2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Multiple map hypotheses for planning and navigating in non-stationary environments

Timothy Morris, Feras Dayoub, Peter Corke, Gordon Wyeth and Ben Upcroft[1]

Abstract— **This paper presents a method to enable a mobile robot working in non-stationary environments to plan its path and localize within multiple map hypotheses simultaneously. The maps are generated using a long-term and short-term memory mechanism that ensures only persistent configurations in the environment are selected to create the maps. In order to evaluate the proposed method, experimentation is conducted in an office environment. Compared to navigation systems that use only one map, our system produces superior path planning and navigation in a non-stationary environment where paths can be blocked periodically, a common scenario which poses significant challenges for typical planners.**

I. Introduction

The main challenge in mapping non-stationary environments for mobile robots comes from the fact that the configuration of the environment can change in unpredictable ways. Therefore, the internal representation which the robot holds about the state of the surrounding environment can easily become invalid and out-of-date. The consequences of this fact can have catastrophic effects on the performance and the efficiency of the planning and navigation of the robot.

A naïve solution to this problem would be simply to rebuild the map from time to time. However, this would lead to a discontinuity in the work of the robot due to the need for repeatedly building a new map. In addition, some other agent, e.g. a human supervisor, would be needed to decide whether the robot's map needed to be rebuilt.

Another solution could be to maintain an off-line database of all past observations of an environment and then choose the best map which matches the current observed configuration of the environment. The problem with this approach is that mobile service robots are required to operate in real-time with finite computational resources, which means that incrementally increasing storage and search requirements could become overwhelming for the resources of the robot. Maintaining compactness in the robot's map requires a balance between learning and forgetting.

In this paper, we argue that the solution to the problem of operating inside a non-stationary environment for long periods of time should meet the following requirements.

- Maintaining accuracy.
- Simultaneous planning using multiple hypotheses.
- Bounded computational requirements.
- Maintaining stability.
- Delayed declaration of outliers.

[1] The authors are with the CyPhy Lab, Queensland University of Technology, Brisbane, Australia `firstname.lastname at qut.edu.au`

The most obvious requirement is to represent faithfully the true state of the environment. In a dynamic environment, the map should accurately reflect changes, while the accuracy of the map should increase with more sensor measurements even when the environment remains static.

The system should also be able to detect multiple configurations of its working place and generate multiple plan hypotheses. In this way the robot can switch between the plans instead of following only one plan which could lead to a blocked path.

The fact that the robot is required to work for long periods of time using its map, which needs to be updated over time, adds another dimension to the problem and introduces further complexity. In particular, actual changes in the environment appear in the first instance as outliers, which can only be identified after more time has passed and more measurements have been made.

Following the above requirements, the contribution of this paper is the introduction of a novel method that enables a mobile robot to localize itself and path plan over multiple map hypotheses simultaneously and switch between the plans and the maps on-the-fly. The hypotheses are generated using a map updating mechanism inspired by the concept of Short-Term Memory (STM) and Long-Term Memory (LTM). The updating mechanism enables the robot to detect different emerging configurations within an environment while filtering out spurious changes.

The rest of the paper is structured as follows. After an overview of the proposed memory model, Section II discusses related work. Section III describes our method for long-term adaptation and multiple map planning. Section IV presents the experiments and results obtained. Finally we draw conclusions in Section V.

II. Related Work

The problem of localization and planning in dynamic environments over a long-term operation has received an increase in attention recently. It is a challenging task as it requires the estimation of the robot pose in a non-static space by way of matching observation to a map which was built at an earlier time. In [1] the author classifies the dynamics in the environment as three elements:

- Static: Unchanging classification of observation.
- Semi-static: Change outside the sensory horizon.
- Dynamic: Change within the sensory horizon.

To segment observation into these elements, some approaches use temporal filtering techniques [2], [3].

The most common treatment of the dynamic elements is to consider them as outliers [4], [5], [6]. However, in [1] it is shown that good localization performance can be achieved over relatively short periods. But over a longer period of time, the semi-static and dynamic observations become noise. This noise can overshadow the available landmarks used for localization and the robot can become lost. Alternatively, other works tried to improve the robustness of the mapping process by detecting and tracking moving objects eparately [7], [8], [9], [10].

To address dealing with both static and semi-static change, [11] proposed a method of reasoning about configurations at an occupancy grid level. Clusters of occupied space were identified as configurations and used as sub-map's in their identified region. The position error in localization provided by these alternative configurations is measured against a ground truth. The measured results are used to demonstrate using a map that represents the current environment provides better localization performance then one that does not. This approach provides useful offline analysis of localization against a known ground truth, although no feedback mechanism is proposed to improve planning given navigation and localization using the current configurations.

III. Methodology

The proposed approach is broken up into two parts Fig. 1, Memory and Planning & Localization. Memory provides the configurations that can be used to construct hypotheses, for this work we use a two stage filtering process of Short Term Memory (STM) and Long Term Memory (LTM). The second part of the approach is the Planning & Localization which uses configurations provided by memory to construct hypotheses of the environment, selecting the best to act on while simultaneously tracking the others.

A. World representation

In this work we use a 3D OctoMap [12] representation for the world built using an RGBD point clouds. OctoMap is a probabilistic 3D occupancy grid with an octree data structure. The octree data structure is a tree with nodes that split the parent node into eight voxels of equal sizes. The leafs of the tree contain probability of occupancy of the space at a minimum resolution size. These proprieties makes Octomap compact and easy to update while accounting for the uncertainty in sensory measurement.

The probability that a leaf n is occupied given the sensor measurements $z_{1:t}$ are computed using a recursive binary Bayes filter [13]:

$$P(n|z_{1:t}) = [1+\frac{1-P(n|z_t)}{P(n|z_t)}\frac{1-P(n|z_{1:t-1})}{P(n|z_{1:t-1})}\frac{P(n)}{1-P(n)}]^{-1}, \quad (1)$$

Where $P(n|z_t)$ is the probability that voxel n is occupied given current measurements z_t, $P(n|z_{t-1})$ is the previous estimate and $P(n)$ is a prior probability.

For faster performance, a log-odds representation is used in the update equation:

$$L(n|z_{1:t}) = L(n|z_{1:t-1}) + L(n|z_t), \quad (2)$$

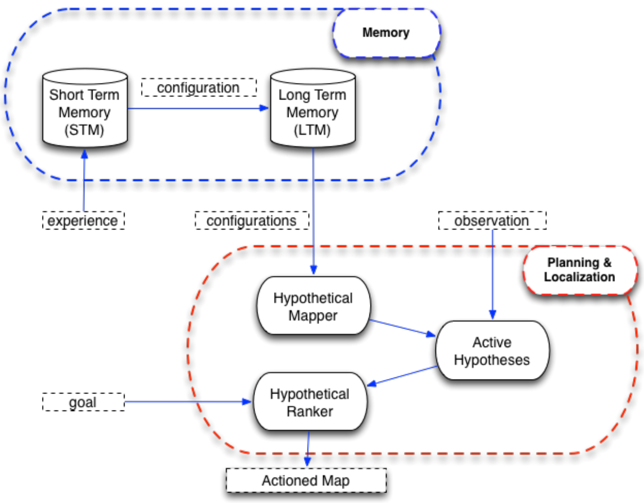

Fig. 1. The interaction between Memory and Localization & Planning. New points are added to STM in blocks associated with an Experience. Points detected to migrate from STM to LTM are flagged as configurations. Map hypotheses are constructed using the currently available configurations from LTM. Hypotheses are simultaneously tracked by replicating observation information to active localizers. The best ranked hypothesis at any moment is promoted to the actioned map, which is used for guiding the robot to the goal location.

where $L(n) = \log \frac{P(n)}{1-P(n)}$. Finally, in order the following clamping update policy is used to ensure a bounded occupancy estimation:

$$L(n|z_{1:t}) = max(min(L(n|z_{1:t-1}) + L(n|z_t), l_{max}), l_{min}), \quad (3)$$

where l_{min} and l_{max} are the lower and upper bound of the log-odds value in the map.

When working in a changing environment, Octomap can update the map very quickly when the robot observes a change whether this change is temporary (e.g. Crowded area) or a permanent change. Although, this might be a good property when mapping a tabletop area for a robotic hand, it can corrupt the map if the update happens during a period of bad self-localization or when the environment is crowded with people.

Another issue with the updating policy of Octomap is that the map cannot represent multiple configuration of the world (e.g. Door open and doors close). This information can be very useful for path planning as we show in section III-D.

In Section III-B we explain how to generate a long- and short memory representation for the environment by modifying the clamping update policy in Eq. 3 so only persistence changes is reflected in the map as long as a multiple configuration of the environment.

B. Short- and long-term memory stores

This work extends a map updating mechanism introduced in [14] to work with dense 3D point clouds. In [14] the LTM–STM filter is used to update spherical views built from omnidirectional images as the nodes to a pose-graph map. The extension to 3D point clouds allows the robot to navigate

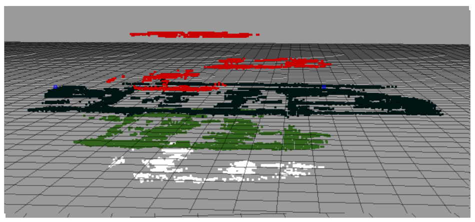
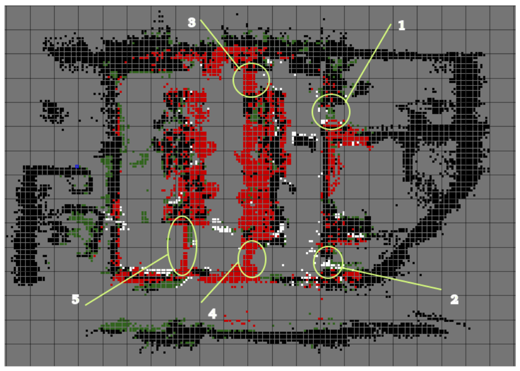

Fig. 2. Top: side view for a different space configuration extracted from LTM: Black: the freshest content in LTM. Green: decaying information. White: forgotten information. Red: different configuration of the place. Bottom: Top view with all the information of LTM: areas 1 and 2 : Doors were closed and then were open. areas 3,4 and 5: three different configurations of the environment representing blocked passages.

in a metric space instead of the previously used topological space.

The updating mechanism uses rehearsal, recall and forgetting process on LTM and STM stores. The information stored in LTM represent the robot's inner representation about the state of its environment. Through the process of rehearsal, information in STM can be committed to LTM to be retained for longer periods of time. In turn the LTM is used to detect novelty in the robot's current view of the world, which are then committed to STM to be rehearsed.

The LTM and STM are built as two Octomaps. An initial mapping stage is used to initialize the LTM, whereas the STM starts as empty. All occupied voxels in LTM are initialized with log-odds value of l_{max} and are updated according to the following policy:

$$L(n|z_{1:t}) = \begin{cases} L(n|z_{1:t-1}) + L(n|z_t), & \text{if } z_t \text{ is a miss} \\ l_{max}, & \text{if } z_t \text{ is a hit} \end{cases} \quad (4)$$

When new points are added to STM, their voxels are initialized with log-odds value of l_{min} and updated according to the following policy:

$$L(n|z_{1:t}) = \begin{cases} l_{min}, & \text{if } z_t \text{ is a miss} \\ L(n|z_{1:t-1}) + L(n|z_t), & \text{if } z_t \text{ is a hit} \end{cases} \quad (5)$$

'Miss' means a ray traced through the voxel. 'Hit' means a ray has ended in the voxel. These updating policies ensure that information in LTM decay slowly and only persistence information is transferred from STM to LTM.

The updating mechanism starts by recalling the content of the LTM using the current view of the world according to the policy from Eq. 4. Then, any points that exist in the current view and not in LTM are used to rehearse the content in STM using the policy from Eq. 5. After that, all voxels in STM with a log-odds values above a certain threshold are transferred to LTM. Finally, all voxels in LTM with a log-odds below certain threshold are deleted (forgotten) from the world representation.

Instead of using the instantaneous point cloud from the sensor to update the map directly, we register instantaneous point clouds over a predefined distance and use this partial view of the world to update the map. In this way we prevent a rapid rehearsal in STM in situations where the robot is turning on the spot due to people blocking its path or when it is standing still.

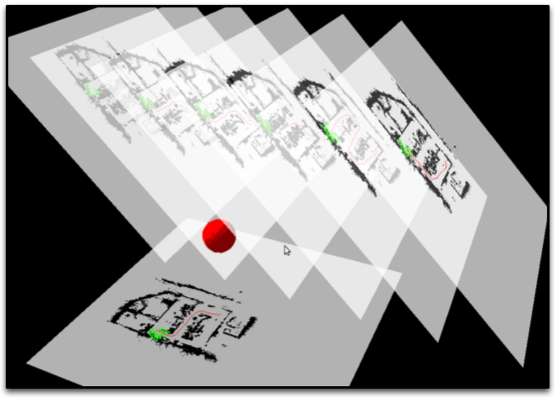

Fig. 3. A path is executed on the most likely map configuration, while other configurations are tracking their individual localization performance. Navigation metrics collected during each experience re-weight the various configurations such that maps that provide good localization and environment prediction are ranked most likely when a new plan is required. Should an alternative map configuration provide greater localization performance and contain a valid plan its plan can replace the current map and plan as the new primary configuration.

C. Multiple space configurations

In order to detect emerging configurations in the environment, we use the migration of occupied voxels that make the transformation from STM to LTM. Each time a transfer occurs between STM and LTM, the new voxels are grouped in LTM as one configuration. These configurations are subject to the same decay policy that governs LTM. Fig. 2 shows the content of LTM at one instance of time projected into multiple layers. The layer in black represents the freshest information about the world (i.e. the voxels with log-odds value of l_{max}). The Green layer is decaying information and the white is forgotten information. The figure also shows three layers of red. Each layer represents a different configuration in the world.

D. Planning and localization on multiple maps

Configurations labelled by their migration into LTM are used to construct various occupancy maps *(Fig. 5)*. Each set of configurations chosen represent a hypothesis of the expected world, whose performance is then measured through applied navigation.

During navigation, confidence in pose estimates will fluctuate in areas where the hypothesis does not reflect the observed world. This fact is used to inform hypothesis selection. In order to measure the confidence in pose estimates, an L2-norm of the localization covariance matrix is used. The localizer in our case is a particle filter localizer [15]. In addition, hypotheses that produce plans conflicting with the world encountered during navigation are also penalized.

Active tracking of navigation performance in a large set of hypotheses can become in-tractable. In order to prevent this, we divide the hypotheses into two sets, active and pending. The active set contains the N best hypotheses which are simultaneously tracked using individual particle filters during navigation. Alternative hypotheses remain in a pending set until an active hypothesis falls below the best pending weight.

Pending hypotheses are assigned an initial weight in case they are moved to the active set. Active hypotheses that are swapped out for a pending hypotheses retain their last known weight and a starting pose of the best active hypothesis. The rate at which hypotheses in the pending set are accessed can be a factor of their initial value, we experimentally found a value of 0.3 to be sufficient for the tested memory configurations and environment.

Algorithm 1 details the process of tracking active hypotheses during navigation. Here we use H for the set of active hypotheses being tracked by current observation and H_a the *actioned* hypothesis used at present to guide the robot. During navigation, should the measured localization uncertainty (L2-norm of covariance) exceed the best available hypothesis in the active set, we re-evaluate our hypotheses to select a new H_a. Before switching, we apply a penalty to the accumulated score of the hypothesis associated with H_a if it was blocked by unexpected obstacles. If a penalty is applied, any hypothesis that is occupied along the action plan predicting visible obstacles, receives a reward to its accumulated weight.

Algorithm 2 details how the sets are maintained during switching. Here we use H_* to represent the pending set for which localization is not tracked. Should an active hypothesis be weighted below the best pending hypothesis they are swapped.

The weights assigned to hypotheses within the pending set do not give information on the localization performance for the present location. This is because the weight is either initial or from when the hypothesis was last active. This value might have been set from anywhere in the map. We therefore condition the swap of pending hypotheses into the active set based on their ability to produce a valid plan. This is unique to the pending set as the active set is tracking the performance at the present location.

```
Plan to goal;
while navigating to goal do
    if $H_a > min(H)$ then
        if plan($H_a$) crosses visible obstacles then
            penalise $H_a$;
            for $H$ do
                if $H_i$ predicted visible obstacles then
                    reward $H_i$;
                end
            end
        end
        goto Hypothesis selection algorithm;
    end
end
```

Algorithm 1: Navigation

```
Sort ascending $H$
for $H$ do
    if $H_i > min(H^*)$ then
        Add $H_i$ to $H^*$;
        Pop $min(H^*)$ to $H_i$;
    end
    if plan($H_i$) is valid and $H_a$ is empty then
        Let $H_a = H_i$
    end
end
if $H_a$ is empty then
    Let $H_a = min(H_i)$
end
goto Navigating algorithim;
```

Algorithm 2: Hypothesis selection

IV. Experimental evaluation

The experiment took place in an office environment. The experimental platform was a MobileRobots' Research GuiaBot. The robot is equipped with a Kinect which provides the 3D point clouds. The system is built under ROS (Robot Operating System). The first map of the environment was built using Octomap server, a ROS nodes provided by [12]. This map is then used to initialize the LTM.

The experimental evaluation is done in two stages: first, point clouds collected by the robot during navigation is used to populate the LTM and generate the multiple configuration hypotheses. Second, the LTM is used to test the planning and navigation while switching between the maps.

A. Populating the LTM

The office environment where the experiment took place is of size $16m \times 20m$. The robot performed 25 episodic navigation tasks where it starts from one side of the map (the top left corner in Fig. 2) and navigate towards the other side (the bottom right corner in Fig. 2). Each time it reaches its goal, all point clouds generated during its navigation are registered to make a current view of the world which is used to update LTM and STM.

While the robot was navigating, the environment was changed manually by using office dividers to block the robot's path and to force the robot to change it's plan to the goal. As well as using dividers to change the structure of the environment, two people were also present and moved around the robot, also chairs where displaced continuously.

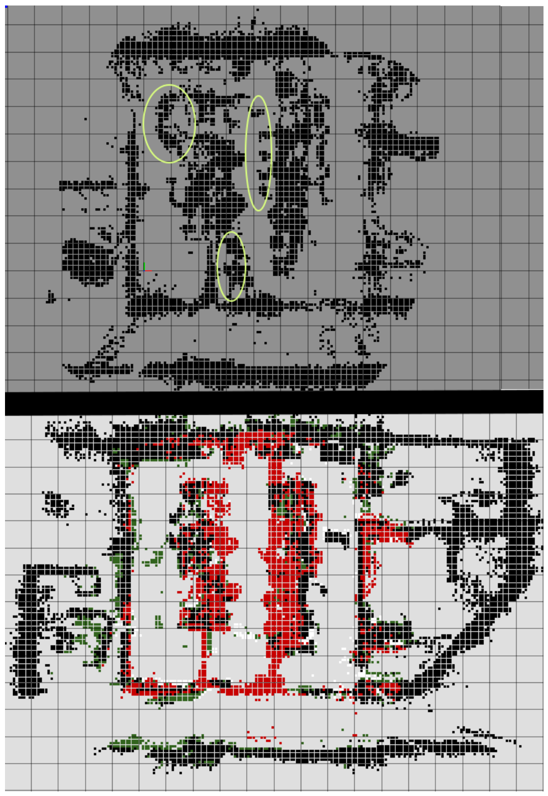

Fig. 4. Top: A current view from one of the navigation runs. The circled areas shows information generated from moving persons and chairs. Bottom: The content of LTM.

However, the change in position of dividers was much slower than the moving people and chairs. Table I contains the parameters used for the LTM and STM updating policies according to Eq. 1 ,Eq. 4 and Eq. 5. These parameters effect the rate by which information added from the current view to the map. As Fig. 2 shows, different configurations

	l_{min}	l_{max}	P_{hit}	P_{miss}
STM	0.40	1.28	0.7	0.4
LTM	-2.9	4.59	0.7	0.4

TABLE I

THE PARAMETERS USED FOR THE UPDATING POLICIES IN LTM AND STM

resulting from the use of office dividers where present in LTM whereas no information from moving people (shown in Fig. 4) migrated from STM to LTM.

B. *Navigation Experiment 1*

In this experiment the robot is tasked with navigating from ‘A’ to ‘B’ *(Fig. 5)* between the corners of the world. It is assumed at this stage that the robot has already been ‘exposed’ to multiple configurations of the world and has accumulated its experiences in the LTM. This experiment gives focus to various configurations taken from LTM and how they perform for a new navigation task. Experimentation

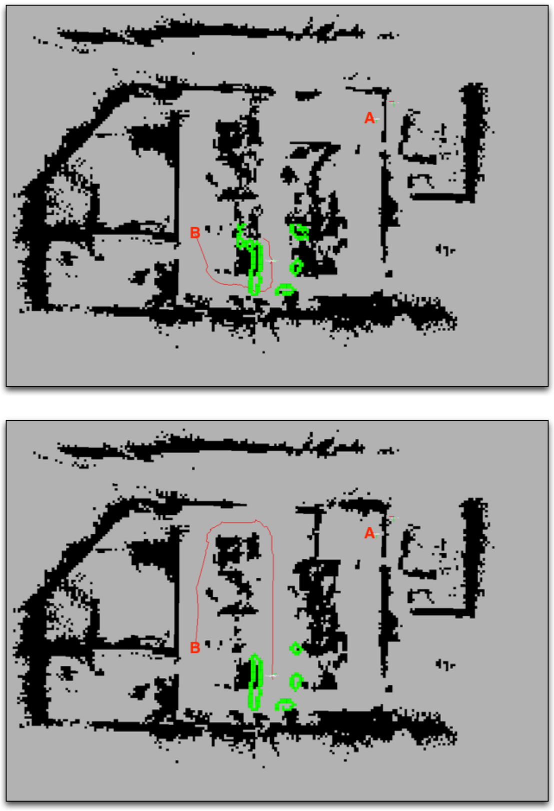

Fig. 5. Two different map hypotheses generated from LTM. The robot was required to navigate from ‘A’ to ‘B’. The top map is initially used to plan to ‘B’ in both single and multiple approaches. Upon reaching a blockage shown in green, the single hypothesis approach can not re-plan to the goal because there is no other alternative route. In contrast the multi hypothesis approach found a map where a path exists as shown in the lower map.

identifies peak performance of various configurations which is used to inform hypothesis switching in Section IV-C.

As shown in Fig. 2 the LTM contains 4 different configurations. These configurations produce 16 combinations each providing a possible map hypothesis. From this set of 16 hypotheses, 3 produced unusable maps that block the proposed goal and starting location from all approaches. Table II shows the localization performance inside multiple map hypotheses form a single navigation run. During that navigation, the world configuration was represented best by Map 2 producing the highest confidence in pose estimation.

	Map 1	Map 2	Map 3	Map 4	Map 5	Map 6
Mean L2-Norm	0.0450	0.0381	0.1190	0.1489	0.0333	0.0423

TABLE II

PERFORMANCE OF INDIVIDUAL HYPOTHESES DURING A SINGLE NAVIGATION RUN WHO’S SETUP CLOSELY MATCHES ‘MAP 2’.

C. *Navigation Experiment 2*

In contrast to Experiment 1 we evaluate multiple configurations simultaneously during navigation, ranking them by performance and switching between the most likely

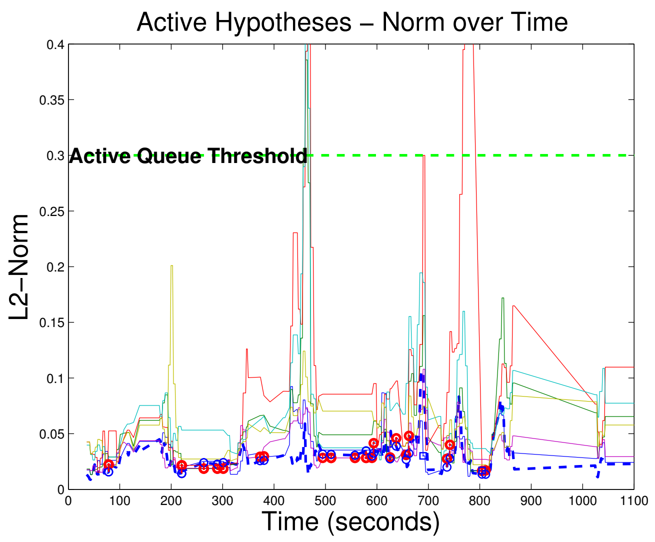

Fig. 6. Navigation performance using 6 active map hypotheses. The green dotted horizontal line represents the threshold at which active hypotheses are substituted for a pending hypotheses. The blue dotted line tracks the confidence in to robot's current estimate of pose used for executing the navigation. Red dots mark hypothesis switching events.

candidates. The multi hypothesis approach is trialled using multiple real world configurations and goals, switching and re-planning as needed. The experiment demonstrates efficient navigation and localization within multiple real world configurations, challenging for single representations as seen in Section IV-B and as shown in Table III. The table demonstrates that compared to navigation systems that use only one map, our system produces better navigation.

Fig. 6 tracks the performance of the active hypotheses for a single navigation run. The green dotted horizontal line represents the threshold at which active hypotheses are substituted for a pending hypotheses. The figure shows that by switching between map hypotheses, the robot maintains a better localization performance compared to each of the hypotheses.

	Run 1		Run 2		Run 3	
Mean	Multi	Single	Multi	Single	Multi	Single
L2-Norm	0.0267	0.1082	0.0333	0.0580	0.0349	0.1489

TABLE III

COMPARISON OF PERFORMANCE BETWEEN SINGLE AND MULTIPLE HYPOTHESES DURING VARIOUS NAVIGATION RUNS.

V. CONCLUSIONS AND FUTURE WORK

This paper introduced a method to enable a mobile robot to plan and navigate using multiple map hypotheses simultaneously while operating inside a non-stationary environment. The map hypotheses are generated using a long-term and short-term memory updating mechanism. The paper demonstrated how to switch between multiple map hypotheses while navigating in environments where paths can be blocked periodically. When compared to approaches that plan using a single representation, the presented method produced a better localization and planning performance. In order to discretise large environments and plan using multiple hypotheses on a topological as well as a local metric level, future work will look at incorporating a topometric representation [16].

ACKNOWLEDGMENTS

This research was supported under Australian Research Council's Discovery Projects funding scheme (project number DP110103006).

REFERENCES

[1] M. Hentschel and B. Wagner, "An adaptive memory model for long-term navigation of autonomous mobile robots," *Journal of Robotics*, vol. 2011, 2012.

[2] G. D. Tipaldi, D. Meyer-Delius, M. Beinhofer, and W. Burgard, "Lifelong localization and dynamic map estimation in changing environments," in *RSS Workshop on Robots in Clutter*, 2012.

[3] J. Saarinen, H. Andreasson, and A. J. Lilienthal, "Independent markov chain occupancy grid maps for representation of dynamic environment," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 3489–3495.

[4] J. Dong, S. Wijesoma, and A. Shacklock, "Extended rao-blackwellised genetic algorithmic filter SLAM in dynamic environment with raw sensor measurement," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Diego, USA, Oct. 29 - Nov. 2 2007, pp. 1473–1478.

[5] Z. Liu, W. Chen, Y. Wang, and J. Wang, "Localizability estimation for mobile robots based on probabilistic grid map and its applications to localization," in *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2012 IEEE Conference on*, Sept., pp. 46–51.

[6] D. Hahnel, R. Triebel, W. Burgard, and S. Thrun, "Map building with mobile robots in dynamic environments," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, Taipei, Taiwan, September 14-1 2003, pp. 1557–1563.

[7] D. Anguelov, R. Biswas, D. Koller, B. Limketkai, S. Sanner, and S. Thrun, "Learning Hierachical Objects Maps of Non-Stationary Environments with Mobile Robots," in *Proc. Conference on Uncertainty in Artificial Intelligence (UAI)*, Edmonton, Canada, August 2002, pp. 10–17.

[8] C. C. Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte, "Simultaneous localization, mapping and moving object tracking," *The International Journal of Robotics Research*, vol. 26, no. 9, p. 889, 2007.

[9] S. Ortega, "Towards visual localization, mapping and moving objects tracking by a mobile robot: a geometric and probabilistic approach," Ph.D. dissertation, Institut National Polytechnique de Toulouse, 2007.

[10] D. Migliore, R. Rigamonti, D. Marzorati, M. Matteucci, and D. G. Sorrenti, "Use a single camera for simultaneous localization and mapping with mobile object tracking in dynamic environments," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, May 12-17 2009, pp. 27–32.

[11] C. Stachniss and W. Burgard, "Mobile robot mapping and localization in non-static environments," in *Proceedings of the National Conference on Artificial Intelligence*, vol. 20, no. 3. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005, p. 1324.

[12] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: an efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, pp. 1–18, 2013.

[13] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[14] F. Dayoub, G. Cielniak, and T. Duckett, "Long-Term experiments with an adaptive spherical view representation for navigation in changing environments," *Robotics and Autonomous Systems*, vol. 5, pp. 285–295, 2011.

[15] D. Fox, "KLD-sampling: Adaptive particle filters," in *In Advances in Neural Information Processing Systems 14*. MIT Press, 2001, pp. 713–720.

[16] F. Dayoub, T. Morris, B. Upcroft, and P. Corke, "Vision-only autonomous navigation using topometric maps," in *IEEE/RSJ International Conference on Intelligent Robots and Systems November 3-8, 2013 at Tokyo Big Sight, Japan*, 2013.