ROBUST SLAM

John B. Folkesson and Henrik I. Christensen [*]

[*] *Royal Inst. of Tech. Sweden*

Abstract: In this paper we describe an approach to simultaneous localization and mapping, SLAM. The method is based on a graphical representation of the information in the map. By consolidating this information intelligently we can eliminate almost all non-linear effects and solve topological constraints on the map very easily. The algorithm has been implemented in real-time on an outdoor robot and we have experimental validation of our ideas. We show how to reduce the graph by combining groups of measurements into a single node which we call a star node. These star nodes act as local sub-maps and are invariant to translations and rotations. These star nodes can then be used to impose topological consistency, such as, closing a loop. This loop closing is fast and could be used on-line.

Keywords: SLAM, Navigation, Localization, Data Association, Mapping

1. INTRODUCTION

This paper presents a new way to look at the SLAM problem, focusing on the issues that have challenged most of the methods reported in the literature. We have tried to combine the best features of existing approaches in a way that allows a robust solution to the problem.

Our guiding principle is to retain all the important information and use it in an exact way with no approximations as long as possible. We can then introduce approximations later when we have a more comprehensive view of the true state.

We represent our world model as a graph with some nodes representing the state and some nodes representing the measurements from our sensors. The updates and approximations can then be formulated as operations on the graph. The interactions of the nodes in a section of the graph can be approximated linearly in a local frame leading to a simplification that does not suffer from non-linear effects as the global map is distorted to impose topological constraints.

Using this general representation it is possible to efficiently evaluate data association and the structure of the model can be evaluated using an Energy based approach.

2. BACKGROUND ON THE SLAM PROBLEM

A number of methods have been proposed to do SLAM. The methods can be characterized as being batch or incremental, feature based or raw data based and topological or metric. All methods can be traced back to some probabilistic interpretation and many are explicitly formulated as maximum likelihood problems and use statistical methods for estimation of a solution.

In many environments the sensors will have many readings that simply are not useful for localization. These could be range scans of a bush or hedge, people, cars, sloping surfaces and so on. By trying to extract good features from the raw data we can significantly improve the signal to noise ratio. Incorrectly matching features from different scans leads to errors in the resulting maps. Once such an error has occurred, many of the existing methods have no way to correct or detect these errors. Matching errors are handled best by expectation maximization, EM, methods, (Thurn *et al.*, 1998). When applied in the pure form EM will find the best set of matches. In on-line systems of EM this desirable property must be partially sacrificed as a consequence of the limited time-windows (Martin and Thurn, 2002).

Global consistency, the problem of closing large loops, is much harder. Many methods simply cannot use the information that the robot has closed a loop to fix the map. Some methods are designed to use this type of information explicitly (Lu and Milios, 1997), (Gutmann and Konolige, 1999), (Golfarelli *et al.*, 1998). The idea of combining topological and metric information in a unified approach was shown in (Bosse *et al.*, 2003) to be a powerful one. Our method is similar to those methods and can also use the global consistency constraints to improve the map.

3. MAIN IDEA

We have the robot moving along a path periodically taking measurements. That is, a number of features are measured along with their uncertainty relative to the current pose of the robot. Between these measurements the robot has moved and the change in pose is also estimated.

Initially we make no approxamations by maintaining a state vector that includes all the pose estimates. The result is a long state vector but very sparse relationships between the elements. We can then exploit the simple structure to reduce the computational burden. After having collected data from an area for a while we can begin to consolidate the state vector but we never carry out the consolidation all the way.

We start by introducing the 'energy' of a measurement, which is nothing less than the log-likelihood of the measurement given the current state vector for the poses and features. This energy gives rise to a force which for Gaussian measurement models looks like a spring with a spring constant proportional to the inverse covariance.

Each measurement is described by its energy function which normally depends on part of the state vector. We form a graph with the state contained in 'state nodes' and the energy in 'energy nodes'. We have edges between the energy nodes and those state nodes that are needed to calculate that energy. The state nodes can be pose or feature nodes. The energy nodes can be due to dead-reckoning or feature measurements. Since the total energy of the system is a measure of the goodness of the fit one can use it to test the data association of measurements with features.

The map is represented by a graph. The energy nodes are measurements of the relative positions of two or more state nodes. Thus each pose node will be connected to the previous and next pose nodes through energy nodes representing the dead-reckoning of the robot motion between them. Each pose can also be connected to feature nodes if the feature was seen from that pose.

Each state node will need some world coordinates to specify its position. For the pose nodes these will typically be $\mathbf{x}_n = \begin{pmatrix} \theta_n, & x_n, & y_n \end{pmatrix}^T$. The features will also have some world representation. For example, we use the x-y of the endpoints to represent walls. The full state vector would be a concatenation of all the pose and feature vectors.

4. POSE-POSE ENERGY

The dead-reckoning estimate of the robot movement from pose n-1 to n will give our graph an energy node between pose nodes n-1 to n. We will then have a vector $\xi_{\mathbf{n}}$ expressing the measured displacement in the relative frame of the robot at pose n-1. We also have an estimate of the inverse of the covariance of $\xi_{\mathbf{n}}$, which we will call $\mathbf{k}_n$. We can form the difference between our measured displacement and the displacement according to the current state.

$$\eta_n = \begin{pmatrix} 1 & 0 \\ 0 & R_n \end{pmatrix} \cdot (\mathbf{x}_n - \mathbf{x}_{n-1}) - \xi_n \tag{1}$$

$$\mathbf{R}_n = \begin{pmatrix} \cos(\theta_{n-1}) & \sin(\theta_{n-1}) \\ -\sin(\theta_{n-1}) & \cos(\theta_{n-1}) \end{pmatrix}. \tag{2}$$

The energy of the measurement will then be:

$$E_n = (1/2)\eta_n^T \cdot \mathbf{k}_n \cdot \eta_n \tag{3}$$

The gradient of this energy is non-linear due to the rotation term which depends on θ_{n-1}. The other measurements will give rise to similar non-linear energy terms which we will simple denote by E_{nm} for a measurement from pose n to feature m. These full non-linear energy formulas are used to calculate the energy and its derivatives whenever the energy node is asked for such information.

5. FEATURE MATCHING

In addition to the energy in each energy node, we introduce a negative energy for each matched measurement,

$$E_m = (-\lambda)(N_m - 1), \tag{4}$$

where N_m is the number measurements associated with feature m. The parameter λ sets how much we can distort the graph edges to make a match and still lower the energy. It will be important when making the data associations. If the energy after matching two measurements is more then before, the match is probably wrong. The λ corresoponds to the size of the validation gates used in Kalman based estimation. The total energy is then given by,

$$E = \sum E_n + \sum E_{nm} + \sum E_m \qquad (5)$$

In general, finding the set of matches that will minimize this energy is not feasible. The problem is that trying more than a few match hypothesizes is computationally expensive. We can, however update the map such that the energy in each new pose-feature interaction does not exceed λ. We do this by first adding the node for the feature measurement, then calculating the new equilibrium position. One can then compare the total energy before and after adding the node using the equation above.

Thus, the data association problem is reduced to checking the change in energy. We need not be very careful when adding measurements to our graph. We use a relatively loose matching requirement and rely on this check of the energy to uncover any mistakes after recalculating the equilibrium.

We should mention here one of the major advantages of this representation of the map. We can at any time and with little effort go in and make changes to the graph. We can for example merge features by simply moving the edges from one feature node to the other and then discarding the node. We can check the energy before and after the merge to see if the features really should be merged. We can add information to existing energy nodes if we find for example more data can be associated with the feature. We can also remove measurements if we find that they no longer make sense. And, of course, the initialization of the features is trivial.

6. MAP UPDATE

We will now outline how the graph is built up as the robot moves about. At the k^{th} step we will need to add the k^{th} pose node and possibly some new feature nodes to the graph. We will also need to add energy nodes and edges between the k^{th} pose node and other nodes. The nodes can be added at their equilibrium positions assuming the positions of the old nodes are fixed.

Having added the new nodes and edges we must calculate the change induced on the rest of the map. We have to find the minimum of eq. (5) with respect to the pose and feature positions. We expand eq. (5) about the current state out to the quadratic terms. We thus get a gradient and a Hessian term contributed from each edge in our graph. Armed with these quantities we can devise several ways to drive the system towards its equilibrium position quickly. The terms associated with a particular node, call it A, look like:

$$E_A = \sum_i [E_i(\bar{\mathbf{x}}_A, \bar{\mathbf{x}}_i) + \bigtriangledown E_i(\bar{\mathbf{x}}_A, \bar{\mathbf{x}}_i) \cdot \begin{pmatrix} \Delta\mathbf{x}_A \\ \Delta\mathbf{x}_i \end{pmatrix} + (1/2) \begin{pmatrix} \Delta\mathbf{x}_A, & \Delta\mathbf{x}_i \end{pmatrix} \cdot \bigtriangledown^T \bigtriangledown E_i(\bar{\mathbf{x}}_A, \bar{\mathbf{x}}_i) \cdot \begin{pmatrix} \Delta\mathbf{x}_A \\ \Delta\mathbf{x}_i \end{pmatrix}], \qquad (6)$$

where $\Delta\mathbf{x}_A = \mathbf{x}_A - \bar{\mathbf{x}}_A$, etc. Some more notation:

$$\mathcal{G} = \bigtriangledown E_A = \begin{pmatrix} \mathcal{G}_A \\ \mathcal{G}_i \end{pmatrix} \qquad (7)$$

$$\mathcal{H} = \bigtriangledown^T \bigtriangledown E_A = \begin{pmatrix} \mathcal{H}_{AA} & \mathcal{H}_{Ai} \\ \mathcal{H}_{iA} & \mathcal{H}_{ii} \end{pmatrix} \qquad (8)$$

Here we just call everything x and make no distinction between pose and feature nodes. The subscript i indicates all the coordinates that are not tied to node A but share an energy node with A. An important number for us is the prediction of the energy gain from moving a node to the minimum position holding all other nodes fixed. This can easily be calculated. First the new node position would be:

$$(\mathbf{x}_A - \bar{\mathbf{x}}_A) = -\mathcal{H}_{AA}^{-1} \cdot \mathcal{G}_A \qquad (9)$$

Then the change in energy would be:

$$\triangle E_A = -(1/2)\mathcal{G}_A \cdot \mathcal{H}_{AA}^{-1} \cdot \mathcal{G}_A \qquad (10)$$

This $\triangle E_A$ will be used to make decisions as to what optimization method to use and whether an update is needed.

The simplest and slowest method is to use steepest decent. We use this as a last resort when near a saddle point. This is the situation that eq. (10) gives $\triangle E_A > 0$. In that case, we need to move the node away from the saddle point, hopefully on the correct side. We move in the direction of minus the gradient a small step and then recalculate and repeat until no significant change occurs. The step size is increased if the change in energy seems to indicate a flat region, (ie. the change is given by the gradient times the change in x), and decreased if the curvature is too high.

The next simplest method is to use eq. (9) directly. This will move the node to the bottom of the energy surface but the higher order terms in the energy might require us to iterate this a number of times until there is no significant change in the energy. We will refer to the use of these two methods, (steepest descents and eq. (9)), for per node minimization, as relaxing the node.

Having implemented only this much we found that the maps were already looking better than our previous SLAM maps. The updates were rather slow however. One must relax the new node then all the adjacent nodes then if any of those changed one must relax any adjacent to them and so on. This causes the update to move back and forth between nodes a lot when what is needed is to move a set of nodes simultaneously. We were able to significantly speed up the procedure by doing what we call chained updates on the pose nodes.

For that we exploit the special nature of our problem. We have a chain of pose nodes connected by the dead-reckoning edges. Let us consider pose node A with the previous pose node labeled B. We will consider the features and the next pose node as fixed while nodes A and B are variable. In this subspace, we can always eliminate pose node A's coordinates in terms of B's like this:

$$(\mathbf{x}_A - \bar{\mathbf{x}}_A) = -\mathcal{H}_{AA}^{-1} \cdot [\mathcal{H}_{AB} \cdot (\mathbf{x}_B - \bar{\mathbf{x}}_B) + \mathcal{G}_A] \quad (11)$$

Making this substitution we find the edge AB now contributes some extra terms to B's gradient and Hessian. We now move to B and repeat the procedure using the gradient and Hessian with the extra terms added to them. First test if B needs an update, (ie. if eq. (10) gives $\triangle E_B <$ some threshold). Then, if an update is needed, eliminate B in terms of the previous pose, C. Then move to C.

When we get to a node that doesn't need an update we can then use its coordinate values to update its next node, then the next node's coordinates to update its next node, and so on until we get back to node A.

Using these techniques we could grow the graph in real time as long as major inconsistancies were not encounterd.

7. REDUCING THE GRAPH

An optimization would be to simplify the graph by combining nodes. For a linear system one can reduce the graph with an exact formula. One simply chooses a state node that one wants to remove, call it node A. Then eliminates the state variables for A using the constraint equations for the equilibrium state. One then ends up with a sub-system with an energy that is a function of all the state nodes that were tied to the node A. The energy of this sub-system can be represented as a single energy node, which we call a star node, with edges radiating out to all these remaining nodes. So we combine all the energy nodes into star node and use gaussian elimination to remove the variables of node A.

In our case we have a nonlinear system. We must therefore expand the equations around the current equilibrium point. We chose to only eliminate pose nodes. We can start with eliminating $\mathbf{x}_1$. We then end up with a star node with edges to $\mathbf{x}_0$, $\mathbf{x}_2$ and $\mathbf{z}_i$ for the features, i, connected to $\mathbf{x}_1$. We can then continue by eliminating $\mathbf{x}_2$ and so on. If we continue all the way to the current pose we end up with a fully connected graph with only one pose variable. This is similar to what happens in the extended Kalman filter.

However, we should not carry the reduction though the whole set of poses. The resulting system would be too hard to update. That is because it is no longer sparsely connected. Therefore, we stop at some point and start a new star node instead.

If we first chose to eliminate every other pose node, we can then merge two star nodes separated by a pose node into a larger star node by eliminating the pose node between them. The details are trivial but numerous so we just say that it must be done correctly.

One continues in this way forming a sort of binary tree until there is only one star node representing the interactions of a section of the graph. The advantage of forming a tree rather than just a long chain is numerical stability as we end up with a sum of terms with a small number of multiplications in each term rather than multiplying a single matrix many times. The level of the tree gives a simple stopping criteria so we can for instance go to 7 levels (127 reductions).

We need not do the combining for the most recent poses. One can wait, say, 50 poses before starting to combine. So as each new pose is added to the graph we try to eliminate the pose 50 steps earlier. The advantage is that the linearization will be done about a better point and the match energy will have been observed for some time. This can be compared to the EKF where the linearization and matching takes place at once.

We need to deal with any invariance, zero eigen values, that the Hessians of the star nodes have or we will suffer numerical instability. We treat the translation/rotation invariance more carefully than the other invariances, (such as, sliding end-points tangent to walls). That is because we later will want to translate and rotate by arbitrary amounts when imposing topological constraints. We reduce the dimensionality of the star node by first changing variables to coordinates relative to one of the two pose nodes attached to the star. This is done exactly with the full Jacobian and so on. For the other invariances we simple project the relevant part of the state space with fixed projection matrices in the relative space. For example, for walls these might be the normal vectors.

We then have reduced the star node's dimension to something like half. We are able to calculate back to the original world coordinates by inverting the procedure. Thus we can move the star node and recalculate the energy without any linearization error. We also have an explicitly invariant representation that can be reduced using principle component analysis to a set of eigen values and eigen vectors. By adjusting the linearization point in the reduced space we can eliminate the gradient terms. The advantage of this clean representation will be apparent when we close the loop.

$$E_* = E_*(\bar{\mathbf{q}}) + (1/2)\sum_j E_{*j}[(\,\Delta\mathbf{q}\,)\cdot\mathbf{V}_{*j}]^2 \quad (12)$$

Here q are the final reduced coordinates, E_{*j} are the eigen values and V_{*j} are the eigen vectors. The q are expanded around the equilibrium point, $\bar{q}$.

The ideas here result in a representation of the map very similar to the sparse extended information filter (Lui and Thurn, 2003). The difference here is that we do not linearize once at the time of the observation and lock the approximation in that frame. Instead we linearize a section of the graph in a relative frame and at a time when it is more mature. This avoids most of the non-linear effects that would otherwise result from the local frame being rotated in the world frame. Another difference is that Lui and Thurn 'throw away' some information in order to prevent a fully connected system. We instead leave some pose nodes around which achieve the same end without loss of information.

8. CLOSING THE LOOP

Closing a loop presents two separate problems. First, how does the robot realize that a loop has been closed. Second how can the map be changed to close the loop. We will show a way to solve the second problem. Thus, we supply the algorithm with a list of pairs of features that are to be matched, (ie. one feature from the beginning of the loop and one from the end). The map will then be recalculated with the new constraints added.

The key insight that makes loop closing trivial is found in eq. (12). Imagine that one did not assume that the features seen from one star node were the same as those seen from another. Then the Δq=0 for those features. This in turn reduces the total energy in the remaining relative pose-pose q's to be block diagonal with 3x3 blocks. These blocks contain the information on the features seen from each star but not the information from associating the features from different stars. This then looks like the strong links of (Lu and Milios, 1997). We can impose topological constraints on this system by the method of Lagrange multipliers. For a simple loop this involves inverting N+1 3x3 matrices. Where N is the number of star nodes around the loop. More complex constraints can also be solved in this way.

After moving all the pose nodes using the method above we simply put the feature associations back in to fine tune the map. This strategy is highly effective and can be done on-line.

9. EXPERIMENTS

We have implemented our method on an ATRV2 outdoor robot. The map consists of walls measured with a SICK laser scanner. The dead-reckoning data fusion and the feature detection are described in an earlier paper, (Folkesson and Christensen, 2003). The output of the feature detection module is simply fed to the grapher.

The loop closing was tested by miss-tuning the algorithm to cause various loop closing errors. The algorithm worked very satisfactorily making very nice adjustments that improved most parts of the maps. The data had 7500 iterations. The average times per iteration was about 20-25 msec. This is only a little slower than our compressed Kalman filter. However since the time for each update is variable it is necessary to collect the data in a queue with separate process and then have the SLAM program read the data from the top of the queue and process it. We use a separate feature extractor/tracker to maintain this queue in our real-time implementation.

In figure 1 we show a typical map that failed to close. The result is shown in figure 2. This loop closing calculation took about 1 second.

10. DISCUSSION

We believe that the results of the experiments confirm the validity of our approach. We can make good quality maps in this straight forward way. The graphs contain information and computational machinery that allows us to do things like imposing global constraints on the map.

Our goal in this paper is to introduce the concept of graphical SLAM and to draw attention to the aspects of the SLAM problem that are the

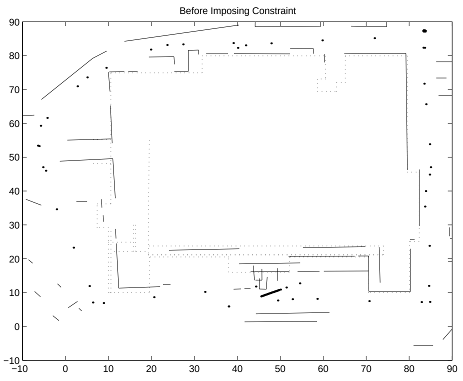

Fig. 1. This is an example of a map that did not close properly. The doted lines show the actual building outline. The path was traversed counterclockwise. The dots show the pose nodes that remain.

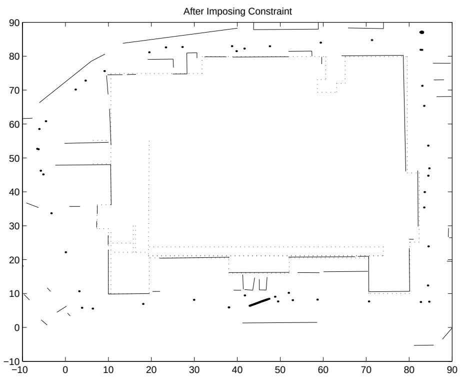

Fig. 2. Here we show the result of applying the loop closing constraint on the previous map.

most difficult. Those aspects are characterized by the appearance of inconsistancies in the map. A solution to the SLAM problem must be able to survive and profit from these inconsistancies when they are discovered.

We can compare the graphical representation of the map to the Kalman Filter type representation. The Kalman filter represents the probability density of the map as a Gaussian while the graphical method, even if it uses a Gaussian for the measurements, does not result in a Gaussian for the state space. So that the graph can represent much more complex probability surfaces.

CONCLUSIONS

We have presented an approach to solving the SLAM problem which avoids its three most troublesome stumbling blocks. The non-linear effects are eliminated by postponing the linearization and linearizing in a local frame. The data association errors are detected and corrected automatically. And finally large loops can be closed in a way that is consistent with all the information collected.

We have outlined an efficient implementation that is running the algorithm in real-time on an outdoor robot. Experimental results have also been shown to confirm that the algorithm does produce good maps. A comparison to a Kalman filter SLAM implementation on the same platform shows that the maps are in many ways superior.

The energy in the graph gives a useful measure of goodness for the map and allows one to compare two solutions. The graph gives a representation of a general map probability distribution and is not limited to 'Gaussian maps'.

REFERENCES

Bosse, M., P. Newman and J. Leonard et al. (2003). An atlas framework for scalable mapping. In: *Proc. of the IEEE International Conference on Robotics and Automation (ICRA03)*. Vol. 1. pp. 1899–1906.

Folkesson, J. and H. I. Christensen (2003). Outdoor exploration and slam using a compressed filter. In: *Proc. of the IEEE International Conference on Robotics and Automation (ICRA03)*. Vol. 1. pp. 419–427.

Golfarelli, M., D. Maio and S. Rizzi (1998). Elastic correction of dead-reckoning errors in map building. In: *Proc. of the IEEE International Conference on Intelligent Robots and Systems*. Vol. 1. pp. 905–911.

Guivant, Jose E. and Eduardo Mario Nebot (2001). Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *IEEE Transactions on Robotics and Automation* **17**(3), 242–257.

Gutmann, J. and K. Konolige (1999). Incremental mapping of large cyclic environments. In: *Proc. of the 1999 IEEE International Symposium on Computational Intelligence in Robotics and Automation*. Vol. 1. pp. 318–325.

Lu, F. and E. Milios (1997). Globally consistant range scan alignment for environmental mapping. *Autonomous Robots* **4**, 333–349.

Lui, Y. and S. Thurn (2003). Results fo outdoor-slam using sparse extended information filters. In: *Proc. of the IEEE International Conference on Robotics and Automation (ICRA03)*. Vol. 1. pp. 1227–1233.

Martin, C. and S. Thurn (2002). Real-time acquistion of compact volumetric 3d maps with mobile robots. In: *Proc. of the IEEE International Conference on Robotics and Automation (ICRA02)*. pp. 311–316.

Thurn, S., D. Fox. and W. Burgard (1998). A probalistic approach to concurrent mapping and localization for mobile robots.. *Autonomous Robots* **5**, 253–271.