Using Erroneous Prior map Information for Active Exploration and Mapping

Sri Ramana Sekharan

November 2016

1 Introduction

The problem of Active Simultaneous Localization and Mapping involves an autonomous robot navigating an unknown environment, simultaneously estimating the map and its pose such that the uncertainty in both the estimates are minimized. There are many approaches for ASLAM in the absence of prior map and in arbitrary environments. [CDKC15],[CCC+16],[Sta09],[SGB05] In general these approaches consists of the following three steps

- **Generate a set of trajectories using a procedure**. Typically, it is done by finding clusters of unseen cells in the map and having the center point of each clusters as the goal and generating one trajectory per cluster. Another important consideration here is to generate trajectories with loop closures.
- **Select a single trajectory from the previously generated set that maximizes a utility function** The utility function usually penalizes uncertainty along the trajectory. The predicted estimate of pose and map is usually obtained by simulating maximum likelihood measurements.
- **Execute selected trajectory**. Follow the trajectory and update the location and map estimate using incoming observations. Do this till some stopping criterion and repeat step 1.

All of these methods are online and hence calculate the path online. Since it is online, the final exploratory path that is obtained from these methods are sub-optimal.

There are many, indoor environments that contain repeated monotonous structures and highly narrow passage ways with distinct topological properties. Applying purely exploratory ASLAM in this case is inefficient, since once could exploit the known prior information of the environment and pre-compute a more optimal path for exploration and reduce the online overhead.

The key idea behind the approach is to take advantage of the structure of the environment and model planning exploratory path as a graph search problem. This allows the algorithm to exploit discrete optimization methods to calculate the optimal path. In some real world environments, the topology of the actual map changes. In case the topology of the actual map has changed, our framework allows re-planning only the portions of the graph that have changes and thus reusing unchanged portions of the graph.

2 Background and Related Work

All ASLAM approaches follow the steps listed in the above section, with its own variation and improvements for each step and the setting in which it is applicable to (like FastSLAM filter slam, graphSLAM[TM06]). [CCC+16] offers an excellent survey of all ASLAM approaches with detailed discussion on unsolved/open problems.

Theory of Optimal Experimental Design(TOED) has been applied to active SLAM to form utility functions based on uncertainty encountered during the execution of a trajectory. [LG11] compares different kinds of the TOED based utility function that can be used, their specific properties and proves one of them offers the best performance. [CLNC12] provides an algorithm that takes in a pose graph and calculates a minimum uncertainty path between a start goal and an end goal using the best utility function used in [LG11].

Information theoretic approaches to ASLAM choose trajectories that gives the maximum information gain and hence minimize entropy. [SGB05] adopts a particle filter based SLAM framework and the utility function is the information gain over the estimate of poses and map. Since, estimating the joint distribution of map and pose is difficult they assume both are independent and hence the utility function is a linear combination of entropy over map and pose estimates. The disadvantage of this is that both are of different scales and has to weighed differently and hence manual tuning is necessary. [CDKC15] overcomes this problem by introducing a novel utility function. This formulation does not require manual tuning.

[MCdFB$^+$09] formulates the ASLAM problem as a modified continuous POMDP with the utility function depending on belief state and the policy as a parametrized trajectory that is independent of belief state. With this formulation, they adopt a policy search method to find the optimal trajectory using Bayesian Optimization methods.

[ICD15] formulates a framework where instead of selecting a set of trajectories they plan in the continuous space of trajectories while also taking into account all possible predicted measurements instead of the maximum-likelihood measurement. As a result their algorithm results in more natural smooth trajectories.

All the above approaches assume partial or no prior map information. But our approach is different in that, it takes advantage of highly structured environment and the prior knowledge could be erroneous. By reducing the problem of calculating the optimal path to a graph search problem, we could take advantage of discrete optimization methods.

[ABRS99],[RDD99] these approaches deal with topological exploration. But they do not account for any kind of uncertainty during exploration and do not use prior information.

Our work extensively uses a SLAM library called GTSAM [Del12]. GTSAM is a library of C++ classes that implement smoothing and mapping (SAM) in robotics and vision, using factor graphs and Bayes networks as the underlying computing paradigm.

The uncertainty model used in our approach is explained as follows. As the robot moves the uncertainty in the relative pose estimate increases due to the inherent uncertainty of the odometry sensor. Given the pose of the robot at a time t, $X(t) = (x(t), y(t), \theta(t))$, and $t+\Delta t$, $X(t+\Delta) = (x(t+\Delta t), y(t+\Delta t), \theta(t+\Delta t))$. The relative pose estimate $\Delta X = (\Delta x, \Delta y, \Delta\theta)$ after a time delta Δt is shown below. For our purposes the covariance is hard-coded as a diagonal matrix whose entries are proportional to the square root of distance travelled,

$$\Delta X = (x(t+\Delta t) - x(t), y(t+\Delta t) - y(t), \theta(t+\Delta t) - \theta(t)) \tag{1}$$

$$\Sigma(\Delta X) = DiagonalMatrix(\sigma * \sqrt{(}\Delta x), \sigma * \sqrt{(}\Delta y), \sigma * \sqrt{(}\Delta\theta)) \tag{2}$$

So σ is our uncertainty parameter and increasing σ indicates more uncertainty and vice-versa.

3 Problem Statement

Given a prior map of a highly structured environment, we propose a framework to effectively use that information to explore and map the environment. The overall idea of our framework is explained in figure 1. We assume that there could be obstacles blocking the robot and the prior map could be erroneous. Our approach will be able to handle these problems.

The first step of our approach is to extract a topological map from the prior map environment. This topological map is modelled as a weighted undirected graph. We assume the robot starts from a vertex in the graph, our initial objective is to pre-plan a path in the graph that traverses all the edges and at the same time reduces localization error i.e robot pose uncertainty while doing so. The uncertainty of a robot pose increases as it moves according to the odometry motion model as shown in 4. With high uncertainty, it would be very hard to do mapping. When the robot visits a previously visited position and a loop closure is done, the uncertainity reduces as shown in 5. So in order to bound the uncertainty, the robot path must contain equally spaced loop-closures i.e it should revisit previously visited vertex in the graph.

For the given prior map, the pre-planned path would be the optimal path that would always have sufficient loop-closures and also map the whole environment. In case when following the path, the topology of the environment changes, our framework should allow for reconstructing the part of the graph that has changed. After reconstruction it should be able to re-plan a path online. This new path would join an already pre-planned path, thus using the information previously calculated.

The expected advantage of our system over existing ASLAM methods is that, by exploiting prior knowledge of highly structured environments, we can have more optimal exploration trajectories. The improved performance of our algorithm in comparison with purely exploratory active SLAM is expected to be a function of the percentage of similarity of the actual map to the prior map.

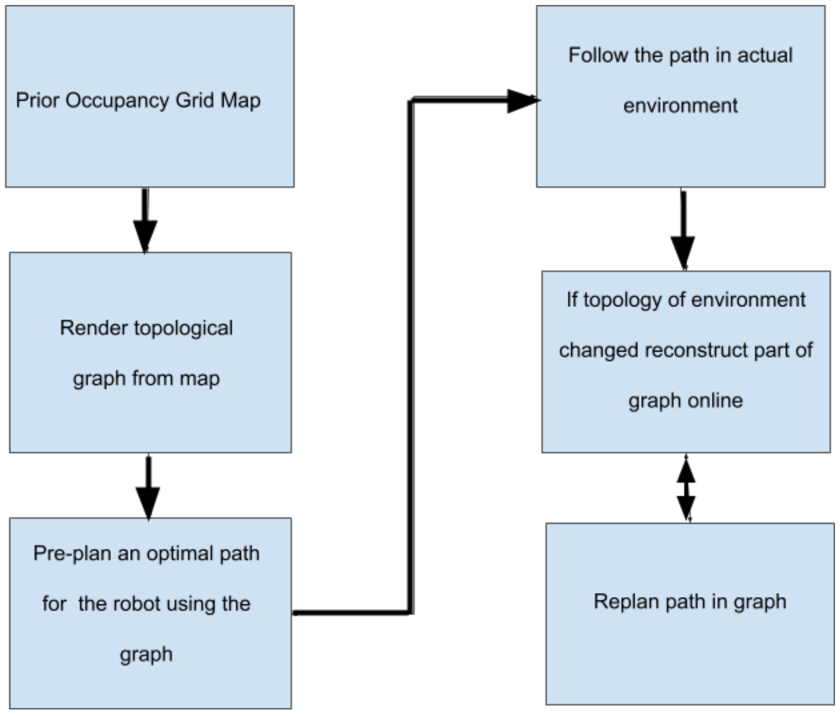
Prior Occupancy Grid Map
Render topological graph from map
Pre-plan an optimal path for the robot using the graph
Follow the path in actual environment
If topology of environment changed reconstruct part of graph online
Replan path in graph

Figure 1: Overall Block diagram of the framework

4 Graph Extraction

From a prior occupancy grid map, a graph is extracted as shown in figure. Our approach first extracts a topological map from the prior map environment. This topological map is modelled as a weighted undirected graph. The edges correspond to distinct empty space (like long aisles and corridors) in the environment and vertexes correspond to intersection of two or more distinct empty spaces in the environment. Each edge represents a trajectory in the environment the robot could take. It is assumed that, loop closure can be done in each of the vertexes of the graph. And it is assumed if the robot traverses all the edges the entire environment will be mapped. The graph is extracted in such a way that traversing all the edges of the graph will result in the whole environment being mapped. So we could take into account the robot sensor range for this purpose. The weight of the edges is the actual length of the trajectory. For now, we assume such a graph has been extracted and our main objective is to find an optimal edge path in the graph.

5 Pre-planning path

5.1 Mathematical Formulation

We have extracted the graph G of the environment. A weighted undirected graph is defined as $G = (V, E, W)$ where $V = \{v_1, v_2, ..., v_{\|V\|}\}$ is the set of vertices and $E = \{e_1, e_2, .., e_{\|E\|}\}$ is the set of edges and each $e_i = \{v_i, v_j\}$ for some $v_i, v_j \in V$ and $W : E \to R$ is a set that maps edges to real numbers. Our objective

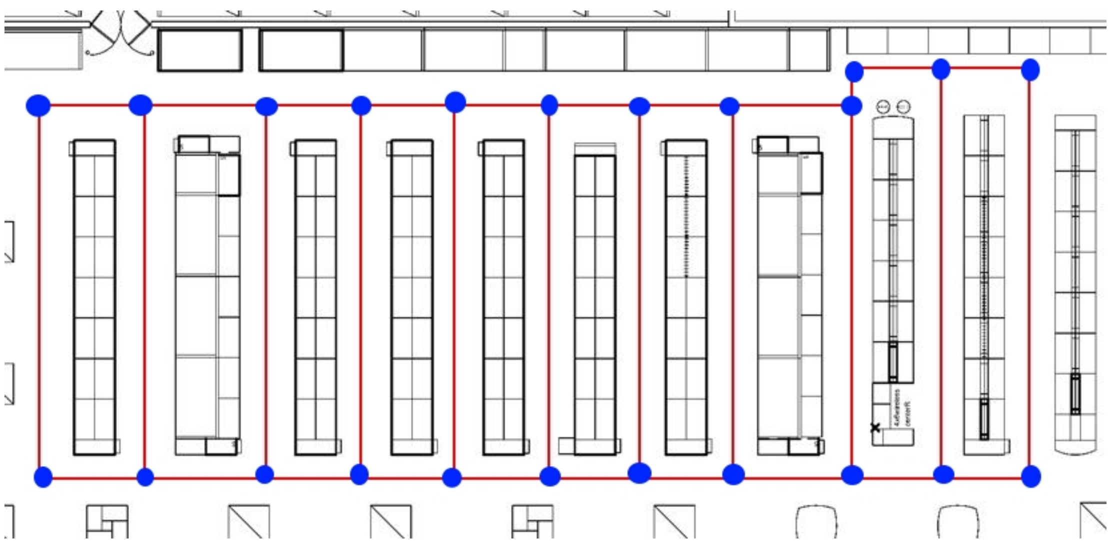
Figure 2: A possible topological graph extracted from the occupancy grid map

is to come-up with an optimal path that visits all the edges at-least once while also having sufficient loop closures so as not to let the uncertainty go out of hand. Let us denote a feasible path in a graph as an ordered set $p = \{e_i\}$ where $e_i \in E$ and each consecutive pair of edges contain at-least one common vertex. Let $\mathcal{P}$ be the infinite set of all possible feasible paths in the graph G.

Mathematically, our optimal path p^* should minimize an objective cost function $F(p)$.

$$p^* = \arg\min_{p \in \mathcal{P}} F(p) \tag{3}$$

The cost function $F(p)$ is linear combination of three function,

$$F(p) = U(p) + D(p) + E(p) \tag{4}$$

where $D(p)$ is the total distance traversed so far,

$$D(p) = \sum_i W(e_i), \forall e_i \in p \tag{5}$$

$E(p)$ the Unexplored Penality is a term proportional to the number of unexplored edges. Let us denote the unexplored edge set $e'(p)$ as

$$e'(p) = E - p \tag{6}$$

Now we can define $E(p)$ mathematically as

$$E(p) = \alpha * \|e'(p)\| \tag{7}$$

where α is a user defined constant large enough to encourage exploration.

$U(p)$ is the total pose uncertainty cost. We define two types of uncertainty cost

5.1.1 Overall Path Maximum Uncertainty Cost (OPMU)

The idea behind this criterion is that, we bound the maximum possible uncertainty that can arise even-though the uncertainty could be reduced by a subsequent loop closure. Let us define 'path subset' $S(p)$ of a path p, as an ordered set

$$S(p) = \{p_1, p_2, ..., p\} \tag{8}$$

where

$$p_i = \{e_1, e_2, .., e_i\} \tag{9}$$

The path subset is the set of all intermediate paths that start from e_1, encountered by p.

Given a path p, and a start vertex $v_s \in V$, we assume the start node is the origin of the robot and we simulate odometry readings for the path. We form a set of robot poses $X(p) = \{x_i\}$ at regular intervals throughout the path. Then we use the odometry readings to form a pose-graph [KFL01] consisting of elements of X with relative pose constraints. We then find the marginalized estimate of the robot poses in the set X using the pose graph using a non-linear optimizer, currently GTSAM [KIRD10],[Del12]. As a result of marginalization we get

$$\forall x_i \in X(p), (\mu(x_i), \Sigma(x_i)) \tag{10}$$

where $\mu(x_i)$ is the mean of the robot pose and $\Sigma(x_i)$ is the Covariance. This Covariance is a measure of uncertainty. To quantify the extent of uncertainty we use the D-opt criterion i.e. the determinant of the Covariance $|\Sigma|$ since this is proved to be the most optimal measure of uncertainty[LG11]. So $U(p)$ is the maximum possible uncertainty that has been accumulated throughout the execution of path p

$$U(p) = \max_{p_i \in S(p)} \max_{x_i \in X(p_i)} |\Sigma(x_i)| \tag{11}$$

We take the maximum of all path subsets because a subsequent loop closure may reduce the covariance of the entire path. This criterion is always increasing, since we take the maximum over all possible determinant values.

5.1.2 Current Path Maximum Uncertainty Cost (CPMU)

In this case the maximum of the current path uncertainty is taken

$$U(p) = \max_{x_i \in X(p)} |\Sigma(x_i)| \tag{12}$$

As in the previous criterion, the maximum of determinant of covariance is taken. But unlike taking max over all paths we use only the current path. Note this value is non-monotonic, it will increase with path and as soon as the path encounters a loop closure this value decreases.

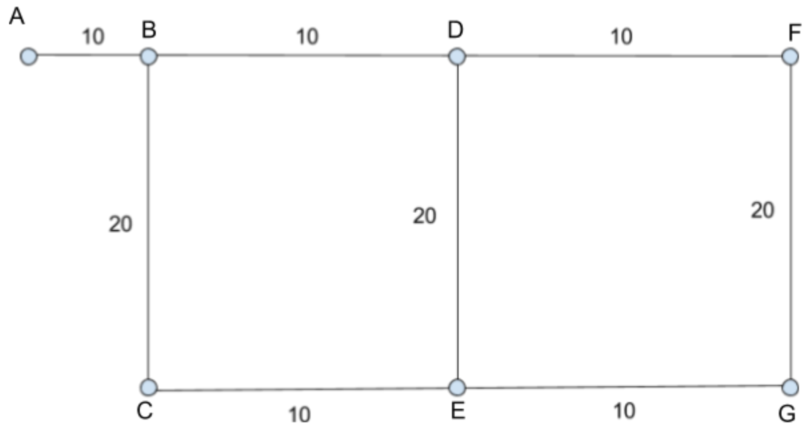

Figure 3: Topological graph (Numbers are distances in meters)

5.2 Exhaustive search

A naive way to solve the problem is to do an exhaustive search. We start from the start vertex v_s, and expand all the adjacent edges in a breath first search fashion. Each expanded child is evaluated and the cost $F(p)$ is calculated. This is put in a priority queue with the priority being the $F(p)$. The top of the queue is popped and expanded. Note that unlike normal graph search which terminates when a previously visited node is visited again, in our case, infinite number of node re-visits are allowed. Since this makes $\mathcal{P}$ an infinite set, the search does not end. A naive way of getting the path soon is to run the algorithm for sometime and

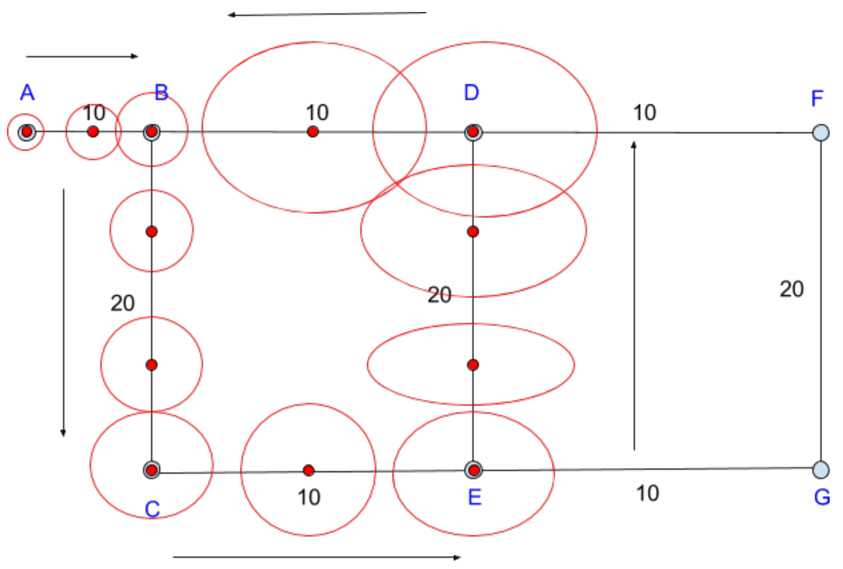

Figure 4: Red points are simulated robot poses. The arrows show the direction in which the path is simulated. The red ellipses are marginalized covariance. It is seen the open loop odometry uncertainty is increasing as the robot traverses. The shown path corresponds to $[A, B, C, E, D, B]$ in the graph

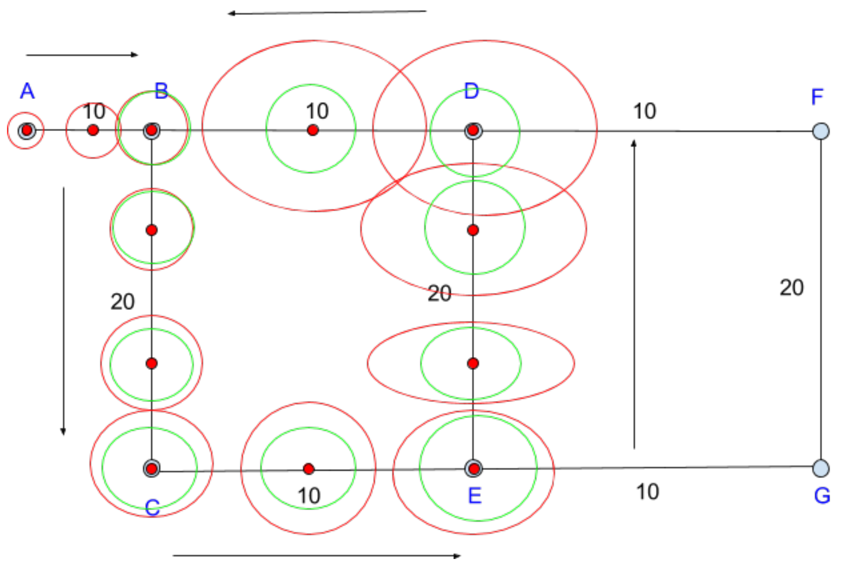

Figure 5: This shows the figure after the robot reaches a previous pose. The green circles shows the reduced covariance after loop closure

return the minimum cost obtained so far as the best path. So the longer we run the algorithm, the more optimal we get.

To overcome the above limitation we limit the number of times we are allowed to revisit a node. Let's call this parameter η. This still has the limitation, that we do-not know the maximum revisit count of p^* beforehand. Another strategy is to use informed heuristic search. So in this case the priority queue would be given $C(p) = F(p) + H(p)$ as the priority, where $H(p)$ is any heuristic. If the heuristic is good, then it converges towards the true solution faster.

All the above methods are still exhaustive search and obtaining p^* may not be possible. So we would have to run the algorithm for a fixed time and use the current best solution.

5.3 Branch and Bound formulation

The graph search is an instance of discrete optimization, so a branch and bound[LW66] formulation is an excellent approach to solve this problem. Branch and bound is a systematic way to search for the optimal solution in a discrete optimization problem. The procedure recursively splits the solution space and searches for the optimal solution. Let's us consider we are trying to find the minimum x of the function $f(x)$. The branch and bound algorithm can be summarized in the following steps

1. Using a heuristic find a solution to the problem, x_h and set it as the current upperbound $B = f(x_h)$
2. Initialize the queue Q with the starting node N
3. Repeat until queue is empty
   (a) Pop the node N from the top of the queue
   (b) If the solution x in N is a candidate solution and if $f(x) < B$ then Record x as the best solution and set $B := f(x)$
   (c) Split the node N into child nodes N_i
      i. If $g(N_i) > B$ do nothing
      ii. else Push N_i onto Q

Our formulation of Branch and Bound is given in algorithm, 1 .As we see from above, our objective function is $F(p)$ and we are searching over $\mathcal{P}$. For this case every solution $p \in \mathcal{P}$ is a candidate solution. To use the above algorithm we need to define a good lower bound function $g(p)$. Also let us denote the set of all possible paths with p as the prefix by $P(p)$.The lower bound function $g(p)$ should have the following properties

$$g(p) \leq f(p'), \forall p' \in P(p) \tag{13}$$

$$g(p') \geq g(p), \forall p' \in child(p) \tag{14}$$

In case we use the OPMU as U(p), then we define our lowerbound $g(p)$ as the sum of the distance cost D(p), distance of unexplored edges $D(e'(p))$ and the odometry cost U(p)

$$g(p) = D(p) + D(e'(p)) + U(p) \tag{15}$$

It can be seen that

$$\forall p, p' \in P(p), D(p) + D(e'(p)) \geq D(p') + D(e'(p')) \tag{16}$$

$$\forall p, p' \in P(p), U(p) \geq U(p') \tag{17}$$

For the case of CPMU, we use the lower bound as

$$g(p) = D(p) + D(e'(p)) \tag{18}$$

5.3.1 Chinese Postman Path

The branch and bound requires a solution to be obtained first using some heuristics and then use the cost of that solution as the upperbound. If the solution is as tight as possible then our search will be very fast. To do that we use the Chinese Postman algorithm [COW14].

An euler tour of a graph is a path that traverses all the edges only once. A graph for which such a path exists is called an eulerian graph. The problem of finding an Euler's tour of a non-eulerian graph with the most minimum number of edge revisits (i.e. finding the minimum distance path to cover all edges) is called the Chinese Postman Problem. Our problem is a special case in which we also need to satisfy the uncertainty requirement. So we use the Chinese Postman Algorithm to find an initial path and use its cost as the initial upperbound.

Algorithm 1 Branch and Bound

```
Input: $(G, v_s)$
$C_{pp} \leftarrow ChinesePostmanPath(G, v_s)$
B $\leftarrow F(C_{pp})$
path $\leftarrow v_s$
bestpath $\leftarrow$ path
node.path $\leftarrow$ path
PriorityQueue, Q.push(node)
while Q.empty() do
  node $\leftarrow$ Q.pop()
  if $F$(node.path) $\leq$ B then
    B $\leftarrow F$(node.path)
    bestpath $\leftarrow$ node.path
  end if
  for v in node.path.neighbourVertices() do
    newpath $\leftarrow$ node.path.append(v)
    if g(newpath) $\leq$ B then
      continue
    end if
    newnode.path $\leftarrow$ newpath
    Q.push(newnode)
  end for
end while
return bestpath
```

6 Experiments and Results

6.1 Graph Search Simulation

Pre-planning algorithms described above were tested for the graph in 3. First a basic uninformed exhaustive search is tested on the graph. As we saw in the section above, that the odometry uncertainty is parameterized by σ. Since naive exhaustive search is intractable, the number of times a node can be visited is constrained to two. For the exhaustive search we test the CPMU criteria. The different path obtained for different settings for a node revisit limit of two (i.e. $\eta = 2$) are as follows

1. When $\sigma = 0.1$ we obtain the path , $[A, B, C, E, D, F, G, E, D, B, A]$. This path has well spaced apart loop closure revisits and also not exceeding the limit visit of two
2. When $\sigma = 0$ we obtain, $[A, B, D, B, C, E, G, F, D, E]$. Here we see that, all the edges are explored but it doesn't focus on revisiting any new node i.e it doesn't care about loop-closures
3. When $\sigma = 1.0$ and the α associated with the unexplored edge cost $E(p)$ is low, we obtain $[A]$. This means the algorithm is afraid to gather uncertainty and hence remains in the same node. So if we increase the value of α to a large degree, we obtain $[A, B, C, E, D, E, G, F, G, F, D, B, A]$. Here we see that the path does more loop closure than the first and second case.

So the path obtained depends on the values set for the various terms in the cost function.

The informed search was also done with the heuristics. The heuristics used is, latest node revisit distance. It is nothing but a number proportional to the distance of the latest node from its previous visit. For example, $[A, B, C, E, C]$ would have a higher heuristic cost than $[A, B, C, E, D, B]$ since the first path revisits a node that is one distance away in path, but the second path revisits a node that is three distance away.

It was seen that for $\sigma = 0.1$ it took 696 iterations to find the optimal path while with heuristics it took 513 iterations. Similarly $\sigma = 1$ optimal path took 750 iterations while using heuristics it took 614 iterations to find the path. It shows that if we have good heuristics then the probability of finding good solutions sooner is increased.

The branch and bound algorithm was tested. There was no limit on the number of revisit, so it is searching for the optimal path p^*. We also test for the two kinds of uncertainty criterion, OPMU and CPMU. The results for CPMU are as follows

1. When $\sigma = 0.1$ the path obtained is $[A, B, C, E, D, F, G, E, D, B, A]$ and this turns out to be the optimal solution.

2. When $\sigma \geq 0.3$ the algorithm took too long to terminate. It is mainly due to the lack of strong lower bound for CPMU

The results for OPMU are as follows

1. When $\sigma = 0.1$ the path obtained is $[A, B, D, F, D, B, A, B, D, E, C, B, D, F, G, E]$ and this turns out to be the optimal solution (p^*). It can be seen that the path is too conservative to let the overall uncertainty to go out of hand.

6.2 ASLAM Simulation

A simulation of ASLAM algorithm was started in turtlebot in gazebo. The simulation uses GTSAM as its SLAM algorithm and a frontier based algorithm for generating exploration trajectories. Local planner from ROS was used and ROS costmaps was used for navigation. SBPL library was used for global trajectory planning. Problems were encountered in navigation where due inaccurate map estimation and bugs in the local planner. But due to the direction of the main problem formulation, debugging the issue was not pursued further.

Another simulation of turtlebot in Gazebo performing coverage mapping as in chapter 3 of [Sta09]. It was simulated in the playground world as shown in the fig 6. ROS gmapping was used for mapping purposes. The plot obtained for number of cells covered vs time is given in 7. The main idea behind this method is that, the robot computes view-points from which the entropy of the map is reduced and then navigates to that location. The effect of sensor reading on entropy of map is not something that we consider in our current formulation. We assume traversing the path will map the entire environment. But it should be possible by the introduction of the term to the objective function $F(p)$.

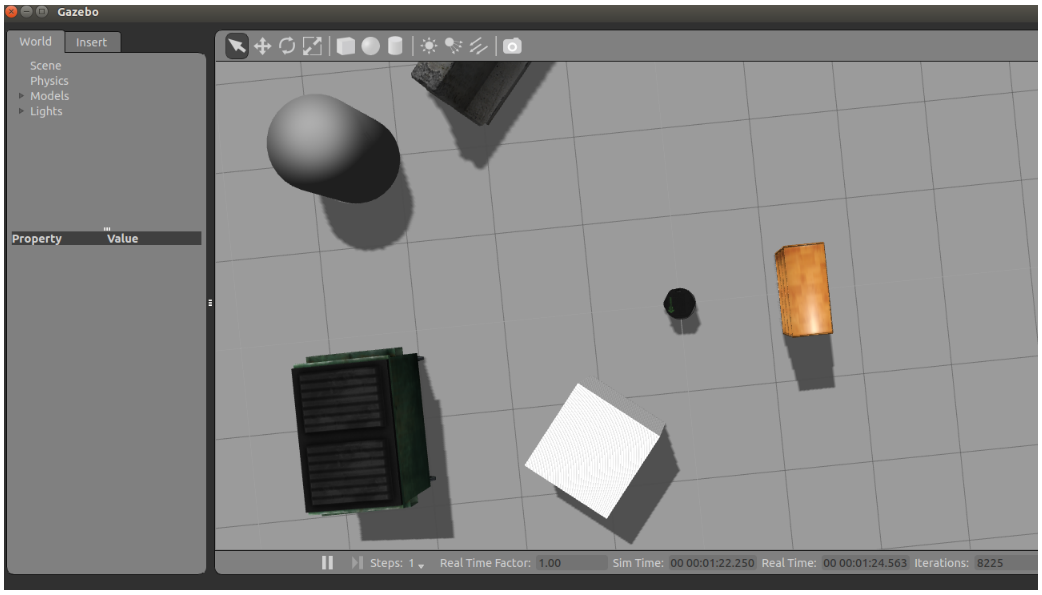

Figure 6: The playground environment in turtlebot gazebo

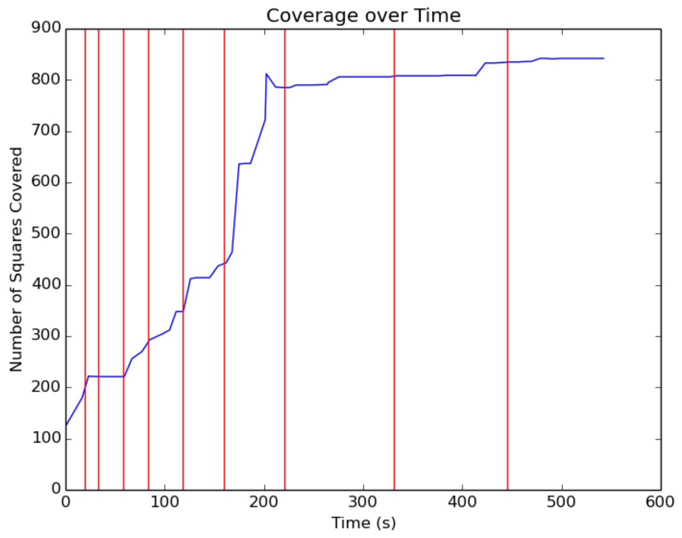

Figure 7: Plot between Coverage vs Time

7 Conclusion and Further Work

We see that the CPMU criterion allows for increased uncertainty as long as a subsequent loop closure reduces the maximum uncertainty, while OPMU caps the maximum uncertainty. The ideal choice would be a combination of both, with a threshold on the maximum uncertainty tolerated, while allowing varying uncertainties to exist within that threshold. The rationale is that if the uncertainty is greater than a certain bound then it is not possible to close the loop reliably. The threshold can be determined by methods in [GS07]

The results obtained so far are very sensitive to weights we give to various terms of the cost function. The next step is to put everything in a strong mathematical formulation so as to choose them naturally. Secondly we need to improve the run time performance of branch and bound with a tighter lowerbound that includes uncertainty. Also ways of using heuristics in branch and bound should be explored.

Once a pre-planned path is formed, we need the robot to follow the path in the environment. If the environment is the same as the prior map, then following the edge trajectories associated with the preplanned edge path is optimal. In case a path is blocked by an obstacle, an edge is disconnected in the graph, then an intelligent way of re-planning that portion of the trajectory should be designed. Also in case the topology of the graph changes locally, also then, replanning should be done online. Our subsequent step would be to device such an algorithm that re-plans online in an efficient way and prove, that the path obtained as a result is no worse than the path obtained by existing pure exploratory ASLAM methods and evaluate the complexity of the algorithms.

References

[ABRS99] Baruch Awerbuch, Margrit Betke, Ronald L Rivest, and Mona Singh. Piecemeal graph exploration by a mobile robot. *Information and Computation*, 152(2):155–172, 1999.

[CCC$^+$16] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[CDKC15] Henry Carrillo, Philip Dames, Vijay Kumar, and José A Castellanos. Autonomous robotic exploration using occupancy grid maps and graph slam based on shannon and rényi entropy.

In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 487–494. IEEE, 2015.

[CLNC12] Henry Carrillo, Yasir Latif, José Neira, and José A Castellanos. Fast minimum uncertainty search on a graph map representation. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2504–2511. IEEE, 2012.

[COW14] Gauvain Chaste, Aurélien Ooms, and Robin Walravens. Chinese postman problem. 2014.

[Del12] Frank Dellaert. Factor graphs and gtsam: A hands-on introduction. 2012.

[GS07] Juan Pablo Gonzalez and Anthony Stentz. Planning with uncertainty in position using high-resolution maps. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 1015–1022. IEEE, 2007.

[ICD15] Vadim Indelman, Luca Carlone, and Frank Dellaert. Planning in the continuous domain: A generalized belief space approach for autonomous navigation in unknown environments. *The International Journal of Robotics Research*, 34(7):849–882, 2015.

[KFL01] Frank R Kschischang, Brendan J Frey, and H-A Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2):498–519, 2001.

[KIRD10] Michael Kaess, Viorela Ila, Richard Roberts, and Frank Dellaert. The bayes tree: An algorithmic foundation for probabilistic robot mapping. In *Algorithmic Foundations of Robotics IX*, pages 157–173. Springer, 2010.

[LG11] Henry David Carrillo Lindado and José Ángel Castellanos Gómez. On the comparison of uncertainty criteria for active slam. 2011.

[LW66] Eugene L Lawler and David E Wood. Branch-and-bound methods: A survey. *Operations research*, 14(4):699–719, 1966.

[MCdFB$^+$09] Ruben Martinez-Cantin, Nando de Freitas, Eric Brochu, José Castellanos, and Arnaud Doucet. A bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot. *Autonomous Robots*, 27(2):93–103, 2009.

[RDD99] Ioannis M Rekleitis, Vida Dujmovic, and Gregory Dudek. Efficient topological exploration. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 1, pages 676–681. IEEE, 1999.

[SGB05] Cyrill Stachniss, Giorgio Grisetti, and Wolfram Burgard. Information gain-based exploration using rao-blackwellized particle filters. In *Robotics: Science and Systems*, volume 2, pages 65–72, 2005.

[Sta09] Cyrill Stachniss. *Robotic mapping and exploration*, volume 55. Springer, 2009.

[TM06] Sebastian Thrun and Michael Montemerlo. The graph slam algorithm with applications to large-scale mapping of urban structures. *The International Journal of Robotics Research*, 25(5-6):403–429, 2006.