Using Incomplete Online Metric Maps for Topological Exploration with the Gap Navigation Tree

Liz Murphy and Paul Newman, Oxford Mobile Robotics Research Group
{liz,pnewman}@robots.ox.ac.uk

***Abstract*— This paper presents a general, global approach to the problem of robot exploration, utilizing a topological data structure to guide an underlying Simultaneous Localization and Mapping (SLAM) process. A Gap Navigation Tree (GNT) is used to motivate global target selection and occluded regions of the environment (called *"gaps"*) are tracked probabilistically. The process of map construction and the motion of the vehicle alters both the shape and location of these regions. The use of online mapping is shown to reduce the difficulties in implementing the GNT.**

I. INTRODUCTION

The goal of this work is to facilitate the exploration of an unknown environment by a single robot system under uncertainty. Using an existing SLAM system as a basis, we overlay a topological data structure which motivates exploration by identifying occluded regions and seeking to add them to the map. The location of these regions is tracked probabilistically.

Exploration occurs when a robot is placed in an unknown environment and is asked to construct a map, which can be used for subsequent navigation, as it moves through the world. The decision as to *where to go next?* is informed only by data contained in the partially complete map. In this work it is the presence of gaps in the robot's field of view that motivates exploration. The resulting robot trajectory covers the local environment in a structured, deterministic manner.

To achieve this we take an algorithm that has been shown to produce optimal navigation paths in unknown planar environments and adapt it for real world implementation. In [1] [2] [3] the use of a data structure called the *Gap Navigation Tree* (GNT) enables locally optimal navigation in terms of Euclidean distance travelled in the exploration of a simple planar environment. Underpinning the algorithm is the assumption of the availability of an abstract *gap sensor*, able to track and uniquely identify discontinuities in the depth of the range information at the robot's current viewpoint. These discontinuities are of interest as they correspond to regions not yet visible to the robot.

This paper presents an extension to the GNT algorithm and an implementation of the gap sensor. It directly addresses the issue of exploration under uncertainty. We adapt the GNT algorithm so that it is suitable for use in motivating next-target selection in an exploratory SLAM system. Experimental results from simulations are presented, showing that this approach results in coverage of indoor environments given a robust implementation of a gap tracker based on data from a 2D Laser scanner - certainly not an "ideal infinite resolution sensor".

II. RELATED WORK

Previous approaches to exploration within the context of SLAM can be broadly characterized as *grid based*, *feature/view based* or *topological*.

Grid based methods provide a continuous metric representation of the environment in which each grid cell stores the probability of a cell being occupied by an obstacle. Many methods inherit from the original *frontier based* approach of [4], where the robot is repeatedly directed toward the nearest frontier between open explored space and unexplored space. In contrast, the grid based approach in [5] [6] decomposes the entropy of the SLAM posterior into two terms, one representing the entropy in the pose posterior and the other representing the expected entropy of the map averaged over all paths. Possible control sequences are evaluated by adding the two entropy terms and selecting the control that minimises the resulting entropy (or uncertainty). This algorithm exhibits loop closing behaviour to improve the pose estimate, as well as seeking to explore unmapped terrain.

Feature/View based methods use landmarks extracted from the environment to guide exploration. In [7] the landmarks generate goals for exploration which are then biased towards enabling local exploration of sparse regions. A utility function is used in [8] to trade off between information gain, the cost of moving to the next sensing location, and the utility of localization based on the covariance matrix in selecting the next pose. Information gain is also the motivating factor in [9] where minimizing the trace of the covariance matrix and hence the average uncertainty of the SLAM map is used as the objective function for exploration. Possible next sensing locations are drawn from a discretized grid and a maximally informative trajectory is chosen by integrating observations along the trajectory to a particular point.

Topological exploratory strategies [10] model the structure of the environment using a graph. Although providing a compact representation, the lack of metric information in a strict topological representation makes localization extremely difficult. Hence most topological approaches are hybrid methods which also incorporate geometric maps [11].

All three approaches have significant disadvantages: feature based methods are reliant on the use of feature detection to extract uniquely identifiable landmarks from the environment during mapping. Such landmarks are not always present. Grid based methods do not scale well to large environments, and topological mapping hinders localization. By combining the topological approach of the GNT with the

view-based SLAM system we seek to ameliorate some of these problems.

Outside of SLAM but akin to the sensing strategy presented in this paper, Landa et al [12] [13] present a theoretical approach to exploration focused on the use of point clouds from a laser scanner to detect gaps in the environment. They present an exploration strategy similar to Frontier Based Navigation in support of the validity of their method.

III. GAP NAVIGATION TREE

We shall now briefly summarize the GNT of [1] [2] [3] as it is central to our work. The algorithm constructs a topological representation of the environment in the form of a tree with the aid of an abstract *gap sensor*. The gap sensor reports the cyclical order of depth discontinuities of the boundary relative to the robot's current position. In [2] the authors suggest a number of possible physical implementations of the sensor, including sonars, cameras, a laser pointer used in conjunction with an omnidirectional camera, and their own implementation using two laser range finders. Each of the gaps returned by the sensor should correspond to a region of free space that is occluded, and it is assumed that the gap sensor is able to track and distinguish between the gaps at all times, even when the robot moves across a non-smooth part of the boundary and the location of the gap jumps discontinuously. Importantly, no geometric information is returned by the gap sensor, the gap sensor output is simply an ordered list of unique identifiers assigned to each gap.

The robot is equipped with a single motion primitive which enables it to rotate itself toward the location of a gap and approach the gap at a constant speed. This is referred to as *chasing* the gap. The *chase gap* operation can only terminate when the gap disappears from the gap sensor.

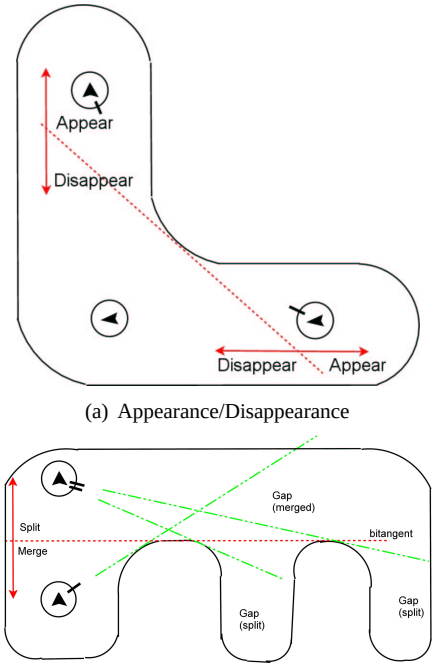
Construction of the GNT begins with the addition of *n* nodes as children of the root of a tree, where *n* is the number of gaps returned by the gap sensor in its initial observation of the environment - one node for each gap. As the robot moves through the environment, changes to the tree are triggered by the occurrence of *gap critical events*. These events occur when the robot crosses either a *generalized inflection* or a *generalized bitangent* of the environment boundary. As illustrated in Figure 1, the *appear* and *disappear* events are associated with crossing a line of inflection, and crossing a bitangent line will trigger a *split* or *merge* of two gaps, depending on the direction of crossing. Each event requires updating of the GNT:

Appear A node *g* is added as a child of the root node, its neighbouring nodes are those that it currently neighbours in the gap sensor.

Disappear By definition any disappearing gap must be represented by a leaf node. The node is removed from the tree.

Split If the gap *g* which has split into gaps g_1 and g_2 is a leaf node, then two new vertices are added to the tree in place of *g*. Otherwise, they must already exist in the tree as children of *g* and become children of the root node upon the removal of *g*.

Merge When gaps g_1 and g_2 merge into gap *g*, *g* is added to the tree as a child of the root node, preserving the order of the gap sensor. The existing nodes g_1 and g_2 become children of *g*.



(a) Appearance/Disappearance

(b) Splitting/Merging

Fig. 1. Gap Visual Events: The red dotted lines in both (a) and (b) extend from the point(s) of inflection on the environment boundary. When the robot (triangular marker) crosses this line gaps appear or disappear depending on the direction of travel. The green dotted lines in (b) project the line of sight to observable gaps. The marked dash on the circle surrounding the robot indicates the direction of the observed gap.

These four operations are sufficient to represent all feasible changes to the environment. Although more complex tritangents may exist, both [2] and [14] argue that tritangents would not survive even a small deformation of the environment whereas the bitangent or inflectional tangent would merely change position. In [14] *generic* curves and surfaces are defined as not being affected by small deformations, and it is stated that the boundaries of all real objects are generic.

At any time τ, the children of the root of the GNT reflect the gaps currently visible to the gap sensor. Any other gaps in the tree were visible at some time $t < \tau$ but are now obscured due to merging.

Nodes in the tree are only used to represent gaps, and gaps fall into two categories. *Non-primitive* nodes are used to motivate exploration, they correspond to unexplored occluded regions and arise as a result of appearing in the gap sensor's initial observation, or from the *splitting* of one of the initial nodes or its non-primitive children. *Primitive* nodes are added to the tree as a result of a previously visible area becoming occluded and hence causing a gap to reappear. Chasing a primitive gap will only result in revisiting previously covered territory.

A *complete* GNT encodes a path from the robot's current position to any other location in the environment. To construct the complete GNT the robot must achieve sensor coverage of the entire environment, and the GNT is forced into completeness by iteratively chasing the non-primitive leaves of the tree until they, and any children that result from their subsequent splitting into other gaps, disappear. When a leaf disappears, another non-primitive leaf is selected for chasing and the procedure is repeated until all leaves are primitive.

We believe the simplicity of the GNT approach to exploration means the algorithm is well suited for use in next action selection when used in conjunction with a metric SLAM system. As outlined in [2], the most important practical issue prohibiting the use of the GNT in a real setting lies with the gap tracking process. Implicit in the GNT algorithm is the assumption of the perfect gap sensor/gap tracker. Sparse sampling and non-smooth environment boundaries as illustrated in Figure (5) are common examples of cases where gaps may be falsely detected or prove difficult to track, especially with a simple 'one-shot' sensor. Should the gap sensor not prove robust to these situations, the robot will fail to detect important visual events leading to malformations of the tree structure, or causing the robot to chase the 'wrong' gap. The GNT does not have a probabilistic basis and our experience shows it to be fragile when driven with real sensors.

This paper details an implementation of the gap sensor which utilizes a partially-complete SLAM map a basis for gap detection and tracking. The success of view-based SLAM methods, in particular those based on mobile robot systems equipped with laser scanners, has lead us to focus on the application of the GNT to a SLAM system producing point cloud maps. A SLAM system using point clouds will still lead to the sparse sampling of some surfaces, depending on the current orientation of the sensor relative to the surface and also due to the sensor's distance from the surface. However, we show the use of a SLAM map not only reduces the number of *faux gaps* due to sparsity, but also allows us to derive a model for expected gap motion hence allowing for robust tracking.

IV. GAP SENSOR IMPLEMENTATION

A. Gap Detection

Before gaps can be tracked, we must first define a capability that allows the detection of gaps in the accumulated SLAM map. We introduce a visibility based formulation similar to [12] [13].

To compute the visibility map of Figure 2 we take a point cloud **P** sampled from the environment Ω. We limit the view direction from the current vehicle position x_0 to the set of angles $\theta \in [0, \theta_d, 2\theta_d ... 2\pi)$where θ_d is the desired angular resolution of our gap sensor. The view direction from the vehicle to any given point x is calculated as

$$v\left(x_0, x\right) = \left(\frac{x - x_0}{|x - x_0|}\right)$$

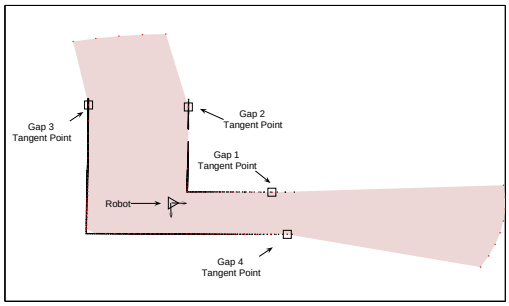


Fig. 2. A Visibility Map showing an approximation to the area visible from the robot's current position. The visible region is shaded in red.

We then define the visibility function ρ_{x_0} which provides a piecewise constant approximation to the visible region at the robot's current viewpoint.

$$\rho_{x_0}\left(\theta\right) = \begin{cases} min_{x \in \Omega}\{|x - x_0| : v\left(x_0, x\right) = \theta\} & \text{if exists} \\ r_{max} & \text{otherwise} \end{cases}$$

where r_{max} is the maximum range of the sensor.

The visibility map is then differentiated to locate gaps in the visible region. The differential is evaluated only at points $\theta \in [0, \theta_d, 2\theta_d ... 2\pi)$ so we arrive at a piecewise constant approximation to the differentiated visibility function.

$$\frac{d\rho_{x_0}}{d\theta} = \frac{\rho_{x_0}\left(\theta_{k+1}\right) - \rho_{x_0}\left(\theta_k\right)}{\theta_{k+1} - \theta_k}$$

Gaps are classified as existing at those locations where $\frac{d\rho_{x_0}}{d\theta}$ exceeds a threshold **T**. Note that this threshold is set to be dependent on characteristics of the environment (say known doorway widths or corridor widths in an indoor environment, or spaces between buildings in an urban setting). Prudent selection of this threshold aids in limiting the appearance of *spurious gaps*. We assume each gap can be uniquely parameterized by a *tangent point* $[x_g, y_g]$; the closest point on the boundary which intersects a line drawn in the view direction from the robot's current position. Figure 3 illustrates the gap detection process using the visibility and difference functions together with the gap detection threshold.

It is important to note that due to the discretization of the data and the discontinuous representation of the environment offered by a point based SLAM map, the gap detector often returns gaps in locations when there are none. For this reason, we employ the notion of persistence (explained in Section IV-C) before adding gaps to the gap tracker or gap navigation tree.

In addition to the tangent point, the gap detector also records whether the gap results from a discontinuity which stretches from a near point on the environment boundary to a distant point or conversely from far to near, relative to the anti-clockwise direction zeroed at the robot's current heading. This information enables us to identify whether the

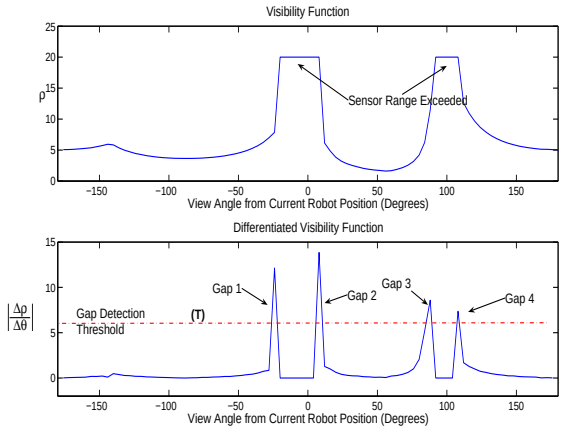


Fig. 3. Visibility Function (top) and Difference Function (bottom) corresponding to Figure (2). In this example four gaps are detected.

occluded region lies to the left or right of the tangent point, aiding in selection of the best viewpoint to travel to next and expediting the gap's disappearance.

B. GAP TRACKING

To introduce our gap tracking algorithm we present the gap tracking problem from a probabilistic point of view. We are interested in maintaining the distribution $p(g_k|D^k, m_k)$, where g_k represents the location $[x_g, y_g]$ of gap g at time k given the history of our gap detector outputs D^k and the matrix of points m_k representing the current state of the map.

We assume that our gap tracking problem is a Markov process, such that knowledge of the state at $k-1$ provides all the information needed to propagate the state at k. This allows us to write the *gap motion model* $p\left(g_k|g_{k-1}, m_k\right)$ as a probabilistic models adhering to the Markov property. We also define the *gap observation model* $p\left(d_k|g_k, m_k\right)$ which defines the likelihood of the next measurement d_k given the current state of the map and the last known gap position. The graphical model of Figure 4 illustrates the relationship between the map, measurements and state of the gaps.

Given the current state of the map at time k and the set of previous observations D^{k-1}, we can write an expression which predicts the position of the gap at time k (prior to making the observation d_k at time k):

$$\begin{aligned} p\left(g_k|D^{k-1}, m_k\right) &= \\ &= \int p\left(g_k|g_{k-1}, m_k\right) \times \\ &\quad p\left(g_{k-1}|D^{k-1}, m_k\right) dg_{k-1} \end{aligned} \tag{1}$$

Once we have the relevant measurements we are able to update the state representation:

$$\begin{aligned} p\left(g_k|D^k, m_k\right) &= \\ &= \frac{p(d_k|g_k, m_k)p(g_k|D^{k-1}, m_k)}{p(d_k|D^{k-1}, m_k)} \end{aligned} \tag{2}$$

Combining (1) and (2) allows us to write

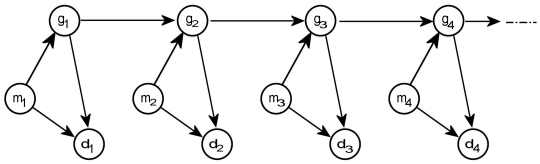


Fig. 4. A graphical model showing the relationship between the measurements, map and gap position

$$\begin{aligned} &p(g_k|D^k, m_k) = \\ &\frac{p(d_k|g_k, m_k)\int p(g_k|g_{k-1}, m_k)p\left(g_{k-1}|D^{k-1}, m_k\right)dg_{k-1}}{p(d_k|D^{k-1}, m_k)} \end{aligned} \tag{3}$$

The above equation is recursive and allows well known (Kalman) closed form updates if we assume gaussian distributions.

Critically important to our gap tracking solution is the gap motion model $p\left(g_k|g_{k-1}, m_k\right)$. This function represents the gap's change in position from time $k-1$ to k. We assume it is a gaussian of the form

$$p\left(g_k|g_{k-1}, D^{k-1}, m_k\right) \sim N(g_{k-1}, P_{k-1} + \Gamma(g_{k-1}, m_k)) \tag{4}$$

where Γ is a function parameterized by the gap and the SLAM map and P_{k-1} is the gap's covariance at time $k-1$. Γ is expressed as

$$\Gamma(g_k, m_k) = VD\Lambda V^T \tag{5}$$

where V is a rotation matrix and Λ is a diagonal matrix of eigenvalues resulting from the eigenvalue decomposition of the covariance matrix of the nearest neighbours, calculated according to

$$V\Lambda V^T = cov[K(m_k, g_k)] \tag{6}$$

$K(m_k, g_k)$ is the neighbourhood operator on m_k, returning the n points in the map which are closest to the estimated gap position g_k. D is a diagonal scaling matrix.

While *genuine* gaps that appear in the gap sensor as a result of generalized inflections and bitangents discussed in Section III can be adequately represented and tracked using a circular covariance function, we are also required to maintain representations of *faux gaps* that result from the limited range of our sensor. These gaps tend to jump with the arrival of new data. Take for example a gap marking the extent of our vision along a wall. As the next chunk of data comes in, we see an extra *x* metres of the wall and the gap moves by *x* metres in accordance.

Note that as these tangent points are located at the map extremity, they are also surrounded by few data points. We allow the local map shape and point cloud density to dictate the motion of perceived gaps by examining the nearest neighbours of the tangent point (within radius *R*), calculating the covariance of these points as in (6) and exaggerating the largest axis with an appropriate choice of scaling matrix D in (5). This method, when applied to *genuine* gaps tends

to produce ellipses of small axes due to the agglomeration of data corresponding with their typical physical presence as corners in a structured environment and reflecting the belief that these gaps are unlikely to move when next observed. In contrast, our *faux gaps* tend to lie on the visible extremity of a wall, so the direction of minimum variance is perpendicular to the wall (and unlikely to change) while the maximum variance is in the direction parallel to the wall. The large covariance parallel to the wall reflects the belief that subsequent observations will see this gap shift along the wall with the addition of new data.

C. DATA ASSOCIATION

A nearest neighbour χ^2 test is used to find the most probable association between measurements $D = [d_1, d_2, ...d_n]$ returned by our gap detector and $\hat{G} = [g_1, g_2, ...g_m]$ the predicted locations of the gaps currently being tracked. This association is used to update the gap positions.

We also maintain an association matrix A of all measurements that gate with current gap positions given a less stringent bound γ.

$$A_{ij} = \begin{cases} 0 & \chi^2_{ij} > \gamma \\ 1 & \chi^2_{ij} \leq \gamma \end{cases}$$

where $\chi^2_i j$ is the Mahalanobis distance between the detected gap d_i and known gap g_j.

This association matrix is used to spot splitting and merging of gaps for use in the construction of the gap navigation tree. A split corresponds to multiple measurements being associated with one gap, and conversely a merge equates to multiple gaps associating with the one measurement.

Unexplained measurements (those that do not associate with any existing gaps) are used in the initiation of new gaps. As explained in Section III, spurious gaps are problematic and we do not want to initialize a gap based on a single appearance in the gap sensor. A list of putative gaps that have associated with prior putative gaps is maintained, these gaps are added to the gap tracker and the gap navigation tree once they have been observed a threshold number of times and we are hence confident they are actual gaps.

V. RESULTS

We focus here on the role of a gap motion model rather than the completion properties of the GNT. Figure 5 shows the progress of the robot through a simulated 2D world where our gap sensor is informed only by noisy, sparse data from a 2D laser scanner. The current state of the tree is shown at each time step, the *L* or *R* notation at each node shows whether the associated gap occludes an area to the left or right of the robot as defined in Section IV-A. The accumulated point cloud map is shown together with square markers indicating the position of currently tracked gaps. The covariances of each gap are shown as red ellipses. The dotted lines emanating from the robot indicate the direction of gaps detected by the gap detector (note that in some instances these gaps may have recently appeared and are hence not yet tracked by the gap tracker).

In Figure 5(a) four gaps are being tracked and the robot is in pursuit of Gap 3. All four gaps lie at the extent of known walls and hence the covariances (in red) are maximal in the direction parallel to the wall and are large in size, reflecting the belief that these gaps are likely to move as the map is incremented.

Figure 5(b) shows that as the robot moves past the corner, Gaps 2 and 4 merge to form Gap 12. Gap 3 has disappeared as a result of the map being *filled in* as the robot explores and sees more of its environment. A new, unassociated gap has appeared at the top right extent of the map, corresponding to the visible extent of the far wall. The robot is now in pursuit of Gap 6.

In Figure 5(c) the robot has turned around a second corner point but has still managed to track Gap 12. As the right side of the map is now completely known, Gap 12 is the sole remaining gap and now represents the entire occluded region of the known map located to the right of the tangent point. Note the altered shape of the covariance of Gap 12 at this point - it is now located at a well-defined corner of the map with a greater point cloud density than in the previous examples.

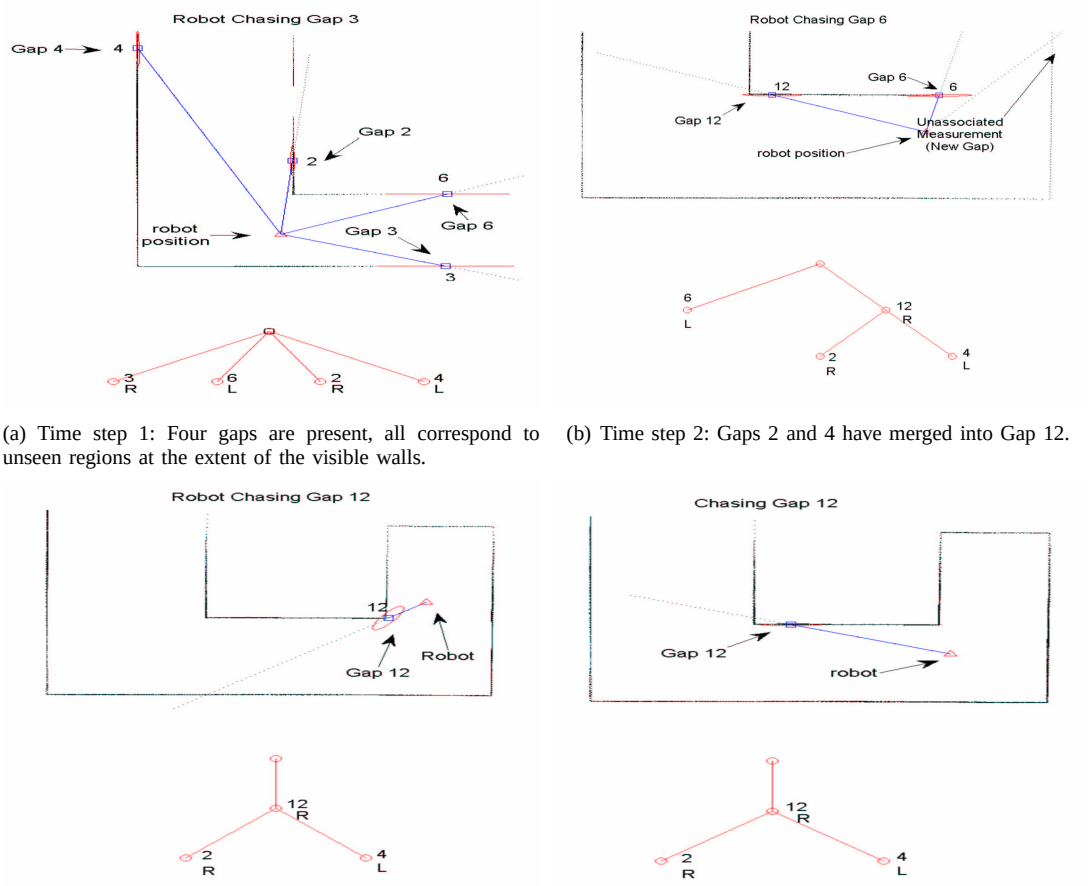
Note that Gap 12 is a *non-primitive* gap, meaning at least part of its occluded region is unexplored. In Figure 5(d) the robot is retracing its trajectory in pursuit of Gap 12. As expected, the location of the tangent point to Gap 12 travels back along the wall towards its previous position in Figure 5(b). The correct association between the region represented by Gap 12 is maintained throughout Figures 5(a)-5(d).

We found the primary effect of adapting the GNT to real world conditions is the addition of *faux gaps* to the tree. They were not found to adversely affect the operation of the GNT however, as they tend to be associated with simple geometric structures such as walls and simply disappear (rather than split) when chased. If anything, they have the tendency to inbuild 'wall following' behaviour into the tree. Gap 3 of Figure 5(a) is an example of such a gap, as the infinite range gap sensor of [2] at the robot position would detect the entirety of the wall on which Gap 3 is located as well as the Wall at the end of the corridor that is evident in Figure 5(c). No discontinuity would be ever be detected in the region of Gap 3.

Weaknesses of the algorithm include the dependency of the gap detector sensitivity on the structure of the local environment. Despite offering improved robustness over non-probabilistic implementations of the GNT, the data structure is still prone to failure a result of failing to detect the splitting or merging of gaps. A rollback/journalling facility could be added to overcome this and provide capability to maintain multiple hypotheses about the possible tree and environment structure at any one time.

VI. CONCLUSION

This paper presented a probabilistic approach to tracking occluded regions of the environment while exploring an unknown environment using the Gap Navigation Tree. It was shown that the use of online metric mapping reduces the



(a) Time step 1: Four gaps are present, all correspond to unseen regions at the extent of the visible walls.

(b) Time step 2: Gaps 2 and 4 have merged into Gap 12.

(c) Time step 3: The robot pursues gap 6 until it disappears, the top right corner of the U-shaped map is now complete.

(d) Time step 4: Gap 12 corresponds to the unexplored area obscured by gaps 2 and 4, the robot turns around and begins to pursue this uncharted territory.

Fig. 5. Tracking a Gap

practical difficulties associated with using the Gap Navigation Tree in real environments with sensor uncertainty, as it allows a model of expected gap motion to be derived and used in gap tracking. The robust approach to gap sensing and tracking presented here positions the hybrid combination of the topological Gap Navigation Tree and metric SLAM as a viable method for use in mobile robot exploration.

REFERENCES

[1] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006.

[2] B.Tovar, R. Murrieta-Cid, and S. M. LaValle, "Distance-optimal navigation in an unknown environment without sensing distances," *IEEE Transactions on Robotics*, June 2007.

[3] B. Tovar, L. Guilamo, and S. M. LaValle, "Gap navigation trees: Minimal representation for visibility-based tasks," in *Proc. Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2004.

[4] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *CIRA '97: Proceedings of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation*. Washington, DC, USA: IEEE Computer Society, 1997, p. 146.

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

[6] C. Stachniss, G. Grisetti, and W. Burgard, "Information gain-based exploration using rao-blackwellized particle filters," in *Proc. of Robotics: Science and Systems (RSS)*, Cambridge, MA, USA, 2005.

[7] P. Newman, M. Bosse, and J. Leonard, "Autonomous feature-based exploration," in *IEEE International Conference on Robotics and Automation*, Taiwan, September 2003.

[8] A. Makarenko, S. Williams, F. Bourgault, and H. Durrant-Whyte, "An experiment in integrated exploration," in *In Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS), Lausanne, Switzerland, 2002.*, 2002.

[9] R. Sim and N. Roy, "Global a-optimal robot exploration in slam," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Barcelona, Spain, 2005.

[10] H. Choset and K. Nagatani, "Topological simultaneous localization and mapping (slam): toward exact localization without explicit localization," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 2, pp. 125 – 137, April 2001.

[11] A. Victorino and P. Rives, "An hybrid representation well-adapted to the exploration of large scale indoors environment," in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 3, 2004.

[12] Y. Landa, R. Tsai, and L.-T. Cheng, "Visibility of point clouds and mapping of unknown environments." in *ACIVS*, ser. Lecture Notes in Computer Science, J. Blanc-Talon, W. Philips, D. Popescu, and P. Scheunders, Eds., vol. 4179. Springer, 2006, pp. 1014–1025.

[13] Y. Landa, D. Galkowski, A. Huang, Yuan R.and Joshi, C. Lee, G. Leung, Kevin K.and Malla, J. Treanor, V. Voroninski, and Y.-H. R. Bertozzi, Andrea L.and Tsai, "Robotic path planning and visibility with limited sensor data," in *ACC '07*, 2007, pp. 5425–5430.

[14] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.