Virtual Reconstruction Using an Autonomous Robot

Matthew McGill, Rudino Salleh, Timothy Wiley,
Adrian Ratter, Reza Farid and Claude Sammut
School of Computer Science and Engineering
The University of New South Wales
Sydney, Australia 2052
Email: {mmcgill, rudinos, timothyw, adrianr, rezaf, claude}
@cse.unsw.edu.au

Adam Milstein
National Robotics Engineering Center
Pittsburgh, Pennsylvania 15201
Email: ahpmilstein@gmail.com

***Abstract*—Advances in sensing technology and algorithm design make it possible for a robot equipped with a laser range-finder to generate a map and localise itself within the map as the robot explores its environment. We describe a system for mapping and virtual reconstruction developed as part of a robot for urban search and rescue. The process of mapping and, at the same time, localising the robot within the map, is called Simultaneous Localisation and Mapping (SLAM). In many applications, such as urban search and rescue, information from wheel encoders is inaccurate and cannot be used for odometry to obtain a position estimate. However, iterative closest point scan matching algorithms make it possible for a robot to perform accurate positioning in unstructured environments where wheel slip is common. When this positioning is combined with a mapping algorithm such as FastSLAM, the robot can construct an accurate map in real-time as it moves. Given the generated map and the robot's position within it, a variety of exploration algorithms allow the robot to autonomously explore its environment. The robot is also equipped with an RGB-D camera. The 3D information as well as the colour video images are incorporated into the map to produce a 3D virtual reconstruction of the environment as the robot explores. This robot won the award for best autonomous robot in three successive RoboCup Rescue Robot competitions [1], 2009 - 2011.**

I. Introduction

One of the fundamental problems in robotics is that of mapping and tracking a robot's location within the map. A robot must be able to localize itself within a map so that it knows where objects of interest can be found and how to safely navigate to the objects. In robotics there are often times when a map isn't available at the beginning so the robot has to both generate the map and track its location within the map. This problem is simultaneous localization and mapping(SLAM).

A. Position Tracking

Most solutions to the SLAM problem require sensor readings and an estimate of the robot's position.

The simplest method for tracking a robot's position is to use the encoders for its method of locomotion. In the case of a wheeled robot this means tracking how far each wheel has rotated in each time step and using the relative and absolute movements to calculate the forward, lateral and rotational changes to the robot's pose.

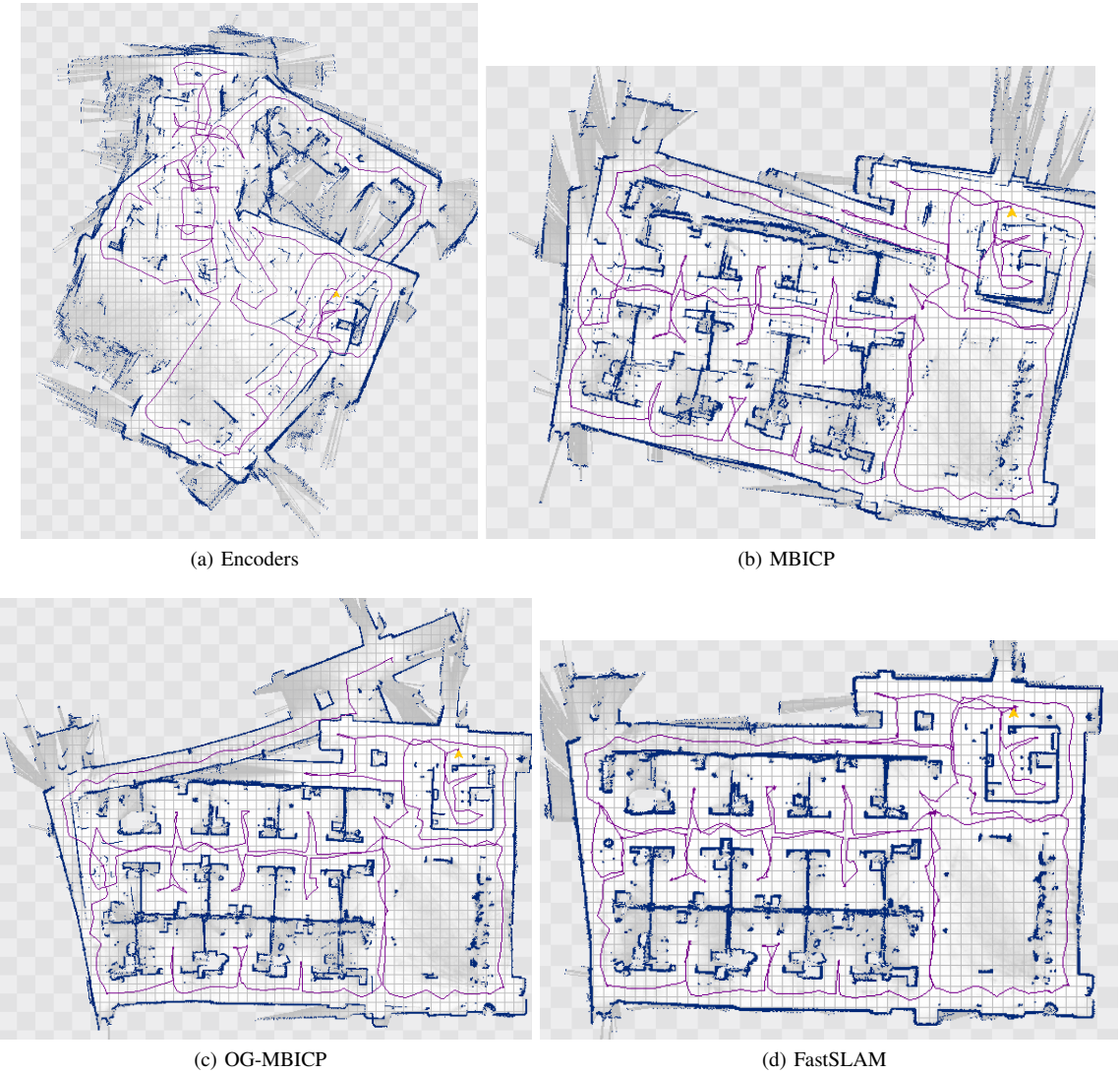
The problem with using encoders is that they only work when the robot's movement is directly related to the rotation of its wheels. If the robot slips or slides as it moves, possibly due to environmental factors such as debris or uneven and slippery surfaces, then its encoders can no longer track its position accurately.

A laser range finder is often used for position tracking and mapping. Metric-Based Iterative Closest Point scan matching (MBICP) [2] is an algorithm that takes two consecutive laser scans and calculates the translational and rotational motion of the sensor between the scans. MBICP works by finding points in the second scan that correspond to points in the first scan. These points are used to find a correction to an estimate of the sensor's motion that better fits the correspondences. MBICP iteratively finds corresponding points and corrects its motion estimate until either the correction falls below a threshold or it overruns a loop counter indicating it cannot accurately match the two scans.

MBICP requires there to be a number of corresponding points between each consecutive pair of scans so the scan must overlap. This is most likely to occur when the laser sweeps through the same plane in each scan, which can be most easily accomplished by ensuring the laser sweeps horizontally. This can be done by mounting the laser horizontally and only driving on flat ground or by mounting the laser on an auto-level mount that keeps the laser level as the robot traverses rough terrain. When using an auto-levelled laser over rough ground it is likely the laser will be moving vertically. In many environments this is not a significant problem as there is often little change vertically, especially in structured environments such as buildings.

Occupancy Grid MBICP [3] is an extension of MBICP which uses an occupancy grid. Searching for matching points can be made more efficient by storing them in an occupancy grid as near by points will mostly be located in neighbouring cells but can be anywhere within a laser scan. The number of observations of a cell can also be taken into account, along with the time since the cell was last observed. Once a scan is aligned, it is placed into the grid keeping the grid up to date. Another benefit of using this grid based method is that a scan is not compared to just the previous scan but to a number of earlier scans. This allows some robustness to position tracking

978-1-4673-1954-6/12/$31.00



(a) Encoders (b) MBICP

(c) OG-MBICP (d) FastSLAM

Fig. 1. Comparison of accuracy between position tracking alternatives and FastSLAM

when a dynamic obstacle such a person passes by the sensor. This is because as the obstacle moves through the the sensor's field of view it will obscure some more distant objects, but as it moves these objects become visible again and their points can be compared to points from scans received before the moving obstacle hid them.

B. FastSLAM

Having a reasonably accurate method for tracking a robot's trajectory is the first step in being able to build a map. An unfortunate reality with all position trackers is that in every interval there is some error and over time these errors accumulate. To compensate for this unavoidable error, a solution to the SLAM problem is required.

We use an implementation [4] of the FastSLAM [5]–[7] algorithm to generate the final map. Figure 1 shows a comparison of maps generated using three position trackers with raw sensor data and FastSLAM.

FastSLAM is a particle filter algorithm that can run in real-time. A FastSLAM map consists of a population of particles where each particle is an estimate of the robot's position and the environment that has been sensed. When new sensor data are available, each particle has its position updated by the change in position reported by the position tracker plus a small random noise corresponding to the expected error in the position track. Each particle is then given a weight relative to the likelihood of the robot getting that sensor data in its new position in the particle's map of the environment. The sensor data are then used to update the particle's map. A new population of particles is then created by randomly selecting from the current generation, with each particle's chance of selection being dependent on its weight. At this point, FastSLAM awaits the arrival of new sensor data.

There are several advantages to the use of FastSLAM. It makes no assumptions about the robot's environment, other than it being reasonably static. While it is possible to detect features and use them to increase the algorithm's accuracy

(a) Emu (b) Negotiator (c) Packbot

Fig. 2. Some Autonomous Robots

it is not a requirement. Also, as it is particle filter based it can be multi-threaded allowing it to be sped up if more than one processor is available. It is possible for FastSLAM to be initialized with a preexisting map and for map updates to be disabled to provide simple localization within a known environment.

The are some drawbacks to using FastSLAM. As the infinite state space of possible positions and possible maps is represented by a finite set of particles, unlikely possibilities tend to die out. This is most apparent when traversing long loops in the environment, in which at time the robot arrives back at the start of the loop, no particles have survived at the correct location. Also as each particle has its own copy of the map the FastSLAM algorithm can use significant amounts of memory.

Other SLAM algorithms such as Gmapping [8] could potentially be used. Gmapping combines a scan matching algorithm with a particle filter SLAM implementation to produce reasonably accurate maps. In our implementation we have elected to explicitly separate these components to allow our autonomy to work with local dynamic sensor readings as well as over the more stable global map.

II. Autonomous Exploration

Once a robot has a map and knows where it is located within its environment, it is possible to endow the robot with the ability to explore autonomously.

A. Voronoi Grid

Rather than allowing the robot to go anywhere it likes in the map, it is generally better to mark areas where the robot should and should not wander. We use a type of voronoi occupancy grid to mark areas that are safe for the robot to maneuver through. Figure 3 is an example of the voronoi grid that we use. It is based on the concept of voronoi diagrams [9]. Occupied cells in the map are clustered to form walls. These walls are then expanded out. Where two walls expand into the same cell at the same distance, the cell is marked as belonging to the voronoi skeleton. Our implementation actually has two phases of expansion. The first is the region in which the robot

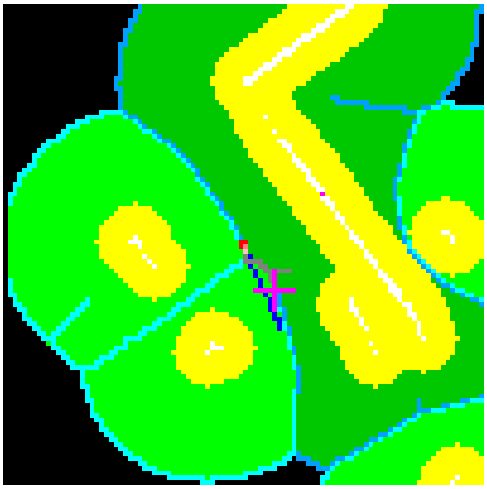
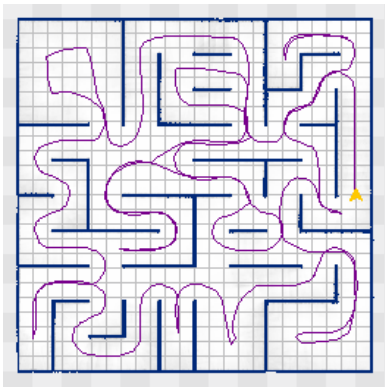


Fig. 3. Voronoi grid showing traversible regions during an autonomous exploration

cannot go, as entering it would require that part of the robot be inside an occupied cell. This forbidden region is the yellow area in figure 3. The second phase expands out to produce the voronoi skeleton. This phase is represented by the green area in figure 3. An optimization is to limit the second expansion phase to some safety margin that we would like the robot to maintain from walls and other obstacles where it is possible. Points at the limit are also added to the voronoi skeleton. Points outside the limit (the black regions in figure 3) are considered safe to travel through. The blue lines in figure 3 represent the voronoi skeleton which is the mid-line between walls and the edge of the safety margin in open areas.

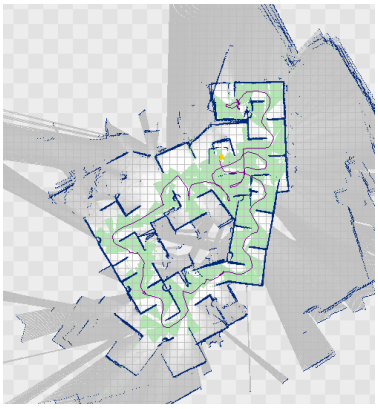
B. Wall Following

Now that the robot knows where it can safely go it needs an method for determining where it should go. For the sake of simplicity we choose to use a simple wall following algorithm. When it starts, it chooses to either follow the left or right wall. If its following the left wall it looks to its left to find a wall. It then finds the section of the voronoi skeleton associated

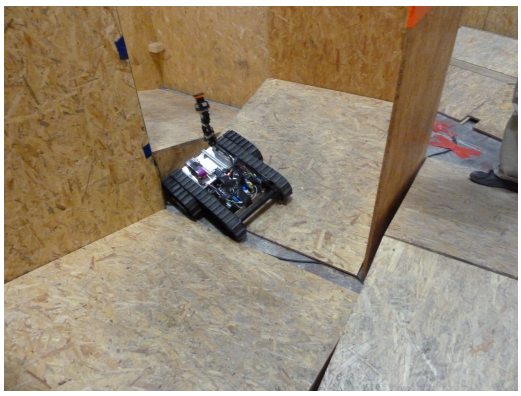


(a) Left wall following with loop breaking in a simulated maze

(b) Right wall following with loop breaking in the lab



(c) An autonomously generated map of the rescue robot arena from RoboCup 2011



(d) Negotiator navigating the rescue robots arena autonomously

Fig. 4. Challenges to autonomous exploration and some results

with that wall. It traces along the skeleton section to some predetermined distance and that location is set as a waypoint it can head towards as it follows the wall. This waypoint is used to calculate the robot's forward speed and turn rate as the robot attempts to head directly towards it. The voronoi grid is updated whenever the map it is based on changes, and as the robot's position is updated the wall following algorithm updates its waypoint, usually by selecting a new point further along the skeleton. In figure 3, the darker green and blue cells are the expansion area around the wall being followed. The purple + marks the current waypoint.

This simple algorithm is useful because it does not require the full map, just the immediate area around the robot. This means only the voronoi grid of the area around the robot needs to be calculated. It is also able to be generated just using the position tracker and the sensor data. This allows the exploration algorithm to react to short lived obstacles that don't exist in the map. These obstacles can include people walking past the robot.

C. Loop Escape

The biggest problem with using a wall following algorithm is that it is possible for it to get caught in a loop. This can happen if the robot follows the external wall of a building and circles around to its starting point or if it starts next to a section of wall that is separated from the rest of the walls.

To escape from this situation our algorithm makes use of the robot's path to detect if it has returned to an area it has previously been. If it has returned to a place it has been before it searches the local area for a section of the voronoi skeleton that it has not yet followed. If no such section can be found then the voronoi grid for the entire map is generated. An A* [10] search is then run over the entire voronoi grid from the robot's current position to any areas that have not yet been fully explored. This A* search produces a path that the robot tries to follow to get to the unexplored area. As the robot moves and the map is updated, the A* search must be rerun and the path being followed changed. When the robot enters an area that it hasn't yet explored it returns to wall following.

The benefits of this approach are that it allows an area to be fully explored while limiting the processor intensive A* to only be run when it is absolutely necessary. This exploration method was used successfully at RoboCup 2011 and combined with the mapping won best in class autonomy in the rescue robot competition. Figure 4c shows an autonomously generated map from the rescue robot competition at RoboCup

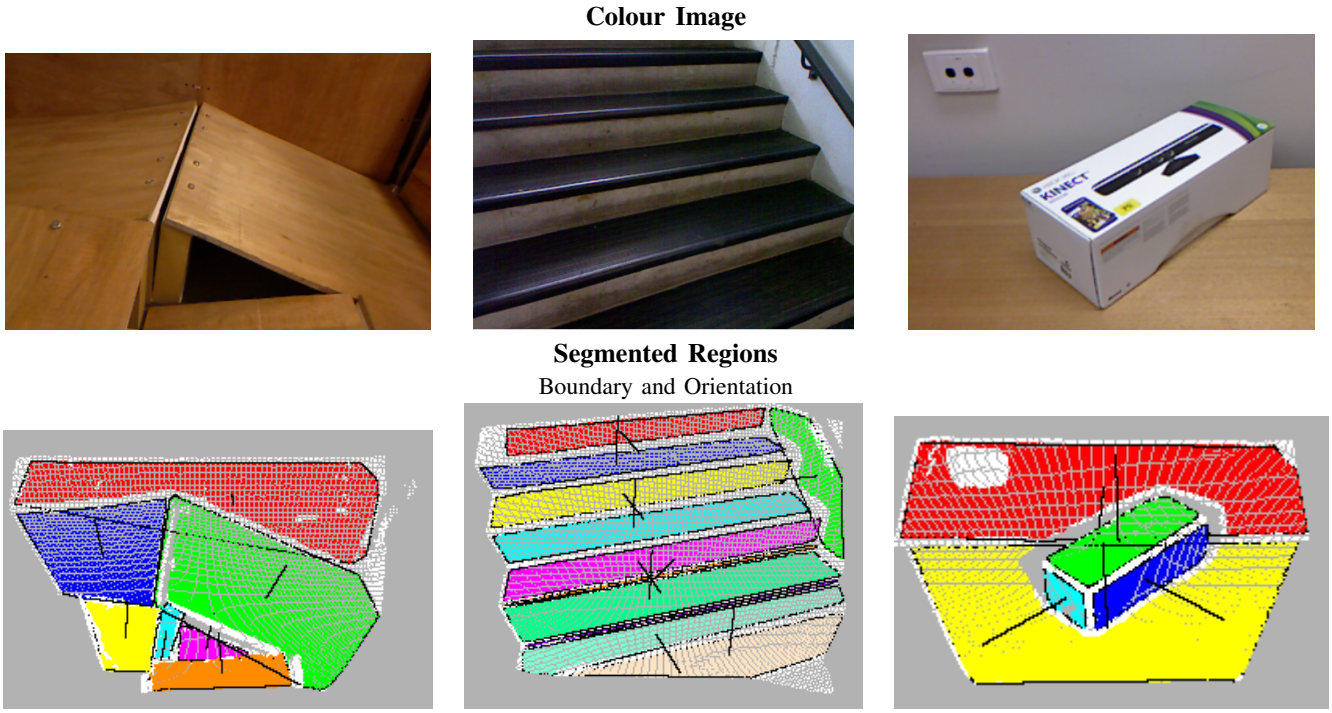


Fig. 5. Planar Segmentation

2011.

III. Hardware

Our software is designed to be platform independent although most robots are equipped with a scanning laser. On robots intended for use in rugged environments, these are mounted on auto-levels. The robots are also equipped with a variety of colour and thermal cameras. Figure 2 shows some of the robots that can run autonomously. They are described in more detail below.

A. Emu

Emu has been developed as a predominantly autonomous robot. It is constructed using a Volksbot [11] base. On top of this base is an orientation sensor that is used to detect the roll and pitch of the robot so that the laser can be auto-levelled. Also attached to the base is a 4 degree of freedom arm that supports Emu's main sensor package, which includes a Microsoft Kinect, wide angle camera and thermal camera.

B. Negotiator & Packbot

Negotiator and the Packbot are a pair of tracked robots designed for maneuvering over rough terrain from iRobot [12]. They have both been equipped with auto-levelled lasers and arms. At the end of their arms are sensor heads that include RGB-D and thermal cameras. Though the Negotiator and Packbot are usually teleoperated, they both have been given autonomous capabilities to allow them to perform tasks while their drivers are otherwise occupied.

C. Base Station

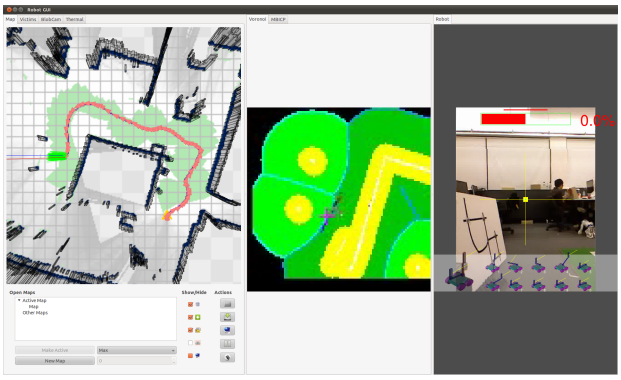
We connect to the robots using a laptop as a base station as this allows us to have a graphical user interface (GUI) [13] so that we can control the robots and it allows us access to the maps as they are generated.

IV. Virtual Reconstruction

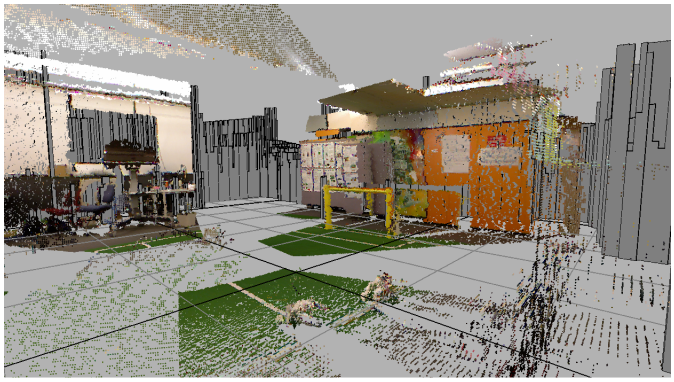
The FastSLAM implementation that we use allows for locations of interest to be marked in the map. During RoboCup Rescue competitions these marks are usually the location of victims that have been found. These landmarks can include captured images and point clouds.

By equipping a robot with an RGB-D camera, such as a Microsoft Kinect, we enable the robot to scan its environment and collect coloured point clouds. As the robot's location is being tracked within the map it is then possible to add the clouds into the map based on the robot's location. In our implementation we periodically stop the robot and then collect the coloured point cloud. We do this to blur in the images from the cameras due to motion and to compensate for the delay in the position tracking and map in determining the robot's pose. The robot's current pose is then combined with the position of the servos controlling its arm to determine the position of the camera when the point cloud is captured allowing the cloud to be placed into the map with reasonable accuracy. These localized point clouds are then used to produce a virtual reconstruction of the environment that the robot finds itself in. The reconstructions can be generated autonomously by setting a robot to periodically scan using the RGB-D camera as it explores a new environment.

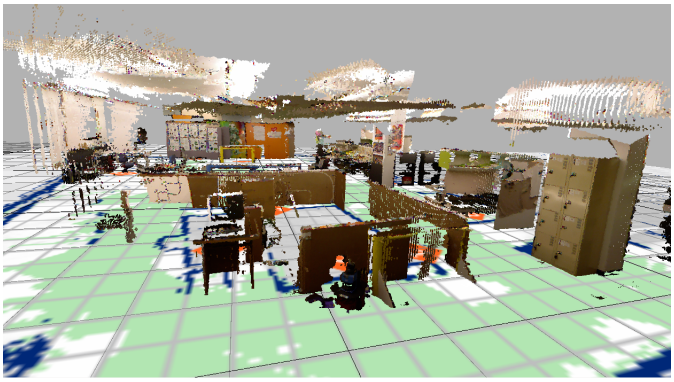
(a) Emu exploring the lab



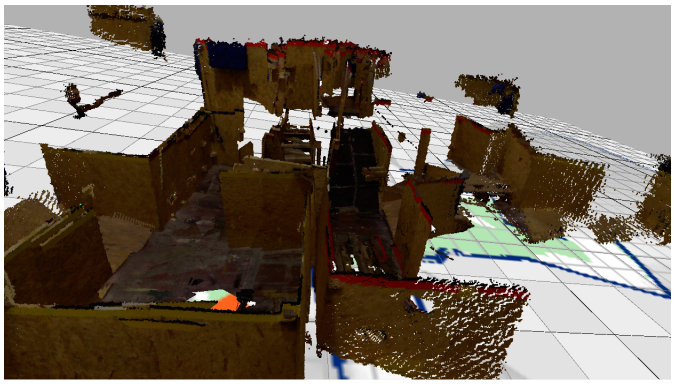
(b) GUI showing real-time map as Emu follows the left wall



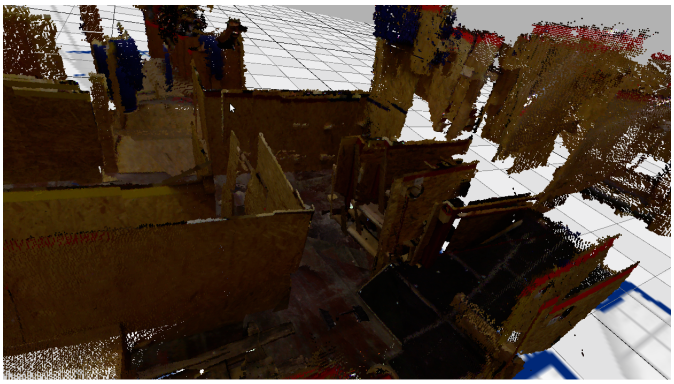
(c) Fusing of map with point clouds of soccer field



(d) Reconstruction of laboratory



(e) Reconstruction of the rescue robot arena from RoboCup 2011



(f) Another view of the rescue robot arena reconstruction

Fig. 6. Virtual reconstructions generated using an autonomous robot

Another approach is to use a higher level representation such as planes rather than points. Using this representation, each point cloud is segmented to a set of planar regions. The boundary of each region can be represented by four points or a convex hull. Thus, each point cloud can be reconstructed using only a few feature points. We have used this segmentation algorithm to generate features for generic object recognition of common objects in an urban search and rescue arena such as 'step', 'staircase', 'wall', 'box' and 'pitch/roll ramp' [14]. Figure 5 shows the result of this planar segmentation. Each region is shown as a different colour; the boundary is a convex hull and the orientation is represented by a 3D vector. The colour image corresponding to the main point cloud is also shown for more clarity.

V. Results

Our exploration algorithms were heavily tested in simulation before they were installed on a robot. Figure 4a shows the path travelled by a simulated robot using left wall following with loop breaking in a simulated maze. The simulated robot was required to complete this exploration task before the algorithm could be transferred onto a real robot. Figure 4b shows the path a real robot chose using the same algorithm doing a right wall follow in our robotics laboratory. The algorithms were tested

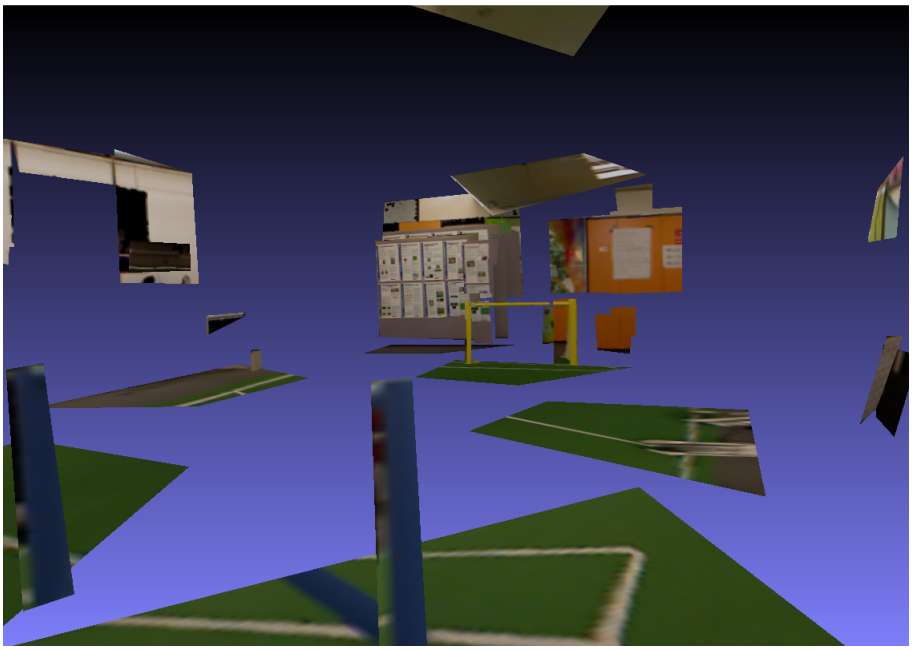


Fig. 7. Planar reconstruction of soccer field from Figure 6c

further during competition in the rescue robot competition at RoboCup 2011. Not only did the robots have to contend with sloped floors, the surface was remarkably slippery causing the robots to regularly slip down the ramps. The elements that are used to construct the arena for the rescue competition a taken from a standard set of components that are used by the National Institute of Standards and Technology (NIST) to evaluate robots for use by emergency services and the military.

A. Virtual Reconstruction

As there is no reliable means of generating a ground truth for virtual reconstructions, their evaluation is subjective. It mainly consists of exploring the virtual environment and comparing a viewpoint in the virtual world with the same viewpoint in the real world.

Two virtual reconstructions can be seen in Figure 6. While not perfect these reconstructions do give some sense as to the environments that the robots found themselves in. There is some misalignment visible in Figure 6 between the map and the cloud and even between the clouds in a scan. These misalignments are due to a combination of error in the robot's localization and error in calculating the kinematics of the robot's arm when computing the pose of the RGB-D camera relative to the robot. Figure 7 shows a planar reconstruction of the soccer field that has been created using the point clouds visible in Figure 6c.

It is possible to reduce the error in alignment of the point clouds by using an ICP or similar algorithm to align the clouds with each other and to align them to the map. This is some thing that we are currently looking into.

VI. Conclusion

Our experiences show that it is possible to build a robot capable of operating in moderately hostile environments that can build a reasonably accurate map of its surrounding. It has also been demonstrated that such a robot can be equipped with the ability to autonomously explore its environment. While exploring this environment the robot can use sensors such as an RGB-D camera to generate extra information that can be stored within the map. This information can then be used to identify objects of interest within the environment, such as the barrels and stairs, that can be recognized by running a planar segmentation algorithm over a point cloud collected by an RGB-D camera.

It is possible for FastSLAM to use data from sensors such as a RGB-D camera in building and testing its maps. In future research we will investigate bypassing the laser and using the Kinect for mapping as well as visualization.

Even when mapping with a laser scanner it is possible to actively compare point clouds from an RGB-D camera to the laser-built map, so that the map and the point clouds can be better aligned.

Acknowledgment

This research was made possible thanks to the support of the Australian Center of Excellence for Autonomous Systems.

References

[1] (2012, Aug.) Robocup 2011 - istanbul. [Online]. Available: http://www.robocup2011.org

[2] J. Minguez, F. Lamiraux, and L. Montesano, "Metric-based scan matching algorithms for mobile robot displacement estimation," in *Proc. IEEE ICRA '05*, Barcelona, Spain, Apr. 18–22, 2005, pp. 3557–3563.

[3] A. Milstein, M. McGill, T. Wiley, R. Salleh, and C. Sammut, "Occupancy voxel metric based iterative closest point for position tracking in 3d enviroments," in *Proc. IEEE ICRA '11*, Shanghai, China, May 9–13, 2011, pp. 4048–4053.

[4] ——, "A method for fast encoder-free mapping in unstructured environments," *Field Robotics*, vol. 26, no. 6, pp. 817–831, Nov. 2011.

[5] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam: A factored solution to the simultaneous localization and mapping problem," in *Proc. NCAI '02*, Edmondton, Canada, Jul. 2002, pp. 593–598.

[6] A. Eliazar and R. Parr, "Dp-slam: Fast, robust simultaneous localization and mapping without predetermined landmarks," in *Proc. IJCAI '03*, Acapulco, Mexico, Aug. 2003, pp. 1135–1142.

[7] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA: MIT Press, 2005.

[8] G. Grisetti, C. Stachniss, and W. Burgard, "Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling," in *Proc. IEEE ICRA '05*, Barcelona, Spain, Apr. 18–22 2005, pp. 2432–2437.

[9] (2012, Aug.) Voronoi diagram. [Online]. Available: http://en.wikipedia.org/wiki/Voronoi_diagram

[10] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, no. 2, pp. 100 –107, Jul. 1968.

[11] (2012, Aug.) Fraunhofer iais: Volksbot. [Online]. Available: http://www.iais.fraunhofer.de/4821.html?&L=1

[12] (2012, Aug.) irobot corporation: Robots that make a difference. [Online]. Available: http://www.irobot.com

[13] M. Kadous, R. Sheh, and C. Sammut, "Effective user interface design for rescue robotics," in *Proc. Human Robot Interaction Conference*, Salt Lake City, USA, Mar.2–4 2006.

[14] R. Farid and C. Sammut, "A relational approach to plane-based object categorization," *RSS 2012 Workshop on RGB-D Cameras*, Jul. 2012. [Online]. Available: http://www.cs.washington.edu/ai/Mobile_Robotics/rgbd-workshop-2012/papers/farid-rgbd12-object-categorization.pdf