Persistent Navigation and Mapping using a Biologically Inspired SLAM System
Michael Milford and Gordon Wyeth
The International Journal of Robotics Research published online 21 July 2009
DOI: 10.1177/0278364909340592

The online version of this article can be found at:
http://ijr.sagepub.com/content/early/2009/07/21/0278364909340592

Published by:
SAGE
http://www.sagepublications.com

On behalf of:
Multimedia Archives

Additional services and information for The International Journal of Robotics Research can be found at:

Email Alerts: http://ijr.sagepub.com/cgi/alerts

Subscriptions: http://ijr.sagepub.com/subscriptions

Reprints: http://www.sagepub.com/journalsReprints.nav

Permissions: http://www.sagepub.com/journalsPermissions.nav
Abstract

The challenge of persistent navigation and mapping is to develop an autonomous robot system that can simultaneously localize, map and navigate over the lifetime of the robot with little or no human intervention. Most solutions to the simultaneous localization and mapping (SLAM) problem aim to produce highly accurate maps of areas that are assumed to be static. In contrast, solutions for persistent navigation and mapping must produce reliable goal-directed navigation outcomes in an environment that is assumed to be in constant flux. We investigate the persistent navigation and mapping problem in the context of an autonomous robot that performs mock deliveries in a working office environment over a two-week period. The solution was based on the biologically inspired visual SLAM system, RatSLAM. RatSLAM performed SLAM continuously while interacting with global and local navigation systems, and a task selection module that selected between exploration, delivery, and recharging modes. The robot performed 1,143 delivery tasks to 11 different locations with only one delivery failure (from which it recovered), traveled a total distance of more than 40 km over 37 hours of active operation, and recharged autonomously a total of 23 times.

1. Introduction

The simultaneous localization and mapping (SLAM) problem is often identified as one of the key challenges in mobile robotics that must be solved before robots can become part of everyday life in domestic homes and the workplace. Over the past two decades, a significant proportion of the research in the robotics community has been dedicated to solving this problem. Typically, research in the SLAM field has been performed based on datasets gathered from robots or vehicles that have been manually driven through an environment, with many researchers making use of commonly available datasets (Stachniss et al., 2007; Howard and Roy, 2008). SLAM algorithms developed in this manner are generally assessed by the quality of the map they produce, and by the size and complexity of the environment that they are capable of mapping. There are currently several SLAM algorithms that enable a robot to form accurate maps of environments ranging from office buildings (Grisetti et al., 2007) to mine shafts (Thrun et al., 2003). The advent of competent SLAM algorithms has enabled investigations of autonomous robot operation based on the localization resource provided by the algorithm. For example, the MobileRobots MAPPER software package1 can cre-

KEY WORDS—persistent navigation and mapping, SLAM, RatSLAM, biologically inspired

ate a map by wandering an environment during a user-defined mapping phase. Subsequently, the acquired map can be used for autonomous goal-based navigation. This is typical of the use of maps built using SLAM algorithms: there is an initial mapping phase after which the map is fixed, followed by autonomous operation that assumes that the world will remain unchanged.

The challenge of persistent navigation and mapping is to break the assumption that the world remains static by continuously and autonomously performing SLAM throughout the robot’s lifetime. The difficulty of this challenge lies in the range of timescales over which change can occur, as well as the variation in what can change in an environment. The static world assumption means that most current SLAM algorithms will fail in the long term from accumulation of errors as the world changes about the robot. Some research has extended existing SLAM algorithms to handle certain classes of dynamic objects such as people (Wang et al., 2003), or to detect and incorporate changes in a map (Biber and Duckett, 2005; Wolf and Sukhatme, 2005). Despite these advances, the goal of producing an autonomous robot that can concurrently maintain and use a map over its lifetime is yet to be achieved.

Biological systems, on the other hand, demonstrate the ability to build and maintain spatial representations that are used as the basis of goal directed navigation throughout the lifetime of the organism. A rat, for example, forages over a wide territory that is constantly changing due to the activity of other animals, the weather and the season. Despite the constant state of flux of sensory cues and available paths, the rat can reliably return to known sources of food and return home to its own nest. Based on this insight, we have investigated a biologically inspired approach to the persistent navigation and mapping problem.

This paper demonstrates the performance of the RatSLAM algorithm as the core of a persistent navigation and mapping system. RatSLAM is a biologically inspired localization and mapping algorithm based on computational models of parts of the rodent brain. The parts of the brain believed to underpin navigation in the rodent are centered on the hippocampus, which has been shown to form spatial representations based on the integration of self-motion signals with calibration from external sensory cues (O’Keefe and Conway, 1978; Ranck, 1984; Hafting et al., 2005). RatSLAM has been demonstrated in previous work to be able to perform real-time visual SLAM in areas as large as an entire suburb (Milford and Wyeth, 2008a).

The challenge set for this paper was to develop an autonomous mobile robot system which could perform mock delivery tasks over a two-week period in a real-world, non-static environment with a minimum of human intervention. The environment was a busy, functioning floor of an office building with cubicles, laboratories, corridors, offices and a kitchen. There were no modifications to the environment to aid mapping or localization, such as placing of navigation beacons, and no instructions given to the occupants of the building. Apart from providing the robot with locations for deliveries, there was no human intervention in robot operation or map building and maintenance. As a further challenge, the robot was transferred without notice to a different office building with an array of corridors. New delivery locations were specified in the new building, and the robot was asked to perform delivery tasks. After a period of time the robot was returned to the original environment.

1.1. Contributions of This Paper

Our key contribution is to demonstrate a system for “out of the box”, long-term mobile robot autonomy with vision-based lifelong SLAM. We achieve this by using the biologically inspired RatSLAM system (Milford and Wyeth, 2008a), and achieve long-term stability through a simple but effective map maintenance procedure. Unlike past autonomous mobile robots, SLAM is run continuously in parallel to whatever task the robot is performing, whether it is exploring, delivering, or recharging. There are no stages of learning and the robot’s map of the world is never considered “complete” and made static, removing the need for human input. As SLAM is always running, the robot is also able to learn changes in the appearance of the environment, enabling operation over day–night cycles and in the long term. Map maintenance ensures that an appropriate amount of information is retained at any one location in the map; enough to enable localization but with an upper bound to limit map size. Consequently, computational demand is dependent only on the size of the environment(s) and is stable over time. RatSLAM also provides the robot with a spatial awareness of its entire environment, providing instantaneous and continual information about where the robot is currently located as well as sufficient information to plan and execute effective paths to any part of the environment.

The paper draws on earlier work published in conference papers (Milford et al. 2004, 2005, 2006), and which has been used in offline dataset SLAM experiments (Milford and Wyeth, 2008a, b). We present for the first time a new system for integrating continuous SLAM with autonomous behaviors, a system of map maintenance, and new results demonstrating the use of the entire system in live delivery experiments over a two-week period across two different office buildings.


In this section we review the current state of the art in research that contributes to solutions for persistent navigation and mapping, including an overview of neuro robotic navigation systems based on rodents, such as the RatSLAM system.
2.1. Autonomous SLAM: Map Building and Maintenance

The SLAM problem captures the dual challenges faced by a navigating robot when placed in an unknown environment. In order to navigate effectively, the robot must build up a representation or map of the environment. However, new information from sensor observations of the environment can only be integrated into the map if the robot knows the location in the map from which the observations were made, so the robot must also localize within this evolving map. The current state of the art solutions to the SLAM problem work well in environments that are “static, structured and of limited size” (Cheeseman and Smith, 1986; Smith et al., 1990). Self-motion estimates and observations of landmarks update the state vector and covariance matrix. For EKF SLAM to work the data association problem must be solved; the system must be able to recognize whether an observed landmark matches one of the landmarks in the map. Graphical SLAM techniques represent the robot and landmark locations as nodes in a graph (Lu and Milios, 1997; Duckett et al., 2002; Folkesson and Christensen, 2004; Thrun and Montemerlo, 2006). Nodes experienced in sequence are connected by links that encode the movement of the robot given by odometry. Graph relaxation can ensure the best overall map (Duckett et al., 2002). The third paradigm uses particle filters, where particles sample the posterior distribution. One of the most successful implementations using particle filters is FastSLAM (Montemerlo et al., 2002, 2003). Implementations of all three SLAM paradigms have been used to successfully create various maps of large real-world indoor and outdoor environments.

The maps created from any of these paradigms are generally used as static resources. The robot performs SLAM during a designated mapping phase, and the map is subsequently used for autonomous operation. A well-known example of this approach was the Minerva museum tour guide robot (Thrun, 2000), which operated using a pre-acquired static occupancy grid map and ceiling mosaic map. Dynamic features of the environment were ignored or filtered and only features known to be static were used for localization. If the static features were to be changed (for example, when the displays were re-arranged), the robot would need to be told to re-map the environment. In a museum application this is perhaps acceptable, but for the wide range of applications where change is more accumulative and less structured, the required level of human intervention is not acceptable.

The problem of lifelong mapping in a dynamic environment is very challenging, and compared with the static SLAM problem only a small amount of work has been done. Such work that has been done has focused on specific aspects of the problem. For example, work by Wolf and Sukhatme (2005) showed that, with some help from artificial beacons with known positions, a robot was able to learn separate occupancy grid maps of the environment, one each for static and dynamic sections of the environment. Work by Wang et al. (2003) addressed detection and tracking of moving objects (DATMO) in conjunction with SLAM and demonstrated success over more than 100 miles of vehicle travel through urban areas. Biber and Duckett (2009) represented the environment over multiple timescales using parallel occupancy maps, where laser-scan memories faded at different rates depending on timescale. Repeated manually controlled robot traverses through an environment demonstrated that such an approach could enable robust map updating in the face of changes ranging in timescale from seconds (moving people) to days (permanent installation of new radiators). A vision-based version was also successfully demonstrated (Dayoub and Duckett, 2008).

2.2. Rat-based Neurorobotic Navigation and Mapping Systems

Animals, particularly mammals, have innate spatial understanding that enables reliable navigation to known food and water sources, territorial boundaries and guides the return to the den or nest. In this section we focus briefly on robot mapping or navigation systems based on the neural mechanisms thought to underlie these processes.

The rodent family has been extensively studied with regard to the neural underpinnings of navigation and mapping. A part of the rat brain known as the hippocampus contains spatially responsive cells which usually form the basis for rat-based neurorobotic systems. Place cells are cells that fire only when the rat is located at a certain place in the environment (O’Keefe and Conway, 1978), while head-direction cells fire only when the rat is facing in a certain direction (Ranck, 1984). Taken together, the firing of these cells encodes the complete state of the rat within a two-dimensional plane over the environment. More recently, a new type of cell known as a grid-cell was discovered, which encodes multiple rat locations arranged in a tessellating hexagonal grid over the environment (Hafting et al., 2005).

Arleo and Gerstner (2000) developed a computational model of place and head-direction cells which was demonstrated using a Khepera robot in a small (0.5 x 0.5 m$^2$) arena with bar-coded walls. The model was coupled strongly to the biology of place cells, and showed how place cell firing can enable mapping and goal navigation within the arena (Burgess et al., 1997). The Psikharpax project is building an artificial robotic rat driven by mapping and navigation algorithms mimicking place cells (Meyer et al., 2005). More recent approaches...
have also used grid cells (Giovannangeli and Gaussier, 2008) to enable navigation and mapping on a robot in a larger room-sized arena. Barrera and Weitzenfeld (2008) have developed a biologically inspired robot architecture with spatial cognition and navigation capabilities. Their system is able to learn and unlearn designated goal locations and navigate to them from any point within their map. The system was demonstrated in an eight-arm radial maze and a single and double T-Maze, with artificial visual cues on the walls of the maze.

In our previous work, we have shown that a neurorobotic system based on models of hippocampus can map large unmodified areas based on natural visual cues. The RatSLAM system has been demonstrated mapping office environments (Milford et al., 2006), a university campus (Prasser et al., 2005) and an entire suburb (Milford and Wyeth, 2008a). RatSLAM produces topological maps that have some local metric properties, and shares similarities with established systems such as the spatial semantic hierarchy (Kuipers and Byun, 1991). As a primarily appearance-based SLAM system, it is one of a number of recent appearance-based systems (Andreasson et al., 2008; Angeli et al., 2008; Cummins and Newman, 2008; Mahon et al., 2008) that have been catching up to large-scale metric mapping systems. RatSLAM is the basis of the algorithm at the core of persistent navigation and mapping system described in this paper, and is described in detail in Section 4.

2.3. Summary

Clearly, state-of-the-art SLAM algorithms are capable of producing accurate maps of large indoor or outdoor environments. One of the main challenges now appears to be how best to incorporate SLAM algorithms into autonomous robot systems in a manner that provides significant improvements in robot capabilities and autonomy. The key to opening a new range of applications for mobile robots is to provide long-term goal-directed navigation ability without user supervision or intervention. The most significant challenge is to provide a method for keeping spatial representations up to date in environments that change across a range of time scales. The challenge set in this paper was to develop a system where SLAM occurred continually and over the entire operating life of the robot, in order to enable effective navigation performance over the long term.

3. System Architecture

In this section we provide a high-level overview of the system architecture. The autonomous navigation and mapping system can be divided by functionality into three subsystems, SLAM, navigation, and task selection, as shown in Figure 1.

The RatSLAM system takes visual and self-motion sensor data and creates a semi-metric spatial representation known as an experience map, which is used to plan global routes in the navigation system and to facilitate exploration. RatSLAM continuously maintains its spatial representations without intervention from humans or other processes, while providing a localization and planning resource. There are no specific learning or recall phases.

The robot has two separate representations of space available to it; the global semi-metric experience map from RatSLAM and a local obstacle map built from sensor readings. The experience map is set in a fixed world reference frame, and describes the layout and connectivity of all of the navigable regions that have been explored by the robot. The local obstacle map is set in a robot-centered reference frame, and describes the local navigable regions based on a short history of range sensor readings. Long-term goals, such as navigating to a delivery location, are based on the experience map, while short-term behaviors use the local obstacle map.
There are three principal navigation processes. The global navigation process plans and executes a route to a specified goal within the global experience map, specifying the short-term behavior by setting a local path. The local path is interpreted by the local navigation process in the context of the local obstacle map to produce the desired motion commands. The third process, exploration, operates when no delivery or recharging task is specified. Exploration uses global knowledge from the experience map to specify a local path that will improve the robot’s coverage of the environment.

The robot chooses between two tasks depending on the availability of a delivery goal and the battery state. The delivery task is executed by a scheduler that sets the global goal to a location picked at random from a list of user-specified delivery locations. The recharging behavior runs as a fixed two-stage process, first setting and navigating to a global goal near the charger, followed by a local homing procedure to physically dock with the charger. In the following sections, we provide a more detailed description of each of the subsystems and the algorithms that drive them.

4. RatSLAM

To perform SLAM we use RatSLAM, a biologically inspired SLAM system based on models of mapping in the rodent hippocampus. RatSLAM consists of three components; a set of local view cells, a network of pose cells, and an experience map, shown in Figure 2. The following sections describe each of these components in further detail.

4.1. Pose Cells

The pose cells are a three-dimensional continuous attractor network (CAN). CANs have been a popular choice for modeling the spatially responsive cells found in the rodent brain (Samsonovich and McNaughton, 1997; Arleo and Gerstner, 2000; Stringer et al., 2002a, b). The CAN is a neural network that consists of an array of units with fixed weighted connections that can be both excitatory and inhibitory. Unlike most neural networks, the CAN predominantly operates by varying the activity of the neural units, rather than by changing the value of the weighted connections. RatSLAM uses a rate-coded CAN, meaning that each neural unit has a continuous activation value between zero and one. The interpretation of
the unit activation level is generally as a measure of a real biological cell’s firing rate; the higher the activation level, the faster the firing rate it represents. In rodents, spatially responsive cells such as place cells fire fastest when the rat is located at a certain location, and reduce their firing rates as the rat moves away from this location. In RatSLAM, the activation value of a neural unit increases when the robot approaches a location associated with that neural unit.

During operation, the pose cell network will generally have a single cluster of highly active units: the activity packet. The center of the activity packet provides an estimate of the robot’s pose that can be determined from the pose cell network’s structure. The pose cell network is arranged in a rectangular prism structure, as shown in Figure 2. Any three-dimensional structure can be used, but optimal configurations are those where the structure tessellates in all three dimensions, which simplifies the connectivity required across network boundaries. Each of the three dimensions of the network corresponds to one of the three spatial dimensions \( x', y', \) and \( \theta' \). Primed coordinates are used to differentiate the space from that used with the experience map. The location of the active neural units in the rectangular prism structure correlates with the robot’s pose in \((x', y', \theta')\) space.

4.1.1. Attractor Dynamics

The intrinsic attractor dynamics are designed to maintain a single activity packet in the CAN. Local excitatory connections increase the activity of units that are close in \((x', y', \theta')\) space to an active unit, generating the main cluster. Inhibitory connections suppress the activity of smaller clusters of activity. For each pose cell, local excitation and inhibition is achieved through a three-dimensional Gaussian distribution of weighted connections, as shown by the arrows connecting pose cells in Figure 2. The distribution, \( e \), is given by

\[
e_{a,b,c} = e^{-(a^2+b^2)/2k_p} e^{-c^2/2k_d} - e^{-(a'^2+b'^2)/2k'_p} e^{-c'^2/2k'_d}, \tag{1}
\]

where \( k_p \) and \( k_d \) are the variance constants for place and direction respectively, and \( a, b, \) and \( c \) represent the distances between units in \( x', y', \) and \( \theta' \) coordinates respectively. The variances for inhibition are larger than for excitation, creating the so-called Mexican-hat function (Kohonen, 1995). The connections wrap across all six faces of the pose cell network, as shown by the longer arrows connecting pose cells in Figure 2, so the indices \( a, b, \) and \( c \) are given by

\[
a = (x' - l)(\text{mod} n_x),
\]

\[
b = (y' - j)(\text{mod} n_y),
\]

\[
c = (\theta' - k)(\text{mod} n_{\theta'}). \tag{2}
\]

The change in a cell’s activity level \( \Delta P \) is given by

\[
\Delta P_{x', y', \theta'} = \sum_{i=0}^{n_x-1} \sum_{j=0}^{n_y-1} \sum_{k=0}^{n_{\theta'}-1} P_{i,j,k} e_{a,b,c} - \varphi, \tag{3}
\]

where \( n_x, n_y, \) and \( n_{\theta'} \) are the size of the network in number of cells along each of the dimensions \( x', y', \) and \( \theta' \), and \( \varphi \) creates global inhibition. The final network stage thresholds activation levels in \( P \) to non-negative values and normalizes the total activation to one.

4.1.2. Path Integration

Path integration involves shifting the activity packet in the pose cell network based on self-motion information (such as odometry). While a higher fidelity biological model would shift activity by using an appropriate set of weighted connections, RatSLAM simply displaces a copy of the current activity state by an amount based on nominal spatial areas and orientation bands of a pose cell. Copying and shifting activity offers stable path integration performance over a wider range of movement speeds and under irregular system iteration rates. Like the excitatory and inhibitory weight matrices, the path integration process can cause a cluster of activity in the pose cells to shift off one face of the pose cell structure and wrap around to the other, as is shown in both packets in Figure 2, one of which is wrapping across the \( \theta' \) boundary.

Path integration based on self-motion information does not result in the activity in the pose cells spreading to represent an increase in pose uncertainty. Unlike a particle filter, the relocalization process in RatSLAM does not require or benefit from maintaining any notion of uncertainty in the pose state. It is also of interest to note that the biological cells found in real rats do not appear to encode increasing uncertainty under animal movement (Fuhs and Touretzky, 2006).

4.1.3. Local View Cells

The local view cells are an array of rate-coded units used to represent what the robot is seeing. Each local view cell becomes associated with a distinct visual scene in the environment, and becomes active when the robot sees that scene. Multiple local view cells can be active to varying degrees simultaneously, and there are no interconnections or recurrent connectivity amongst the cells. Connections are, however, formed between local view cells and pose cells, as is described in the next section.

4.1.4. Learning and Recalling Visual–Place Associations

The path integration process is subject to the accumulation of errors in odometry, which becomes critical in situations
such as loop closure. To correct odometric drift, RatSLAM increases the strength of connections between local view cells and pose cells that are active simultaneously. In this way, when familiar visual scenes activate local view cells, the learnt connections to the pose cells will move the activity packet towards the correct location, either correcting odometric drift, or possibly closing a loop. The connections between local view cells and pose cells are stored in a connection matrix $\beta_i$ where the connection between local view cell $V_i$ and pose cell $P_{x',y',\theta'}$ is given by

$$\beta_{x',y',\theta'}^{+1} = \max \left( \beta_{x',y',\theta'}^t + \lambda V_i P_{x',y',\theta'}, \lambda V_i P_{x',y',\theta'} \right),$$

where $\lambda$ is the learning rate. When a familiar visual scene activates a local view cell, the change in pose cell activity, $\Delta P_i$, is given by

$$\Delta P_{x',y',\theta'} = \frac{\delta_{P_{act}}}{P_{act}} \sum_{i} \beta_{v,x,y,\theta} V_i,$$

where the $\delta$ constant determines the influence of visual cues on the robot’s pose estimate, normalized by the number of active local view cells $P_{act}$. Figure 2 represents the moment in time when a strongly active local view cell has injected sufficient activity into the pose cells to cause a shift in the location of the dominant activity packet. The previously dominant activity packet can also be seen, which is less strongly supported by a moderately activated local view cell.

When a re-localization event occurs, a discontinuity is often introduced into the pose cells where two places that are close in the physical world are represented by pose cells that are separate in the pose cell network, especially if significant odometric error has accumulated. Consequently, over time the $(x', y', \theta')$ arrangement of the pose cells corresponds less and less to the spatial arrangement of the physical environment. The wrapping connectivity can also lead to pose ambiguity, where a pose cell encodes multiple locations in the environment. Rather than try and correct these discontinuities and ambiguities within the pose cells, we combine the activity pattern of the pose cells with the activity of the local view cells as input to a relaxation algorithm, which creates a topologically consistent and semi-metric map in a separate coordinate space.

4.2. Experience Mapping

The experience map is a semi-metric topological map containing representations of places, called experiences, $e$, and links between experiences describing the transitions, $t$, between these places. Each experience is defined by the conjunction of a certain activity state $P_i$ in the pose cells and an active local view cell $V_i$. However, each experience is positioned at a location $p_i$ in experience space, which is similar to real-world Cartesian space but with connectivity constraints. Consequently, the complete state of an experience can be defined as the 3-tuple:

$$e_i = \{ P_i, V_i, p_i \}.$$

Figure 2 shows the region of pose cells and the single local view cell associated with the currently active experience A.

4.2.1. Experience Creation and Matching

When the combined activity state of the pose cells and local view cells is not sufficiently described by any of the existing experiences, a new experience is created. A score metric $S$ is used to compare how closely the current pose and local view states match those associated with each experience, given by

$$S_i = \mu P |P_i - P| + \mu V |V_i - V|,$$

where $\mu_p$ and $\mu_V$ weight the respective contributions of pose and local view codes to the matching score. If $\min(S) \geq S_{\max}$, a new experience is created, defined by the current pose and local view cell activity states. However, if any experience matching scores are below the threshold, the lowest scoring is chosen as the “active” experience, and represents the best estimate of the robot’s location within the experience map.

4.2.2. Experience Transitions

The creation of a new experience also creates a transition link $l_{ij}$ from the previously active experience $e_i$ to the new experience $e_j$. Transition links encode the change in position, $\Delta p_{ij}$, computed directly from odometry, and the elapsed time, $\Delta t_{ij}$, since the last experience was active:

$$l_{ij} = \{ \Delta p_{ij}, \Delta t_{ij} \}.$$

The odometry information defines the initial location in experience space of a newly created experience:

$$e_j = \{ P^j, V^j, p^j + \Delta p^j \}.$$

The temporal information defines the travel time between places in the environment and is used to create the temporal map which is used in global navigation, described in Section 6.1.

4.2.3. Experience Map Relaxation

In the initial stages of map building the spatial relationships between linked experiences in experience space, defined in Equation (9), exactly match the odometric information contained in the transitions between those experiences. When loop closure occurs, it is likely that the relative position of the two linked experiences in the map will not match the odometric transition information between the two. The experience map relaxation method seeks to minimize the discrepancy between
odometric transition information and absolute location in experience space, by applying a change in experience location $\Delta p'$:

$$\Delta p' = \alpha \left[ \sum_{j=1}^{N_f} (p'_j - p' - \Delta p^{ij}) ight]$$

$$+ \sum_{k=1}^{N_t} (p^k - p' - \Delta p^{ki}) ] ,$$

where $\alpha$ is a correction rate constant, $N_f$ is the number of links from experience $e_i$ to other experiences, and $N_t$ is the number of links from other experiences to experience $e_i$. Equation (10) is applied iteratively at all times during robot operation; there is no explicit loop closure detection that triggers map correction. The effect of the repeated application of Equation (10) is to move the arrangement of experiences in experience map space incrementally closer to an arrangement that averages out the odometric measurement error around the network of loops. This method is an extension of the map relaxation algorithm described by Duckett et al. (2002) with the addition of an ego-rotation parameter.

4.3. Experience Map Maintenance

The experience map algorithm will continue to add experiences about the changing state of the world as the robot operates. In this way, the robot constantly updates and maintains its representation of the world. For example, if the robot proceeds along a corridor on one day with an adjoining door open, then the robot will build an experience that captures the open door. If the door is closed on the next day, then a new experience will be constructed with the door closed. The experiences will overlap in the experience map, by virtue of their connectivity to surrounding experiences, effectively mapping the same overlap in the experience map, by virtue of their connectivity. This is not an extension of the existing algorithm but is a property inherited from the algorithm’s biological origins.

The difficulty with using this method of map maintenance is that the robot will remember all experiences from throughout its lifetime, creating an unmanageable list of experiences to search and update. The list of experiences must be pruned to a manageable level in order to attain the goal of persistent navigation and mapping where the robot can continue to map and navigate indefinitely. Experience pruning is achieved by consolidating parts of the experience map which exceed a maximum spatial density threshold. In this way the number of experiences grows in proportion to the size of the environment that has been explored, but not with time.

Figure 3 illustrates an experience being pruned from the map. The pruning algorithm uses a grid overlaid on the experience map to tag squares that contain more than one experience.

![Fig. 3. Experience map pruning. Experiences are removed to maintain a one experience per grid square density. (b) Transitions to or from removed experiences are either deleted or reconnected to remaining experiences.](image)

Formally, the algorithm prunes an experience when $|k|$ is larger than one, given

$$\{ k \mid k \in e, p \in g \}$$

where $g = \{ x_g < x < x_{g+1} \}$

$$y_g < y < y_{g+1}$$

$$0 < \theta < 2\pi$$

where $e$ is the set of experiences, $p$ is an experience’s $(x, y, \theta)$ coordinates within experience space and $g$ defines the grid square. An experience $r$ is picked at random from the set $k$, and all other experiences are deleted. Other methods for choosing the experience to delete were investigated (using measures such as recency and connectivity) but were found to have no overall beneficial effect on performance (i.e., application of recency metrics led to biases towards short-term mapping cycles). Existing transitions to deleted experiences are either deleted or updated to link to another experience:

$$\{ k \mid k \neq r \} : \begin{cases} l_{ik} = l_{ir}, i = 1 : N_f \\ l_{kj} = 0, j = 1 : N_t \end{cases}$$

where $l_{ik}$ are all of the links from other experiences to each experience $k$ and $l_{kj}$ are all the links from each experience $k$ to other experiences. The odometric information $\Delta p$ for the new link is inferred from the current positions of the experiences in experience space, while the temporal information $\Delta t$ is calculated by dividing the inter-experience distance by the average robot speed.

In the situation illustrated in Figure 3, experience B is removed, one transition is removed and two other transitions are updated and reconnected. If, after the pruning process, any local view cells no longer define any experiences (see Equation
Fig. 4. Vision hardware and vision processing system. A camera and mirror produces panoramas which are unwrapped into 360° by 90° images. The image is then reduced in resolution, patch normalized to enhance local image contrast and correlated with all template images. Template images that are close matches to the current image activate local view cells, which link to the pose cells and experience map.

(6)), the cells and the visual templates associated with them are removed. Any local view–pose cell links from these removed local view cells are also deleted, so that the post pruning local view cell vector $V$ and local view–pose cell link vector $\beta$ are given by

$$V = \begin{bmatrix} V_i & c_i > 0 \\ 0 & c_i = 0 \end{bmatrix}, \quad \beta = \begin{bmatrix} \beta_i & c_i > 0 \\ 0 & c_i = 0 \end{bmatrix}$$

where $c_i = |\{e|V_e = i\}|$, (13)

where $V_e$ is the index of the local view cell that defines experience $e$ and $c_i$ is a count of the number of experiences defined by the $i$th local view cell.

5. Vision Processing and Appearance-based Recognition

The vision system uses panoramic images obtained from a Basler camera, mounted at the central rotation axis of a Pioneer 3DX robot and facing vertically upwards at a parabolic mirror (see Figure 4). Automatic adjustment for global illumination changes is achieved in hardware through gain and exposure control. Local variations in illumination can still be quite severe, such as patches of sunlight on the floor which disappear at night. We use patch normalization to reduce local variation, which has been shown to enable robust scene recognition in varying illuminations (Zhang, 2007), and process only the bottom half of each image to reduce the effects of direct sunlight. The patch normalized pixel intensities, $I'$, are given by

$$I'_{xy} = I_{xy} - \mu_{xy} \frac{\sigma_{xy}}{\sigma_{xy}},$$

where $\mu_{xy}$ and $\sigma_{xy}$ are the mean and standard deviation of pixel values in a patch of size $P_{size}$ surrounding $(x, y)$.

5.1. Scene Recognition

Since this is an indoor robot application, we assume that the ground surface is locally flat and that the robot is constrained to the ground plane. The recognition process starts with the current unwrapped, patch normalized $w$ by $h$ pixels panoramic image, with the $w$ dimension aligned with the ground plane in the real world and the $h$ dimension aligned with the vertical plane. Image similarities between the current image and
template images are calculated using the cross correlation of corresponding image rows. For each row this correlation can be performed efficiently in the Fourier domain, in which multiplication is equivalent to convolution in the spatial domain:

\[ C = F^{-1} \left[ \sum_{y=1}^{b} F\left( t_y \right) \cdot F\left( r_y \right) \right], \]

where \( F(\ldots) \) is the Fourier Transform operator and \( i_y \) and \( r_y \) are the pixels rows at \( y \) in the current and template images, respectively. The value of the maximum real correlation coefficient gives the quality of the match \( m \):

\[ m = \max \left( \text{Re} \left( C \right) \right). \]

A new image template and local view cell is created if the best match for all current image–template image pairings is below a threshold \( m_{\text{min}} \). Fourier transform coefficients are calculated only once for an image and stored by the vision system. As the number of image templates becomes large, computation becomes \( O(m w \log w + m u h) \), scaling linearly with the number of templates. Coding optimizations for the Fourier transform, such as the use of the single instruction, multiple data (SIMD) instruction set extension, are implemented using the freely available fast FFTW C library (Frigo and Johnson, 1998).

5.1.1. Local View Cell Calculation

The match quality scores for each image pair are used to set the activation levels for the corresponding local view cells:

\[ V_i = \max(m_i, 0) \quad \text{for all } i. \]

The local view cell activation level is limited to non-negative values, since the correlation coefficient and hence match quality \( m \) has the range \((-1 \leq m \leq 1)\).

6. Navigation System

In this section we describe the three principal navigation processes: global navigation, local navigation, and exploration.


The global navigation process plans a route to a specified goal location within the global experience map space, and executes this route by setting a local path for the local navigation process. Routes are optimized for travel duration rather than travel distance, meaning that the selected route is not always the shortest in distance, due to factors such as environment clutter. To calculate the fastest route to a goal, a temporal map is created by seeding the current active experience with a timestamp value of zero, and then iteratively parsing the experiences and updating their timestamps \( T^{k+1}_j \) based on the following rule:

\[ T^{k+1}_j = \min(T^k_j + \Delta t_{ij}, T^k_i), \]

where \( \Delta t_{ij} \) is the transition duration between experience \( i \) and \( j \). The fastest route to the goal is calculated by following the descending gradient from the goal location. Figure 5(a) shows a temporal map for the GP South environment, showing the predicted travel times to all locations in the environment from the robot’s current “Start” location. To transfer navigation control to the local navigation module, the immediate section of route
is converted into robot-centered coordinates and passed to the local navigation process. The robot is deemed to have reached the goal when it moves to within a distance $d_{del}$ of the goal experience. Figure 5(b) shows the actual path of the robot in navigating to this goal, showing the close correlation with the predicted travel time.

Figure 6 shows a process flow diagram for global navigation. Goals are pursued until achieved or a timeout occurs. Localization performance (the period of time since a successful localization) and tracking performance (how well the robot is following the global path) are used to monitor the health of the navigation process. Either measure degrading (10 seconds of bad localization or path tracking) can trigger a recovery period of exploration, during which time the robot tries to re-localize in order to update its path to the global goal.

### 6.2. Local Navigation

The local navigation process plans and executes a route to a local goal or path specified in robot centered coordinates, and creates velocity commands for the robot. To aid navigation around obstacles, a short time window of laser and sonar range readings and odometry are used to maintain a robot centered obstacle map (Figure 7(b) and (c)). Certain chairs in the environment were completely undetectable by the laser; a sheet of paper was attached to these chairs to aid local obstacle detection. A tree search is performed through space, with branch propagation ceasing when the depth exceeds $N_l$ or when the minimum clearance at a branch tip drops below $r_{min}$. The resulting tree branches are grouped into candidate pathways (Figure 7(c)), which are then converted into local path arc candidates (Figure 7(d)). Each path arc is compared with either a specified local goal or local path. In the case of a desired local path being specified, each candidate path arc $i$ is given a distance score $d_i$ based on the sum of minimum distances between points on the arc and the specified local path:

$$d_i = \sum_{i=1}^{n} \min (p' - a_i),$$  \hspace{1cm} (19)$$

where $p'$ is a set of $n_{path}$ evenly spaced points along the local path, and $a$ is the corresponding set of points along the candidate path arc. The path arc with the smallest distance score is converted into velocity commands which are sent to the robot. A more complete description of local navigation can be found in Milford (2008).

### 6.3. Exploration

Exploration is primarily a local navigation process which uses the global experience map to help increase environment coverage. The exploration algorithm compares the possible local path arcs with routes traversed previously by the robot from its current location in the experience map, using the distance

---

**Fig. 6.** Flowchart of the global navigation process. The system evaluates the health of the global navigation process by monitoring localization performance and the degree to which the robot is able to follow the global path. Periods of exploration are used to re-localize and to update the planned path to the goal.
7. Robot Tasks: Deliver and Recharge

The robot chooses between performing mock deliveries and recharging, depending on the availability of a delivery goal and the battery state. When neither task is active, the robot defaults to exploration of the environment. In this section we provide process flowcharts of each task.

The mock delivery task uses the global navigation process shown in Figure 8 to guide the robot close to the specified delivery location. If the robot were then to perform a real delivery, it would switch to a local, reactive process using sensor information to home to the precise delivery location. The full mock delivery process is shown in Figure 8.

The recharging process is an example of this type of two-stage process, and illustrates how a robot task can be completely autonomous, requiring no user intervention to set up the task (see Figure 9). Using prior information about the cross-sectional geometry of the recharging station, the algorithm tags the experience in the experience map from which the charging station can best be approached. The two-step recharge task first navigates the robot to the tagged experience using the global navigation system. Upon reaching this experience, the recharging system switches to a local vector field-based homing routine, driven by laser range scans of the docking station profile, which guides the robot onto the docking station and engages the charging plate. The positional accuracy achieved with this local homing was typically of the order of 50 mm.

8. Experimental Setup

The challenge set for the system was to perform the role of a delivery robot in real-world workplaces over a two-week period. The workplaces were floors in two different buildings at The University of Queensland in Brisbane, Australia, shown in Figures 10, 11, and 12. The two buildings, Axon and General Purpose South (GP South), are typical research environments, moderately populated during the day and cleaned by janitors at night. The sizes of the two environments were $43 \times 13 \text{ m}^2$ and $60 \times 35 \text{ m}^2$. The robot had to navigate through open plan office space, corridors, kitchens, and laboratories. The environments were by no means static, with moving people, changing door states, re-arrangement of furniture and equipment, and a range of trolleys that intermittently appeared and disappeared during deliveries, maintenance, and cleaning operations. Perceptually the environments also changed, with the most significant example being the day–night time cycles, which had an especially significant impact in areas with many windows, such as shown in Figure 10(a). Figure 13 shows the typical extent of perceptual change at one of the delivery locations.

We used a Pioneer 3 DX robot equipped with a panoramic imaging system, a ring of forward facing sensors, a Hokuyo laser range finder and encoders on the wheels (shown in Figure 4). Sensors were sampled at 7 Hz. All computation and logging was run onboard on a 2 GHz single core computer running Windows XP. The robot’s typical operation speed was $0.3–0.5 \text{ m s}^{-1}$. The robot operated continuously for two to three hours between recharging cycles. Owing to the need for supervision of the robot during operation (to ensure the integrity of the experiment), the total active time in a typical day was limited to one to four recharge cycles. In order to capture the effect of around-the-clock operation, experiments were conducted across all hours of the day.

8.1. Procedure

The experimenter first placed the charging station at a location in the Axon building. The experimenter then positioned the robot at a random location, turned it on, and initiated the
Fig. 8. Flow chart of the delivery process. After a delivery task is completed, further delivery tasks are suppressed for an arbitrary period of ten seconds, during which time the robot reverts to exploration.

Fig. 9. Flow chart of the recharging process. After navigating to the vicinity of the recharging station, the robot homes in on the station using a movement vector field. A failure to engage with the docking station (such as a bad electrical contact) results in a short period of exploration before restarting the process.

software suite. The robot commenced exploration of the environment. After 117 minutes, the experimenter specified six desired delivery locations, by clicking on these locations in the experience map. Detecting the specification of delivery locations, the robot commenced a process of picking a delivery location at random and navigating to it to make a “mock” delivery. When the robot detected a low battery state, it navigated to the recharging dock and docked to recharge. During re-charging the robot powered down in a fashion that retained the map, but lost localization.

After 8 days and 1,017 delivery trials in the Axon environment, the experimenter powered down the robot and took it to the GP South environment. The experimenter also placed the charging station in the new environment, and then turned on the robot and initiated the software suite. To take advantage of the wider corridors, the experimenter made a single change to a movement parameter governing the obstacle clearance; this change was not required but enabled higher speed operation. The robot was not told that it had been put in a new environment, and since it had no path solutions to any existing delivery locations, commenced exploration. After 68 minutes, the experimenter specified five desired delivery locations, by clicking on the experience map. With delivery goals that were now accessible, the robot commenced random delivery navigation. After 54 delivery trials the robot returned to the charger to recharge, after which the robot was turned off and replaced in the original Axon environment, with no notification that it had changed environments. The robot commenced navigation to the original delivery locations. A further 72 delivery trials were conducted in the Axon building, before the experiment ended after 11 days.
Fig. 10. Photos of the two environments. (a) Cluttered open plan office space in Axon building, note the number of windows. (b) Robotics laboratory in Axon building. (c)–(e) Corridors in GP South building. Panoramas generated using the Autostitch software demo (Brown, 2008).

Fig. 11. Delivery performance in the Axon building. The specified delivery locations are shown by black circles, while the crosses indicate the locations the robot navigated to during each delivery trial. The black point shows the average location. The scattered crosses at (38, 7) show the locations the robot navigated to during recharging, when the goal location was not constant. Doors were kept open.

8.2. Parameters

The parameters for the various robot systems and algorithms were set as shown in Tables 1–5. These parameter values have been obtained from extensive indoor experimentation and are usable for most typical indoor office environments. To transfer the system to a completely different platform and environment (say to a car driving around a city suburb) re-
Fig. 12. Delivery performance in the GP South building. The cross at (17, 16) shows the location the robot navigated to during recharging.

Fig. 13. (a) Collage of 20 snapshots of delivery location number 3 (the kitchen) each time the robot determined it had reached that location. Gain and exposure adjustment has already been performed at this stage. Changes in illumination from the window show the day/night cycles. (b), (c) Change in appearance of delivery location 2 from day to night. Note the people and bike.

requires changes to some parameters, as in Milford and Wyeth (2008a).

9. Results

The results show the performance of the robot in its main task of navigating to delivery locations, and the stability of the per-
Table 1. Pose Cell parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n_x$, $n_y$, $n_0$</td>
<td>60 x 60 x 36</td>
</tr>
<tr>
<td>Nominal pose cell size</td>
<td>0.25 m x 0.25 m x 10°</td>
</tr>
<tr>
<td>$k_p$, $k_d$ (excitation)</td>
<td>Four cells (1.0 m, 40°)</td>
</tr>
<tr>
<td>$k_p$, $k_d$ (inhibition)</td>
<td>Eight cells (2.0 m, 80°)</td>
</tr>
<tr>
<td>$\Phi$</td>
<td>0.00002</td>
</tr>
</tbody>
</table>

Table 2. Local View Cell Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda$</td>
<td>0.36</td>
</tr>
<tr>
<td>$m_{min}$</td>
<td>0.1</td>
</tr>
<tr>
<td>$\delta$</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Table 3. Experience Map Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>0.5</td>
</tr>
<tr>
<td>$\mu_p$</td>
<td>1.0</td>
</tr>
<tr>
<td>$\mu_v$</td>
<td>0.5</td>
</tr>
<tr>
<td>$S_{max}$</td>
<td>1.0</td>
</tr>
<tr>
<td>$s_q$</td>
<td>0.1 m</td>
</tr>
</tbody>
</table>

Table 4. Navigation and Exploration

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_l$</td>
<td>4</td>
</tr>
<tr>
<td>$n_p$</td>
<td>5</td>
</tr>
<tr>
<td>$l_{seg}$</td>
<td>0.3 m</td>
</tr>
<tr>
<td>$\theta_s$</td>
<td>17.2°</td>
</tr>
<tr>
<td>$r_{min}$</td>
<td>0.35 m (0.5 m in GP South)</td>
</tr>
<tr>
<td>$w_{max}$</td>
<td>40 °/s</td>
</tr>
<tr>
<td>$v_{max}$</td>
<td>0.5 m s⁻¹</td>
</tr>
<tr>
<td>$d_{del}$</td>
<td>0.1 m</td>
</tr>
</tbody>
</table>

Table 5. Other Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{size}$</td>
<td>11 pixels</td>
</tr>
</tbody>
</table>

Performance over the 11 days of the trial. Results were obtained from analysis of more than 9 GB of data including video from the robot, logged continually and backed up every time the robot docked to recharge. During fulltime operation the operation time to charging time ratio was approximately one-to-one (typically 2.5 hours operation, then 2.5 hours of charging), however the robot would occasionally be left on the charger for longer periods when the experiment was not running. The robot spent approximately 27 of the 37 hours of active operation performing navigation to delivery goals or the charging dock, 6.5 hours exploring, and 3.5 hours on docking and transitions between processes.

9.1. Delivery Performance

Delivery performance was assessed by examining the delivery success rate and by measuring ground truth of the delivery location. Ground truth was obtained by processing the on-board video frame captured at the time of each delivery location using the known location of three landmarks in the image (see Appendix A).

Table 6 shows the delivery success rates for each delivery location. A success was deemed to be the robot navigating to and logging a delivery completion in the vicinity of the actual delivery location. As discussed in Section 7, this is considered a success because the robot could then use a more accurate local reactive procedure, as demonstrated by the two-stage recharging task. Figure 11 shows the accuracy of the global navigation system with respect to ground truth when making deliveries. The global navigation system uses the parameter $d_{del} = 0.1$ m as the acceptable tolerance for a delivery to be deemed complete. Perfect localization at delivery would result in a delivery error of 0.1 m. Figures 11 and 12 show the distribution of the delivery points with respect to the specified delivery location in the Axon and GP-South buildings, respectively. The accuracy of delivery tasks is shown in Figure 14.

The robot failed to complete delivery trial number 579 (that is, failed to move close to the delivery location). This failure was due to an extended localization failure in room 505 (see Figure 10(b)), resulting in the robot erroneously localizing to the corridor outside the room when in reality it was in the corner of the room. Consequently, the global navigation system provided a local path to the local navigation system which could not be fulfilled, and the robot became “stuck” in the corner, before eventually timing out the delivery attempt and reporting a delivery failure. Room 505 was one of the most challenging environments for the mapping system, because robot motion was relatively unconstrained in the large open space, leading to unique template sequences. The delivery accuracy and repeatability was worst for the delivery location in this room.
Fig. 14. Accuracy of delivery tasks in (a) Axon building and (b) GP South building, shown using box whisker plots. Whiskers encompass the full range of delivery errors for each goal location, box bounds show upper and lower quartiles.

Table 6. Delivery Success Rates

<table>
<thead>
<tr>
<th>Goal location</th>
<th>Number of navigation trials</th>
<th>Number of successes/failures</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Axon</td>
<td>155</td>
<td>154/1</td>
</tr>
<tr>
<td>2 Axon</td>
<td>186</td>
<td>186/0</td>
</tr>
<tr>
<td>3 Axon</td>
<td>186</td>
<td>186/0</td>
</tr>
<tr>
<td>4 Axon</td>
<td>197</td>
<td>197/0</td>
</tr>
<tr>
<td>5 Axon</td>
<td>186</td>
<td>186/0</td>
</tr>
<tr>
<td>6 Axon</td>
<td>179</td>
<td>179/0</td>
</tr>
<tr>
<td>Docking station</td>
<td>34</td>
<td>34/0</td>
</tr>
<tr>
<td>7 GP South</td>
<td>14</td>
<td>14/0</td>
</tr>
<tr>
<td>8 GP South</td>
<td>6</td>
<td>6/0</td>
</tr>
<tr>
<td>9 GP South</td>
<td>10</td>
<td>10/0</td>
</tr>
<tr>
<td>10 GP South</td>
<td>11</td>
<td>11/0</td>
</tr>
<tr>
<td>11 GP South</td>
<td>13</td>
<td>13/0</td>
</tr>
<tr>
<td>Docking Station</td>
<td>1</td>
<td>1/0</td>
</tr>
<tr>
<td>Totals</td>
<td>1178</td>
<td>1177/1</td>
</tr>
</tbody>
</table>

9.2. Experience Maps

The experience map is the core global representation that forms the basis for navigation performance. In Figure 15, the experiences are plotted as a circle at the \((x, y)\) coordinates in the stored position \(p\). Linked experiences are joined with a line. The spatial integrity of the map is important for efficient operation of the pruning algorithm, while topological integrity is important for path planning and navigation. The spatial coherence of the maps is evident from a comparison with the floor plans shown in Figures 11 and 12. The maps are not strictly accurate in a Cartesian sense, but need not be for effective navigation. For clarity, the map of GP South in Figure 15(e) has been represented with an offset to the map of Axon. Note that there are no topological connections between the two buildings. Were the maps to overlap, system function including map maintenance would not be affected, due to this lack of connections. Multimedia Extension 1 is a video showing the progression of the experience map over the entire experiment duration.

9.3. Stability

In order to address the question of the long-term stability of the persistent navigation and mapping system, various measures were assessed to investigate trends in performance over time. The first indicator is the average time taken to navigate to each delivery location, plotted against time as in Figure 16. While this figure is obviously dependent on the distance from the random robot starting location and the delivery location, the large number of trials smoothes the data sufficiently to obtain a measure of whether there is any increasing trend in navigation times. The statistics show that there is no significant performance trend.

The second indicator of stability used is the number of experiences and visual templates retained by the system over time, shown in Figure 17. The number of experiences and visual templates in use is stable after the first two hours, although the robot does gradually learn extra sections of the environment. In particular, room 505 (see Figure 11) continued
to build new experiences as the robot visited new areas of the room on subsequent visits. The extended coverage of the room is illustrated by the map evolution shown in Figure 15.

About 1,200 new experiences and 800 new visual templates were learnt in 90 minutes when the robot was moved to GP South (31.5–33 hours), after which the number of experiences reaches a plateau of 2,500. When placed back into Axon building at 35 hours, a small number of new experiences and templates are learned. The drop in numbers at 35 hours is due to map maintenance while the robot was charging on the dock.

A third indicator of system stability is drift in the delivery accuracy over time (Figure 18). Accuracy for delivery goal 3, which was in Room 505, drifts over time, but all other locations are stable. The drift in delivery location in Room 505 is attributable to the continuing growth of experiences in the open room.

Computational resources are constrained to those available onboard the robot and as such computation time cannot afford to grow outside acceptable bounds in the long term. To assess the computational load over time, the time allocated to map correction and other system processes was plotted against time, as shown in Figure 19. Map correction is the last stage of processing in each system iteration, and is performed using whatever spare compute time is available after a sleep period where other system processes are allowed to run. Another indicator of system tractability is the number of map correction processes performed per second, which determines the speed at which the experience map is rearranged. The rate of map correction depends both on spare computational time and the size of the map, and is also shown in Figure 19.

10. Discussion

The problem of persistent navigation and mapping can be approached in a markedly different fashion to the problem of
Fig. 16. Navigate-to-goal durations for the (a) 1,123 navigation trials (deliveries and recharging) in the Axon building and (b) 55 navigation trials in the GP South building. Titles show the regression fit equations, the $P$ values, and the 95% lower and upper ranges on the $n$ coefficient, which include zero for both environments.

SLAM. Where SLAM is concerned with the Cartesian accuracy of human readable maps, persistent navigation and mapping is concerned with reliable goal-directed navigation over long operational periods in constantly changing environments. The performance metrics for persistent navigation and mapping must therefore be concerned with the reliability and stability of navigation performance, rather than Cartesian accuracy of the underlying spatial representations. The results shown in this paper illustrate that consistent, stable, and reliable goal-oriented navigation can be achieved without highly accurate maps. The performance metrics listed in this paper show more than 99.9% navigation reliability, although the ex-
Fig. 18. Accuracy at each delivery location over time for (a) Axon building and (b) GP South building. Only delivery goal 3 showed a degradation in accuracy, likely caused by further mapping of the open space after the first few hours.

Fig. 19. The experience map correction update frequency (top, dashed line) and spare compute time per second allocated to map correction (bottom, solid line). (a) Stage 1 in Axon building. The map compute time and number of updates is fairly stable after the first five hours. (b) Stage 2 in GP South building and stage 3 back in Axon building. Computation stabilizes after about 2 hours in the new environment, and remains similar after the transition back to Axon building at 35 hours.

Experience map that underpins navigation is quite crude in terms of spatial accuracy with respect to ground truth.

10.1. Navigation from Experience Maps

While the experience map lacks in spatial accuracy, it nevertheless contains all of the necessary information for navigating from one place to another. Rather than attempting to map the occupied regions of the environment, the experience map contains the locations that the robot can navigate given the constraints imposed by the local navigation routines. The links between navigable locations contain the necessary information for path planning, showing the sequence in which locations can be navigated, and the time taken to navigate a given sequence. The global spatial information of the experience plays no role.
except to aid pruning, and to help an operator specify delivery locations.

The experience map is driven by the behavior of the local view cells and the pose cells. The local view cells generate an index from a list of remembered views by measuring their perceptual similarity with the robot’s current view. The local view cells are subject to aliasing, where more than one place has a similar appearance, and to recognition failure, where a previously visited place may not be recognized. The role of the pose cells is to robustly filter the sequence of local views in both space and time to provide a best estimate of robot position. Without the pose cells, the noise in local view matching would significantly degrade the topological coherence of the experience map, rendering it unsuitable for global navigation. This filtering is a significant point of difference from many existing appearance-based systems, where the focus is on removing any bad data association in the pre-processing stage.

### 10.2. Maintaining the Experience Map

The nature of the experience map as a set of linked locations provides the flexibility and maintainability required to perform persistent navigation and mapping. New experiences are constantly learnt over time, and linked to previous experiences by persistent navigation and mapping. New experiences are presented a method of pruning based on spatial proximity. The strength of this pruning system is that it requires no semantic knowledge about types of dynamic objects or timescales of change, both of which can vary almost infinitely in real-world environments. If the robot moves through an area of the map that has been pruned, it will learn and integrate fresh experiences which encompass any recent changes to the environment. The weakness is that the system deals rather inefficiently with cyclic changes such as day–night time cycles. Over a full night of operation, the pruning process gradually develops the experience map representation into one suited to localization at night time, somewhat hindering localization in the morning. We would not expect the experience map maintenance and navigation procedures to handle major changes to the topology or geometry of the environment; this capability would require additional maintenance methods, such as initially investigated by Milford et al. (2006). However, while our approach might not suit large-scale geometric change, its performance in real-world office buildings suggests it would be sufficient for a range of indoor environments.

### 10.3. Architecture of Robot Tasks

The architecture of the system presented in this paper provides the robot with two resources for navigation. The global experience map is a resource to plan paths to locations that are beyond the range of the robot’s sensors. The local obstacle map is a resource that captures the relevant aspects of the environment within the immediate range of the robot’s sensors. The recharging task is an excellent example of how the two resources complement one another for long-term autonomous operation without human intervention. The local map provides a robust measure of the location of the recharger with respect to the robot for the purposes of docking, and provides the global map with a location from which the recharger can be approached. The global map and the associated global planning process can be used to bring the robot near the recharger when required, before handing over to the docking process driven by the local map.

The two-step process of using the global map to bring the robot to near the correct location before handing over to processes driven from more immediate sensor information is robust and efficient. It is interesting to note that the complementary nature of the two representations helps overcome the most significant weakness exposed by our study. It was noted that target locations in open environments had a greater tendency to drift in position over time than target locations in more confined spaces. Open environments, on the other hand, are more accessible to local sensing than confined spaces, providing easier detection of targets through the local map representation. Indeed, if the location of the target was updated from the local map each time the robot visited the target, the drift in open environments could be prevented altogether.

### 10.4. Concluding remarks

This paper shows one possible solution to the challenge of persistent navigation and mapping: illustrating the type of world representation that is suited to continual SLAM, how maps can be maintained in the long term, and the map characteristics that enable reliable navigation and task-orientated behavior. We have demonstrated that our mapping algorithm, RatSLAM, can serve as a reliable mapping resource for an autonomous system with validation over a two-week period in multiple environments. Furthermore, our analysis shows that performance and computation remain in proportion to the area mapped, not changing over time in a constantly changing environment, suggesting the potential for operation over much longer periods of months or perhaps even years.

### Appendix A: Delivery Ground Truth

To obtain ground-truth delivery locations, three known landmarks (1, 2, and 3 in Figure A1) were clicked in the panoramic camera image captured at each delivery time. Angular separation between each pair of landmarks (two pairs in total, one pair comprising landmarks 1 and 2, the other landmarks 2 and...
Fig. A.1. The ground truth location of the robot was calculated by clicking on three known landmarks (marked by asterisks in (a) and circles in (b)) in the panoramic image captured at each delivery time.

3) provided a circle of possible robot locations, and the intersection of the two circles provided the robot’s unique location. Two people processed every delivery location independently, and the results were averaged.

Appendix B: Index to Multimedia Extensions

The multimedia extension page is found at http://www.ijrr.org.

Table of Multimedia Extensions

<table>
<thead>
<tr>
<th>Extension</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Video</td>
<td>Experience map evolution</td>
</tr>
</tbody>
</table>

References


