Local Map Based Graph SLAM with Hierarchical Loop Closure and Optimisation

Adrian Ratter and **Claude Sammut**
School of Computer Science and Engineering
University of New South Wales, Sydney, Australia
adrianr@cse.unsw.edu.au and claude@cse.unsw.edu.au

Abstract

We present a Graph SLAM system for use with a laser range finder suitable for low powered mobile robots, that is capable of producing globally consistent maps despite large errors in ICP registration. We introduce two types of loop closure constraints to allow the global map to be corrected as soon as possible, while retaining robustness to false positives in the presence of errors in position tracking. By using a two step optimisation procedure operating on a tree-based graph of local maps, only a subset of the global map is updated after a loop closure. Our system can generate globally consistent maps in challenging scenarios such as long featureless corridors and complex office environments.

1 Introduction

We present a robust and efficient Graph based *Simultaneous Localisation and Mapping* (SLAM) algorithm designed for use with a laser range finder based *Iterative Closest Point* (ICP) position tracker. We focus on loop detection and graph building strategies to bridge the gap between the popular research areas of front end position tracking and back end graph optimisation. Our algorithm, called *Hierarchical Graph SLAM* (HG-SLAM), is designed to maintain an accurate global position and generate a globally consistent map in the presence large ICP registration errors. These errors could be from drift or loss of tracking due to areas where motion is not fully observable, such as featureless corridors. The algorithm is suitable for use on low powered and motion encoder-free mobile robots, and can be integrated with an RGB-D sensor to generate 3D maps of an environment. Specifically, we propose:

- Two types of loop closure detection. *Full loop closures* to ensure large errors in position tracking can be corrected, and *Partial loop closures* to ensure partially overlapping local maps align and that the global position is corrected as soon as possible.
- Using the covariance of individual scans inside local maps to determine the constraint information for graph optimisation. This allows large errors in position tracking to be corrected without introducing errors in the global map.
- A two level optimisation procedure based on a tree of local maps to allow the global structure to be efficiently corrected, and to avoid discontinuities between overlapping local maps.
- A complete algorithm for global map building, loop closing and graph optimisation suitable for mobile robots with limited processing power.

Graph SLAM algorithms represent the global map as a series of nodes, each representing a previous position of the robot and observations of the local area from this position. Edges in the graph represent *constraints* in the positions between nodes, and are generated from motion estimates or loop closure detection. The resulting graph can be optimised to find the configuration of nodes that maximises the likelihood of the observations, thereby correcting past mistakes in movement estimation. Many Graph SLAM solutions described in literature take the graph as input and focus on the optimisation process [Grisetti *et al.*, 2007b] [Kummerle *et al.*, 2011] [Sunderhauf and Protzel, 2012]. In this paper we instead focus on using the output from the laser range finder and position tracking algorithm to generate the graph needed for optimisation, and how the optimisation process can be modified to account for uncertainties in the graph structure while it is built dynamically as the robot moves around the environment. We present a complete loop detection, graph building and optimisation algorithm.

Each node in our graph combines many scans into a local map of an area. Local maps are significantly more differentiable than individual scans, greatly reducing the likelihood of false positive loop closures. While existing Graph SLAM algorithms use the fit between local maps

to determine the covariance of the constraints between local maps [Bosse and Zlot, 2008] [Ratter *et al.*, 2013], we calculate the covariance of each scan added to a local map and use the resulting internal covariance of the local maps instead (Section 3.1). This enables the optimisation process to correct for significant errors in ICP registration.

A potential drawback with Graph SLAM approaches is that to reduce false positives, loop closures have to be delayed until a very good match with an existing node is found. This means a robot can visit a partially overlapping area of the existing map without closing a loop. We address this problem by adding constraints to align local maps that are already partially overlapping, and removing these constraints during optimisation if they are found to be false positives (Section 3.2). The additional constraints generated by this secondary type of loop closure, called *partial loop* constraints, increases the accuracy of the map optimisation.

While local maps improve the accuracy of loop closure detection and make the optimiser more efficient, errors in position tracking made inside local maps are not corrected during the optimisation process. We address this by introducing a two step optimisation algorithm (Section 3.3), which firstly optimises across the graph of local maps, and then corrects for position tracking errors inside each local map. Our graph construction minimises the number of nodes involved in the optimisation. RGB-D frames can be attached to laser scans inside local maps. The second stage of the optimisation allows the positions of the frames to be corrected, resulting in an accurate 3D map that can be generated on resource limited robots.

By implementing our HG-SLAM algorithm in the *Robot Operating System* (ROS) framework [Quigley *et al.*, 2009], we compared its accuracy and efficiency against laser range finder based SLAM solutions using datasets designed to be challenging. As shown in Section 4, our algorithm was able to accurately and efficiently generate global maps despite large errors in ICP registration, such as from the robot driving down a featureless corridor.

2 Related Work

One of the most common approaches to real-time SLAM has been particle filters. GMapping [Grisetti *et al.*, 2007a] uses a combination of laser readings and odometry to estimate the global map, and selectively carries out particle re-sampling to reduce the possibility of the correct map being deleted. Milstein et al. [2011] use an occupancy grid ICP algorithm as input into a FastSLAM system, thereby avoiding the need for motion encoders. However, to generate an accurate map, a large number of particles are required, making it computationally expensive. Additionally, maps with motion encoder or ICP failures can lead to no particles containing the correct map unless an infeasible number of particles are used.

Graph SLAM, first introduced in the seminal paper by Lu and Milios [1997], and in particular efficient graph optimisation algorithms, have become increasingly popular. Olson et al. [2006] parameterise the global position of the local maps as a function of the distance to their neighbours, and apply a stochastic gradient descent to optimise the graph. Grisetti et al. [2007b] represent the global position of maps as a tree structure. This minimises the number of nodes each loop closure affects during optimisation, increasing both the speed and accuracy of the algorithm. We use a similar structure for our graph, but apply it to a least squares optimisation strategy. Least squares optimisation is used by recent state of the art approaches in Graph SLAM, such as the g2o framework [Kummerle *et al.*, 2011], as it converges faster than gradient descent. Sunderhauf and Protzel [2012] introduce a switch variable into the optimisation formula to remove the influence of false positive loop closures at the cost of substantially increasing the problem complexity. We apply a similar weighting of constraints, but use knowledge of the map locations to find false positives and then modify the graph.

These algorithms all take a graph as input, leaving loop detection and graph building to a front end solution, as opposed to our algorithm, which links front end graph loop detection and graph building with the back end optimisation process. The Hierarchical SLAM algorithm [Estrada *et al.*, 2005] also does this by using a series of feature based maps to construct a graph of the world. If loop closures are detected through maps containing similar features, a global optimisation procedure corrects the global map.

Instead of using features, Bosse and Zlot [2008] detect loop closures by correlating histograms representing local maps. This approach enables loop closures to be rapidly detected regardless of alignment errors. In our previous work [Ratter *et al.*, 2013], we presented an extension to this algorithm, with loop closures detected using a three stage process of covariance estimation, histogram correlation and ICP matching to efficiently and accurately detect loop closures. Our algorithm further extends these ideas by creating new local maps through considering the uncertainty in the alignment of laser scans, and by using secondary loop closure and optimisation strategies. These allow our algorithm to close loops when only partial traversing a previously visited area, to correct for alignment errors made inside local maps, and to correctly work in more challenging environments, such as featureless corridors.

The introduction of the Microsoft Kinect has seen a surge in RGB-D mapping solutions. Endres et al. [2012]

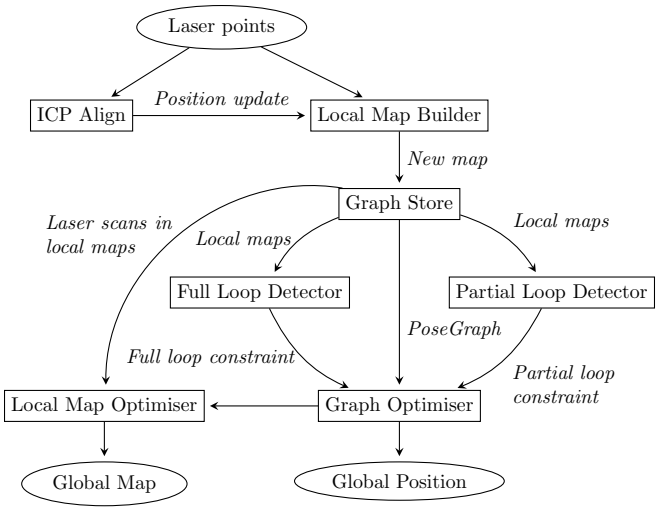


Figure 1: Overview of the architecture of our HG-SLAM algorithm

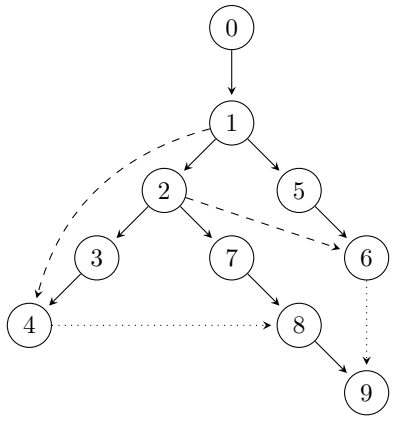


Figure 2: An example of the tree graph structure. Solid lines represent constraints from position tracking, dashed lines are full loop closures, and dotted lines are partial loop closures.

use visual features for position tracking and loop closing. However, this requires a feature rich environment. Kinect Fusion [Newcombe *et al.*, 2011] aligns successive 3D scans to an estimation of the surface generated from a truncated signed distance function. The Kintinuous Framework [Whelan *et al.*, 2013a] allows Kinect Fusion to work over long distances by shifting the voxel grid and extracting point cloud slices. Whelan et al. [2013b] combine Kintinuous with feature based loop closure. While many of these techniques produce accurate maps, they all are computationally intensive and unsuitable for low powered mobile robots. Additionally, they require slow and steady motion to maintain tracking, which is often an invalid assumption for robots.

3 Approach

Our SLAM system combines an ICP front end for position tracking, such as POGMBICP [Ratter and Sammut, 2013] or Hector SLAM [Kohlbrecher *et al.*, 2011], and our proposed HG-SLAM algorithm for loop closure detection, graph construction and graph optimisation. Fig. 1 shows the major components of our system. Once a local map is built (Section 3.1), it is added to the graph and checked for loop closures (Section 3.2).

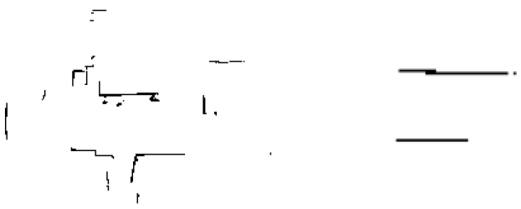
We connect nodes to each other such that the constraints from motion estimation form a tree of local maps. Normally, the parent of a new local map is the previous local map. If the previous local map closed a loop, the parent of the new local map is set to the node that was matched with the previous map, as shown in Fig. 2. For example, when the full loop closure is found between local maps 1 and 4 in Fig. 2, the normal motion estimation constraint from local map 4 to 5 is instead added to the graph to be from local map 1 to 5. This is achieved by using the loop closing constraint to transform the estimated motion between local maps 4 and 5 to be relative to local map 1. Fig. 2 also shows that our secondary type of loop closure, partial loop closures, do not cause a new branch to be created in the tree. We use this underlying tree structure of our graph to lessen the number of nodes involved in the optimisation (Section 3.3). If either type loop closure is found, the graph is optimised.

A graph edge, normally referred to as a *constraint*, is defined as δ_{ba}; an observation of node b from node a. Constraints also contain the information matrix, Ω_{ba}, describing the uncertainty of the constraint, δ_{ba}.

3.1 Local Map Generation

As the robot moves around an environment, laser scans aligned by the ICP front end are added to the current local map. Representing nodes in the graph as local maps containing many scans reduces the computation time needed to find loop closures and perform optimisation. This approach makes our algorithm suitable for mobile robots with low computing power. Additionally, many scans of a local area are more descriptive compared to single scans, making it easier to prevent false positive loop closures.

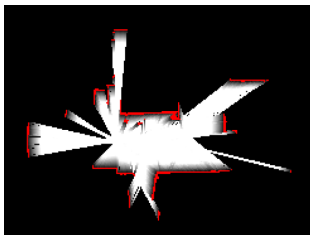
To build a local map, as shown in Fig. 3(a), laser points in a scan are added to an occupancy grid as they are observed. Every cell in the occupancy grid maintains an average of the positions of the points inside the cell, and the average normalised gradient of the laser points contained in the cell. Due to sensor noise, it is not adequate to estimate the gradient around a laser point by comparing it to its neighbour. Instead, the gradient around each point in the scan is estimated by using endpoints of a sliding window of the laser points in the



(a) Typical local map

(b) Featureless local map

(c) Local map from a revisit of the area near the local map in (a)



(d) Probability that cells in a map are free. The lighter the shading, the higher the probability. Black is unknown state.

Figure 3

scan.

This gradient is used to estimate the covariance between a new scan and the scans already in the local map. The current position of the robot relative to the start of the local map is $\kappa = (x, y, \theta)$. For a point p_i in the laser scan, $T_i(\kappa)$ is the coordinate of the scan point relative to the local map, and is calculated according to:

$$T_i(\kappa) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} p_{i,x} + x \\ p_{i,y} + y \end{pmatrix} \quad (1)$$

The value of the occupancy grid cell at this point is $h_i(\kappa) = M(T_i(\kappa))$. Differentiating $h_i(\kappa)$ yields the gradient of the match, allowing a Hessian matrix to be constructed according to $H_i = h_i'(\kappa)^T h_i'(\kappa)$,

$$H = \frac{1}{N} \sum_{i=1}^{N} \left(\nabla M(T_i(\kappa)) \frac{\partial T_i(\kappa)}{\partial \kappa} \right)^T \left(\nabla M(T_i(\kappa)) \frac{\partial T_i(\kappa)}{\partial \kappa} \right) \quad (2)$$

where $\nabla M(T_i(\kappa))$ is the gradient of the occupancy grid cell $M(T_i(\kappa))$. It is inverted to give the covariance of the scan. The sum of the covariance of each scan gives the total internal covariance of each local map, Σ_a. This represents the degree of uncertainty between the start of the current global map and the start of the next global map (b). The inverse of Σ_a gives the information matrix, Ω_{ba}, of the constraint. This is stored in the graph and is used as part of the optimisation process. The covariance between each scan is also used in the second stage of the optimisation.

It is desired that each local map be large enough to reduce the number of local maps and to make each local map sufficiently distinct. However, each local map should be small enough so that a robot revisiting the same area, regardless of the path it traverses, will generate a similar local map. In our previous work [Ratter *et al.*, 2013], we created new local maps after the robot had traveled a fixed distance from the starting point. There are cases though where this is not ideal, such as when robot motion is not detected while it traverses a featureless area (Fig. 3(b)) as the local maps could contain a large part of the global map. Therefore, we propose creating new local maps either when the robot has moved a certain distance or when the internal covariance of the local map indicates that the robot is unsure of its position inside the local map. If a RGB-D camera is being used to generate a globally consistent 3D map, each frame is attached to the nearest laser scan in time, allowing its position to be moved as the graph is optimised.

Local maps also store the probability a cell of the occupancy grid is free. The probability is approximated while the local map is being built. When the first laser point is added a cell in the occupancy grid, the cells on the line between the first laser point and the robot are marked with the probability the cell is free according to the formula

$$P_{cell} = \frac{\log(1 + d_c)}{\log(1 + d_r)} \quad (3)$$

where d_c is the distance between the cell and the laser point, and d_r is the distance between the robot and the laser point. As shown in Fig. 3(d), this equation means that cells that are close to the laser point are marked with a lower probability of being free. It also models the increased uncertainty of the position of laser points that are far away from the robot. This is used to evaluate the likelihood of a loop closure, as described in the following Section.

3.2 Loop Closure Detection

We use two loop closure strategies in our algorithm. Our *full loop* closure method accurately and quickly detects loop closures even if large errors have been made in ICP registration. We use the same three step process as in our previous work [Ratter *et al.*, 2013]. The covariance between maps is used to estimate maps potentially in the same area. Histogram correlation then compares the structural similarity of potential matches. If successful, fine alignment is made by ICP. We increase the robustness of this method by calculating the matching probability of the maps. This is given by

$$P_{match} = 1 - \frac{1}{N} \sum_{i=1}^{N} M_Y(T(p_i)) \quad (4)$$

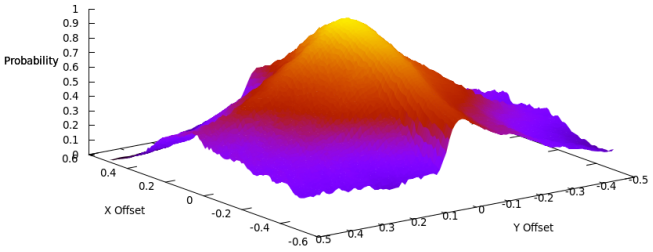


Figure 4: Graph of the matching probability of local maps of the same area at different offsets of their true match at (0,0)

where $T(p_i)$ transforms a point p_i into the reference frame of the candidate map, and $M_Y(T(p_i))$ gives the probability of the cell matching p_i being free, calculated from Eq. 3. Fig. 4 shows how the probability of the match between Fig. 3(a) and Fig. 3(c) changes at different alignment offsets once they have been aligned by ICP.

While this method is robust to incorrect loop closures, if the maps are only partially overlapping the histogram detection will eliminate them as a possible loop closure to avoid false positives. This means that the global map will not be corrected if the robot is only partially traversing a previously visited area. This delay in loop closing can cause a problem for many higher level robotic behaviours, such as path planning. We address this by adding a second type of loop closing constraint. These *partial loop* constraints allow small errors in position tracking to be corrected by the optimiser to produce a globally consistent map when full loop closure is not possible. Partial loop constraints also increase the accuracy of the SLAM position and quality of the global map after optimisation as they add extra constraint information.

Partial loop closures are detected using the process:

1. Local maps whose current global position is within an area where the local maps could overlap are found.
2. If laser points overlap cells marked with a probability of being free or occupied cells, the matching probability is calculated using Eq. 4.
3. An ICP match is performed using the POGMBICP algorithm [Ratter and Sammut, 2013].
4. If the ICP algorithm converges, the matching probability is recalculated according to the new alignment. If the probability increases above a threshold by which few laser points could overlap free cells in the candidate map, a partial loop closure constraint is generated.

As opposed to full loop closures, partial loop closures do not change the connection of the next local map in the graph as they are considered to be less reliable (shown by the dotted lines in Fig. 2). This is due to false positives being possible if large errors in position tracking cause local maps to incorrectly overlap. As will be described in the following Section, the optimisation process removes incorrect closures. Once either type of loop closure is identified, the information matrix of the constraint, Ω_{ba}, is calculated using Eq. 2.

3.3 Graph Optimisation

When either type of loop closing constraint is added to the graph, the graph is optimised using a two step process. First, the positions of the local maps are changed to reduce the errors in the graph caused by loop closing constraints. Secondly, the positions of the laser scans inside local maps are optimised to remove the error between the last scan added to a local map and the first scan added to the child local map.

The global optimisation of the graph can be expressed as a energy minimisation problem according to:

$$F(\mathbf{x}) = \sum_{\langle a,b\rangle \epsilon \mathbf{C}} w_{ba} e_{ba}(\mathbf{x})^T \Omega_{ba} e_{ba}(\mathbf{x}) \tag{5}$$

The goal of the optimisation is to find the configuration of global positions of the local maps, $\mathbf{x^*}$, that minimises Eq. 5. Here, $\mathbf{C}$ is the set of all constraints δ_{ba}, w_{ba} is an optional parameter to weight the influence of the constraint, and e_{ba} is the error given by the constraint, calculated

$$e_{ba} = x_b \ominus x_a - \delta_{ba} \tag{6}$$

where $\ominus$ is the inverse of the motion composition operator $\oplus$ as defined by Lu and Milios [1997], and x_b and x_a are the global positions of local maps b and a respectively.

We solve $F(\mathbf{x})$ by formulating it as a least squares minimisation, using a method similar to the g2o framework [Kummerle *et al.*, 2011]. This is done by approximating the error function using the first order Taylor expansion around an initial guess $\breve{x}$

$$e_{ba}(\breve{x} + \Delta x) \approx e_{ba}(\breve{x}) + J_{ba}\Delta x \tag{7}$$

where J_{ba} is the Jacobian of $e_{ba}(\breve{x})$, and is calculated from the first order partial derivatives of Eq. 6. The approximation of the error can be substituted into Eq. 5 to give:

$$F(\mathbf{x}) = \sum_{\langle a,b\rangle \epsilon \mathbf{C}} c_{ba} + 2d_{ba}\Delta x + \Delta x^T H_{ba} \Delta x \tag{8}$$

where $c_{ba} = w_{ba} e_{ba}^T \Omega_{ba} e_{ba}$, $d_{ba} = w_{ba} e_{ba} \Omega_{ba} J_{ba}$ and $H_{ba} = w_{ba} J_{ba}^T \Omega_{ba} J_{ba}$. This is quadratic in Δx, so can

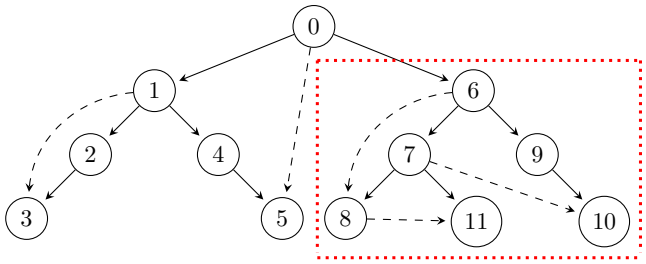


Figure 5: An example graph structure for optimisation. When the loop is closed between nodes 8 and 11, only the nodes in the dotted rectangle are optimised.

be minimised by solving the linear system $H\Delta x* = -d$ using sparse Cholesky decomposition, with $H = \sum H_{ba}$ and $d = \sum d_{ba}$. The increments $\Delta x*$ are then added to the initial guess $\breve{x}$, $x* = \breve{x} + \Delta x*$, and the algorithm repeated until it converges, according to the Gauss-Newton algorithm. However, as Gauss-Newton will sometimes not converge, we use an additional damping term according to the Levenberg-Marquardt algorithm [Lourakis and Argyros, 2009], and solve $(H + \lambda I) = -d$.

It is desirable to only optimise a small section of the global map if possible, both to increase the efficiency of the optimiser itself, and to prevent the need to re-project all laser points from all local maps into the global map. If a full loop closure between the current local map b and a previous map a was performed, we connect the next local map created to map a, forming a tree. This minimises the nodes affected by a constraint. While this doesn't directly affect the complexity of the optimisation, unlike in the gradient descent method proposed by Grisetti et al. [2007b], we use the tree structure to work out the minimum number of local maps that will be moved by the addition of the new constraint. Every node in the tree stores the minimum optimisation level, l_{min}, by which all nodes in the sub-tree with its root at l_{min} will need to be included in the optimisation. When a new loop closure is added to the tree, the l_{min} value of all nodes on the path of the constraint, $E_{i \to j}$, is set to the l_{min} value of the node on the path that is closest to the root of the tree. This is shown in the example in Fig. 5, where when the loop closing constraint is added between local maps 8 and 11, only the local maps in the sub-tree with root local map 6 are optimised.

As partial loop closures only affect the recent heading and map structure, only the constraints that have been added to the graph since the last full loop closure are optimised when a partial loop closure is detected. In the example shown in Fig. 2, when the partial loop closure between node 6 and 9 is added, only nodes 2, 7, 8 and 9 are included in the optimisation. If the partial loop closure involves a node that is not in the current branch of the tree, as it does in this example, the constraint is transformed to be from the root of the current branch instead. The partial constraint between node 6 and 9 would be transformed to be from nodes 2 to 9, using the current offset in position between nodes 2 and 6.

We use the parameter w_{ba} in Eq. 5 for weighting the confidence of a partial loop closing constraint. It is the percentage overlap of maps a and b. A lower overlap increases the possibility that the constraint is a false positive. w_{ba} is always 1 for full loop closures. False positives are detected by firstly temporarily setting w_{ba} to 0 for all partial loop constraints when the graph is optimised after the addition of a full loop constraint. If this optimisation results in the local maps of a partial loop closure no longer overlapping, it is considered a false positive, and the constraint is removed.

In spreading the errors of constraints across nodes depending on their internal covariance, the amount of error between a local map and the local map that was created immediately following is determined by the degree of uncertainty inside the local map. Once the map has been globally optimised, there can be discontinuities between the final scan added to a local map and the first scan added to the following map. The second stage of the optimisation process therefore involves spreading the error, e, between the final scan, N, in a map and the first scan in the following map over all the scans in the map according the the covariance of each scan. The change in position of a scan c, Δx_c is given by:

$$\Delta x_c = \frac{\sum_{n=1}^{c}(\mathtt{cov}_n)}{\sum_{n=1}^{N}(\mathtt{cov}_n)} e \tag{9}$$

where the covariance of the local maps Σ_n is written as $\mathtt{cov}_n$ for clarity.

If a global 3D map is being created using a RGB-D sensor, each frame is attached to the nearest laser scan in time inside a local map. The global position of each Kinect frame can be updated to the new global position of its associated laser scan after the second stage of the optimisation.

4 Evaluation

To evaluate our approach, we captured several datasets using a Turtlebot running the ROS [Quigley *et al.*, 2009] framework, and compared it against SLAM algorithms publicly available as ROS packages. The Turtlebot is equipped with a Hokuyo URG-04LX-UG01 laser range finder and a Microsoft Kinect. This laser range finder generates 1024 points per iteration and has a field of view of 240° with a maximum range of 5.6m. The datasets were run at the capture frame rate on a test rig consisting of a 3.4GHz Core i7 CPU and a NVIDIA 570GTX GPU. This GPU has 480 CUDA cores.

Table 1: Comparison of the accuracy of SLAM Methods. Final position error is the error at the end of the robot's trajectory. The trajectory error is the average error around the entire path compared to the ground truth.

Algorithm	Final Position Error		Trajectory Error	
	Dist (m)	Angular (°)	Dist (m)	Angular (°)
HG-SLAM	**0.03**	6.87	**0.04** ± 0.05	**0.57** ± 1.15
POGMBICP	11.06	14.88	0.44 ± 1.61	**0.57** ± 1.72
HG-SLAM (no partial loop closures)	0.20	**6.15**	0.06 ± 0.07	**0.57** ± 1.15
HG-SLAM (no per scan covariance)	10.68	12.26	0.19 ± 0.42	1.72 ± 2.86
Hector SLAM [Kohlbrecher *et al.*, 2011]	15.84	31.51	2.17 ± 3.75	14.32 ± 25.78
HG-SLAM (with Hector SLAM)	0.10	11.45	0.14 ± 0.38	4.01 ± 15.47
FastSLAM (with POGMBICP) [Milstein *et al.*, 2011]	8.82	19.3	2.89 ± 3.09	8.02 ± 14.90
GMapping (with wheel odometry) [Grisetti *et al.*, 2007a]	18.46	59.0	1.76 ± 3.04	1.72 ± 2.29
GMapping (with POGMBICP)	10.83	14.8	0.28 ± 0.59	5.16 ± 16.04

Table 2: Timing comparison of SLAM methods.

Algorithm	Time Per Iteration (ms)
HG-SLAM	0.5
HG-SLAM (no partial loop closures)	0.4
HG-SLAM (no per scan covariance)	0.2
HG-SLAM (Loop close + Optimise)	8
POGMBICP	3.3
HG-SLAM + POGMBICP	3.8
Hector SLAM [Kohlbrecher *et al.*, 2011]	2.5
Fast SLAM [Milstein *et al.*, 2011]	165
GMapping [Grisetti *et al.*, 2007a]	450

The datasets were designed to include many different environmental challenges. They were captured by driving the robot around an office environment in around three loops totaling 80 - 120m of distance traveled, arriving precisely back in the starting position. The datasets contain several large loops that allow drift errors in ICP to accumulate, and several areas with few objects in the range of the laser. One of the loops includes a long featureless corridor, which presents a difficult challenge for any SLAM solution as ICP algorithms cannot detect any movement down the corridor. Parts of the office were visited several times to show alignment errors between matching local maps. Table 1 shows the accuracy averaged across each dataset of the algorithms. We measure the accuracy of the algorithms in two ways. The error in the final position of the robot gives an indication of how accurately the loops were closed. The trajectory error measurements were calculated using the SLAM comparison algorithm by Burgard et al. [2009]. Over 1000 relations (displacements between scans) were aligned by hand for each dataset. The errors in Table 1 represent the average absolute errors and the standard deviations across all the relations, and give an measure of the accuracy of the path around the map. A map of one dataset, generated by our HG-SLAM algorithm, and the route taken, is shown in Fig. 6.

4.1 HG-SLAM

We used the *Parallel Occupancy Grid Metric Based* (POGMBICP) algorithm [Ratter and Sammut, 2013] to generate ICP position updates for our HG-SLAM algorithm, with no wheel odometry used. As can be seen in Table 1, POGMBICP by itself had an error of just over 11m. Fig. 7 shows the global map. While it shows the ICP is for the most part accurate, it drifts around the large loop in the top left of Fig. 7, and substantially underestimates the distance traveled down the featureless corridor in the middle right.

Running our HG-SLAM algorithm alongside POGMBICP reduced the error to just 3cm, and the final map is shown in Fig. 6. It was able to correct for drift around the large loop. The use of per-scan covariance meant that the error introduced by the featureless corridor was correctly spread almost entirely over the local maps in that corridor, with the other maps in the path of the constraint barely moving. This is shown by Figs. 9(a) and 9(b), which depict the area around the featureless corridor in the global map immediately before (a) and immediately after (b) the loop closure was identified and the map optimised. As the relative positions of the local maps containing the featureless corridor change considerably, the optimisation of the scans inside each local map ensure that the corridor is accurately formed. Fig. 9(c) shows the global map just before the second level of the optimisation is performed, with discontinuities between the local maps clearly visible, and Fig. 9(d) shows the correction after the second level optimisation finishes.

Aside from allowing the global position and map to be corrected earlier, which may be important for autonomous tasks, Table 1 shows that the use of partial loop closures increases the accuracy of HG-SLAM. This happens because the extra constraints, particularly in areas that the robot repeatedly visits, ensure that overlapping maps stick together even when some of them are involved in a large loop closure. This can be visually seen in the final map, with Fig. 9(e) showing an area of the map visited several times in different loops when partial loop closures are not used, and Fig. 9(f) showing that the use of partial loop closures improves the alignment between the local maps in that area.

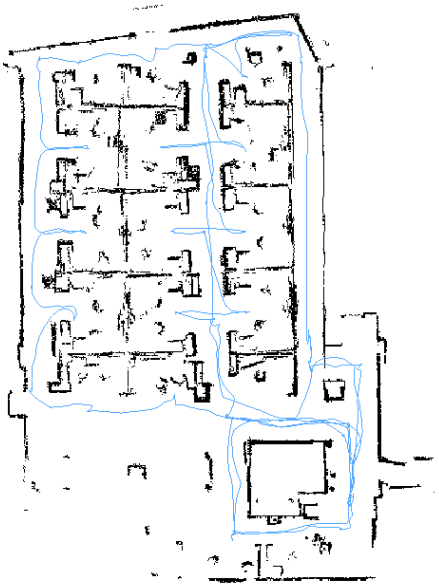


Figure 6: Map generated using our HG-SLAM algorithm. The blue line shows the robot's trajectory.

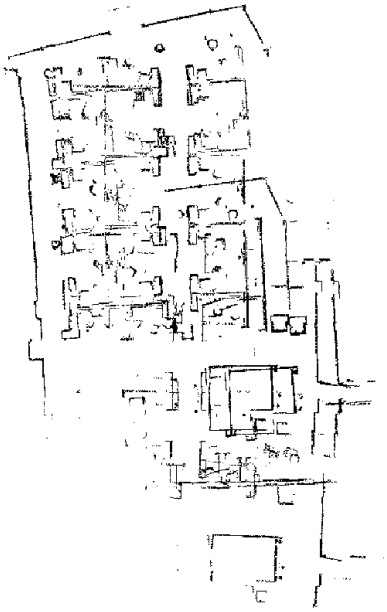


Figure 7: Map generated using only POGMBICP.

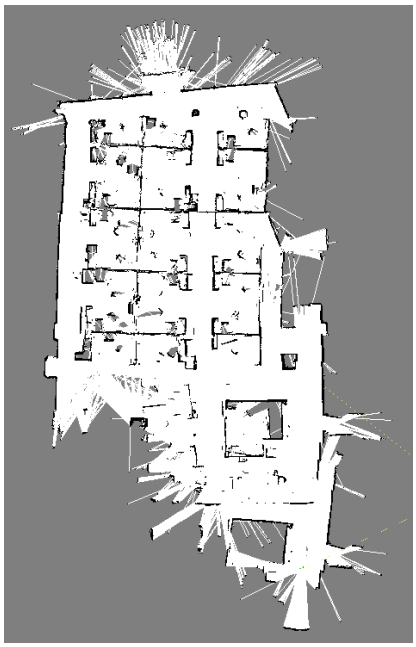
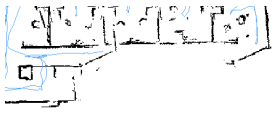


Figure 8: GMapping with POGMBICP as odometry.



(a) Before a loop including a featureless corridor has been closed.

(b) After a loop including a featureless corridor has been closed.

(c) End of featureless corridor without correcting errors inside a local map.

(d) End of featureless corridor correcting for errors inside a local map.

(e) Repeated visits to an area with no partial constraints.

(f) Repeated visits to an area with partial constraints.

Figure 9

As shown in Table 2, HG-SLAM is extremely efficient, adding only 0.5ms to the execution time of an iteration of the position tracking algorithm. The only processing performed per iteration is calculating the per scan covariance and adding the scan to the local map. Only when a local map is finished and a potential loop closure is identified does it take a non-trivial amount of time to run, taking an average of 8ms.

4.2 Comparisons to Other Algorithms

We tested the particle filter based GMapping algorithm [Grisetti *et al.*, 2007a] using either wheel odometry or POGMBICP as input. Table 1 shows that GMapping is unable to correct for the inaccuracy in the wheel odometry, and becomes lost. GMapping was able to correctly adjust for drift in ICP, but was unable to correct for the ICP failures in the featureless corridor. Figure 8 shows the most accurate global map we were able to obtain through tuning the GMapping parameters. We also ran an experiment with the FastSLAM implementation by Milstein et al. [2011], and received similar results to GMapping. Both algorithms were unable to correct for the featureless corridor because the large error in ICP meant that no particles contained an accurate map. If the gain was set such that the particles were spread widely enough, they were spread too thinly to have a good chance of containing an accurate map. Table 2 shows that both particle filter approaches are an order of magnitude slower than HG-SLAM.

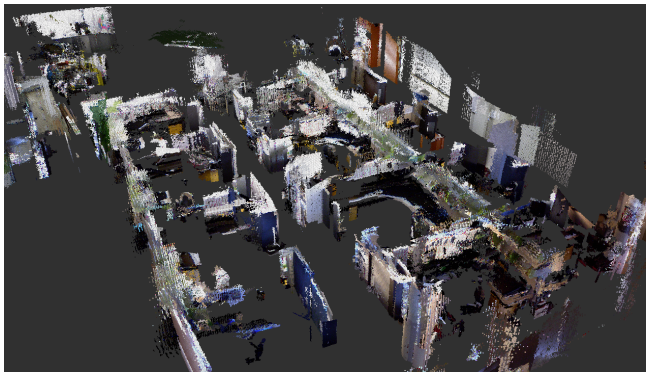


Figure 10: Screenshot of the 3D map from a dataset. The floor and the ceiling are not drawn for clarity.

One of the main laser range finder based ICP algorithms available in ROS is Hector SLAM [Kohlbrecher *et al.*, 2011]. We found that errors in its position tracking appeared in similar places to POGMBICP, albeit with slightly more drift. We used Hector SLAM as the ICP front end for our HG-SLAM algorithm, and these mistakes were corrected.

4.3 RGB-D Mapping

We also used the RGB-D data captured in our datasets to generate globally consistent 3D maps of the environment. An example map is shown in Fig. 10, with points on the floor and ceiling not drawn for clarity. The map was generated on dual core 2.8GHz computer with a very low range NVS300 graphics card. This shows that attaching RGB-D scans to laser scans inside local maps can produce large and accurate 3D maps on low powered hardware.

5 Conclusions

We have presented an efficient Hierarchical Graph SLAM algorithm that combines loop detection, graph building and graph optimisation to create a complete SLAM algorithm for mobile robots. Our algorithm is able to produce a globally consistent and undistorted map despite large errors in ICP by using the internal covariance of each local map to determine when to create a new local map, and by using this internal covariance to estimate the covariance of the constraints between adjacent local maps for graph optimisation.

Our use of partial loop closures, which are able to identify loops when only partially revisiting areas, increase the accuracy of the optimisation as they add more constraints, while enabling their influence to be removed if they are found to be false positives. The accuracy of the final map produced is further enhanced by the two level optimisation process, which allows errors made inside the local map to be corrected. We showed that these novel components of our HG-SLAM algorithm allowed it to efficiently produce globally consistent maps using datasets designed to be challenging.

References

[Bosse and Zlot, 2008] M. Bosse and R. Zlot. Map matching and data association for large-scale two-dimensional laser scan-based slam. *The International Journal of Robotics Research*, 27(6):667–691, 2008.

[Burgard *et al.*, 2009] W. Burgard, C. Stachniss, G. Grisetti, B. Steder, R. Kummerle, C. Dornhege, M. Ruhnke, A. Kleiner, and J.D. Tardós. A comparison of SLAM algorithms based on a graph of relations. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2089–2095. IEEE, 2009.

[Endres *et al.*, 2012] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. An evaluation of the RGB-D SLAM system. In *IEEE International Conference on Robotics and Automation*, pages 1691–1696. IEEE, 2012.

[Estrada *et al.*, 2005] C. Estrada, J. Neira, and J.D. Tardós. Hierarchical SLAM: Real-time accurate mapping of large environments. *Transactions on Robotics*, 21(4):588–596, 2005.

[Grisetti *et al.*, 2007a] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *Robotics, IEEE Transactions on*, 23(1):34–46, 2007.

[Grisetti *et al.*, 2007b] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard. A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In *Robotics: Science and Systems*, 2007.

[Kohlbrecher *et al.*, 2011] S. Kohlbrecher, O. von Stryk, J. Meyer, and U. Klingauf. A flexible and scalable SLAM system with full 3D motion estimation. In *Safety, Security, and Rescue Robotics*, pages 155–160. IEEE, 2011.

[Kummerle *et al.*, 2011] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *IEEE International Conference on Robotics and Automation*, pages 3607–3613. IEEE, 2011.

[Lourakis and Argyros, 2009] M. Lourakis and A. Argyros. SBA: A software package for generic sparse bundle adjustment. *ACM Transactions on Mathematical Software (TOMS)*, 36(1):2, 2009.

[Lu and Milios, 1997] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous robots*, 4(4):333–349, 1997.

[Milstein *et al.*, 2011] A. Milstein, M. McGill, T. Wiley, R. Salleh, and C. Sammut. A method for fast encoder-free mapping in unstructured environments. *Journal of Field Robotics*, 28(6):817–831, 2011.

[Newcombe *et al.*, 2011] R. Newcombe, A. Davison, S. Izadi, P. Kohli, et al. KinectFusion: Real-time dense surface mapping and tracking. In *Symposium on Mixed and Augmented Reality*, pages 127–136. IEEE, 2011.

[Olson *et al.*, 2006] E. Olson, J. Leonard, and S. Teller. Fast iterative alignment of pose graphs with poor initial estimates. In *IEEE International Conference on Robotics and Automation*. IEEE, 2006.

[Quigley *et al.*, 2009] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng. ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, volume 3, 2009.

[Ratter and Sammut, 2013] A. Ratter and C. Sammut. GPU Accelerated Parallel Occupancy Voxel Based ICP for Position Tracking. In *Australasian Conference on Robotics and Automation*, 2013.

[Ratter *et al.*, 2013] A. Ratter, C. Sammut, and M. McGill. GPU accelerated graph SLAM and occupancy voxel based ICP for encoder-free mobile robots. In *International Conference on Intelligent Robots and Systems*, pages 540–547. IEEE, 2013.

[Sunderhauf and Protzel, 2012] N. Sunderhauf and P. Protzel. Towards a robust back-end for pose graph SLAM. In *IEEE International Conference on Robotics and Automation*, pages 1254–1261. IEEE, 2012.

[Whelan *et al.*, 2013a] T. Whelan, H. Johannsson, M. Kaess, J. Leonard, and J. McDonald. Robust real-time visual odometry for dense RGB-D mapping. In *IEEE International Conference on Robotics and Automation*, 2013.

[Whelan *et al.*, 2013b] T. Whelan, M. Kaess, J. Leonard, and J. McDonald. Deformation-based loop closure for large scale dense RGB-D SLAM. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, 2013.