office Environments

Clayton Kunz
Computer Science Dept.
Stanford University
Stanford, CA 94305
clay@cs.stanford.edu

Thomas Willeke
Computer Science Dept.
Stanford University
Stanford, CA 94305
twilleke@cs.stanford.edu

Illah R. Nourbakhsh
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
illah@cs.cmu.edu

Abstract

We present a robot, InductoBeast, that greets a new office building by learning the floorplan automatically, with minimal human intervention and *a priori* knowledge. Our robot architecture is unique because it combines both abductive and inductive mapping methods to solve this problem. We present experimental results spanning three office environments. We hope the breadth of these results helps to establish a performance benchmark against which robust and adaptive navigator robots of the future may be measured.

1 INTRODUCTION

The goal of this work has been to produce a robot that greets a new office building by learning the floorplan automatically, with minimal human intervention and *a priori* knowledge. We present a robot architecture that solves this problem by combining automatic mapping, inductive learning and reliable, long-term navigation.

In addition to describing this architecture in detail, we present initial empirical results based on our ongoing effort to test InductoBeast in a variety of office buildings at Stanford University. The empirical tests demonstrate the ease with which InductoBeast can be introduced to a new environment, as well as long-term navigation reliability.

2 GOALS

The primary goal of this work is to develop a robot navigation system that is truly autonomous. In this section we describe corresponding subgoals in more detail.

2.1 Criteria for autonomous navigation

• Dynamic Worlds
An autonomous navigator must function reliably in a dynamic office environment: doors close and open, transient obstacles may block entire hallways, ordinary human traffic causes sensor noise. We require our robot to work in these ordinary conditions, so all of the results in Section 5 were recorded in unmodified, 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 block them off 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. When possible, however, even that navigation task will be transformed 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*).

• 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.

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.

• 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. Open areas such as foyers, however, may be of any shape, provided that they abut hallways that satisfy the rectilinearity constraint.

• Hallway Widths
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.

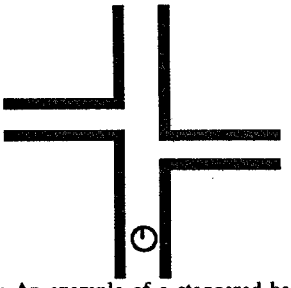

Figure 1: An example of a staggered hallway, a topology assumed out of existence by InductoBeast.

Together, the above set of assumptions and the goals in Section 2.1 define the problem InductoBeast is designed 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 graphically. 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.

In contrast to behavior-based architectures, the architecture we present has a single control thread, as described in Section 4. This control loop calls upon the modules in Figure 2 as necessary without the need for any form of arbitration.

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

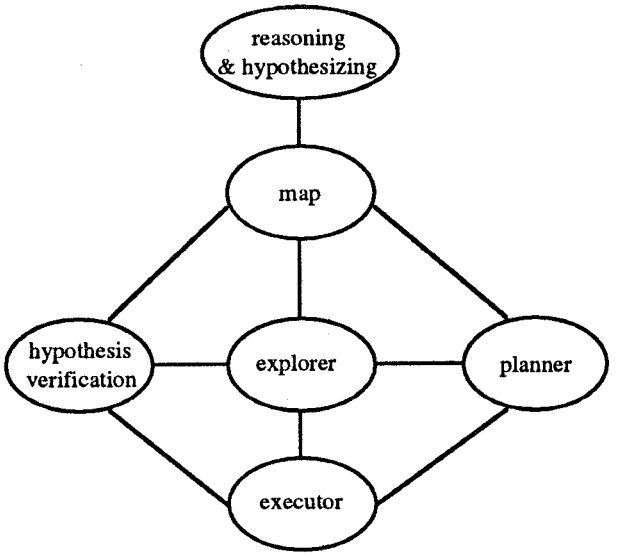

Figure 2: Abstract architecture of InductoBeast.

coincidental, the robot must actively pursue the goal of moving to the frontier of its map, then stepping beyond it. For this reason, the exploration module calls upon the planner and plan executor to take the robot to the edge of its map so that it can uncharted territory.

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

• Executor
Plan execution is made difficult because it is very closely linked with hypothesis verification. The robot will attempt to traverse hypothetical hallways, and the executor

must ensure that such behavior is graceful when the hypothetical hallway does not exist.

• 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.

• Reasoning and hypothesizing
This portion of the architecture is also relatively simple, since it does not involve any robot control. Reasoning (including cycle detection) is performed on the current state of the map, and hypotheses are made as appropriate when the robot believes unexplored hallways may exist due to office building symmetries. Because this module only examines the map, additional ways to create hypotheses can be added without a great deal of code modification.

• 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, which 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.

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 from the robot's sensors. This reliance upon behavior rather than sensor values also 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.

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 is able to recover from hallways that are completely blocked off during navigation, even if its (incomplete) map indicates that no routes exist.

3.3 Learning and planning

Reasoning and learning appear in several guises throughout this work. InductoBeast performs induction by proposing hypothetical hallways based on office building symmetry and the colinearity 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 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 lead to the expected destination.

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. 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 sufficiently changed

- an opening long enough to travel through is detected by examining a series of side sonar 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 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. InductoBeast's path may include a hypothetical connection, however, so that during execution, the verification component must monitor navigation and replan 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 avoids obstacles as it travels. Our system for obstacle avoidance is case-based, directing the robot toward perceived free space. This method approximates a potential field without explictly computing one (Khatib 1986; Nourbakhsh et al. 1995). At the high level we need only to execute a *move* function which controls all of the robot's forward motion and obstacle avoidance. When it is in hall-follow mode, InductoBeast also attempts to remain centered in the hallway. 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.

Our localization system is unusual in that it ignores sonar data almost 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 inches after the robot has traveled the length of a hallway. Our strategy of localizing after each leg eliminates potential 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.

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 in 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 significant of which was the ability to navigate through open areas by wall following. During this phase of development, which spanned approximately 160 hours, InductoBeast collided with walls fewer than five

times. During subsequent experimental runs, there were zero collisions (see table 1).

Figure 3 is a partial map of Gates generated by InductoBeast.

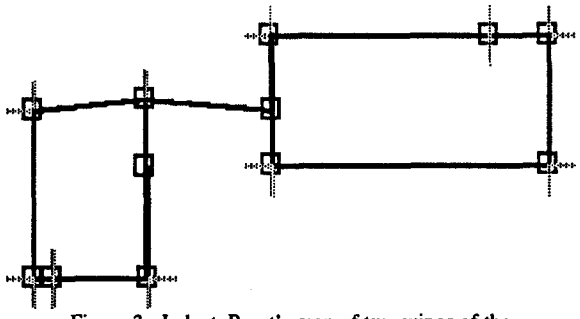

Figure 3: InductoBeast's map of two wings of the second floor of the Gates Computer Science building at Stanford.

• Foreign environnment 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.

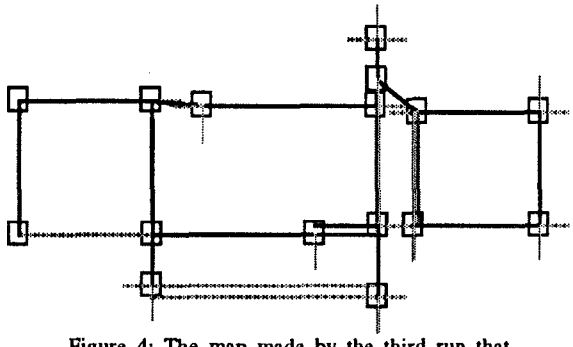

Figure 4: The map made by the third run that InductoBeast made in the 2nd floor of the Psychology building at Stanford, after the hallway width parameter was adjusted.

• Artificial benchmark environment at AAAI
The AAAI '96 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. 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.

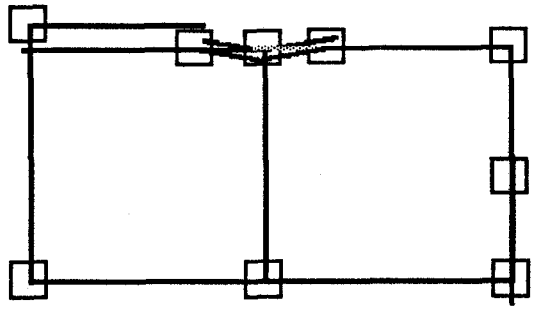

Figure 5: One of the maps generated in the AAAI '96 contest arena.

• Building and navigation statistics[1]
Table 1 summarizes the results of experiments conducted with InductoBeast during normal business hours with human traffic.

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 adjacent nodes
contact Non-fatal touch between robot and environment

[1]Real world experiments are ongoing; we plan to conduct mapping and navigation tasks in up to 10 buildings.

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

Mobile roboticists have been focusing primarily on one subproblem of the general mobile robotics problem: robot navigation. This specialization has resulted in a great deal of experimentation in controlled and limited environments. Recently, probabilistic approaches such as probabilistic state set progression and partially observable Markov decision processes have shown good potential as reliable navigation systems (Simmons & Koenig 1995; Gutierrez-Osuna & Luo 1996; Nourbakhsh et al. 1995). The probabilistic approach has been motivated by a combination of incomplete information and perceived encoder inaccuracies. However, the research has limitations in its ability to offer robust navigation in the face of moving obstacles and, more to the point, dynamic environments.

Navigating in dynamic environments necessitates building systems that can alter their representation of the world. Automatic mapping is one starting point for research on this subject. However, surprisingly little work has been done in this area lately. (Galles 1993) implemented a Nomad 100 robot control system that mapped part of Stanford's computer science department using a neural network to recognize "landmark" locations, such as intersections, and TR-trees to control navigation between them. (Kuipers et al. 1993) also focused on identifying distinctive positions and noting control methods for navigating between them. Both of these works are limited by assuming a static map topology and a lack of moving obstacles. Furthermore, real-world empirical results are difficult to identify because much of the experimentation has involved robotic simulations. An exception to this trend is (Dean et. al., 1990), who used influence diagrams both for feature recognition and for control (including map building) on Huey, a working robot.

Another important differentiating characteristic of automatic mapping research is that other robot mapping strategies have been essentially abductive (Shanahan 1995) and not inductive. Inductive learning for robotics has been the focus of several projects, which have applied machine learning methods to pieces of the mobile robot problem. (Racz & Dubrawski 1995) applies neural network methods to the problem of robot localization, transforming localization into a static matching problem between expected sensor values and measured sensor values (this research assumes a static environment and a static robot). (Thrun 1995) applies explanation-based neural network methodology to the problem of learning a control program to achieve a particular goal position in a static environment.

The InductoBeast project has diverged from these researchers' goals in that we intend to combine inductive learning with a representation that enables robust navigation, over the long term, in various dynamic environments. It is this application of learning to the problem of "guessing" the topology of an office building, and its subsequent execution on real-world robots, that distinguishes this work. We believe our approach is directly applicable to the Robot Challenge Problem proposed by Nils Nilsson (Nilsson 1996), since the resulting system can be trained to navigate a new building with a minimum of human effort.

7 CONCLUSIONS

In a largely theory dominated robotics community, we feel it is important to encourage real robot experimentation. Simulation, while useful in many circumstances, can lead to unfortunate setbacks if the goal of eventual implementation is not kept firmly in sight. We hope that that work like InductoBeast will encourage researchers to contemplate the tasks they could make real robots do today. Our technology and understanding of robotics is close to the point where useful and interesting applications of autonomous robots are possible, if concentrated effort is put into implementation.

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.

In contrast, 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. Given the success of InductoBeast at mapping and navigation, we hope that it will be used as a benchmark against which future office robots will be compared.

BIBLIOGRAPHY

Dean, T. et. al. 1990. Coping with Uncertainty in a Control System for Navigation and Exploration. *Proceedings, Eighth National Conference on Artificial Intelligence*. pp. 1010-1015, AAAI Press.

Galles, D. 1993. Map Building and Following Using Teleo-Reactive Trees. *Proceedings, Third International Conference on Intelligent Autonomous Systems*, Washington DC., IOS Press.

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.

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.