Auton Robot (2013) 34:149–176
DOI 10.1007/s10514-012-9317-9

OpenRatSLAM: an open source brain-based SLAM system

David Ball · Scott Heath · Janet Wiles · Gordon Wyeth · Peter Corke · Michael Milford

Received: 28 May 2012 / Accepted: 29 December 2012 / Published online: 21 February 2013

Abstract RatSLAM is a navigation system based on the neural processes underlying navigation in the rodent brain, capable of operating with low resolution monocular image data. Seminal experiments using RatSLAM include mapping an entire suburb with a web camera and a long term robot delivery trial. This paper describes OpenRatSLAM, an open-source version of RatSLAM with bindings to the Robot Operating System framework to leverage advantages such as robot and sensor abstraction, networking, data playback, and visualization. OpenRatSLAM comprises connected ROS nodes to represent RatSLAM's pose cells, experience map, and local view cells, as well as a fourth node that provides visual odometry estimates. The nodes are described with reference to the RatSLAM model and salient details of the ROS implementation such as topics, messages, parameters, class diagrams, sequence diagrams, and parameter tuning strategies. The performance of the system is demonstrated on three publicly available open-source datasets.

Keywords RatSLAM · OpenRatSLAM · SLAM · Navigation · Mapping · Brain-based · Appearance-based · ROS · Open-source · Hippocampus

D. Ball (✉)· G. Wyeth · P. Corke · M. Milford
School of Electrical Engineering and Computer Science,
Queensland University of Technology, Brisbane, Australia
e-mail: david.ball@qut.edu.au

S. Heath · J. Wiles
School of Information Technology and Electrical Engineering,
The University of Queensland, Brisbane, Australia

M. Milford
e-mail: michael.milford@qut.edu.au

1 Introduction

Appearance-based Simultaneous Localization And Mapping (SLAM) systems have advanced rapidly in recent years (Maddern et al. 2012; Sibley et al. 2010; Cummins and Newman 2010; Milford and Wyeth 2008; Konolige et al. 2008; Konolige and Agrawal 2008; Strasdat et al. 2010; Davison et al. 2007; Newman et al. 2009; Cummins and Newman 2009; Andreasson et al. 2008; Labbe and Michaud 2011). These appearance-based systems rely on the visual similarity between images taken of the environment from discrete locations. At their core, most approaches are based on detecting conventional visual features, such as SIFT (Lowe 1999) or SURF (Bay et al. 2006), extracted from relatively high resolution monocular or stereo vision images. RatSLAM (Milford 2008) is an alternative SLAM system based on the neural processes underlying navigation in the rodent brain, and functions with both low resolution (Milford et al. 2011) or intensity profile visual data (Milford and Wyeth 2008). The system's neural filtering—which builds localization hypotheses by accumulating sensory evidence—enables it to function even with perceptual ambiguity. Consequently, RatSLAM works with a wide range of visual processing systems which would cause some conventional appearance-based SLAM systems to degrade or fail completely. Seminal results achieved using RatSLAM include the vision-only mapping of an entire suburb using only a web camera (Milford et al. 2011; Milford and Wyeth 2008) and a two week long service robot experiment in which a robot performed SLAM and deliveries at all times of day and night over a period of 2 weeks (Milford and Wyeth 2010).

The original RatSLAM codebase has been developed for almost a decade with few software design or sustainability considerations. The code base is relatively large, partly due to

the diverse number of environments and vehicles RatSLAM has been applied to, and also because frameworks and tools such as OpenCV and Robot Operating System (ROS) did not exist. Consequently, the resultant system, while functional for both offline and online applications, is unsuitable for distribution to the wider research community. As a result of increasing interest in the RatSLAM approach, researchers from a range of laboratories (see Table 1) have coded segments of the original system using information provided in RatSLAM research papers, with varying levels of contact with the RatSLAM creators. There are also other partial implementations of RatSLAM components which have been used in research but never released (Smith and Dodds 2009; Kyprou 2009). However, most are incomplete, and only one MATLAB version has been properly released as an open source project. This MATLAB version (Ball 2009) is an offline-only implementation and is too slow to apply in real-time to large environments. Most critically, it doesn't support generic robot frameworks such as the ROS (Quigley et al. 2009) which would allow other researchers to more readily integrate RatSLAM into their existing systems. ROS offers many advantages such as hardware abstraction, transparent online or offline operation, transparent networking and the ability to mix and match user contributed modules.

In this paper we present OpenRatSLAM, an open source, easily reconfigurable modular version of RatSLAM integrated with ROS and capable of online and offline operation. Using OpenRatSLAM we repeat the seminal suburb mapping experiment (Milford and Wyeth 2008), present new results running RatSLAM on Oxford's well known New College dataset (Smith et al. 2009) and on a rat-like robot called the iRat (Ball et al. 2010). The demonstration of an open source rat-based navigation system on a commercially available rat-like robot is of particular relevance to a number of biological laboratories around the world which are conducting interdisciplinary work combining aspects of robotics and neuroscience. The paper and open source software package make the following specific contributions:

- an open source, modular implementation of RatSLAM for both online and offline use, integrated with ROS and its visualization tools such as *rviz*,
- a planning system that returns the quickest path to a user supplied goal position,
- detailed UML class and sequence diagrams showing the inner workings of RatSLAM,
- intuitive and technical explanations of the key parameters that drive the RatSLAM algorithms, and the means to easily change these through a single configuration file,
- a step by step process for tuning the parameters to function optimally for a given dataset,
- visualization scripts for MATLAB that recreate the key figures used to present and analyze the results of an experiment, including analyzing spatial matching errors using ground truth information,
- demonstration of OpenRatSLAM on the *St Lucia 2007* dataset, on the Oxford *New College 2008* dataset and on a new iRat rodent-sized robot platform in a rat-like maze (the Australia *iRat 2011* dataset), with visualization and analysis of the results using the provided MATLAB scripts,

Table 1 Publically available versions of RatSLAM

Institution	Programming language	Description	License
Rowland Institute at Harvard	Python	In progress offline Python port of the algorithms	Undefined
URL	http://github.com/coxlab/ratslam-python		
Originally the University of Queensland, now Queensland University of Technology	MATLAB	Offline only vision and odometry datasets	GNU GPL v3
URL	http://wiki.qut.edu.au/display/cyphy/RatSLAM+MATLAB		
University of Queensland	RobotC	For use on a LEGO NXT brick using sonar sensors	(a) GNU GPL (b) Artistic License
URL	http://code.google.com/p/rsnxt08/		
Queensland University of Technology	**C++ with ROS bindings**	**Online and offline version integrated into ROS. Also includes MATLAB scripts to generate plots for results.**	**GNU GPL v3**
URL	http://code.google.com/p/ratslam/		

This paper describes the release of the final entry in the table (in bold)

- provision of the *St Lucia 2007*, *New College 2008,* and*iRat 2011* datasets in an accessible form ready to be processed with the provided software, and accessory information such as ground truth for the iRat dataset, and
- discussion of several modules currently in development and areas of future work.

The datasets are available online as ROS bag files, which is the standardized method for storing ROS message data. There are a variety of tools to record, play, analyze and visualize the message data in ROS bag files.

The paper proceeds as follows. In Sect. 2 we describe the core RatSLAM components, before describing the software implementation of these components in Sect. 3. Section 4 describes the key system parameters and their effect on system behavior, and provides a parameter tuning process. Section 5 presents the two experimental datasets processed in this paper, with results and the relevant configuration information given in Sect. 6. Section 7 discusses current and future work, with a brief conclusion in Sect. 8.

2 RatSLAM

In this section we describe the RatSLAM algorithms on which OpenRatSLAM is based. RatSLAM is a SLAM system based on computational models of the navigational processes in the *hippocampus*, a part of the mammalian brain. The system consists of three major modules—the *pose cells*, *local view cells*, and *experience map*. We provide an overview of the function of each of these modules, including mathematical descriptions of the major algorithms, in order to contextualize the modular structure employed in OpenRatSLAM. Further technical details on RatSLAM can be found in Milford and Wyeth (2008, 2010).

2.1 Pose cells

The pose cells are a Continuous Attractor Network (CAN) of units (Samsonovich and McNaughton 1997), connected by excitatory and inhibitory connections, similar in characteristics to a navigation neuron found in many mammals called a *grid cell* (Hafting et al. 2005). The network is configured in a three-dimensional prism (Fig. 1), with cells connected to nearby cells by excitatory connections, which wrap across all boundaries of the network. The dimensions of the cell array nominally correspond to the three-dimensional pose of a ground-based robot—x, y, and θ. The pose cell network dynamics are such that the stable state is a single cluster of activated units, referred to as an *activity packet* or *energy packet*. The centroid of this packet encodes the robot's best internal estimate of its current pose. This dynamical behavior is achieved with locally excitatory, globally inhibitory connectivity, described by the distribution ε:

$$\varepsilon_{a,b,c} = e^{-(a^2+b^2)/k_p^{exc}} e^{-c^2/k_d^{exc}} - e^{-(a^2+b^2)/k_p^{inh}} e^{-c^2/k_d^{inh}} \quad (1)$$

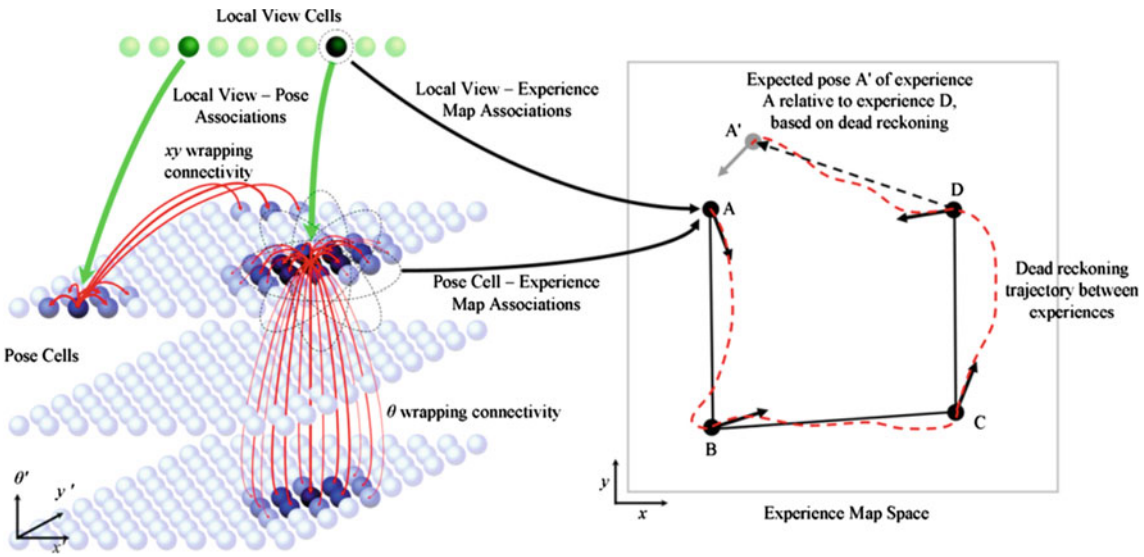

Fig. 1 The major modules of the RatSLAM system reproduced from Milford and Wyeth (2010). The local view cells represent learnt unique scenes in the environment. The pose cells represent the belief about the current pose. The experience map is a topological representation encoding the pose cells and local view cells in nodes and links

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 θ' co-ordinates respectively. The variance constants are fixed as a result of extensive tuning and should not require tuning. Connections wrap across all six faces of the pose cell network, as indicated by the longer arrows in Fig. 1. The change in a cell's activity level ΔP due to internal dynamics is given by:

$$\Delta P_{x',y',\theta'} = \sum_{i=0}^{S_{xy}-1} \sum_{j=0}^{S_{xy}-1} \sum_{k=0}^{35} P_{i,j,k} \varepsilon_{a,b,c} - \varphi \tag{2}$$

where s_{xy} is the side length of the square (x, y) plane of the pose cell network and ϕ is a global inhibition amount.

Self-motion information provided by odometry input shifts activity in the pose cells to represent the robot's movement based on a nominal spatial scale for each pose cell. Injection of activity by local view cells provides a mechanism to perform loop closure. This vision-driven activity injection is one of the critical processes on which system performance depends and tuning details are provided in Sect. 4.

2.1.1 Local view cells

The local view cells are an expandable array of units, each of which represents a distinct visual scene in the environment. When a novel visual scene is seen, a new local view cell is created and associated with the raw pixel data in that scene. In addition, an excitatory link β is learnt (one shot learning) between that local view cell and the centroid of the dominant activity packet in the pose cells at that time. When that view is seen again by the robot, the local view cell is activated and injects activity into the pose cells via that excitatory link:

$$\Delta P_{x',y,'\theta'} = \delta \sum_i \beta_{i,x',y',\theta'} V_i \tag{3}$$

where the δ constant determines the influence of visual cues on the robot's pose estimate. A saturation process ensures each visual template can only inject activity for a short period of time, to avoid spurious re-localizations when the robot is stationary. If a sufficiently long sequence of familiar visual scenes is experienced in the correct sequence, the constant injection of activity into the pose cells will result in re-localization, that is, the dominant activity packet occurring at the same pose as the first time the scene was viewed.

2.1.2 Experience map

Initially the representation of space provided by the pose cells corresponds well to the metric layout of the environment a robot is moving through. However, as odometric error accumulates and loop closure events occur, the space represented by the pose cells becomes discontinuous—adjacent cells in the network can represent physical places separated by great distances. Furthermore, the pose cells represent a finite area but the wrapping of the network edges means that in theory an infinite area can be mapped, which implies that some pose cells represent multiple physical places. The experience map is a graphical map that estimates a unique estimate of the robot's pose by combining information from the pose cells and the local view cells. Each node in the experience map can be defined as a 3-tuple:

$$e_i = \left\{ P^i, V^i, \mathbf{p}^i \right\} \tag{4}$$

where P^i and V^i are the activity states in the pose cells and local view cells respectively at the time the experience is formed, and $\mathbf{p}^i$ is the location of the experience in the experience map space (the space in which graph relaxation is performed).

A new experience is created when the current activity state in the pose cells P^i and local view cells V^i is not closely matched by the state associated with any existing experiences. 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:

$$S^i = \mu_p \left| P^i - P \right| + \mu_v \left| V^i - V \right| \tag{5}$$

where μ_p and μ_v weight the respective contributions of pose and local view codes to the matching score. If $\min(\mathbf{S}) \geq S_{\max}$, a new experience is created, defined by the current pose and local view cell activity states.

As the robot transitions between experiences, a link l_{ij} is formed between the previously active experience e_i to the new experience e_j:

$$l_{ij} = \left\{ \Delta \mathbf{p}^{ij}, \Delta t^{ij} \right\} \tag{6}$$

where $\Delta\ \mathbf{p}^{ij}$ is the relative odometry pose between the two experiences, and Δt^{ij} is the time taken to move between the two experiences. The robot uses this temporal information to plan paths from its current location to a desired goal location. Using Dijkstra's algorithm (Knuth 1977) with an edge path cost set to the stored transition times Δt^{ij}, the system can find the quickest path to a goal location.

A graph relaxation algorithm distributes odometric error throughout the graph, providing a map of the robot's environment which can readily be interpreted by a human. The change in an experience's location is given by:

$$\Delta \mathbf{p}^i = \alpha \left[\sum_{j=1}^{N_f} (\mathbf{p}^j - \mathbf{p}^i - \Delta \mathbf{p}^{ij}) + \sum_{k=1}^{N_t} (\mathbf{p}^k - \mathbf{p}^i - \Delta \mathbf{p}^{ki}) \right] \tag{7}$$

where α is a correction rate constant set to 0.5, N_f is the number of links from experience ei to other experiences

and N_t is the number of links from other experiences to experience e_i.

3 OpenRatSLAM

This section describes the OpenRatSLAM implementation of RatSLAM, which consists of four nodes. This split enhances the modularity of the algorithms and permits each node to run in a separate process, and therefore in a continuous pipeline, making efficient use of multi-core CPUs. The four nodes and the connections between them are shown in Fig. 2.

- Pose Cell Network—This node manages the energy packet that represents pose in response to odometric and local view connections. In this implementation this node also makes decisions about experience map node and link creation.
- Local View Cells—This node determines whether a scene given by the current view is novel or familiar by using image comparison techniques.
- Experience Map—This node manages graph building, graph relaxation and path planning.
- Visual Odometry—For image only datasets, this node provides an odometric estimate based on changes in the visual scene. The other nodes are unaware of the source of the odometric information.

In previous descriptions of RatSLAM, the decision process for creation of nodes and links was handled by the experience mapping module. In OpenRatSLAM, the Pose Cell Network node handles the decision on when to create new nodes and links because it requires knowledge of the internal workings of the Pose Cell Network, which is no longer available due to the split into separate nodes.

The RatSLAM system has several configurable parameters which affect system performance. Parameter descriptions and a tuning process are given in following sections.

3.1 Visual odometry

The Visual Odometry node determines camera motion by comparing successive images. The node makes implicit assumptions that the camera motion is limited to the possible motion of a car, and that the car travels at a relatively constant speed. It allows for separate regions in the image to be specified for determining forward translational and rotational speeds. Full details on how the node determines translational and rotational speeds are provided in Milford and Wyeth (2008).

The rotational velocity is estimated by determining what relative horizontal offset of two consecutive scanline profiles minimizes the mean of absolute differences between the two profiles. Scanline profiles are generated by summing the images in the vertical direction. The translational velocity is estimated by multiplying the minimum difference by a scaling factor, and limited to a maximum value to prevent spurious results from large changes in illumination. This visual odometry implementation is cruder than many other popular packages such as *libvisio2,* and in particular favors simplicity and lack of camera parameterization over accuracy and

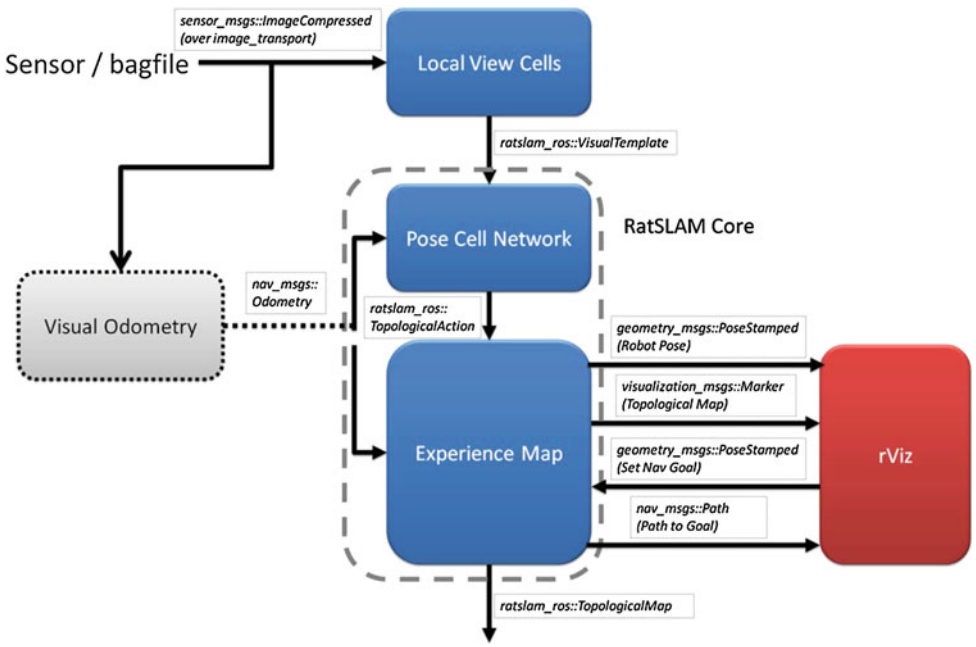

Fig. 2 The node and message structure for OpenRatSLAM for the *St Lucia 2007* dataset configuration. When the odometry is already provided by the dataset or robot, the Visual Odometry node will not be required as indicated by the *dotted lines*. In this case the odometry messages will come directly from the sensors or dataset file. The *St Lucia 2007* dataset is an image only dataset, hence the need for the Visual Odometry node. As the *iRat 2011* and *New College 2008* datasets provide a topic with odometric messages that are derived from the robot's wheel encoders, it does not require the Visual Odometry node

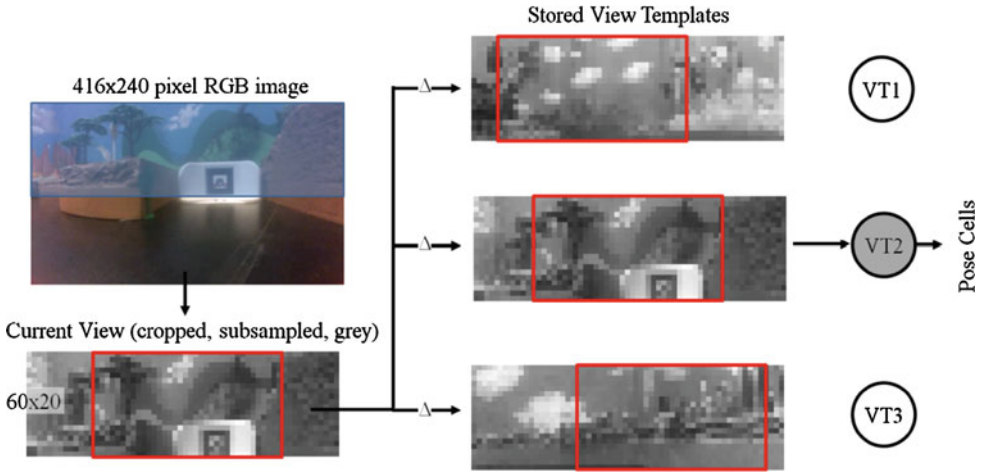

Fig. 3 Place recognition using view templates for the iRat dataset. The *bottom half* of the image is discarded by cropping, as it contains perceptually ambiguous black floor. The Local View node compares the copped, sub-sampled and grayscale current view to all of the stored view templates to find the best match. The delta operator indicates that comparisons are made while shifting the current view and visual templates relative to each other. The result is the currently active view template, which may be a new view template

calibration. However, as demonstrated in Milford and Wyeth (2008) when used with SLAM system it is sufficient to yield a topological map that is representative of the true environment.

The visual odometry node subscribes to a sensor _msgs::CompressedImage topic using the image_transport library and registers a topic that publishes nav_msgs:: Odometry.

3.2 Local view match

The local view match node preprocesses the current image into a *visual template* representation and then decides if this is a new or previously seen visual template. The following preprocessing steps, also shown in Fig. 3, produce the visual template.

1. If required, the image is converted into a mono grayscale format.
2. The image may then be cropped to bias the templates towards visually interesting areas of the camera images. Visual interesting areas are those that will bias towards areas that are visually unique across the environment. For example, this process is used to remove visually bland features such as roads.
3. The cropped region may then be subsampled to defined height and width parameters. The scanline intensity profile is a generalization by subsampling with the height parameter equal to one.
4. The subsampled region may then undergo global and local normalization steps which attempt to alleviate changes in illumination. Global normalization considers the mean and range of the entire image and addresses global changes in illumination. Local normalization preserves contrast in small patch regions by subtracting from each pixel the mean intensity of the surrounding patch region and then dividing by the patch's pixel intensity standard deviation (Zhang and Kleeman 2009).

After pre-processing, the local view match node compares the visual template that represents the current camera image with all previously learnt templates. A similarity measure based on the Sum of Absolute Differences (SAD) between the current visual template and each previously learnt visual template is calculated. If the smallest difference is less than a threshold, then the corresponding learnt template is selected. Otherwise, the current visual template is added to the database of templates.

For forward facing cameras, the comparison process handles small rotational offsets by finding the minimum SAD while shifting the stored templates relative to the current view in the horizontal direction by a parameterized amount. For the panoramic camera's views SAD is calculated while shifting the visual templates and current view through a full rotation.

The local view match node subscribes to a topic that publishes sensors_msgs::CompressedImage (using ROS's image_transport library) and publishes a custom ratslam_ros:: ViewTemplate message that is listed below. This custom message includes a header and template id, where the template id refers to the currently active local view cell. The message also includes a relative angle, which for panoramic images, is the angle between the original stored template and the agent's current angle.

ViewTemplate.msg

```
Header header

uint32 current_id
float64 relative_rad
```

3.3 Pose cell network

The pose cell network node responds to two types of input; odometry and view templates. This input is received as ROS messages.

The action on a view template input depends on whether this is a new or existing view template. For new view templates the id is associated with the centroid of the current peak activity packet in the pose cell network. For existing view templates, activity is injected into the previously associated location in the pose cells. The injected activity for consecutive matches of the same view template decays rapidly but is gradually restored over time. Because RatSLAM has no explicit motion model, this decay process is necessary to avoid potentially incorrect re-localizations when the robot is motionless for long periods of time.

For odometry input these steps are performed in the following order. Note that the Pose Cell node is unaware of the source of the odometry input, and for example could be from visual odometry or wheel odometry, and either from the dataset file or calculated live. Also note that each of the flowing steps account for the wrapping across each face of the pose cell network rectangular prism.

1. Local excitation where energy is added around each active pose cell.
2. Local inhibition where energy is removed around each active pose cell. These first two steps ensure the stabilization of the energy packets.
3. Global inhibition where energy is removed from all active pose cells but not below zero.
4. Network energy normalization to ensure the total energy in the system is equal to one. This stage ensures stability of the global pose cell system.
5. Use the odometric information for path integration by shifting the pose cell energy.
6. Identify the centroid of the dominant activity packet in the network.

After performing these steps, the node must determine an action for the experience map's topological graph. The possible actions are: create a new node (which implicitly includes creating an edge from the previous node), create an edge between two existing nodes or set the location to an existing node. This action is sent within a new custom message, ratslam_ros::TopologicalAction.

TopologicalAction.msg

```
# actions
uint32 CREATE_NODE=1
uint32 CREATE_EDGE=2
uint32 SET_NODE=3

Header header

uint32 action

uint32 src_id
uint32 dest_id

float64 relative_rad
```

3.4 Experience map

The Experience Map node uses the received actions to create nodes and links, or to set the current node. In this implementation, each experience has an associated position and orientation. Creating a new node also creates a link to the previously active node. Graph relaxation is performed on each action interaction. A link encapsulates the pose transformation and time between nodes based the odometric messages. The agent's state is given by the current experience map node and the agent's rotation relative to the node's orientation.

The experience map node publishes three messages to expose the state of the experience map. The first is a complete representation of the experience map as a topological map in a custom message as described below, which consists of a list of nodes and edges. The second is the pose of the robot within the experience map. The third message is a Marker message suitable for rendering the map within *rviz*.

The experience map node will also generate a path to a user supplied goal pose message, connecting with the *rviz 2D goal nav* command. The experience map node responds with a Path message which includes the path from the robot to the goal. Generating the path works transparently in offline and online mode. The node and edge count parameters are included to address the limitation in using ROS's *rostopic* command to export the data in a plotting friendly

format where no information is provided on the size of the array.

TopologicalMap.msg

Header header
uint32 node_count
TopologicalNode[] node
uint32 edge_count
TopologicalEdge[] edge

TopologicalNode.msg

uint32 id
geometry_msgs/Pose pose

TopologicalEdge.msg

uint32 id
uint32 source_id
uint32 destination_id
duration duration
geometry_msgs/Transform transform

3.5 Visualization

There are several options available to visualize the live state of the OpenRatSLAM system, as shown in Fig. 4. As shown in Fig. 4e, the ROS tool *rviz* can be used to visualise the experience map, robot's pose and (not shown in the figure) the path from the robot to a goal. Apart from the visual odometry node, each node has an associated visualization. Although not shown in the figure, we use the ROS command *rxplot* for visualizing the live state of variables. In particular, we use this tool to visualize the growth of templates and experiences to help investigate false loop closures.

3.5.1 Post visualization

We provide one shell script file and four MATLAB script files for post experiment visualization and analysis. During processing, the messages sent from the OpenRatSLAM nodes can be recorded into ROS bag files. This recording is enabled by default in the included ROS launch files. The shell script file is used to export the recorded messages in a MATLAB readable format. A brief description of the MATLAB scripts is given in Table 2, while more extensive comments can be found in the body of the script. The files provide the means to visualize many of the key figures provided in RatSLAM research papers (Milford and Wyeth 2008; Milford et al. 2011; Milford and Wyeth 2010).

3.6 Software architecture

This section describes the technical software engineering details of the ROS nodes that comprise OpenRatSLAM and the interactions between these nodes at both a message passing and major function call level of detail. Figure 5 shows a UML diagram of the classes, associate wrapper interface and underlying data structures. Figures 6 and 7 contain sequence diagrams that show the creation of new experience nodes and links through the interaction between OpenRatSLAM nodes. The three main classes (LocalViewMatch, PosecellNetwork, and ExperienceMap) are independent of each other and may be individually compiled without including ROS or any other OpenRatSLAM components. ROS interfaces to each class are then provided as abstractions, allowing OpenRatSLAM to interact with other ROS packages. Callback functions provide a ROS wrapper interface to each class, so that the classes do not depend on ROS. Each class and associated ROS interfaces are compiled into an executable ROS node.

3.7 Code specifications

The OpenRatSLAM source is released under the GNU GPL version 3 license and is available at Google code (http://code.google.com/p/ratslam/). In the repository is an implementation with bindings for ROS and another version with bindings for OpenCV. The technical description in this paper describes only the ROS version. The ROS code is written in C++ and uses the Irrlicht engine (http://irrlicht.sourceforge.net) for rendering and the boost library for configuration. The code has been tested against ROS Electric and Fuerte on Ubuntu 11.10 and 12.04. Full instructions for downloading, compiling and running the code are described in Appendix B.

4 OpenRatSLAM parameters and tuning

In this section we list the names of all the key system parameters and give an intuitive description of their purpose and the effect of changing their value. These parameters are parsed using boost's parameter tree library. Using the parameter tree library over the ROS parameter is required to maintain the multiple bindings between interface and the core RatSLAM code. The parameters are listed in Table 3.

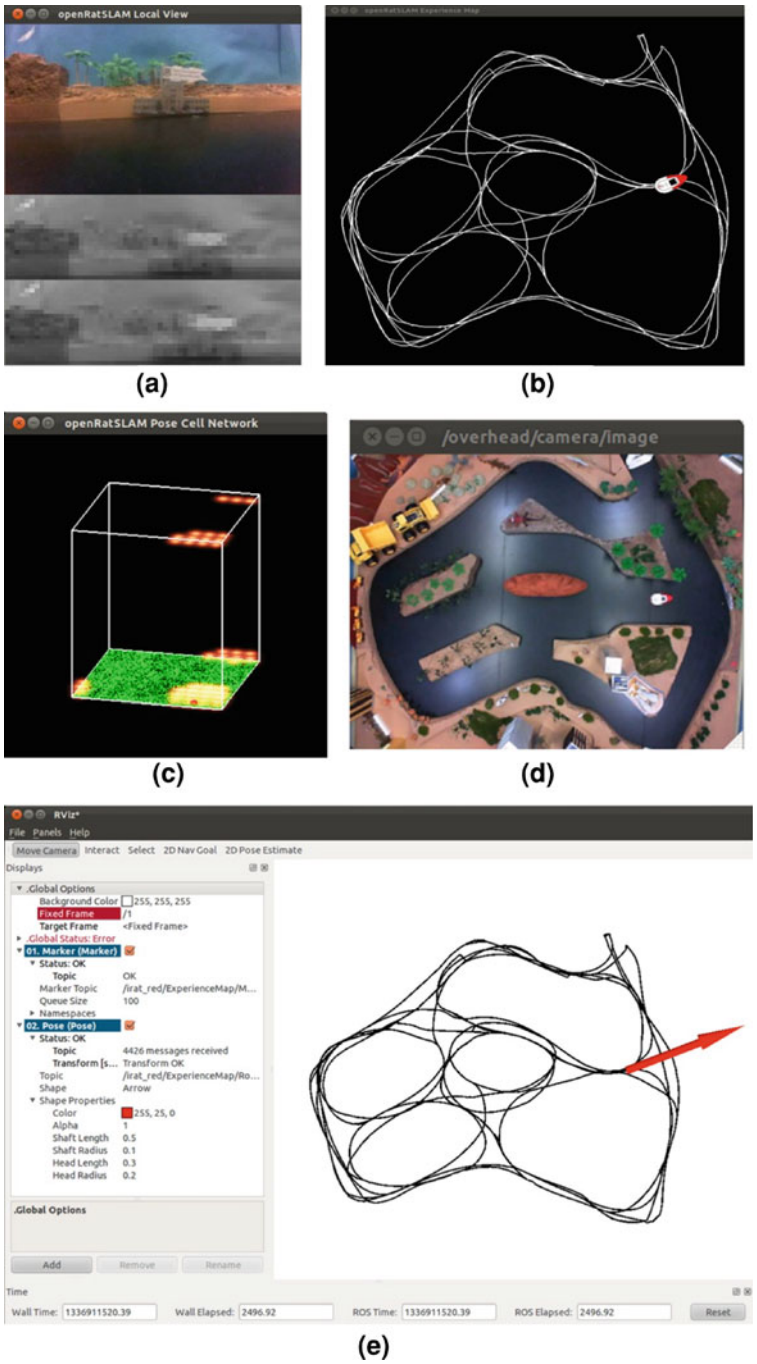

Fig. 4 Screenshots of OpenRatSLAM in action. **a** Local View Cells showing the robot's camera image and current-matched template pair, **b** Experience Map, **c** Pose Cell Network showing the activity packet wrapped across the prism faces, and tracking of the activity packet's centroid movement in the (x, y) plane in *green*, **d** overhead image rendered by ROS image_viewer, **e** the topological map and pose rendered by ROS *rviz*. The *red arrow* shows the pose of the robot which matches the robot's location in the overhead image and in the experience map (Color figure online)

Table 2 Offline post-processing scripts

Script name	Description
export_bagfile.sh	A shell script that exports the view template, topological action, robot pose and map messages out of the recorded ROS bag data file given an input file and topic root. The data is exported in a format suitable for the following MATLAB scripts to process
show_id.m	Plots the active experience and visual template ID against time over the course of the experiment
show_em.m	Plots the experience map on an (x, y) plane. Run once to generate indices into the large map.dat file, then run again to generate the actual plot
plot_matches.m	Plots any false positive matches on a ground truth plot and provides statistical data on the false positive matches including mean and maximum error and number. Synchronizes any asynchronous logging timestamps
frame_matches.m	Extracts the frame match pairs from the video using the reported visual matches, visualizes them side by side and saves to separate numbered image files which can then be imported into video creation software such as Virtualdub

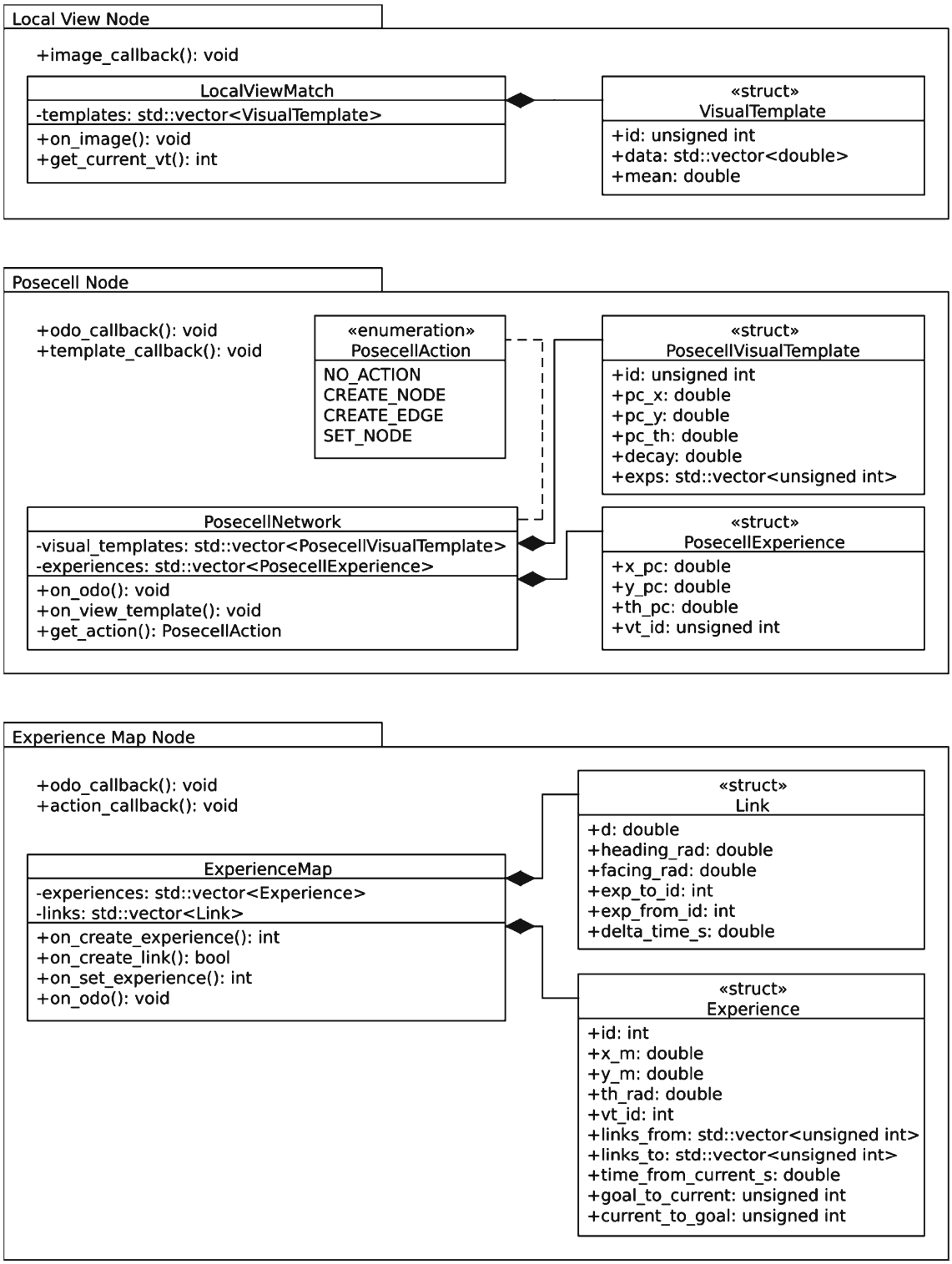

Fig. 5 UML diagram of the classes and structures of the three ROS nodes comprising OpenRatSLAM. Each node has a ROS callback wrapper around its class for incoming messages. This wrapper abstracts ROS away from the OpenRatSLAM classes. The diagram shows each node's data structures in detail. The ExperienceMap class has the Links and Experiences stored in separate vectors for performance reasons. This diagram serves as a reference for the signal flow diagrams in Fig. 6

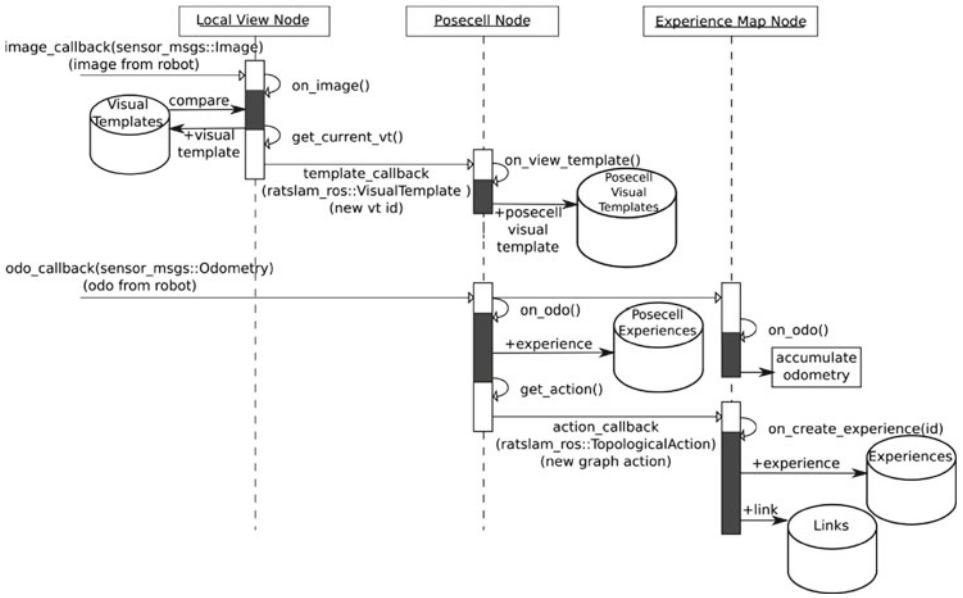

Fig. 6 Sequence diagram showing how interactions between OpenRatSLAM nodes create a new experience. Data flow through the three nodes is initiated by asynchronously arriving image and odometry ROS messages. The ROS callbacks process the messages (shown as *solid white rectangles*) and then transfer them to the OpenRatSLAM class methods (*solid black rectangles*). *Cylinders* represent data storage that relate to the structure shown in the previous UML diagram. *Plus signs* indicate new data is being added to storage. The creation of a new experience starts with the creation of a new visual template, followed by the creation of pose cell representations for both new visual templates, and lastly the creation of a new node and link to the previously active node in the experience map

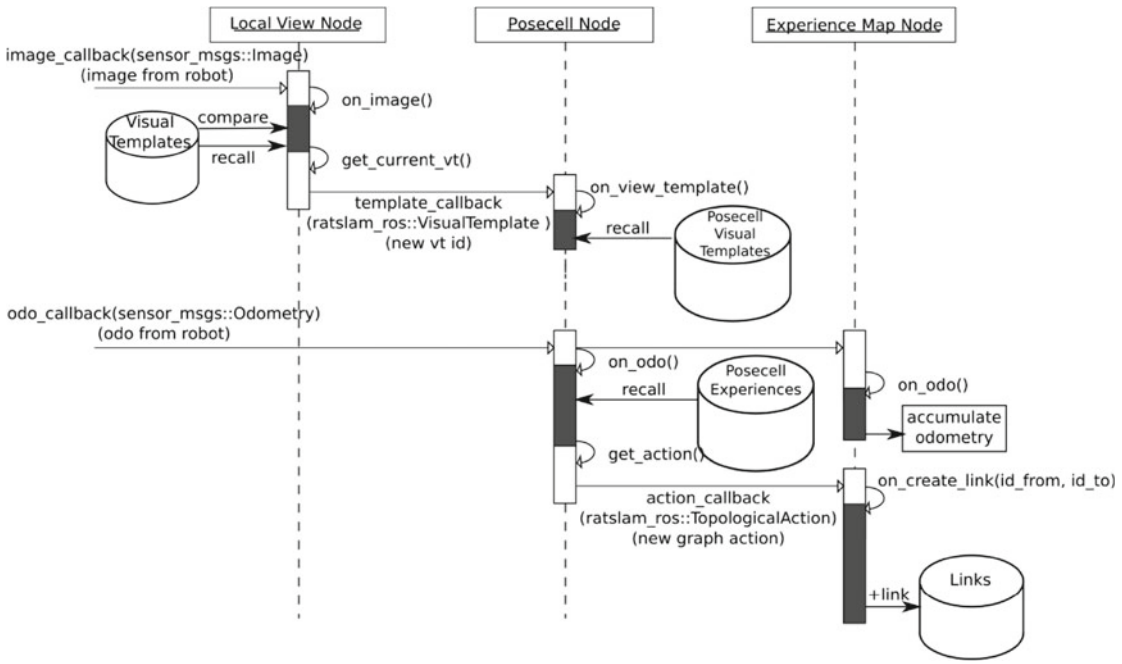

Fig. 7 Sequence diagram showing new link creation between existing experiences through interactions between OpenRatSLAM nodes. This process is similar to sequence shown in Fig. 6. The major difference is that the current view is matched to a previously learnt visual template, which activates a previously learnt experience node, in this case creating a new link. Note that the matching of a previously learnt visual template doesn't guarantee that a new link is created

Table 3 Parameter descriptions

Parameter name	Description and properties
Visual odometry	
[vtrans_image_x_min, vtrans_image_y_min], [vtrans_image_x_max, vtrans_image_y_max]	These four parameters allow the specification of the cropping region for translational velocity
[vrot_image_x_min, vrot_image_y_min], [vrot_image_x_max, vrot_image_y_max]	These four parameters allow the specification of the cropping region for rotational velocity
camera_fov_deg	The horizontal camera field of view which is used to scale the rotational velocity
camera_hz	The camera frame rate which is used to scale the velocities by accounting for the time between frames
vtrans_scaling	This parameter directly scales the translation velocity into meters per second
vtrans_max	This parameter limits the maximum translation velocity to handle large changes in illumination
Local view parameters	
vt_panoramic	Set this to 1 if the images are panoramic
vt_shift_match	The range (in pixel units) of horizontal offsets over which the current image is compared to all learnt image templates. Unused in panoramic mode
vt_step_match	The number of pixels to increment the shift match offset
[image_crop_x_min, image_crop_y_min], [image_crop_x_max, image_crop_y_max]	These four parameters allow a cropping region of the original camera image to be specified. Cropping is a useful tool for specifying image regions that are salient for place localization. For example, carpet or road can be removed from the image. Note these are defined from the top left of the image
[template_x_size, template_y_size]	The horizontal and vertical size in pixels of the 'subsampled' template that represents the camera view. For a single intensity profile set *template_y_size* to 1
vt_match_threshold	The sensitivity parameter that determines the boundary between the current visual scene being considered novel and being matched to an already learnt visual template
vt_normalisation	All templates are normalized by scaling their mean to this parameter. This addresses global changes in illumination. Note that values are clipped between zero and one
vt_patch_normalisation	This effectively increases the local contrast of patch regions across the current view to handle local changing light conditions and bring out more image details. The parameter sets the size of the patch in pixels from its centre
Pose cell parameters	
pc_dim_xy, s_{xy}	The side length of the square (x, y) plane of the pose cell network. The larger the network size, the greater the computation, but the lower the likelihood of a hash collision in the pose cell network and local view cells resulting in a false positive loop closure in the experience map
exp_delta_pc_threshold	The radius within the pose cell network which can be associated with a single experience—if the centroid of the pose cell activity packet moves more than this distance a new experience is generated, *regardless of whether the visual scene has changed*
pc_cell_x_size	A scaling factor that can be adjusted to suit the translational velocity range of the robot or sensor platform. For efficiency reasons the pose cell dynamics do not scale ad infinitem and as such this parameter can be adjusted to ensure the pose cell network is within its normal operating range. The normal operating range for this implementation of OpenRatSLAM is to limit the movement of the energy in the network to one cell per iteration
pc_vt_inject_energy	Determines the amount of energy that is injected into the pose cell network when a familiar visual scene is recognized. Setting this to a very high value ensures one shot localization but makes the system brittle to false positive visual matches. Conversely, setting this parameter low means the system is very robust to false positive matches but may require long sequences of familiar visual input in order to perform loop closure
vt_active_decay	A local view cell saturation mechanism uses this parameter to rapidly attenuate the amount of activity that is injected by repeated exposure to the same visual scene. This mechanism ensures the robot is less likely to perform false positive loop closures while stationary. The higher the value, the shorter period of time over which a single local view cell will inject activity into the pose cell network, and the longer the sequence of *different* visual matches required to perform loop closure
pc_vt_restore	Determines the rate at which a local view cell is restored to its original state after being attenuated due to repeated activations
Experience map parameters	
exp_loops	The number of complete experience map graph relaxation cycles to perform per system iteration

4.1 Parameter tuning

The current RatSLAM implementation is the result of years of tuning, especially of the pose cell network parameters, in order to generate stable system dynamics that appropriately filtered ambiguous visual sensory input. The parameters given for the experiments performed in this paper serve as a good basis for experiments in large outdoor and smaller indoor environments. Most importantly, the core pose cell parameters do not need to be changed unless one is interested in investigating the effect of varying network dynamics. In general, the only parameters that *may* need tuning from the defaults given in this paper are the *vt_match_threshold* (M) and, in some circumstances, *pc_vt_inject_energy* (δ).

Here we outline a brief general tuning procedure for producing functional OpenRatSLAM performance for a generic dataset. The following process first ensures that the velocity scaling constant V_s is approximately correct, before tuning the relative rate at which new experiences and visual templates are generated to be between 1:1 and 2:1, and finally ramps up the amount of activity injected into the pose cells by an active visual template until loop closure is achieved over the shortest distance the user expects the system to close a loop.

Parameter Tuning Process

Pick any "in motion" segment of a dataset

$R = \dot{T}_{ID} / \dot{E}_{ID}$ where $\left(\dot{T}_{ID}, \dot{E}_{ID}\right)$ are the average rates at which templates and experiences are generated

M = use the vt_match_threshold value in the dataset with most similar visual processing paradigm (St Lucia 2007 or AusMap iRat 2012). Noting that this parameter will scale with the normalization until clipping.

run OpenRatSLAM

if R << 0.5 or R >> 1

  adjust V_s

endif

while $0.5 < R < 1$

  run OpenRatSLAM

  if R < 0.5 **then**

    decrease M

  else

    increase M

  endif

end while

Select a segment of the dataset which involves the shortest loop closure you want the system to successfully close.

δ = delta threshold change initialized to low value or zero

do

  run OpenRatSLAM

  increase δ

while no loop closure

5 Using OpenRatSLAM

In this section we describe the experimental setup for the three datasets. These datasets are available online as ROS bag files.

5.1 St Lucia Suburb Dataset

The *St Lucia 2007* dataset (Fig. 8) was first used in Milford and Wyeth (2008) and has since been used in other biologically inspired or GIST-based mapping such as Sunderhauf (2012), Sunderhauf and Protzel (2010). The dataset consists of web camera footage from a 66 km car journey around an Australian suburb, through a wide range of environment types (Fig. 8b–d) at a variety of speeds. The bag file for this dataset contains the large majority of the original video as a set of compressed images. The dataset is slightly trimmed at the start and end to remove the operators working on the computer and a particularly washed out brightly white section due to massive illumination changes.

5.2 Oxford's New College 2008 dataset

The New College dataset is a well known dataset from the Oxford University taken in England in 2008 (Smith et al. 2009). The full dataset includes, laser, odometry, stereo camera images, panoramic images, and GPS recordings in a custom format. Data collection was performed outdoors on the 2.2 km path shown in Fig. 9 using a Segway RMP200 robot. In order to run the dataset with OpenRatSLAM the panoramic images and odometric information have been re-encoded into a ROS bag file. Timestamps were extracted from the original dataset to ensure proper timing. The odometric information has been integrated to match the panoramic image rate of 3 Hz.

5.3 iRat 2011 Australia dataset

The iRat—intelligent Rat animat technology—is a small mobile robot that has a similar size and shape to a large rodent (Fig. 10). The robot has differential wheel drive, a forward facing wide field of view camera, speakers and microphone, IR proximity sensors, and wheel encoders. It also has an onboard 1 GHz ×86 256 MB RAM computer running Ubuntu and 802.11 g/n WLAN. The iRat has center and wall following algorithms that allow it to autonomously explore this environment. For this dataset, the iRat's exploration was guided by a human who gave directives on which way to turn at each intersection.

The dataset was obtained while the iRat explored a road movie set based on Australian geography, contain-

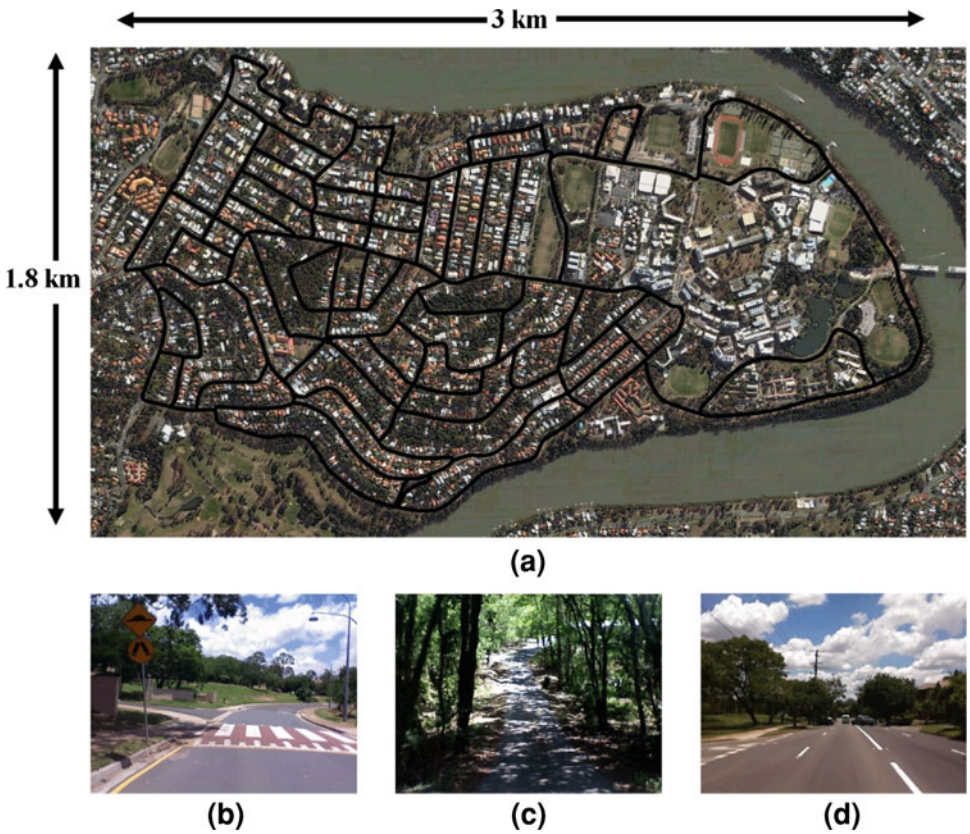

Fig. 8 **a** The *St Lucia 2007* Dataset, and **b** representative frames from the video dataset showing the variety of places encountered in the dataset. Imagery ©2012 Cnes/Spot Image, DigitalGlobe, GeoEye, Sinclair Knight Merz & Fugro, Map data ©2012 Google, Whereis(R), Sensis Pty Ltd

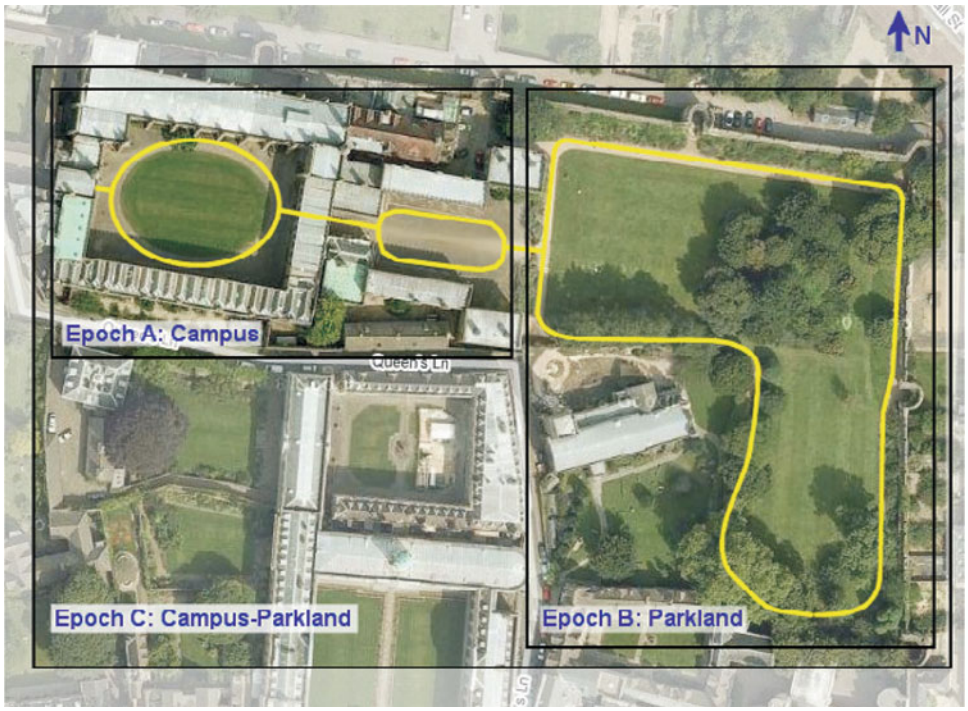

Fig. 9 This figure shows the Oxford New College dataset. The path the robot follows is marked in *yellow*. Image reproduced from Smith et al. (2009) (Color figure online)

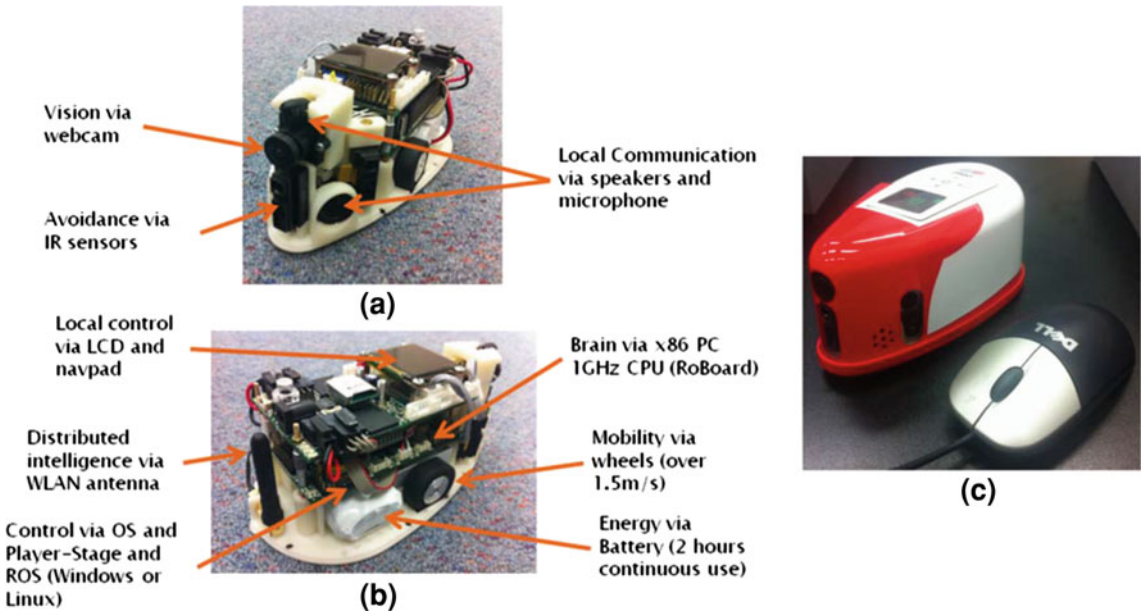

Fig. 10 **a–b** The labeled iRat robot internals. **c** The iRat alongside a standard computer mouse to show scale

ing prominent Australian landmarks such as the Sydney Opera House and Uluru. A camera mounted overhead provided images that allowed us to extract ground truth information. The iRat ROS bag dataset is ∼16 min long and includes the iRat's camera images (shown in Fig. 11b–c, range and odometry messages, the overhead camera's images (shown in Fig. 11a) and tracked pose information.

6 Experimental results

In this section we present experimental results for the two datasets. Table 4 provides the values of the key parameters that we used to generate the results. Parameters relating to visualization are not listed here as they do not influence the map. All figures and results are generated using the MATLAB

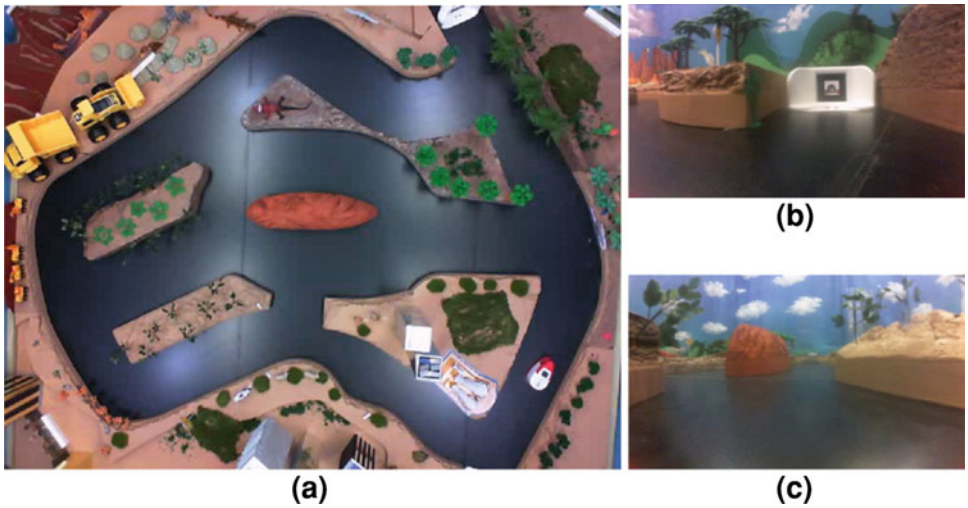

Fig. 11 The *iRat 2012* dataset **a** overhead view and **b**–**c** sample frames from the onboard camera

Table 4 Parameter values

Parameter	St Lucia	New College	iRat
image_crop_x_min	40	N/A (0)	N/A (0)
image_crop_x_max	600	N/A (image width)	N/A (image width)
image_crop_y_min	150	N/A (0)	N/A (0)
image_crop_y_max	300	N/A (image height)	150
template_x_size	60	60	60
template_y_size	10	10	20
vt_panoramic	0	1	0
vt_shift_match	5	N/A	4
vt_step_match	1	1	1
vt_match_threshold	0.085	0.054	0.03
vt_active_decay	1.0	1.0	1.0
vt_normalisation	0.4	N/A (0)	N/A (0)
vt_patch_normalisation	N/A (0)	2	N/A (0)
pc_dim_xy	30	30	11
pc_vt_inject_energy	0.15	0.2	0.1
pc_cell_x_size	2.0	1.0	0.015
exp_delta_pc_threshold	1.0	2.0	2.0
exp_loops	20	100	20
exp_initial_em_deg	N/A (90)	N/A (90)	140
vtrans_image_x_min	195	N/A	N/A
vtrans_image_x_max	475	N/A	N/A
vtrans_image_y_min	270	N/A	N/A
vtrans_image_y_max	430	N/A	N/A
vrot_image_x_min	195	N/A	N/A
vrot _image_x_max	475	N/A	N/A
vrot _image_y_min	75	N/A	N/A
vrot _image_y_max	240	N/A	N/A
camera_fov_deg	53	N/A	N/A
camera_hz	10	N/A	N/A
vtrans_scaling	1000	N/A	N/A
vtrans_max	20	N/A	N/A

script files accompanying this paper. For each dataset we present an experience map plot, a template-experience ID plot, showing the growth over time of visual templates and experience nodes, and sample scene matches. In addition, for the iRat dataset, we show a false positive plot overlaid on ground truth, identifying minor place recognition errors in the vision system.

6.1 St Lucia 2007 dataset

Figure 12 shows the evolution of the experience map for the *St Lucia 2007* dataset, for intervals of one quarter of the dataset, ending in the final map show in Fig. 12d.

Figure 13 shows a graph of the active experience and visual template over the duration of the experiment. Experiences are learnt at approximately double the rate of visual templates, consistent with the tuning process indicated in Sect. 4.1. Figure 14 shows that the simple method of comparing images works despite traffic.

6.2 Oxford New College dataset

Figure 15 shows the evolution of the experience map over time for the *New College 2008* dataset. The final experience map matches approximately to the ground truth path marked in yellow on Fig. 9. While the map is topologically correct, the rightmost loop, Epoch B—Parkland, is twisted relative to the rest of the map. This twist is because of the reliance on odometry through a single path through a gate that provides the only link between the two map sections. The experience map shows that the Local View panoramic matches during traversals of the same path in different directions.

Figure 16 shows a graph of the active experience and visual template over the duration of the experiment. As in the St Lucia dataset, experiences are learnt at approximately double the rate of visual templates, consistent with the tuning process indicated in Sect. 4.1. The forward backward matching of the panoramic images can be seen by the segments of increasing and decreasing visual templates and experiences. An example of panoramic image matching is shown in Fig. 17.

6.3 iRat 2011 Australia dataset

Figure 18 shows the evolution of the experience map over time for the *iRat 2011* dataset. Although the iRat has wheel encoders, the nature of the maze floor surface means that wheel odometry is error prone, in particular due to wheel slip, and this is apparent when looking at the shape of the map before and after loop closures. Once the entire environment has been explored, the map becomes more stable and there are only minor adjustments (Fig. 18c–d).

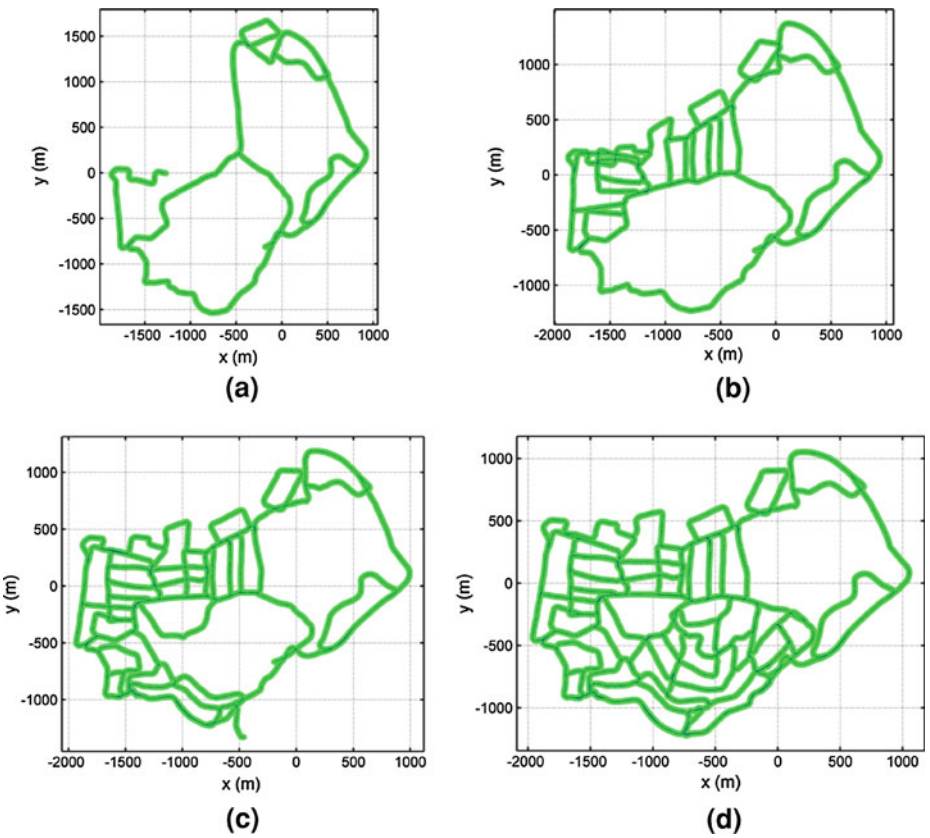

Fig. 12 Experience map, showing the topological map at regular intervals up to the (**d**) final map. The final map is similar to that in the original paper (Milford and Wyeth 2008) and the map in Fig. 8a

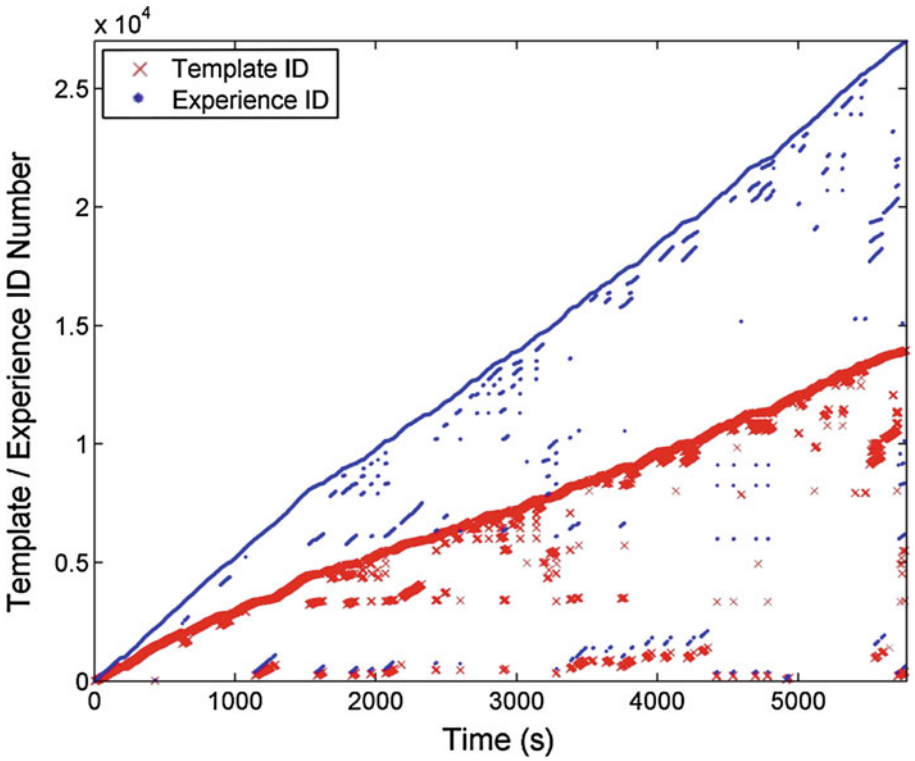

Fig. 13 Graph of the active experience (*blue*) and visual template (*red*) over the duration of the experiment. Experiences are learnt at approximately double the rate of visual templates. The graph shows the continuous creation of new experiences and view templates as the upper 'bounding line'. The *graph* shows the recognition of previously learnt experiences and view templates as the short segments under the bounding line. The beginning and end of the experience segments correspond to where new links, typically loop closures, will be added between the previous experience and a previously learnt experience (Color figure online)

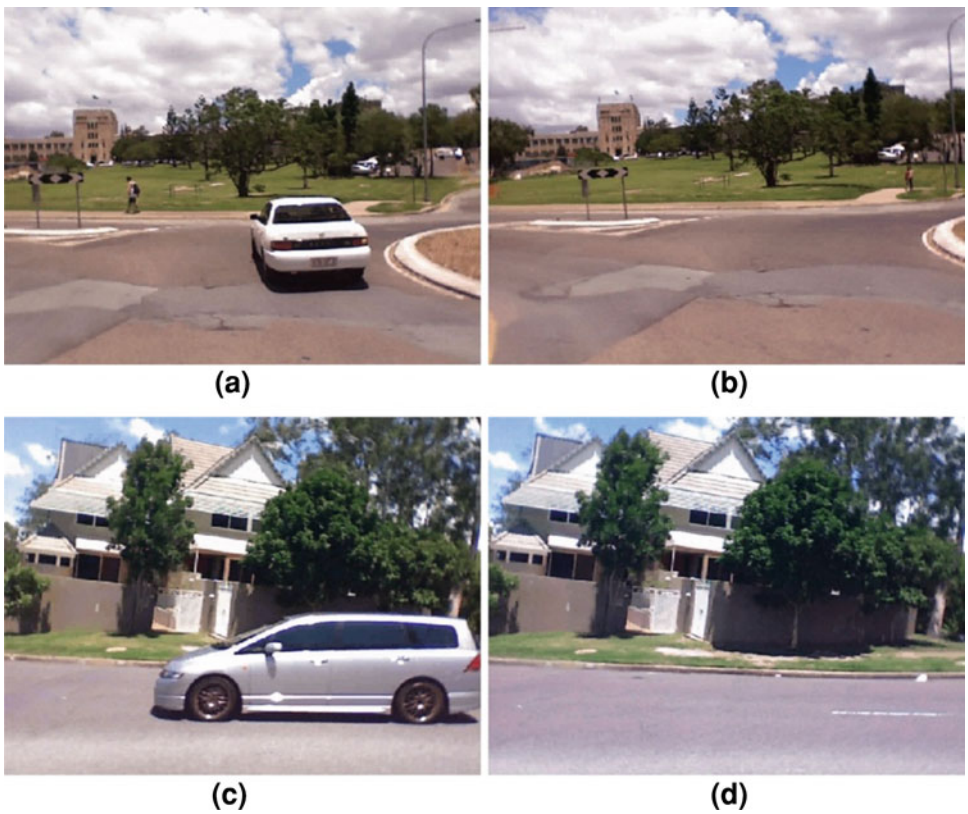

Fig. 14 Sample frame matches from the St Lucia dataset, as output by the vision system. Scenes are recognized despite traffic and the simple visual matching process

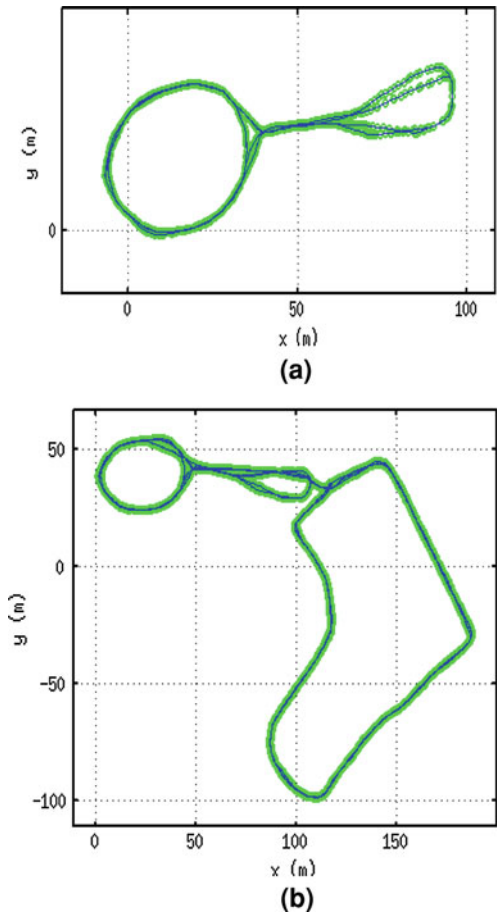

Fig. 15 Experience map, **a** showing the topological map after epoch A then **b** at the end of the experiment. The final map is similar to the ground truth map, except for a twist at the single entry point between the large loop on the *right* and the rest of the map

Figure 19 shows a plot of the active experience and visual template over the duration of the experiment. In the later stages of the dataset there are multiple quite short sequences of image template matches, caused by the robot's path taking it only momentarily over a previously traversed path (Fig. 20).

The final map is similar to the ground truth trajectory shown in Fig. 21. Figure 21a shows a plot of false positive image template matches, generated using the ground truth data and MATLAB script files provided with this paper. For illustrative purposes we have set the error threshold to only

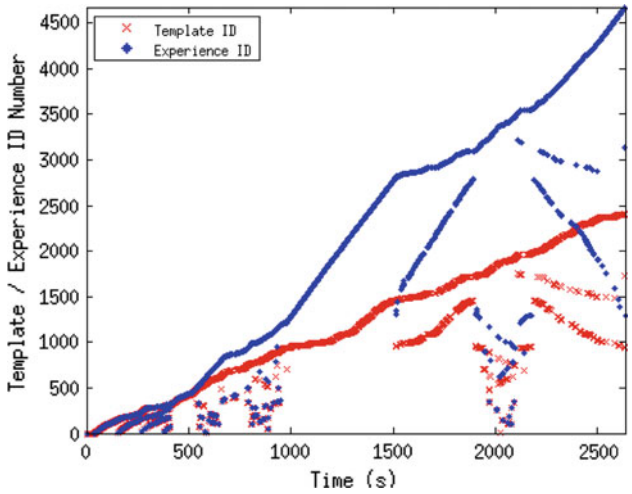

Fig. 16 Graph of the active experience and visual template over the duration of the experiment. OpenRatSLAM has learnt approximately double the number of experiences as visual templates. See the Fig. 13 caption for a full explanation of the *graph*. Specific to this *graph* due to the panoramic matching are the segments of decreasing IDs which show the robot traversing in the other direction along a previously learnt path

Fig. 17 Sample frame matches from the *New College 2008* dataset. The panoramic images have been matched despite at different orientations

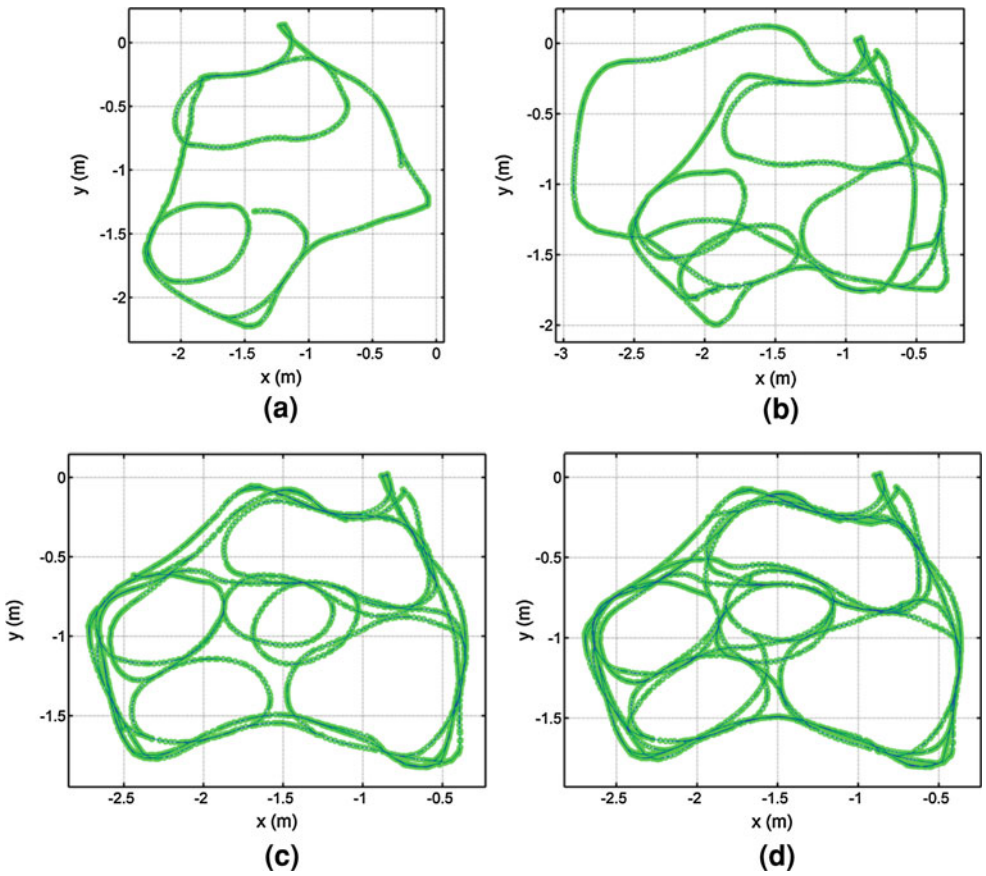

Fig. 18 Experience map evolution for the AusMap iRat 2012 dataset. The wheel odometry is poor due to the slippery floor and regularly leads to large localisation errors which are corrected by loop closures

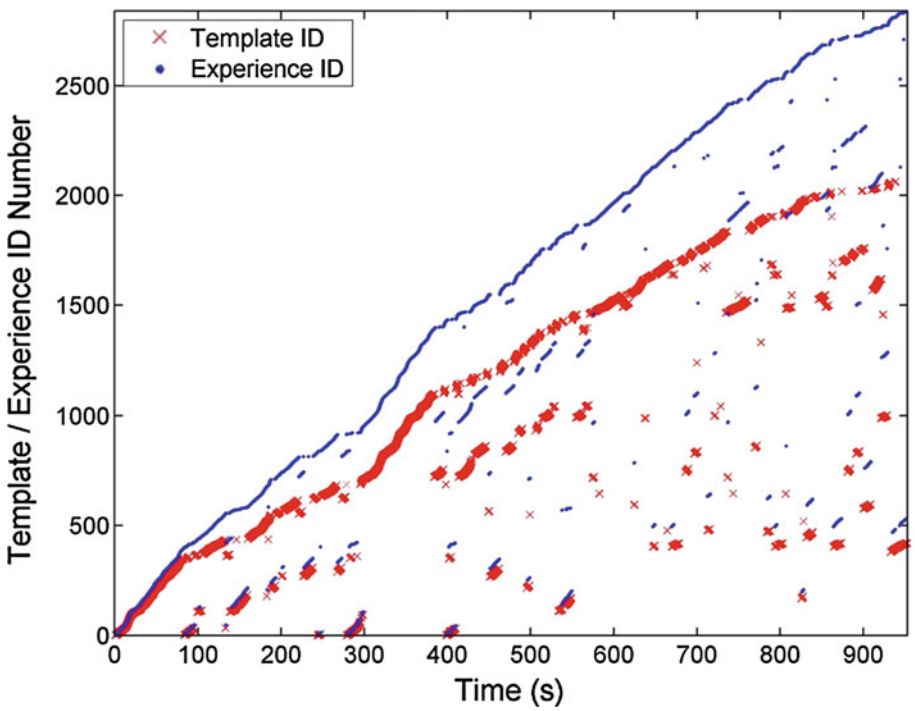

Fig. 19 Graph of the active experience and visual templates over the duration of the experiment. See the Fig. 13 caption for a full explanation of the *graph*

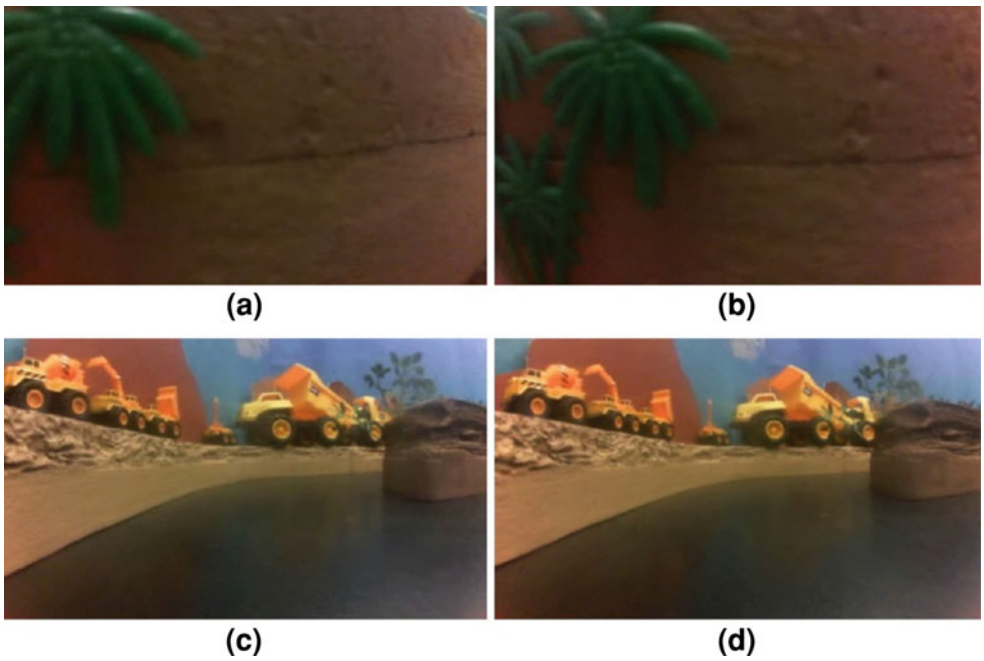

Fig. 20 Sample frame matches from the *iRat 2011* dataset, as output by the vision system. **a**–**b** Turning very close to a wall and **c**–**d** moving down a slightly curved corridor. There were no significant false positives in the this dataset (see Fig. 21)

0.05 m (the robot is 0.18 m long)—each of the thicker solid black lines indicates the 'false' match between two places as suggested by the visual template system. The zoomed in section in Fig. 21b shows how the visual template system has some local spatial generalization—the place where the image template was first learned is matched over a short range of places during subsequent visits to that place. The MATLAB script files also provide general statistics for the dataset—there were 575 'false' positives out of 16656 frames analyzed, but the maximum error was only 0.13 m and the mean was 0.063 m. This mean error is approximately the same as the iRat's width.

6.4 Compute

The datasets presented in this paper can be processed at real-time speed or faster by the OpenRatSLAM system on a standard modern desktop computer. The current implementation scales linearly with the number of image templates and experience nodes in the experience map (experience maps are sparsely connected leading to only linear growth in the computation required).

Base runtime performance is primarily dependent on the following three parameters. The first is that the size of the visual template effects the time to calculate the SAD between the current and all the previously stored visual templates. Optimizations have been added that first compare template means as well as checks during the comparison process to see if the error is already larger that minimum error already found. The second is that the pose cell computation time is linearly related to the size of the pose cell environment due to the dense implementation. A sparse pose cell implementation is possible, however at the cost of a dramatic increase in code complexity. The third is that the experience map computation time is linearly related to the number of correction loops per cycle. While the speed of the correction process doesn't affect the topological correctness, it is directly related to the metric accuracy.

6.5 Comparison with the original code

This section has results of software engineering metrics for comparison between the OpenRatSLAM code base and the original code base. Table 5 compares the two code bases across several metrics. The results show the more that order of magnitude reductions in the number of functions, defines, lines of code and sources files between the two code bases which implies that the new OpenRatSLAM code base is easier to understand and manage. The other metrics demonstrate that the new code base is more portable and due to the ROS wrapper, is designed to work with a wider range of robot platforms.

7 Discussion and future work

In this paper the performance of OpenRatSLAM has been demonstrated across three different datasets. The datasets are from three different robot platforms with varying sensor

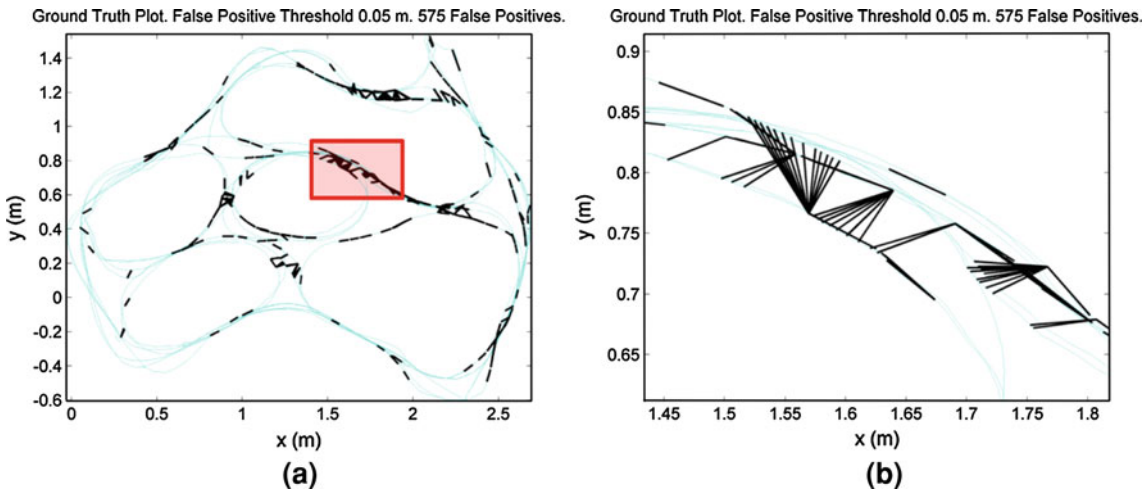

Fig. 21 Automated true-false positive analysis of the iRat dataset using the provided ground truth and MATLAB .m script files. **a** 'False' positive image matches based on a small 0.05 m threshold, and **b** a zoomed in section showing the local spatial generalization of a single image template

Table 5 Comparison of the original RatSLAM and OpenRatSLAM code base demonstrating the software engineering and usability improvements

Metric	Original RatSLAM	OpenRatSLAM
Functions	1099	135
Defines	580	13
Lines of code	∼32, 000	∼3, 000
Source files	85	19
Compiler	Visual Studio	gcc using ROS
OS platform	Windows	Windows, Linux
Robot platform	Offline, Pioneer	Offline, robots conforming to ROS (only confirmed on iRat and Pioneer)

Lines of code and source files calculated using CLOC (cloc.sourceforge.net); functions and defines determined using CTAGS (ctags.sourceforge.net) only considering C/C++ source and header files

suites operating in indoor and outdoor environments with significant scale variation. The *St Lucia 2007* dataset was taken outdoors on a road network using a laptop with a cheap inbuilt webcam, mounted on an Ackermann steer vehicle, travelling at tens of meters per second across an area covering several million square meters, with no wheel odometry. The *New College 2008* dataset was taken outdoors along paths in a university campus environment, using a panoramic camera sensor mounted on a rocking mobile robot travelling at approximately a meter per second across an area covering several thousand square meters. The *iRat 2011* dataset is taken indoors on a custom embedded computer using a cheap camera, mounted on a small custom robot, travelling at tenths of a meter per second, across an area covering only a few square meters.

This paper has demonstrated that OpenRatSLAM is able to successfully build topological maps across these significantly different datasets with only minimal parameter changes. Most of the parameters are directly related to measurable and known properties of the data. For example, there is a physical relationship between the size of the environment and vehicle and the parameters that represent the number and size of the cells in the Pose Cell network. The image normalization parameters are directly related to the different lighting conditions between the datasets; the *St Lucia 2007* has large ambient light changes hence requiring global image normalization, the *New College 2008* dataset has dull overcast lighting and hence requires local contrast enhancement to bring out features in some areas, and the *iRat 2011* dataset has controlled indoor lighting conditions and hence requires no extra preprocessing.

The OpenRatSLAM system released with this paper implements the majority of features published across more than two dozen RatSLAM publications. Although this paper does not include results for the path planning algorithm, the underlying code base is relatively unchanged from what was used in prior studies such as Heath et al. (2011) and Radish. These studies have tested the experience map's ability to plan a path to hundreds of goal locations. While these studies report that the robot sometimes fails to reach a goal, this is a limitation of having local obstacle avoidance using three noisy proximity sensors on the robot and an incomplete topological map.

To generate the results for the *St Lucia 2007* dataset, we used a different set of parameters than previously published in Milford and Wyeth (2008), in particular using a 2D template instead of the 1D scanline profile. During the parameter tuning process we found that a 2D template gave more robust scene matching results. While we attempted to make OpenRatSLAM as similar as possible to the original code base there were inevitably a number of minor differences. However, the recreation of past results with different parameters

suggests the differences are not detrimental to system performance. In the following discussion we highlight several areas of future work.

Various map maintenance algorithms will be implemented to facilitate online operation over long periods of time. A simple pruning scheme was implemented in Milford and Wyeth (2010), but there is the potential to integrate other map consolidation algorithms in this rapidly developing field. We are also investigating the feasibility of an autonomous or supervised process for tuning the key system parameters based on a user providing a representative dataset, similar in overall process to (Fast Appearance Based Mapping) FAB-MAP codebook generation (Cummins and Newman 2008). The system would optimize two or three parameters to close the loop on a loop closure segment manually chosen by a user. While the tuning process described in this paper provides a standardized method for achieving good mapping performance, the introduction of such a training scheme would enable complete "hands-off" operation in a range of environments, without the user requiring any understanding of system parameters.

Current work is also enabling multisensory input to OpenRatSLAM and automatic weighting of sensor readings based on their utility for localization (Jacobson and Milford 2012). Results indicate that OpenRatSLAM is capable of successful performing SLAM in changing environments by dynamically weighting sensor readings without any explicit knowledge of the number or type of sensors on a robot platform. In addition to adding new features to OpenRatSLAM, we intend to continue to release existing and new datasets and associated configuration parameters, in order to increase the variety of dataset types and sensory processing schemes that can be handled by the system.

OpenRatSLAM is intended for use not only by roboticists but also by scientists in other fields such as neuroscience and cognitive science. Cognitive scientists have developed a range of neural algorithms for components of navigation, including models of visual cortex, head direction calibration and functional regions of the hippocampus and surrounding areas. The modular nature of OpenRatSLAM will enable researchers to incorporate custom modules into a fully functioning navigation system, and compare performance against the standards reported here. For example, the implementation of a hexagonal pose cell structure resulting in grid-cell-like hexagonal firing fields (Milford et al. 2010) should be easily replicable using the OpenRatSLAM codebase—edits to the pose cell module can take place in isolation, as opposed to requiring an in-depth understanding of the entire original RatSLAM codebase. We believe that one of the most interesting areas for future research will be the incorporation of biologically plausible visual processing algorithms into the local view cell node. The relatively simple vision processing system could be replaced with more sophisticated computational models of rodent vision and object recognition, such as those developed by the Cox laboratory (Zoccolan et al. 2009). Again, researchers wishing to incorporate changes like these need only edit the relevant module. The combination of OpenRatSLAM with the iRat will allow the development and testing of high-fidelity neural and behavioral models for use in computational neuroscience.

8 Conclusion

This paper has described an open-source version of RatSLAM with bindings to the increasingly popular ROS framework. RatSLAM has the particular advantage of being an appearance-based navigation system that works well with low resolution monocular image data, in contrast to other available navigation systems that focus on probabilistic methods, occupancy grids and laser range sensors. To date there has been no complete open-source implementation of RatSLAM, which has inhibited investigation of its merits for different applications and comparisons with other SLAM systems. This is also the first publication to provide in depth technical detail about how RatSLAM algorithm internally works, which will facilitate its use by others on their own datasets and robot platforms. The paper also provides ground truth data and analysis tools for the iRat dataset.

OpenRatSLAM is modular, easy to visualize, tune and integrate into larger robotic systems. We demonstrated the scalable performance of the system with three publicly available datasets (two of which we provide and support); an outdoor suburb scale environment on a full-size car, an outdoor university campus environment on a medium size robot, and an indoor environment on a small custom robot. OpenRatSLAM development and support will continue with the aim of growing an interdisciplinary user base amongst a variety of research fields including robotics, cognitive science and neuroscience.

Acknowledgments This work was supported in part by the Australian Research Council under a Discovery Project Grant DP0987078 to GW and JW, a Special Research Initiative on Thinking Systems TS0669699 to GW and JW and a Discovery Project Grant DP1212775 to MM. We would like to thank Samuel Brian for coding an iRat ground truth tracking system.

Appendix A: Included datasets

The three datasets described in this paper are available online at http://wiki.qut.edu.au/display/cyphy/OpenRatSLAM+datasets . Details of the ROS bag files are provided below.

path: **irat_aus_28112011.bag**
version: 2.0
duration: 15:53s (953s)
start: Nov 28 2011 15:41:38.37 (1322458898.37)
end: Nov 28 2011 15:57:31.47 (1322459851.47)
size: 861.0 MB
messages: 116603
compression: none [1111/1111 chunks]
types: geometry_msgs/PoseStamped [d3812c3cbc69362b77dc0b19b345f8f5]
nav_msgs/Odometry [cd5e73d190d741a2f92e81eda573aca7]
sensor_msgs/CompressedImage [8f7a12909da2c9d3332d540a0977563f]
sensor_msgs/Range [c005c34273dc426c67a020a87bc24148]
topics: /irat_red/camera/image/compressed 16657 msgs : sensor_msgs/CompressedImage
/irat_red/odom 16658 msgs : nav_msgs/Odometry
/irat_red/proximity/range0 16658 msgs : sensor_msgs/Range
/irat_red/proximity/range1 16658 msgs : sensor_msgs/Range
/irat_red/proximity/range2 16658 msgs : sensor_msgs/Range
/overhead/camera/image/compressed 16657 msgs : sensor_msgs/CompressedImage
/overhead/pose 16657 msgs : geometry_msgs/PoseStamped

path: **stlucia_2007.bag**
version: 2.0
duration: 1hr 36:00s (5768s)
start: Jan 01 1970 10:00:00.10 (0.10)
end: Jan 01 1970 11:36:09.00 (5769.00)
size: 2.3 GB
messages: 57690
compression: none [3070/3070 chunks]
types: sensor_msgs/CompressedImage [8f7a12909da2c9d3332d540a0977563f]
topics: /stlucia/camera/image/compressed 57690 msgs : sensor_msgs/CompressedImage

path: **oxford_newcollege.bag**
version: 2.0
duration: 43:57s (2637s)
start: Nov 03 2008 23:43:32.69 (1225719812.69)
end: Nov 04 2008 00:27:29.98 (1225722449.98)
size: 1.2 GB
messages: 16002
compression: none [1458/1458 chunks]
types: nav_msgs/Odometry [cd5e73d190d741a2f92e81eda573aca7]
sensor_msgs/CompressedImage [8f7a12909da2c9d3332d540a0977563f]
topics: /newcollege/camera/image/compressed 7854 msgs : sensor_msgs/CompressedImage
/newcollege/odom 8148 msgs : nav_msgs/Odometry

Note that the data in the New College dataset belongs to the original authors from Oxford University.

Appendix B: Installation instructions and tutorial

Installing dependencies

OpenRatSLAM depends on ROS packages: opencv2 and topological_nav_msgs and also on 3D graphics library Irrlicht. Irrlicht can be installed on Ubuntu with apt-get

sudo apt-get install libirrlicht-dev

Build instructions

Checkout the source from SVN:

svn checkout http://ratslam.googlecode.com/svn/branches/ratslam_rosratslam_ros

Setup ROS environment variables by typing:

. /opt/ros/fuerte/setup.sh

The OpenRatSLAM directory needs to be added to the environment variable ROS_PACKAGE_PATH.

export ROS_PACKAGE_PATH=$ROS_PACKAGE_PATH:/path/to/OpenRatSLAM

Then build OpenRatSLAM with

rosmake

Running OpenRatSLAM

To use one of the provided pre-packaged bag files, download the bag file for either:

- iRat 2011 in Australia
- Car in St Lucia 2007
- Oxford New College 2008 dataset

All datasets are available at https://wiki.qut.edu.au/display/cyphy/OpenRatSLAM+datasets.

Place the dataset in the OpenRatSLAM directory.

Run the dataset and RatSLAM by typing either

roslaunch irataus.launch

rosbag play irat_aus_28112011.bag

or

roslaunch stlucia.launch

rosbag play stlucia_2007.bag

or

roslaunch oxford_newcollege.launch

rosbag play oxford_newcollege.bag

Using rviz

The map created by OpenRatSLAM will be periodically published to rviz. To run rviz:

rosrun rviz rviz

Click on the ”Add” button down the bottom left of the window. Choose ”MarkerArray” from the list. In the field ”Marker Array Topic” on the left, click on the button with 3 dots. Choose the topic <my_robot>/ExperienceMap/MapMarker

Using OpenRatSLAM with a custom dataset

Creating a bag file

The easiest way to tune RatSLAM is using an offline dataset. Any robot providing camera images as sensor_msgs/CompressedImage and odometry as nav_msgs/Odometry can be used to create a dataset. The images and odometry must be in the form <my_robot>/camera/image and<my_robot>/odom.

To convert topic names to the correct format run:

rostopic echo <path/to/my/robot/camera> | rostopic pub <my_robot>/camera/image sensor_msgs/CompressedImage & rostopic echo <path/to/my/robot/odom> | rostopic pub <my_robot>/odom nav_msgs/Odometry &

Start recording into a bag file:

rosbag record -O <my_robot> .bag <my_robot>/camera/image <my_robot>/odom

Start the robot and collect the dataset. Press Ctrl-C at the terminal to finish recording.

Running the bag file

To run a custom bag file, a new config file and launch file are required.

Creating a new config file

In a terminal type

cd ratslam_ros

cp config/config_stlucia.txt config/config_<my_robot> .txt

gedit config/config_<my_robot> .txt

Change the first line

topic_root=stlucia

to

topic_root=<my_robot>

Creating a new launch file In the same terminal type

cp stlucia.launch <my_robot> .launch

gedit <my_robot> .launch

Replace all references to "../config/config_stlucia.txt" with "../config/config_<my_robot> .txt"

Comment out the visual odometry node to prevent it from running. Replace

<node name="RatSLAMVisualOdometry" pkg="ratslam_ros" type="ratslam_vo" args="../config/config_<my_robot> .txt
_image_transport:=compressed" cwd="node" required="true" />

with

<!– <node name="RatSLAMVisualOdometry" pkg="ratslam_ros" type="ratslam_vo" args="../config/config_<my_robot> .txt
_image_transport:=compressed" cwd="node" required="true" />–>

Running your dataset Your dataset can now be run the same way is the provided datsets:

roslaunch <my_robot> .launch

rosbag play <my_robot> .bag

Tuning parameters

Open the created config file

gedit config/config_<my_robot> .txt

Edit the settings under [ratslam]. Refer to Sect. 4 for parameter details.

References

Andreasson, H., Duckett, T., & Lilienthal, A. (2008). A minimalistic approach to appearance-based visual SLAM. *IEEE Transactions on Robotics*, *24*, 1–11.

Ball, D. (2009). RatSLAM, 1.0 ed. ratslam.itee.uq.edu.au. The University of Queensland, Brisbane.

Ball, D., Heath, S., Wyeth, G., & Wiles, J. (2010). iRat: Intelligent rat animal technology. In *Australasian conference on robotics and automation*. Brisbane, Australia.

Bay, H., Tuytelaars, T., & Van Gool, L. (2006). SURF: Speeded up robust features. In *Computer Vision—ECCV 2006* (pp. 404–417).

Cummins, M., & Newman, P. (2008). FAB-MAP: Probabilistic localization and mapping in the space of appearance. *International Journal of Robotics Research*, *27*, 647–665.

Cummins, M., & Newman, P. (2009). Highly scalable appearance-only SLAM—FAB-MAP 2.0, in Robotics: Science and Systems, Seattle, United States.

Cummins, M., & Newman, P. (2010). Appearance-only SLAM at large scale with FAB-MAP 2.0. *The International Journal of Robotics Research*, *30*(9), 1100–1123.

Davison, A. J., Reid, I. D., Molton, N. D., & Stasse, O. (2007). MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*, 1052–1067.

Hafting, T., Fyhn, M., Molden, S., Moser, M.-B., & Moser, E. I. (2005). Microstructure of a spatial map in the entorhinal cortex. *Nature*, *11*, 801–806.

Heath, S., Cummings, A., Wiles, J., & Ball, D. (2011). A rat in the browser. In *Australasian conference on robotics and automation*. Melbourne, Australia.

Jacobson, A., & Milford, M. (2012). *Towards brain-based sensor fusion for navigating robots, presented at the Australasian conference on robotics and automation*. Wellington, New Zealand.

Knuth, D. (1977). A generalization of Dijkstra's algorithm. *Information Processing Letters, 6*.

Konolige, K., & Agrawal, M. (2008). FrameSLAM: From bundle adjustment to real-time visual mapping. *IEEE Transactions on Robotics*, *24*, 1066–1077.

Konolige, K., Agrawal, M., Bolles, R., Cowan, C., Fischler, M., & Gerkey, B. (2008). *Outdoor mapping and navigation using stereo vision* (pp. 179–190).

Kyprou, S. (2009). *Simple but effective personal localisation using computer vision*. London: Department of Computing, Imperial College London.

Labbe, M., & Michaud, F. (2011). *Memory management for real-time appearance-based loop closure detection, presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems*. San Francisco, United States.

Lowe, D. G. (1999). *Object recognition from local scale-invariant features, presented at the proceedings of the international conference on computer vision* (Vol. 2).

Maddern, W., Milford, M., & Wyeth, G. (2012). CAT-SLAM: Probabilistic localisation and mapping using a continuous appearance-based trajectory. *The International Journal of Robotics Research*, *31*, 429–451.

Milford, M. J. (2008). *Robot navigation from nature: Simultaneous localisation, mapping, and path planning based on hippocampal models* (Vol. 41). Berlin: Springer.

Milford, M., & Wyeth, G. (2008). Mapping a suburb with a single camera using a biologically inspired SLAM system. *IEEE Transactions on Robotics*, *24*, 1038–1053.

Milford, M., & Wyeth, G. (2008). Single camera vision-only SLAM on a suburban road network. In *International conference on robotics and automation*. Pasadena, United States.

Milford, M., & Wyeth, G. (2010). Persistent navigation and mapping using a biologically inspired SLAM system. *International Journal of Robotics Research*, *29*, 1131–1153.

Milford, M.J., Wiles, J., & Wyeth, G. F. (2010). Solving navigational uncertainty using grid cells on robots. *PLoS Computational Biology, 6*.

Milford, M., Schill, F., Corke, P., Mahony, R., & Wyeth, G. (2011). Aerial SLAM with a single camera using visual expectation. In *International conference on robotics and automation*. Shanghai, China.

Newman, P., Sibley, G., Smith, M., Cummins, M., Harrison, A., Mei, C., et al. (2009). Navigating, recognizing and describing urban spaces with vision and lasers. *The International Journal of Robotics Research*, *28*, 1406–1433.

Quigley, M., Gerkey, B., Conley, K., Fausty, J., Footey, T., Leibs, J., et al. (2009). *ROS: an open-source Robot Operating System, presented at the IEEE international conference on robotics and automation*. Kobe, Japan.

Radish: The Robotics Data Set Repository [Online]. Available: http://radish.sourceforge.net/

Samsonovich, A., & McNaughton, B. L. (1997). Path integration and cognitive mapping in a continuous attractor neural network model. *The Journal of Neuroscience*, *17*, 5900–5920.

Sibley, G., Mei, C., Reid, I., & Newman, P. (2010). Vast-scale outdoor navigation using adaptive relative bundle adjustment. *International Journal of Robotics Research*, *29*, 958–980.

Smith, D., & Dodds, Z. (2009). Visual navigation: Image profiles for odometry and control. *Journal of Computing Sciences in Colleges*, *24*, 168–179.

Smith, M., Baldwin, I., Churchill, W., Paul, R., & Newman, P. (2009). The new college vision and laser data set. *The International Journal of Robotics Research*, *28*, 595–599.

Strasdat, H., Montiel, J. M., & Davison, A. J. (2010). *Scale drift-aware large scale monocular SLAM, in robotics science and systems*. Spain: Zaragoza.

Sunderhauf, N. (2012). Towards a robust back-end for pose graph SLAM. In *IEEE international conference on robotics and automation*. St Paul, United States.

Sunderhauf, N., & Protzel, P. (2010). Beyond RatSLAM: Improvements to a biologically inspired SLAM system. In *IEEE international conference on emerging technologies and factory automation* (pp. 1–8). Bilbao, Spain.

Zhang, A. M., & Kleeman, L. (2009). Robust appearance based visual route following for navigation in large-scale outdoor environments. *The International Journal of Robotics Research*, *28*, 331–356.

Zoccolan, D., Oertelt, N., DiCarlo, J. J., & Cox, D. D. (2009). A rodent model for the study of invariant visual object recognition. *Proceedings of the National Academy of Sciences of the United States of America*, *106*, 8748–8753.

Author Biographies

David Ball is currently a Research Fellow at the Queensland University Technology in Australia working on robotic solutions for agriculture. He completed his undergraduate degree in Computer Systems Engineering in 2001 and his PhD titled "Reading the Play: Adaptation by prediction in robot soccer" in Mechatronic Engineering in 2008, both at the University of Queensland. His first research position was on an Australian

Special Research Initiative, Thinking Systems, a cross-disciplinary team which investigated navigation across real and conceptual spaces.

Scott Heath received a dual degree in Engineering (Electrical) and Information Technology from the University of Queensland in 2010. He is currently a graduate student at the University of Queensland where he works on the Lingodroids project—an investigation of robots evolving language for space and time. His research interests include symbol grounding, computational modelling and telerobotics.

Janet Wiles received the B.Sc. Hons. and Ph.D. degrees from the University of Sydney, Sydney, Australia, in 1983 and 1989, respectively. She is currently a Professor of Complex and Intelligent Systems at the University of Queensland, Brisbane, Australia. She is the Project Leader of the Thinking Systems Project, supervising a cross-disciplinary team studying fundamental issues in how information is transmitted, received, processed, and understood in biological and artificial systems. Her research interests include complex systems biology, computational neuroscience, computational modeling methods, artificial intelligence, and artificial life, language, and cognition.

Gordon Wyeth is Head of the School of Electrical Engineering and Computer Science and Professor of Robotics at the Queensland University of Technology. Prior to 2010 he was at the University of Queensland where he was co-Director of Mechatronic Engineering. Professor Wyeth's main interests are in spatial cognition and biologically inspired robotics, with more than 150 papers published in leading journals and conferences. He has served as President of the Australian Robotics and Automation Association 2004–2006, chaired the Australasian Conference on Robotics and Automation in 1999, 2004 and 2011, chaired the IEEE Robotics and Control Systems Queensland Chapter 2010–2011, and is currently Chair of the Research and Education College of the Cooperative Research Centre for Spatial Information.

Peter Corke is Professor of Robotics and Control at the Queensland University of Technology. Previously he was a Senior Principal Research Scientist at the CSIRO ICT Centre where he founded and led the Autonomous Systems laboratory, the Sensors and Sensor Networks research theme and the Sensors and Sensor Networks Transformational Capability Platform. He is a Fellow of the IEEE; Editor-in-Chief of the IEEE Robotics and Automation magazine; founding editor of the Journal of Field Robotics; member of the editorial board of the International Journal of Robotics Research, and the Springer STAR series. He has over 300 publications in the field and has held visiting positions at the University of Pennsylvania, University of Illinois at Urbana-Champaign, Carnegie-Mellon University Robotics Institute, and Oxford University.

Michael Milford holds a PhD in Electrical Engineering and a Bachelor of Engineering from the University of Queensland, awarded in 2006 and 2002 respectively. He recently joined the Queensland University of Technology as member of Faculty, having previously worked as a Postdoctoral Research Fellow at QUT and before that for three years as a Research Fellow at the Queensland Brain Institute and in the Robotics Laboratory at the University of Queensland. His research interests include Simultaneous Localisation And Mapping, vision-based navigation, cognitive modelling of the rodent hippocampus and entorhinal cortex, biologically inspired robot navigation and computer vision. In January 2012 he commenced a fellowship on enabling visual navigation for "sunny summer days and stormy winter nights", and is also leading a project on Brain-based Sensor Fusion for Navigating Robots.