The Office Marathon:
Robust Navigation in an Indoor Office Environment

Eitan Marder-Eppstein, Eric Berger, Tully Foote, Brian Gerkey, Kurt Konolige

Willow Garage Inc., USA
{eitan,berger,tfoote,gerkey,konolige}@willowgarage.com

***Abstract*— This paper describes a navigation system that allowed a robot to complete 26.2 miles of autonomous navigation in a real office environment. We present the methods required to achieve this level of robustness, including an efficient Voxel-based 3D mapping algorithm that explicitly models unknown space. We also provide an open-source implementation of the algorithms used, as well as simulated environments in which our results can be verified.**

I. Introduction

We study the problem of robust navigation for indoor mobile robots. Within this well-studied domain, our area of interest is robots that inhabit unmodified office-like environments that are designed for and shared with people. We want our robots to avoid all obstacles that they might encounter, yet still drive through the tightest spaces that they can physically fit. We believe that reliable navigation of this kind is a necessary prerequisite for any useful task that an indoor robot might perform.

While many robots have been shown to navigate in office-like environments, existing approaches invariably require some modification of the environment, or do not allow the robot to negotiate tight spaces. Most indoor robots rely on a planar or quasi-planar obstacle sensor, such as a laser range-finder or sonar array. Because vertical structure dominates man-made environments, these sensors are positioned on the robot to detect obstacles along a horizontal slice of the world. The result is a robot that can easily avoid walls but will miss chair legs, door handles, table tops, feet, etc. Collisions are avoided by either modifying the environment (e.g., removing chairs, covering tables with floor-length tablecloths), or adding artificial padding (e.g., inflate obstacles by the maximum expected length of a person's foot), which prevents the robot from traversing tight spaces.

Three-dimensional obstacle sensing is increasingly common, usually in the form of an actuated laser range-finder (e.g., a horizontally-mounted laser that tilts up and down) or a stereo camera pair. Some sensors combine laser and camera technology, in a pulsed or line-stripe fashion. The availability of such devices provides an opportunity for an indoor robot to sense and avoid nearly all hazards that it might encounter.

The challenge lies in interpreting, storing, and using the data provided by the three-dimensional sensors. In this paper, we present a robot navigation system that exploits three-dimensional obstacle data to avoid the smallest obstacles that can be perceived with the available sensors, yet drives through the tightest spaces that the robot can fit. At the core of our work is an efficient technique for constructing, updating, and accessing a high-precision three-dimensional Voxel Grid. This structure encodes the robot's knowledge about its environment, classifying space as free, occupied, or unknown. Using this grid, the robot is able to plan and execute safe motions in close proximity to obstacles.

Fig. 1. The PR2 avoiding a table after passing through a narrow doorway.

Through extensive experimentation, we have established that our approach is safe and robust. During endurance runs with the PR2, we regularly left a robot running unattended overnight in our office building, which was not modified to accommodate the robot. The software described in this paper is available under an open-source license,[1] and we encourage others to experiment with and use our code.

II. Related Work

A wide variety of robots have been demonstrated to navigate successfully in human environments. A common theme is robotic museum tour guides. RHINO [1] was the first robotic tour guide and was followed soon after by MINERVA [2], which led tours through the Smithsonian Museum in 1998. Other robots have followed, including Mobots [3], Robox [4], and Jinny [5]. These robots have dealt with environments that are often crowded with people

[1] Instructions for running the experiments presented in this paper, as well as all the source code for the PR2 navigation system can be found here: http://www.ros.org/wiki/Papers/ICRA2010_Marder-Eppstein

with varying degrees of success. Most cite their primary difficulty as robust localization, and have focused on creating localization systems that are reliable in these highly dynamic environments. In attempting to apply the navigation techniques used by these platforms to the PR2, however, another pressing issue emerged that was not addressed: the ability to reason about three-dimensional space, both occupied and unknown, in a principled manner.

Previous techniques for handling three-dimensional obstacles range from restricting the path of the robot to corridors known to be clear in the nominal case [3], to building a three-dimensional octree of the environment and computing bounding boxes around obstacles [6]. The first approach fails whenever a difficult obstacle enters a navigation corridor, and the second approach requires the robot to stop for a prolonged period of time, take a full scan of the environment, create a plan, and then fall back on two-dimensional sensing if a dynamic or occluded obstacle is encountered while executing the plan. In cluttered environments such as offices, both approaches would be highly susceptible to collisions, and have poor performance.

In the outdoor navigation domain, there has also been work on three-dimensional perception. Stanford's autonomous car Junior [7] utilized a three-dimensional free space analysis from its Velodyne laser range finder. This free space analysis was performed radially in the ground plane and allowed dynamic obstacles recorded in the map to be cleared quickly as they moved to a new location. This approach is limited, however, by the fact that only the two-dimensional projections of obstacles were stored. For a sensor like the Velodyne, this works because the laser returns 360-degree scans of the world at a high rate. However, for a three-dimensional sensor with any latency, storing only the two-dimensional projections of obstacles is insufficient. Also, storing obstacle information in two dimensions limits the types of occlusions that can be tracked in an environment. For office environments that contain obstacles that occlude others, this proves problematic.

III. Problem Description

Given any physically reachable goal in an office-like environment,[2] we require a mobile robot to autonomously navigate to that goal, avoiding collision with obstacles on the way. The robot is assumed to be wheeled, and cannot step over obstacles, but only drive around them. The environments that we study are unmodified, and are filled with a variety of obstacles, including tables, chairs, people, and scooters (Figure 1), all of which can change location throughout the day. The key challenge in this problem is detecting and avoiding obstacles with non-trivial three-dimensional structure, while not giving up the ability to fit through narrow passageways.

[2]We scope our effort to institutional environments that conform to modern U.S. building codes, chief among them the Americans with Disabilities Act (ADA), which establishes a variety of constraints, including minimum width on doorways.

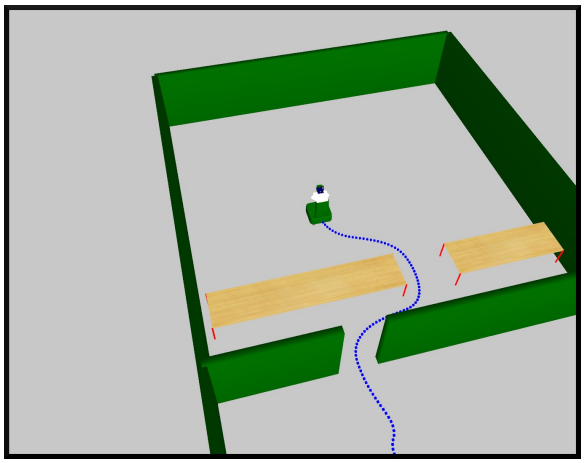

Fig. 2. A narrow doorway with tables.

As a concrete example, consider the case where a robot is tasked to move through the narrow gap between two tables of different heights. The robot must detect the edges of the tables using a sensor capable of generating three-dimensional information, create a plan through the two tables that avoids the edges, and then execute that plan. Next, consider that a person walks in front of the robot while it executes its plan, blocking the robot in, and then moves out of the way. The robot now needs a way to tell the difference between the person that it no longer sees because they have walked away, and the tables it no longer sees because the three-dimensional sensor on the robot takes a long time to make a full sweep of the environment. In our experience, using fixed timeouts for the persistence of observed obstacles resulted in either unreasonably slow and conservative behavior near dynamic obstacles, or dangerous behavior near small obstacles. For example, if the robot is too aggressive in clearing out obstacles, it may mistakenly remove the tables from its map, potentially causing a collision. This is just one of many troublesome situations that a robot will eventually encounter in environments that are not modified for its benefit.

We consider proof of robustness in these difficult scenarios contingent upon completion of long-running autonomous navigation in an office environment, without human intervention. We believe the key to successful navigation in such environments is the robot's ability to reason about its environment in three dimensions, to handle unknown space in an effective manner, and to detect and navigate in cluttered environments with obstacles of varying shapes and sizes. To prove competence in these areas, we require the robot pass a number of tests both in simulation, and a real-world office environment. Summaries of these tests are presented in the sections below.

A. Narrow Doorways and Tables

The first navigation test challenges a robot to move through narrow spaces while also avoiding three-dimensional obstacles. The robot first passes through a narrow doorway, after which it must navigate between two tables placed

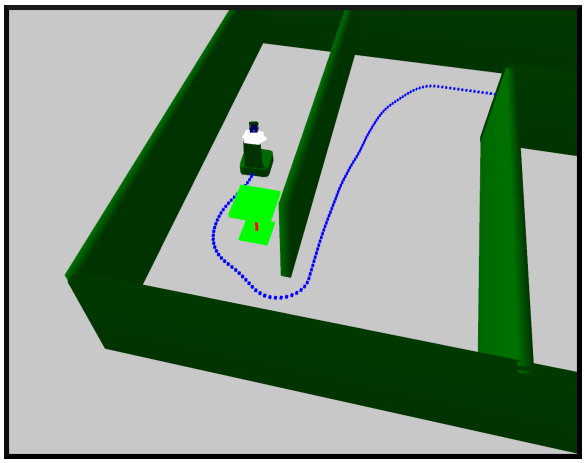

Fig. 3. The robot rounding a blind corner.

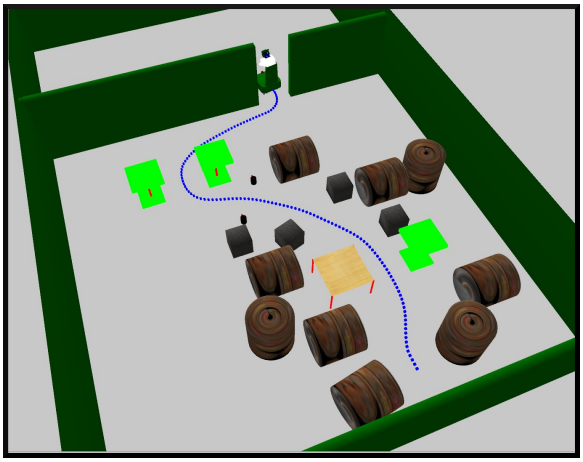

Fig. 4. A cluttered simulation environment.

directly on the other side, as shown in Figure 2. This environment is impossible to navigate successfully using the traditional approach of two-dimensional sensing with large padding values on obstacles. For the robot to pass through the doorway, the padding must be small enough that the robot would also pass between the legs of the tables, causing a collision. If, however, the padding is set large enough to avoid the tables, the robot will be unable to pass between the narrow doorway or the narrow space between the tables. In this way, the test guarantees a robot reasons about its environment in three dimensions.

B. Rounding a Blind Corner

The second test forces a robot to round a blind corner that occludes a table, as shown in Figure 3. To complete this task successfully, the robot must reason about unknown space in a principled manner. Robots that treat the occluded space as free will risk hitting the table.

C. Obstacle Slalom Course

The third test, shown in Figure 4, challenges a robot with obstacles of varying size, shape, and height. This environment stresses the robot's ability to detect different kinds of three-dimensional obstacles in cluttered, tight spaces, as well as its ability to move through a difficult obstacle field.

IV. Platform

Experiments in this paper were performed with an Alpha prototype of the PR2 mobile manipulation platform (Figure 1). The PR2's 8-wheeled omni-directional base allows it to drive in wheelchair-accessible environments. The top speed of the PR2 base is approximately 1 m/s. We do not use the PR2's arms in this study.

A Hokuyo UTM-30LX laser scanner is mounted on the mobile base, providing a 270-degree horizontal view of the world at ankle-height. A second Hokuyo laser scanner is mounted on a tilting platform at shoulder-height, providing a 3D view of the area in front of the robot. The PR2's head is a pan-tilt platform equipped with a high resolution 5 megapixel (Mp) camera and two stereo camera pairs, with different fields of view and focal lengths.

In the present work, we use the base laser and the tilting laser to detect obstacles and navigate, but our approach is compatible with any range sensor that produces a point cloud.

A system as complex as the PR2 is driven by a number of subsystems that must be able to easily communicate with one another: hardware drivers, controllers, perception, planning, higher level control, etc. The computational needs of these components virtually necessitate the use of a distributed computing environment. To accommodate these needs, we use ROS,[3] which provides access to hardware and hides the complexities of transferring data between components.[8]

V. Approach

Office environments offer many interesting challenges for a mobile robot. There are chairs, tables, and small obstacles to avoid, people that get in the way, and obstacles that occlude others. In this section, we discuss the key aspects of our approach that handle many of these challenging problems.

A. Sensor Processing Pipeline

Sensor data is often imperfect. Obstacles may be reported where none exist, or be missed when they should be detected. Because of this, the sensor processing pipeline proved to be an extremely important part of the PR2 navigation system. The pipeline's job is to take in raw sensor data and apply a number of filters to the data before it is converted into obstacle information used for planning.

Our approach infers obstacles strictly from geometric information, relying on range sensors that can produce point clouds. On the PR2, we use two Hokuyo lasers, one mounted on the base of the robot and one mounted on an actuated platform just under the robot's head. These lasers have an observed depth accuracy of only 1.5cm, and are also susceptible to veiling effects off edges hit at high incidence angles or reflective surfaces where the laser reports false

[3]ROS (Robot Operating System - `http://www.ros.org`) is an open-source project to build a meta-operating system for mobile manipulation systems.

hits in free space. This lack of accuracy in readings makes it difficult to differentiate between small objects and the ground, and the veiling effects result in false positives when entering narrow passages such as doorways or when the scan passes near the robot's own body.

To detect small obstacles on the ground, each ground laser scan is passed through a Random Sample Consensus (RANSAC) [9] algorithm that fits a line model to the scan and reports the outliers as obstacles. We can identify and avoid ground obstacles as short as 6cm. To remove the false positives generated by veiling effects, a filter is applied that looks at the angle between consecutive points on a scan and the viewpoint, and throws the reading out as invalid if it is outside of a tolerance. This filter allows traversal of narrow passageways that would have otherwise been blocked off by false hits.

In addition to filtering out sensor readings that are invalid due to laser hardware limitations, it is important to filter out readings that intersect the body of the robot. For example, if the actuated tilting laser sees the base or arm of the robot, the navigation system should not consider those readings as obstacles. Therefore, an additional filter, which checks each laser reading for intersection with the robot body, is applied to all sensor data to remove any intersecting readings from the obstacle information used by the navigation system for planning.

B. Voxel Grid

The Voxel Grid, as shown in Figure 5, is an efficient three-dimensional occupancy grid that serves as the core of our approach to navigation.

Each cell in the grid has one of three states: occupied, free, or unknown; we do not use a probabilistic scheme because of performance considerations. Occupancy information is used to create safe plans for the navigation system even in cluttered environments. There are two main operations that can be performed on the grid: marking and clearing. Marking refers to accessing an element of the grid and changing its status based on an observation from a sensor, and clearing refers to raytracing in the grid from the origin of a sensor to an observation while asserting free space along the way. To run at a reasonable rate, it is important that both marking and clearing operations be extremely efficient. Using a 3GHz Core 2 Duo Processor, our Voxel Grid performs marking and clearing operations on 1,647,058 rays a second out to a distance of 3.0m in a grid with 2.5cm resolution per cell. This means that the Voxel Grid can raytrace through 197,647,059 cells in one second, easily keeping up with the lasers mounted on the robot which require only 17 percent of that computation.

Each grid cell needs only two bits of memory to store its state. The Voxel Grid encodes state efficiently using a two-dimensional array of 32-bit integers, where each integer represents a vertical column of the 3D grid, and each two bits of the integer encodes the state of a cell. Initially, all the cells in the Voxel Grid are marked as unknown because we have no sensor readings about them. Marking a cell as occupied involves a two-dimensional index into the array, and then a single bitwise-or operation with a mask corresponding to the height of the cell. Clearing in the grid uses a modified version of Bresenham's two-dimensional raytracing algorithm [10]. Steps in the X and Y dimensions correspond to changes in the index, but steps in the Z dimension correspond to bit-shift operations applied to an integer mask. Then, for each cell visited in a given raytracing iteration, a bitwise-and operation is applied to clear the cell.

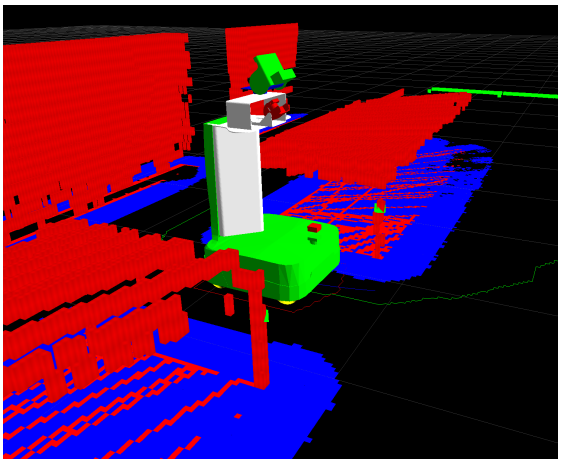

Fig. 5. The robot's Voxel Grid as it navigates between two tables.

This implementation of the Voxel Grid is remarkably efficient, but it does limit the number of grid cells in the Z dimension to 16 to run at the rate reported in this paper. In the case of the PR2, obstacles up to 1.3 meters tall are placed in the grid, which results in a maximum Z-resolution of 8.1cm per cell. On a 64-bit machine, however, this could be further reduced to 4.1cm. The 8.1cm resolution proved sufficient for the PR2 in its office environment, but for a robot that has tight vertical clearances, a finer-grained resolution might be required.

For the horizontal dimensions of the Voxel Grid we use a resolution of 2.5cm for each cell. Such a high resolution is necessary because the PR2 has only 5cm of clearance on each side when traversing a typical ADA-compliant doorway. A coarser grid can cause doorways to appear impassable based on discretization errors that occur upon projection of points into the grid, especially when doorways are not aligned with the grid. The use of such a high horizontal resolution increases the importance of efficiency in the Voxel Grid.

C. Unknown Space

One of the main advantages of the Voxel Grid structure presented above is its ability to track unknown space in the world. There have been many different approaches to handling occlusions in navigation. Some robots limit their speed based on the speed of dynamic obstacles in the environment and two-dimensional sight-lines computed between the robot and occluding obstacles [11]. However, this analytical approach does not scale well to three-dimensional

obstacles. Others limit speeds when making sharp turns, or ignore the danger of rounding a blind corner completely.

In our approach, speed limits arise naturally from treating unknown space as illegal to traverse. Consider the case where the robot must round a corner and a wall occludes the laser from receiving useful obstacle information from around the bend. In this case, the space occluded by the wall will appear as unknown in the robot's three-dimensional occupancy grid. The robot has two qualitative options as it approaches the corner: it can slow down until it is able to see around the bend, or it can take the corner wide at a higher speed, increasing visibility into that space. Both options are reasonable and result in safe navigation behavior for the robot, without setting arbitrary thresholds.

In an ideal world, every unknown grid cell would be treated as an obstacle as described above. This would allow hard guarantees to be made about the safety of an autonomous robot because it would never traverse cells that it had not explicitly seen. In the case of the PR2, however, the tilting laser cannot carve out space quickly enough to allow the robot to move at a reasonable speed with this strategy. Instead, we use a non-zero tolerance for the number of unknown cells allowed in a Voxel Grid column that is considered to be free. Using a low tolerance results in a robot that is safer but moves more slowly, while using a higher tolerance results in a robot that can move more freely but with added risk. In the experiments presented here, the PR2 runs with a tolerance of 4 unknown cells per each 16-cell column; this allows the base to move at reasonable speeds while still causing the robot to slow down for major occlusions.

VI. Implementation

The PR2 navigation system is simple at a high level. It takes in data from sensors, odometry, and a navigation goal, and outputs velocity commands that are sent to a mobile base. The low-level architecture of this system, however, is complex and consists of many components that must work together. The major components of the navigation system and the relationships between them are presented below.

A. Mapping and Localization

The PR2 navigation system can be initialized with or without an a priori, static map. When initialized without a map, the robot only knows about obstacles that it has seen, and will make optimistic global plans through areas that it has not yet visited which may traverse unknown space, potentially intersecting unseen obstacles. As the robot receives more information about the world, it replans accordingly to avoid obstacles. Initialized with a static map, the robot will make informed plans about distant parts of the environment, using the map as prior obstacle information. To navigate efficiently over long distances in an office environment, and to allow humans to task the robot to move to a particular location in the world, having a map can be a significant benefit.

In either case, we require that the robot's pose be tracked in a consistent global coordinate frame. When not using a map, the robot's pose is usually estimated by integrating wheel odometry, possibly fused with data from an inertial measurement unit (IMU). When using a map, the robot is usually localized using a probabilistic technique. We employ both approaches in our experiments, using an extended Kalman filter to fuse wheel odometry and IMU data in the no-map case, and adaptive Monte Carlo localization [12] in the map case.

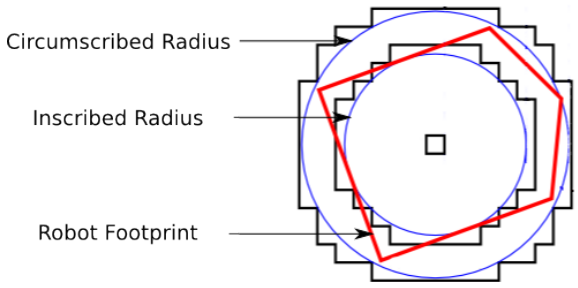

Fig. 6. An illustration of the inscribed and circumscribed radii as they relate to the robot's footprint.

B. Costmap

Because the robots we study are constrained to drive on flat ground, and cannot, for example, step or jump over obstructions, we assemble obstacle data into a planar Costmap on which the planners operate. The Costmap is initialized with the static map (if available), but updates as new sensor data comes in to maintain an up-to-date view of the robot's local and global environment. Although the Costmap is used as a two-dimensional structure by the navigation system, its underlying representation of the world actually consists of the efficient three-dimensional Voxel Grid described above. Each column in the Voxel Grid is projected down into two dimensions where it is assigned a cost. Columns with occupied cells are assigned a lethal cost, meaning that no part of the robot's footprint is allowed to be inside of the corresponding two-dimensional cell. Then, inflation is performed in two dimensions to propagate costs from obstacles out to a user-specified inflation radius. Cells that are less than one inscribed radius of the robot away from an obstacle (see Figure 6) are assigned a uniformly high cost, after which an exponential decay function is applied that will cause the cost to drop off out to the inflation radius used for the Costmap.

C. Global Planner

The global planner is given the obstacle and cost information contained in the Costmap, information from the robot's localization system, and a goal in the world. From these, it creates a high-level plan for the robot to follow to reach the goal location. It is important that this planning process be efficient, so that the navigation system can run at a reasonable rate. Therefore, the global planner used for this navigation system assumes that the robot is circular, uses an A* algorithm that plans directly in the configuration space computed during obstacle inflation in the Costmap, and does

not take into account the dynamics or the kinematics of the robot [13]. While this ensures that the global planner returns quickly, it also means that the planner is optimistic in the plans that it creates. For example, the global planner may produce a path for the robot that is infeasible, such as a plan that turns through a narrow doorway too tightly, causing the corners of the robot to hit the door frame. Because of its shortcomings, the global planner is used only as a high-level guide for navigation in an environment.

D. Local Planner

The local planner is responsible for generating velocity commands for the mobile base that will safely move the robot towards a goal. The local planner is seeded with the plan produced by the global planner, and attempts to follow it as closely as possible while taking into account the kinematics and dynamics of the robot as well as the obstacle information stored in the Costmap. In order to generate safe velocity commands, the local planner makes use of a technique known as the Dynamic Window Approach (DWA) to forward simulate and select among potential commands based on a cost function [14]. The cost function combines distance to obstacles, distance to the path produced by the global planner, and the speed at which the robot travels. The behavior of the robot can be changed drastically by setting the weights on each component of the cost function differently. For example, a robot that is programmed to stay as far away from obstacles as possible might have the weighting factor for distance to obstacles set quite high. This would result in the robot slowing down significantly around obstacles, and perhaps taking a longer route to avoid coming within a certain distance of obstacles. Because the robot must pass through narrow spaces such as doorways, the PR2's cost function is tuned aggressively, guaranteeing only 3cm of clearance between the base and obstacles.

E. Local and Global Coordinate Frames

As discussed by Moore [15], it is important to distinguish between global and local coordinate frames when implementing a navigation system. A global coordinate frame, like the one provided by the localization component described above, is advantageous in that it provides a globally consistent frame of reference, but is flawed in that it is subject to discontinuous jumps in its estimation of the robot's position. For example, when the localization system is struggling to determine the robot's position, it is not uncommon for the robot to teleport, on a single update step, from one location to another that is one meter away. A local coordinate frame, such as the one provided by odometry, has no such jumps, but presents its own flaws in that it is prone to drifting over time, which localization corrects.

Often, all planning and obstacle avoidance is performed in the global frame, but this leads to problems when discrete jumps in localization occur. To illustrate this, consider the case in which a robot is tasked to navigate through a narrow doorway with limited clearance on either side. If the robot attempts to plan through the doorway in a global frame, a localization jump drastically affects the robot's obstacle information. A jump in the robot's position of just a few centimeters to either side, combined with new sensor data, may actually be enough for the robot to see the narrow doorway as impassible. If the robot instead operates in a local frame, nearby obstacles are not affected by jumps in localization, and the robot can traverse through the doorway independent of any difficulties with the localization system.

Thus, we use the global coordinate frame to create high-level plans for the robot, but use a local coordinate frame for local planning and obstacle avoidance. To relate the two frames, the plan produced by the global planner is transformed from the global coordinate frame into the local coordinate frame on every cycle. This seeds the local planner with a path that is not affected by odometry drift, allowing the robot to traverse narrow passageways and crowded environments undeterred by jumps caused by uncertainty in localization.

F. Tilting Laser

The tilting laser used as the three-dimensional sensor for the PR2 navigation system can be actuated at different speeds. A slow tilt profile will give dense obstacle information, but the time between updates for a specific region of the world will be large, limiting the safe speed of the robot. Alternatively, a quick tilt profile will update the full world-model rapidly, but that model will have poor density. By virtue of the modelling of unknown space in the algorithm, neither of these options will create a safety hazard, but both will cause the robot to move slowly.

When the laser is tilted, the worst-case spacing between rays occurs at maximum range. We want a maximum of 3cm of vertical travel between points. The Hokuyo UTM-30LX takes 25 ms to scan a line; at a 2-meter obstacle detection range, this gives a maximum tilting rate of 0.6 rad/sec and, for a 1.9 radian tilt range, gives us a minimum scan time of 3.2 seconds. Although the robot is moving while it scans, we can clear out roughly 2 meters in front of us every 3 seconds, which allows a maximum speed of 0.6m/s. In actual operation, we chose to use a more conservative maximum velocity of 0.45 m/s.

G. Policy

The navigation system as described works well most of the time, but there are still situations where the robot can get stuck. To make the system as robust as possible, a number of recovery behaviors were built into the navigation system. One common cause of failure for the navigation system is entrapment, where the robot is surrounded by obstacles and cannot find a valid plan to its goal. When the robot finds itself in this situation, a number of increasingly aggressive recovery behaviors are executed to attempt to clear out space. First, the robot clears all obstacles in its map that are further than 3m away by reverting to the initial map for all obstacles outside of a 6m$\times$6m local window. If this fails, the robot performs an in-place rotation to attempt to clear out space. If this too fails, a more aggressive map reset is attempted

Simulation Results				
Environment	Trials	Avg. Speed	Distance	Collisions
Doorway and Tables	10	0.39m/s	7m	None
Blind Corner	10	0.35m/s	12m	None
Obstacle Slalom	10	0.41m/s	13m	None
Real World Results				
Environment	Trials	Avg. Speed	Distance	Collisions
Doorway and Tables	5	0.40m/s	6m	None
Blind Corner	5	0.37m/s	7m	None
Obstacle Slalom	5	0.38m/s	12m	1 Grazing

TABLE I

RESULTS FOR THE EXPERIMENT ENVIRONMENTS BOTH IN SIMULATION AND THE REAL WORLD.

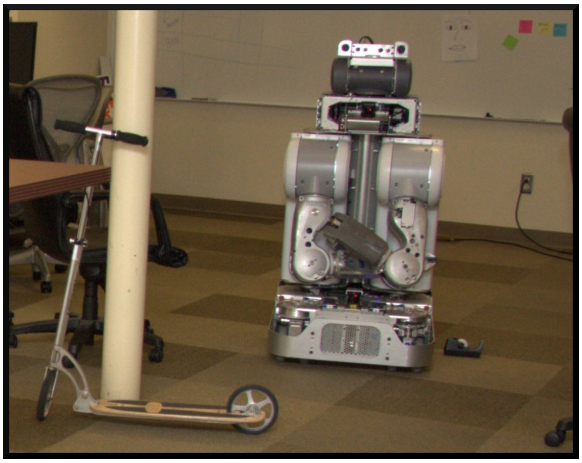

Fig. 7. The robot avoiding a tape dispenser and a scooter in an obstacle slalom course.

where the robot clears all obstacles that are outside of its circumscribed radius. After this, another in-place rotation is performed to clear space, and if this last step fails, the robot will abort on its goal which it now considers infeasible.

VII. EXPERIMENTAL RESULTS

The PR2 navigation system described above was created with the ultimate goal of completing long-running autonomous navigation tasks in an office environment, without human intervention. To achieve this goal, we required that the navigation system operate without incident in our office environment over a 26.2 mile run. As described in Section III, the robot also had to complete a number of pre-defined tasks to prove additional robustness. Experiments on these tasks were run both in the Gazebo Robot Simulator [16] and our local office environment. In each experiment, the robot ran with its maximum velocity set at 0.45m/s and its threshold for unknown space set at 25 percent. This means that the robot was required to see 75 percent of the cells in each column in the Voxel Grid before considering the column traversable. The local planner was configured to forward-simulate 60 different trajectories for 1.7 seconds each.

A. Task Results

The robot completed the navigation tasks described above both in simulation and the real world, avoiding obstacles as small as 6cm in height and driving through passageways with as little as 5cm of clearance on each side. (Figure 7) The robot hit no obstacles in simulation, but did scrape the rubber piece on the bottom of a tripod in one of five runs of the real world obstacles slalom course. This is not entirely unexpected, however, as the piece on the tripod lay below the 6 cm threshold for the minimum obstacle the robot's sensors can detect. A more intelligent perception pipeline might reason that the slanted legs of the tripod must extend to the floor, and avoid the obstacle in that manner, but the PR2 navigation system lacks this kind of object classification. The results from each of the experiments, in both the real world and simulation, are shown in Table I.

B. Marathon Results

The PR2 navigation system successfully completed 26.2 miles of autonomous navigation in an office environment, without human intervention. For the marathon, the navigation system was given a 2.5cm resolution map of the office generated by running a Simultaneous Localization and Mapping system off-line on planar laser data of the building. This means the map included information about the building's structure, but not about three-dimensional obstacles. During the marathon, the robot often encountered people, passed through narrow spaces such as offices, and avoided three-dimensional obstacles such as shelves, tables, and chairs. Due to the underlying Voxel Grid representation, the robot was able to keep a truly three-dimensional view of the world, allowing it to be principled about clearing out obstacles from its map. The robot took approximately 30 hours to complete the marathon, and ran at an average speed of 0.4 meters per second. When the robot ran low on battery power, it returned to a designated charging station and emailed a plug-in request. Then, when charging finished, the robot would email again, requesting to be unplugged. During the marathon, the robot was often harassed by people, surrounded by large groups, and purposely cornered into narrow and difficult positions. Through all of this, the robot proved to be robust and safe.

VIII. LIMITATIONS

While the navigation system works well in many cases, there are some limitations that are important to address. One such limitation involves the RANSAC approach used to detect small ground obstacles. The approach processes single laser ground scans extremely quickly, but can generate false negatives for some obstacles. For example, RANSAC cannot tell the difference between a laser reading on a wall that is 1cm above the ground, and the actual ground plane. Typically, this does not affect the performance of the PR2 since walls tend to be vertical, causing the robot to avoid them based on sensor readings received higher up. However, the PR2 is susceptible to hitting long, low obstacles.

Another shortcoming of the navigation system stems from inconsistencies between the global and local planners. The global planner uses a circular approximation of the robot for planning, while the local planner uses the actual robot

footprint. This means there are cases where the global planner creates plans, such as through a half-closed door, that the local planner cannot execute. Here, the robot will struggle to pass through the doorway until its current goal times out and it receives a new goal in a different location.

The tilting laser also limits navigation. Slow 3D scan times result in the robot either moving slowly, or in a potentially unsafe manner. To run the robot at a reasonable speed, some safety was sacrificed in setting the unknown tolerance in a given column to 25 percent.

Finally, the PR2 has no way of detecting when it is stuck and asking for help. We attempted to use wheel odometry to detect these situations so the robot could email for help, but this technique yielded too many false positives and needs further work. Visual odometry gave more promising results for detection of wheel slippage and stalls, but was not integrated into the navigation system by the time of writing.

IX. Conclusions and Future Work

We have developed a robust navigation system for robots in office environments, and demonstrated 26.2 miles of autonomous navigation in a real office environment. The system uses a Voxel Grid and modelling of unknown space to allow safe behavior in complex environments. The Voxel Grid has a computational burden comparable to 2D grid-based navigation techniques, but allows for better behavior in the presence of 3D obstacles such as tables. The modelling of unknown space also provides a principled approach to determining maximum safe operating speed and required sensor density for the robot. The system performs close to limits set by the update rate of the tilting laser range-finder, and demonstrates that a single tilting range-finder can be sufficient to detect and avoid most obstacles in an office environment.

There are many areas in which the system can be improved, including: explicit modelling of dynamic obstacles such as people, integration of high update-rate sensors such as stereo cameras, improved detection and tracking of the ground plane, improved detection and recovery behaviors for collisions or caster stalls, integration with an online Simultaneous Localization and Mapping system, and use of a topological planner to handle large-scale environments. We leave this and other potential improvements as future work in our continued development of the PR2 navigation system.

X. Acknowledgements

The authors of this paper would like to thank the ROS and PR2 development teams for their hard work towards providing a stable platform for robotics development. Also, we thank Steffi Paepcke for editing the text of this paper, and Vijay Pradeep for his help with laser calibration. Finally, we would like to thank all members of the Milestone 2 team for their extensive testing of the PR2 navigation system.

References

[1] W. Burgard, A. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, "The interactive museum tour-guide robot," in *Proc. of the National Conference on Artificial Intelligence*, 1998.

[2] S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hhnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz, "Minerva: A second-generation museum tour-guide robot," in *In Proceedings of IEEE International Conference on Robotics and Automation (ICRA99)*, 1999.

[3] I. R. Nourbakhsh, C. Kunz, and T. Willeke, "The mobot museum robot installations: A five year experiment," in *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2003, pp. 3636–3641.

[4] R. Siegwart, "Robox at Expo.02: A Large Scale Installation of Personal Robots," *Special issue on Socially Interactive Robots, Robotics and Autonomous Systems*, no. 42, pp. 203–222, 2003.

[5] G. Kim, W. Chung, K. rock Kim, and M. Kim, "The Autonomous Tour-Guide Robot Jinny," in *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2004, pp. 3450–3455.

[6] H. Surmann, A. Nchter, and J. Hertzberg, "An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments," 2003.

[7] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, and S. Thrun, "Junior: The stanford entry in the urban challenge," *J. Field Robot.*, vol. 25, no. 9, pp. 569–597, 2008.

[8] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. B. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *International Conference on Robotics and Automation*, ser. Open-Source Software workshop, 2009.

[9] M. Fischler and R. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Application to Image Analysis and Automated Cartography," *Communications of the ACM*, vol. 24, pp. 381–395, 1981.

[10] J. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Systems J.*, vol. 4, no. 1, pp. 25–30, 1965.

[11] R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand, "On the influence of sensor capacities and environment dynamics onto collision-free motion plans," in *Proc. in IEEE International Conference on Intelligent Robots and Systems*, 2002.

[12] D. Fox, "KLD-sampling: Adaptive particle filters," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds. Cambridge, MA: MIT Press, 2001.

[13] K. Konolige, "A gradient method for realtime robot control," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2000.

[14] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[15] D. C. Moore, A. S. Huang, M. Walter, E. Olson, L. Fletcher, J. Leonard, and S. Teller, "Simultaneous local and global state estimation for robotic navigation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2009.

[16] N. Koenig and A. Howard, "Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, Sep 2004, pp. 2149–2154.