Graphical SLAM using Vision and the Measurement Subspace*

John Folkesson, Patric Jensfelt and Henrik I. Christensen
Centre for Autonomous Systems
Royal Institute of Technology
SE-100 44 Stockholm
[johnf,patric,hic]@nada.kth.se

***Abstract*— In this paper we combine a graphical approach for simultaneous localization and mapping, SLAM, with a feature representation that addresses symmetries and constraints in the feature coordinates, the measurement subspace, *M-space*. The graphical method has the advantages of delayed linearizations and soft commitment to feature measurement matching. It also allows large maps to be built up as a network of small local patches, star nodes. This local map net is then easier to work with. The formation of the star nodes is explicitly stable and invariant with all the symmetries of the original measurements. All linearization errors are kept small by using a local frame. The construction of this invariant star is made clearer by the M-space feature representation. The M-space allows the symmetries and constraints of the measurements to be explicitly represented. We present results using both vision and laser sensors.**

***Index Terms*— SLAM, Vision, Graph, Features**

I. Introduction

Simultaneous localization and mapping, SLAM, is a central problem in mobile robotics. As a robot moves through an environment it builds a map and uses this partially built map to remain localized. Here a map is assumed to be composed of a relatively limited number of high fidelity features. These features need to be characterized in a way that allow them to be used for localization. Thus the geometry of the features takes on a central role in SLAM. Other characteristics can be important for matching the measurements to the features.

Many of the SLAM algorithms have their roots in the seminal work by Smith, Self and Cheeseman [1] where the stochastic mapping framework is introduced. In stochastic mapping, an extended Kalman filter (EKF) is used to estimate the pose of the robot and the feature parameters. Two problems with the standard EKF solution are, i) the complexity is $\mathcal{O}(N^2)$ where N is the number of features and ii) decisions cannot be reversed once made. The first of the problems has been the driving force behind much of the work on SLAM. The CEKF [2], FastSLAM [3] and SEIF [4], [5] are all recent examples of methods that build upon the EKF framework and address the complexity problem. The problem of not being able to revert a decision once it has been made is intimately linked with the problem of data association, i.e. how to associate measurement data to map data. Methods that deal with this problem have typically been working offline with the whole data set. Examples of this are [6] and [7].

Another key issue in SLAM is the ability to close loops. In addition to detecting when a loop is closed, the algorithm also needs to be able to deal with potentially large corrections to the map as a result of the loop closing. Combinations of topological and metric approaches such as [8] address this. This paper describes a graph based technique that addresses all problem mentioned so far. Others have used graphs for SLAM. In [9] the graph consists of places connected to one another by scan matching. In [10] the graph represents a Gaussian belief state implied by the measurements. That graph is closely related to SEIF as is our graph. The main differences between our method and SEIF are that in SEIF one commits to linearization and data association at once, as in EKF. In our method these commitments are softer and delayed. Also in SEIF one must prune weak edges from the graph to maintain sparseness. We achieve spareness by chosing to not carry the elimination of state variables as far as in SEIF. We thus never create the fill that destroys sparseness.

The feature representation used in this paper is similar to the SP-model [11]. In the measurement subspace representation, *M-space*, [12], we can represent the measured parts of the geometry of the feature. The dimensionality of features might increase over time as the accumulated measurement information changes their observability. The symmetries of the measurements are preserved by the representation. Additionally, constraints on the features such as shared corner points between features or horizontal constraints on lines can be made explicit. At the same time the transformation rules under translation and rotation of the coordinate system are well defined. Furthermore, the graphical SLAM implementation could be done without specifying the type of features and sensors that would be used. To verify the utility of our approach we present results using various combinations of camera and SICK laser scan data.

II. The Measurement Subspace Representation

We begin with a quick review the main points of the M-space feature representation [12]. See Appendix for more details. In our previous work we claimed that our representation allows easy interchange of SLAM algorithms. There

*This research has been sponsored by the Swedish Foundation for Strategic Research through the Centre for Autonomous Systems.

we presented an EKF implementation. Here we support our claim with a Graphical SLAM algorithm.

A feature is represented by a set of coordinates, $\{\mathbf{x}_f\}$ (i.e. sets of points) that have well defined transformation rules. The M-space is an abstraction of the information gathered on the feature as measurement data while the robot moves about. This information tells us that some directions of change in the $\{\mathbf{x}_f\}$ obey a known probability distribution. While other directions are as yet not well known. A projection matrix, $B(\mathbf{x}_f)$, is used to project out these 'known' directions. The B matrices are non-linear functions of $\mathbf{x}_f$. The change in the M-space is denoted by $\delta\mathbf{x}_p$. The main idea is to calculate changes in the M-space, $\delta\mathbf{x}_p$, through measurements and then propagate these to changes in the feature coordinates, $\delta\mathbf{x}_f$. The fundamental relations are:

$$\delta\mathbf{x}_p = B(\mathbf{x}_f)\delta\mathbf{x}_f, \tag{1}$$

$$\delta\mathbf{x}_f = \widetilde{B}(\mathbf{x}_f)\delta\mathbf{x}_p, \tag{2}$$

$$I_{pp} = B(\mathbf{x}_f)\widetilde{B}(\mathbf{x}_f). \tag{3}$$

Denote the coordinates of the feature relative to the robot by $\mathbf{x}_o$. The $\delta\mathbf{x}_f$ transform into $\delta\mathbf{x}_o$ under a coordinate change to a robot frame $\mathbf{x}_r$ as:

$$\delta\mathbf{x}_o = J_{of}\delta\mathbf{x}_f + J_{or}\delta\mathbf{x}_r. \tag{4}$$

Here J_{of} and J_{or} are the Jacobians from the transformation function $\mathbf{x}_o = \mathbf{x}_o(\mathbf{x}_f, \mathbf{x}_r)$ with respect to $\mathbf{x}_f$ and $\mathbf{x}_r$. The features are parameterized by a set of coordinates. These coordinates have well defined Jacobians. Thanks to the M-Space representation it is a simple matter of combining partial Jacobians for the coordinates that parameterize a certain feature to derive the above Jacobians.

III. Graphical Formulation of the SLAM problem

Our graphical SLAM method was introduced in [13]. Here we review the ideas there and then expand on them to incorporate the M-space. In our previous work we concentrated on the problem of closing a large loop. Here our concentration is on the features and their symmetries. We therefore use a wider variety of sensors and features. We also show here how to include multiple passes through the mapped area, as opposed to the single pass of our previous paper. We show how we can create junctions of different robot paths in the graph.

We will give an overview of the graph starting with the nodes. The nodes come in two basic types, state nodes and energy nodes. The state nodes are either pose nodes representing some pose that the robot visited and took measurements from or feature nodes that are associated with a feature. The state is then given by $\mathbf{x}_f$ and the measurements define the M-space changes $\delta\mathbf{x}_p$. For the robot pose we define $\mathbf{x}_f$ as the 3-D coordinates plus the three Euler angles and $\delta\mathbf{x}_p$ are the parts that are considered unconstrained, in our case x, y and θ[1]. We often distinguish robot pose coordinates by using an r as subscript rather than p.

The energy nodes represent the probabilistic information from the measurements. The simplest are the measurement nodes that calculate an *energy*, E, based on the current state of a pair of state nodes. This energy and its derivatives are non-linear functions of the state and are re-calculated as needed. The energy is given by:

$$E(\mathbf{x}_f, \mathbf{v}_i) = -\log(P(\mathbf{v}_i|\mathbf{x}_f)) - \Lambda\kappa_i, \tag{5}$$

where κ_i is 1 if the measurement is matched to an existing feature and 0 otherwise. The $\mathbf{v}_i$ is a measurement that may, for example, be odometry between two pose nodes or the measurement of features from some robot pose. The Λ parameter is used similarly to a threshold on the Mahalanobis distance. It gives a lowering of the energy for a match to an existing feature. So if the increase in energy from the first term in eq. (5) is less than this matching gain, the match is considered correct.

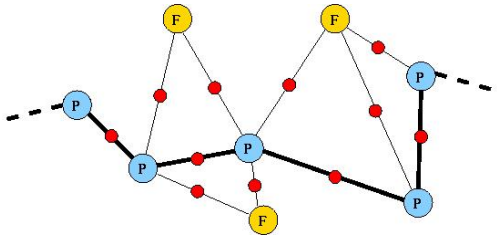


Fig. 1. Small part of a graph. The red/darker nodes are energy nodes. These can be either odometry nodes or measurement nodes. The measurement nodes connect a pose node with a certain feature node i.e. the data association. The lighter nodes are state nodes, where **P** denotes pose nodes and **F** feature nodes.

Figure 1 illustrates a small part of a graph. Typically we add a pose node to the graph connected via an odometry energy node to the previous pose node. Then we add any measurement nodes connecting this pose node to existing features, relaxing the graph after each addition. If any measurement nodes increase the energy we delete them, i.e. we reject that data association. For measurements to new features we create new feature nodes.

We will later see how to minimize the energy in the graph. To do so requires the derivatives of the energy. One normally has a Gaussian model for the feature measurements. We can denote the Gaussian innovation by $\eta(\mathbf{v}_i, \mathbf{x}_o)$ which has expectation value of 0. An incremental change in the innovation can be written:

$$\delta\eta = \begin{pmatrix} \underbrace{J_{\eta v}}_{measurement} & \underbrace{J_{\eta o}J_{or}}_{robotpose} & \underbrace{J_{\eta o}J_{of}\widetilde{B}_f}_{feature} \end{pmatrix} \begin{pmatrix} \delta\mathbf{v}_i \\ \delta\mathbf{x}_r \\ \delta\mathbf{x}_p \end{pmatrix}. \tag{6}$$

[1]In any actual implementation there would need to be some additional robot to sensor transformation. This could even include a pan tilt unit. All that can and was included in the pose node state without any problem. We ignore these points in the text for clarity.

The J's are all Jacobians and are evaluated at the linearization point. For normal measurement nodes all these non-linear terms are re-calculated each time the relevant states change. The covariance of η due to the measurement errors in v becomes;

$$R_{\eta\eta} = J_{\eta v}\sigma_{vv}^2 J_{\eta v}^T \tag{7}$$

The coordinate dependent part of the energy then looks like:

$$E = \frac{1}{2}\eta(\mathbf{x}_o)^T \cdot R_{\eta\eta}^{-1} \cdot \eta(\mathbf{x}_o) \tag{8}$$

and the gradient of energy is,

$$G = \eta \cdot R_{\eta\eta}^{-1} \begin{pmatrix} J_{\eta o}J_{or} & J_{\eta o}J_{of}\widetilde{B}_f \end{pmatrix} \tag{9}$$

A similar expression can be written for the Hessian of the energy. Here we made the assumption that η is small and so we can drop all the terms that look like derivatives of the Jacobians. This assumption is not necessary. We have investigated both ways and there is no major difference. However, this choice leads to positive semi-definite Hessians which is an advantage.

IV. Map Updates

In the previous section we saw how one could calculate an energy for the graph. We would like this energy to be minimized. One way to drive the graph towards lower energy is to chose a state node and relax it. This relaxation is performed by holding all other nodes fixed, so here we refer to only this one state node. The change in state $\Delta\mathbf{x}_p$ can be found from:

$$H_{pp}\Delta\mathbf{x}_p = -G_p \tag{10}$$

where H_{pp} and G_p are the Hessian and gradient of the energy with respect to this one state node. They are calculated by adding up the contributions from each energy node connected to this state node.

One can move systematically through a selection of nodes relaxing each node in turn and repeating until no significant change in energy is achieved. This is similar to doing a Gauss-Seidel iteration on a linear system. We organize this by examining the energy nodes. If the energy of an energy node changes by a significant amount after an iteration we include all its attached state nodes in the next iteration. Thus the selection will naturally move through the graph to the areas most in need of relaxation.

There is a way to speed up the convergence which worked well. This was to solve the sub-system of the chain of pose nodes going back from the current node while keeping the features fixed. This Hessian has a simple block tridiagonal form with 3x3 blocks. One could then relax a chain of poses to agree with the features in one step [2].

[2]A short note on implementation: First change variables to the changes in coordinates between pose nodes. The Hessian is then no longer tridiagonal but has a special form so that the complexity of solution is the same. The advantage is the Hessian is now strongly diagonally dominant.

V. Star Nodes

As the measurements become mature they can be combined together by linearizing them around an appropriate state. Part of the linearization will then be frozen at the chosen state. We show here how to take a set of energy nodes and combine them to form a star node with the correct invariances. The formed star node will be an energy node with edges to all the state nodes of the original energy nodes (see Figure 2). So in this section we restrict the state, $\mathbf{x}_f$, to only these state nodes.

We start by considering the gradient term of the energy calculated by adding up all the gradients from the set of energy nodes evaluated at the current state[3].

$$E = E(\bar{\mathbf{x}}_f) + G(\bar{\mathbf{x}}_f) \cdot \begin{pmatrix} \delta\mathbf{x}_b \\ \delta\mathbf{x}_q \end{pmatrix} + ... \tag{11}$$

Here we have split the state permutation vector $\delta\mathbf{x}_p$ into $\delta\mathbf{x}_b$ and $\delta\mathbf{x}_q$. The $\delta\mathbf{x}_b$ is the special robot pose to become the 'base' pose of the star and $\delta\mathbf{x}_q$ are the other pose and feature coordinates. The bar on the $\bar{\mathbf{x}}_\mathbf{f}$ is to indicate that this is the linearization point. Now consider the transformation of all $\delta\mathbf{x}_q$ coordinates to the base frame so that they become $\delta\mathbf{x}_o$. We can write the following identity:

$$I = \begin{pmatrix} I & 0 \\ -BJ_{of}^{-1}J_{ob} & BJ_{of}^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ J_{ob} & J_{of}\widetilde{B} \end{pmatrix} \tag{12}$$

Using eq. (4) we can write:

$$\begin{pmatrix} \delta\mathbf{x}_b \\ \delta\mathbf{x}_o \end{pmatrix} = \begin{pmatrix} I & 0 \\ J_{ob} & J_{of}\widetilde{B} \end{pmatrix} \cdot \begin{pmatrix} \delta\mathbf{x}_b \\ \delta\mathbf{x}_q \end{pmatrix} \tag{13}$$

$$G\begin{pmatrix} \delta\mathbf{x}_b \\ \delta\mathbf{x}_q \end{pmatrix} = G\begin{pmatrix} I & 0 \\ -BJ_{of}^{-1}J_{ob} & BJ_{of}^{-1} \end{pmatrix}\begin{pmatrix} \delta\mathbf{x}_b \\ \delta\mathbf{x}_o \end{pmatrix}. \tag{14}$$

The B and J_{of} matrices in eq. (14) are block diagonal matrices with blocks for each state node that is connected to the set of energy nodes. The J_{ob} is a column of blocks. So far we have only algebra. Now we argue that the energy of this set of energy nodes cannot change when $\delta\mathbf{x}_b \neq 0$ if $\delta\mathbf{x}_o = 0$. This is due to the fact that the original set of measurements was invariant under transformation of all the state coordinates to a new coordinate frame. In other words moving the base of the star without changing the positions of any states relative to the base should change nothing. Thus the result of multiplying G by the first column of the matrix above must be zero and we can remove part of the gradient without loss of information.

$$G_p = \begin{pmatrix} G_b & G_q \end{pmatrix} = G_q \begin{pmatrix} BJ_{of}^{-1} \end{pmatrix} \begin{pmatrix} J_{ob} & J_{of}\widetilde{B} \end{pmatrix} \tag{15}$$

[3]The Hessian term is also needed but we only show the details for the linear part as the quadratic part is the same thing done from both sides of the Hessian.

We define $Q = BJ_{of}^{-1}$ and $\widetilde{Q} = (J_{ob}, J_{of}\widetilde{B})$. Further we now have separated out the invariant part from G (inside the $\{\}$ below),

$$G_p(\bar{\mathbf{x}}) = \{G_q(\bar{\mathbf{x}})Q(\bar{\mathbf{x}})\}\widetilde{Q}(\bar{\mathbf{x}}). \tag{16}$$

Similarly,

$$H_{pp}(\bar{\mathbf{x}}) = \widetilde{Q}(\bar{\mathbf{x}})^T\{Q(\bar{\mathbf{x}})^T H_{qq}(\bar{\mathbf{x}})Q(\bar{\mathbf{x}})\}\widetilde{Q}(\bar{\mathbf{x}}). \tag{17}$$

Next we need to choose the best value of $\bar{\mathbf{x}}$ to calculate the values of H_{qq} and Q to save. We choose the value that gives $G_q = 0$ by solving the linear equation, moving the state nodes temporarily to the star equilibrium point, recalculating the values and repeating the process if needed. Thus, we can linearize the set of energy nodes at the state that this set indicates is best.

We can see that H_{qq} is the Hessian at the linearization point without the rows and columns of the base node. This matrix is in general singular which compels us to do a singular value decomposition of it keeping only the positive eigenvalues.

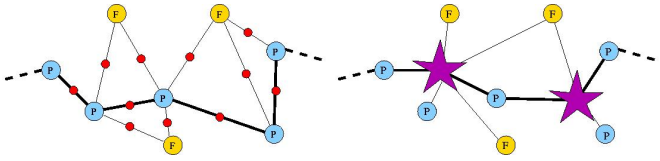
$$H_{qq} = \sum_{k=1}^{m} U_k \lambda_k U_k^T \tag{18}$$

Here λ and $\mathbf{U}$ are the eigenvalues and eigenvectors for the m positive eigenvalues. We can project the equilibrium point and the current state into the eigenvector subspace,

$$\bar{\mathbf{u}}_k = U_k^T Q\bar{\mathbf{x}}_o, \quad \mathbf{u}_k = U_k^T Q\mathbf{x}_o \tag{19}$$

We save $\bar{u}_k$, λ and Q. We define $\Delta u_k = u_k - \bar{u}_k$. The energy at any new point can be calculated by first transforming to the base frame and then projecting out u_k to get:

$$E = E(\bar{\mathbf{u}}_k) + \sum_{k=1}^{m} \frac{\lambda_k}{2}(\Delta \mathbf{u}_k)^2 \tag{20}$$



(a) (b)

Fig. 2. Introduction of a star node. a) initial state, b) Star nodes are introduced in the graph between every other pose node. The stars are connected to all state nodes that the energy nodes it replaced were connected to.

This is then the star node energy equation which gives the energy as a function of the current $\mathbf{x}_f$ variables. We can re-center around any state. So for the re-centered gradient at $\mathbf{x}_f$ we get,

$$G_p(\mathbf{x}_f) = \sum_{k=1}^{m} \frac{\lambda_k}{2}(\Delta \mathbf{u}_k(\mathbf{x}_f))U_k^T Q(\bar{\mathbf{x}})\widetilde{Q}(\mathbf{x}_f) \tag{21}$$

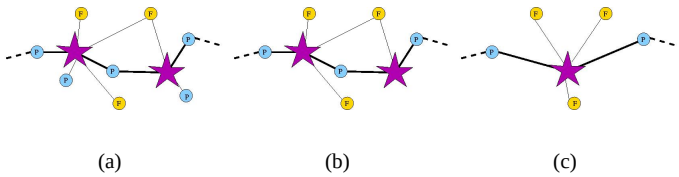
One only needs to recalculate all of $\widetilde{Q}(\mathbf{x}_f)$ if there has been significant changes.[4]

VI. Graph Reduction

Having shown how a set of energy nodes can be combined into a star node we can now reduce the complexity of our graph. We also show how new information collected upon returning to a region can be absorbed into the previously created star node. Thus multiple passes through a region will produce a path that is connected directly to the previous path, as opposed to via feature nodes. This would be important if one needed to impose global constraints on the system using the method in [13]. Here we do not need to impose the global constraints as all the loops are small enough to close naturally.

We start by observing that if a state node has only one edge and that edge is to a star node then the state of that node can be solved for in terms of the other nodes connected to the star. That is, the variables describing the state of the node can be expressed in the other node variables. The node can then be eliminated from the star. In practice one eliminates, (and deletes), any such nodes at the time the star is formed.

We can then select a pose node and form a star out of all its attached energy nodes, (Figure 2). The result will be that the pose node is eliminated (see Figure 3a-b) and the graph simplified. We begin reducing the graph by eliminating every other pose node. Then we get *level* 0 star nodes, (i.e. made from simply energy nodes). These then are separated by pose nodes. By again taking every other one of these pose nodes and forming a star we get level 1 stars, made out of level 0 stars (see Figure 3b-c). We try to reduce the graph by combining two stars at a time when they have the same level. In this way we get a more symmetric procedure which is better numerically and is more efficient computationally.



(a) (b) (c)

Fig. 3. Graph reduction. a) Initial state of graph from Figure 2. b) Pose nodes connected to only one star node can be reduced. c) Two star nodes of level l can be combined into one star nodes of level $l+1$.

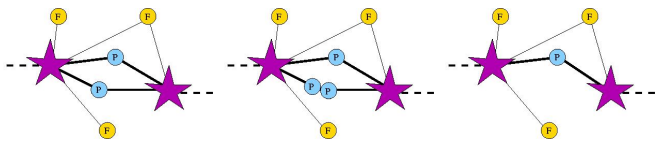
We can limit the size of the stars either by placing a maximum on the level or more fundamentally on M-dimension of the star node. The M-dimension is the total of the dimension of $\delta\mathbf{x}_p$ for all the attached state nodes. The M-dimensions

[4]In practice some rows, (i.e. those for the angle of lines), of $Q(\bar{\mathbf{x}})$ need to be rescaled when there is significant change to the feature coordinates. One can even recalculate the whole B part of Q but that is seldom necessary.

becomes a problem when forming the star if the eigenvalue problem eq. (18) takes too long to solve. We begin to have some problems for 40 or so dimensions. Most stars need not exceed 20 dimensions.

When the robot returns to a region previously visited we would like merge the stars from the previous path with those currently being formed to simplify the graph. Formally this is no problem as we can make a star out of any set of energy nodes. There are however some practical considerations. One is the size of the star node as we just mentioned. Another is that there must be enough feature nodes in common to specify the relative pose between the two base poses of the stars. Without that the two stars are essentially independent and the result of merging them will be a star with two disconnected blocks. In the case of a forward facing sensor this requirement might mean two stars formed when the robot was at the same location but facing opposite directions cannot be combined since the same features were not observed.

It can happen that small loops in the graph develop where there are two pose nodes connecting the same two star nodes. When this happens we chose to eliminate one of the pose nodes. We do this by a trick illustrated by Figure 4. We imagine that the pose node attached to the first star has only one edge and can thus be eliminated there. We then do the same at the second star node. This can be seen to be a good approximation so long as the star nodes are not under too much tension. In cases where there is a lot of tension the two disconnected nodes will snap far apart and the approximation becomes bad. Again this trick is applied at the time of star formation, as soon as the situation has arisen.



(a) (b) (c)

Fig. 4. Elimination of small loops between two star nodes.

VII. Growing Feature Dimensions

The M-space features have the ability to be partially initialized. This means that the B matrix might go from having 0 to 1 row and then after more data, *dense information*, is gathered on the feature it might be able to grow to 3 rows. For example, a line on the ceiling assumed to be horizontal can be used almost immediately for orientation, 1 dimension. After the robot moves enough perpendicular to the line its position can be determined and it will grow to 3 dimensions. The dense information in this case could be the image position of the line at different camera positions.

We can attach features to our graph at the very first observation. They will then have 0 dimensions. If the feature is later initialized these earlier measurements will begin to contributed automatically. We must go back and check the match for these at that time.

While the dense information is being collected it is important that the sensor movements between the observations be smoothly estimated. This is an advantage of the graphical method over methods such as the EKF. Using an EKF estimated pose does not work well when the filter makes large corrections to the current pose as there is no way to correct the collected dense information. For that reason the dense information must be collected in the dead-reckoning frame and needs to be discarded when the dead-reckoning is carried out over too long a distance.

With the graphical method one can continually correct the dense information as the entire path of the robot is being updated, so the dense information need not be discarded until the pose node is reduced away by forming a star node. For that reason, it is important to adaptively change the length of the tail, (see figure 7), the path back to the last star node. The tail consists of pose nodes that still have the original measurement nodes and are connected by the odometry energy nodes. It is the along the tail that the exact non-linear calculations are being done. Therefore, the updates of the graph are faster for shorter tails. However, when dense information is still being collected the tail is kept as long as needed to allow correction to the dense information. If the feature succeeds to increase its dimension then all the measurement nodes along the tail can begin to use the additional measurement dimensions to adjust the state nodes. This is in contrast to the EKF which cannot use new measurement information for old poses.

When the features attached to a pose node are not accumulating dense information, because they either are not being currently observed or have reached full dimensionality, the pose node should be reduced and the tail made shorter.

VIII. Experiments

We have tested these ideas using a robot equipped with a camera pointing straight up at the ceiling and a SICK laser scanner pointing forwards. The camera mounting was chosen to take advantage of the structures on the ceiling as vision features.

Odometry and laser data was also collected throughout the path. Lines were extracted from the laser data. Images were collected at 10Hz and had size 320x240. The camera was calibrated using standard camera calibration software. Each image was undistorted as a first step. Two types of vision features were extracted using the OpenCV library, lines and points. The lines were extracted using the Hough Transform and the points corresponds to circular lamps with high image intensity. Figure 5 shows example images from the environment with the extracted features highlighted. In some parts of the environment a relatively large number of false line measurements are produced due to the fine grid structure in the ceiling (top right image in Figure 5). The data was the same as used in [12].

Figure 6 shows an example map. The robot was driven through several rooms and then turned back on its path

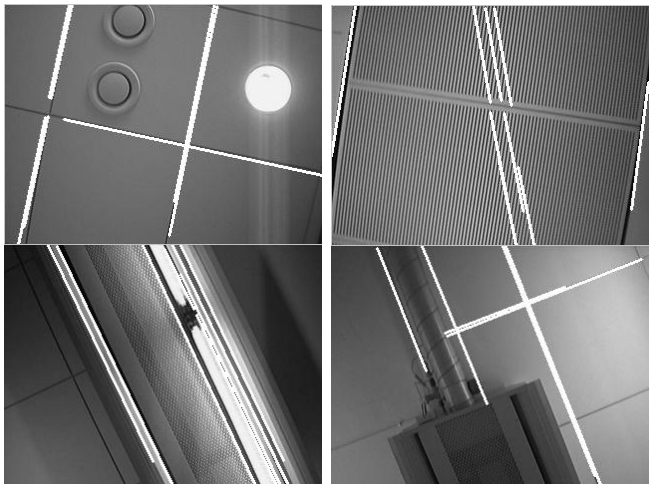


Fig. 5. Snapshots of the ceiling along the path showing the line detection output.

following it to the starting position. The path starts in the lower left room, moves out into the corridor and then into the second door of the room above the first. It then completes a small loop and continues up to the third room.

We could create an accurate map using the walls and the SICK scanner and the vision features, both taken seperately and together. The comparison with the previous EKF implementation[12] for this data is that the maps look equally good but took 2-3 times longer to do the calculation which is not surprising. The graphical method takes longer on iterations that require reconciling new measurements of features observed from previous sections of the path, the usual situation in these experiments. On the other hand extending the map into new areas goes fast. The EKF can reconcile the measurements with a closed formula and is thus quite fast in the first situation but works just as hard when in the easy situation.

As these are really rather small maps and the graphical method's calculation time for an iteration is independent of the size of the map one expects that the graphical method would be faster than EKF on larger maps. This is the next step in our research.

One can see that the graph ends up with a rather intricate topology as many stars from the return path and the small loop were merged with the original path's stars, Figure 8. This shows that information can be added to stars successfully upon return to a region previously visited.

Using the laser we were able to include 2 corner points on the walls (the green square connecting two lines just above the loop on the left side of Figure 8). These walls then shared the point with measurements from both walls effecting the location. We also were able to use partial information on all features. We succeeded in matching the features upon returning to the starting position.

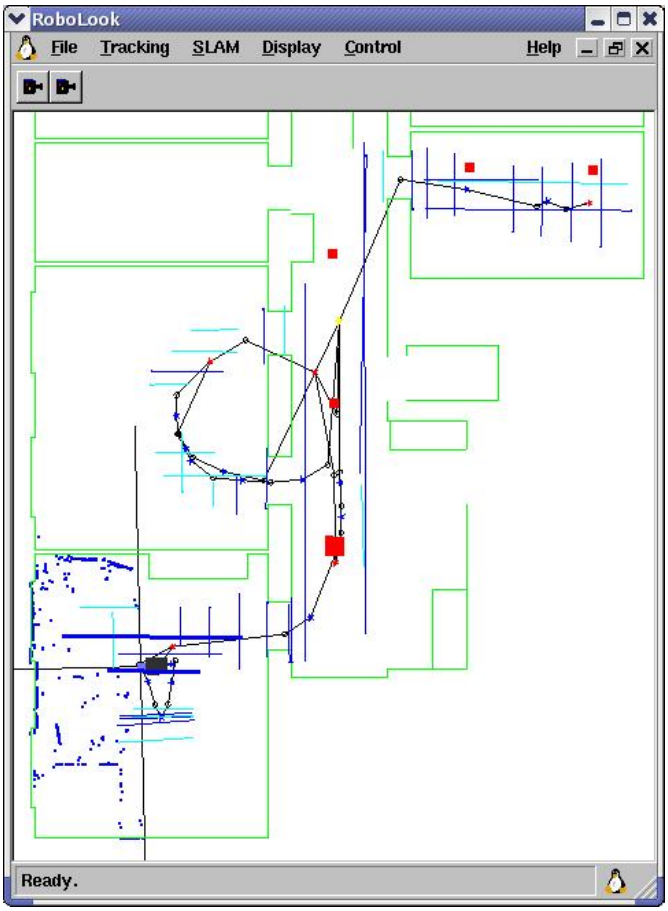


Fig. 6. This this the map obtained using only the vision features. The walls of the building are shown for reference. The final graph has 78 pose, 49 feature and 25 star nodes. 3,377 pose nodes were eliminated by graph reduction which also formed 3 loops. It took 2.5 times as long as the EKF.

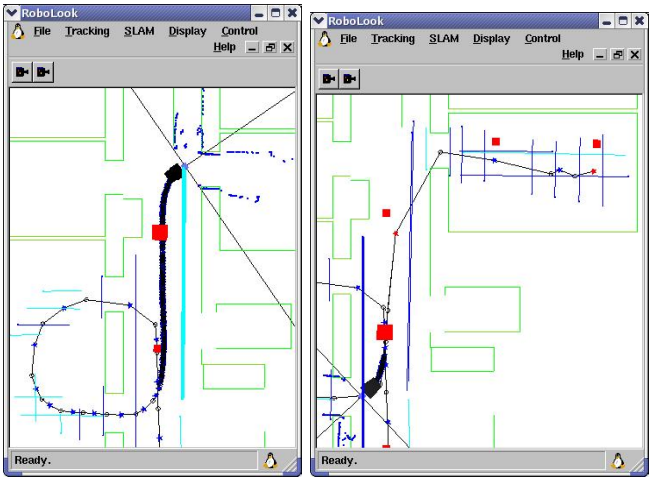


Fig. 7. Here it is shown how the tail needs to be left long while the long line on the ceiling is still accumulating dense information (moving up). As soon as the line become 3 dimensional the tail can be reduced and the entire path can use the full 3-D feature (moving down).

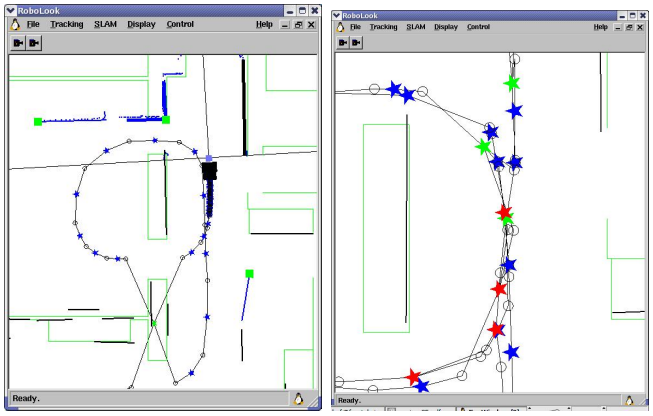


Fig. 8. Here we see that two star nodes have been combined when closing the small loop. We can see a 4-pose star. We can also see the inclusion of a corner point between two walls found using the laser. The final topology of the graph becomes rather intricate upon returning to this region two more times. One sees that stars near each other but with the robot facing the opposite direction were not combined.

IX. Conclusion

We have demonstrated Graphical SLAM with a variety of sensors and features by combining the two ideas of Graphical SLAM and M-space features into a single formalism that allows us to abstract away many of the details. We show that the M-space is indeed a versatile way to represent features. It allowed us to reuse most of the code from our EKF implementation when implementing Graphical SLAM.

We show how to properly linearize the energy in a way that preserves translation and rotation invariances, as well as all symmetries and constraints on the features. This coupled with the fact that all linearization is done in local frames helps to maintain consistency of our approximations.

We demonstrate that the graphical method is not limited to a single pass through an environment. Information can be added to an existing graph energy node. Furthermore, we were able to use any combination of vision and laser data to build maps.

Future work will include making much larger maps with more complicated topologies. We can then see how a graph can be used to solve constraint equations upon closing multiple large loops. We would also like to investigate more closely some of the approximations in the updates of the graphical method.

Appendix: M-Space

The M-Space, or measurement subspace, is an abstraction of the measured subspace of the feature space. The M-Space can deal with constrains and symmetries. If $\mathbf{x}_f$ denotes the feature coordinates, we let $\mathbf{x}_p$ denote the corresponding M-Space coordinates. To go between the feature space and the M-Space, we use a projection matrix $B(\mathbf{x}_f)$ according to eq. (1)-(3). It is important to note that we only use the M-Space coordinates as a means to get to the change in the feature coordinates. That is, the measurements tell us something about the changes in M-Space coordinates and then using eq. (2) we can calculate the corresponding change in feature coordinates. The absolute values of the M-Space coordinates are thus not known.

Parameterization

The parameterization of a map feature in the M-space framework consists of sets of coordinates of three different kinds, i) 3-dimensional $\mathbf{x}^{3D}$, i) 2-dimensional $\mathbf{x}^{2D}$ and iii) scalar $\mathbf{x}^{S}$. Any number of the three types can be combined for a certain map features type. It is also possible to share coordinates between features to represent situations such as two walls sharing a corner point. The parameterization for some different feature types are summarized in Table I.

Varying M-Space Dimensionality

A common issue when dealing with feature based SLAM is that it is often not possible to measure all dimensions of a feature at first sight. Take a line extracted from a wall with a laser scanner for example. If the wall is long, the position of both end points can typically not be determined. What can be measured is the distance and the orientation of the line. In this scenario the dimension of the M-Space for this wall line feature would be 2. As more information is gathered and the location of the end points become known they can be added and the M-Space dimensionality increase first to 3 (one end point) and then 4 (both end points).

To add some defense again spurious measurements it is also common not to add a new feature to the map immediately. In the M-Space framework we can model this as giving the feature M-Space dimensionality 0 and not initialize the first dimensions until enough evidence of its existence has been collected.

Feature	Parameterization	min(dim(M))	max(dim(M))
Point	$\mathbf{x}_f = \{\mathbf{x}^{3D}\}$	3	3
HLine	$\mathbf{x}_f = \{\mathbf{x}^{3D}_S, \mathbf{x}^{3D}_E\}$	1	3
Wall	$\mathbf{x}_f = \{\mathbf{x}^{2D}_S, \mathbf{x}^{2D}_E\}$	2	4

TABLE I

Parameterization of some different features. Also shown is the minimum and maximum dimensions of the M-Space when the corresponding feature is initialized.

Feature Types

In this work we us two types of vision features, points and horizontal lines and one type feature extracted from laser data, lines.

Point Feature: The point feature is the simplest of the features to describe. It is parameterized by a single 3D-point,

$$\mathbf{x}_f = \{\mathbf{x}^{3D}\}.$$

The point feature is initialized directly to 3 dimensions (its full dimension). The M-space coordinates are the same as the

feature coordinates which gives a B-matrix that is the identify matrix (see Table II).

Horizontal Line Feature: The horizontal line feature (`HLine`) is parameterized by two 3D points,

$$\mathbf{x}_f = \{\mathbf{x}_S^{3D}, \mathbf{x}_E^{3D}\},$$

the end points of the line. This feature illustrates the power of the M-space representation in that it can constrain the two points that have the same height. Another advantage is that the `HLine` can be initialized almost immediately with 1 dimension corresponding to the direction of the line, before its location is known. This is important when traveling along a corridor for example. Lines along the corridor are often difficult to find the position of as the robot typically moves down the corridor and thus parallel to the line. However, being able to use the direction of the line helps reduce the accumulation of uncertainty significantly. Table II shows the B-matrix for the `HLine` when it has reached the full dimension. Initially only the first row is used, corresponding to the direction of the line. When the position of the line can be triangulated the dimension goes up to 3, the full dimension. Note that the position tangential to the line is not measured in our work.

Wall Line Feature: The wall line feature is parameterized by two 2D points,

$$\mathbf{x}_f = \{\mathbf{x}_S^{2D}, \mathbf{x}_E^{2D}\},$$

the end points of the line. The 2D points can be seen as vertical lines in 3D, i.e. extending between plus/minus infinity. This comes from the fact that the line is assumed to be on a plane, the wall. When the sensor observing the line is rotated the length will appear to change. The wall is initialized as having 2 dimensions in the M-space, corresponding to the distance to and direction of the line. This results in the first two rows in the B-matrix shown in Table II. The third row in the B-matrix in Table II corresponds to the start point and the fourth to the end point. The dimensionality of a wall line can thus be 2 (no end points), 3 (one end point) and 4 (two end points).

Feature	B-matrix
Point	$B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
HLine	$B = \begin{pmatrix} \frac{\cos\gamma}{L\sqrt{2}} & \frac{\sin\gamma}{L\sqrt{2}} & 0 & \frac{-\cos\gamma}{L\sqrt{2}} & \frac{-\sin\gamma}{L\sqrt{2}} & 0 \\ \frac{\cos\gamma}{\sqrt{2}} & \frac{\sin\gamma}{\sqrt{2}} & 0 & \frac{\cos\gamma}{\sqrt{2}} & \frac{\sin\gamma}{\sqrt{2}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} \end{pmatrix}$
Wall	$B = \begin{pmatrix} \frac{\cos\gamma}{L\sqrt{2}} & \frac{\sin\gamma}{L\sqrt{2}} & \frac{-\cos\gamma}{L\sqrt{2}} & \frac{-\sin\gamma}{L\sqrt{2}} \\ \frac{\cos\gamma}{\sqrt{2}} & \frac{\sin\gamma}{\sqrt{2}} & \frac{\cos\gamma}{\sqrt{2}} & \frac{\sin\gamma}{\sqrt{2}} \\ -\sin\gamma & cos\gamma & 0 & 0 \\ 0 & 0 & -\sin\gamma & \cos\gamma \end{pmatrix}$

TABLE II

SOME B-MATRICES FOR DIFFERENT TYPES OF FEATURES WHEN THEY HAVE REACHED THEIR FULL DIMENSION. THE PARAMETER γ IS THE NORMAL TO THE LINE IN THE XY-PLANE.

REFERENCES

[1] R. Smith, M. Self, and P. Cheeseman, "A stochastic map for uncertain spatial relationships," in *4th International Symposium on Robotics Research*, 1987.

[2] J. E. Guivant and E. M. Nebot, "Optimization of the simultaneous localization and map-building algorithm for real-time implementation," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 242–257, June 2001.

[3] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam: A factored solution to the simultaneous localization and mapping problem," in *Proc. of the National Conference on Artificial Intelligence (AAAI-02)*, Edmonton, Canada, 2002.

[4] Y. Lui and S. Thrun, "Results fo outdoor-slam using sparse extended information filters," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA03)*, vol. 1, 2003, pp. 1227–1233.

[5] S. Thrun, Y. Liu, D. Koller, A. Ng, Z.Ghahramani, and H. Durrant-White, "Simultaneous localization and mapping with sparse extended information filters," *International Journal of Robotics Research*, vol. 23, no. 8, pp. 690–717, 2004.

[6] F. Lu and E. Milios, "Optimal global pose estimation for consistent sensor data registration," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA'95)*, 1995, pp. 93–100.

[7] S. Thrun, D. Fox., and W. Burgard, "A probalistic approach to concurrent mapping and localization for mobile robots." *Autonomous Robots*, vol. 5, pp. 253–271, 1998.

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

[9] U. Frese and T. Duckett, "A multigrid approach for accelerating relaxation-based slam," in *"Proc. of the IJCAI-03 Workshop on Reasoning with Uncertainty in Robotics (avail at http://www.aass.oru.se/Agora/RUR03/")*, 2003.

[10] M. A. Paskin, "Thin junction tree filters for simultaneous localization and mapping," in *Proc. of the 18th Joint Conference on Artificial Intelligence (IJCAI-03)*, G. Gottlob and T. Walsh, Eds. San Francisco, CA: Morgan Kaufmann Publishers, 2003, pp. 1157–1164.

[11] J. A. Castellanos, J. Montiel, J. Neira, and J. D. Tardós, "The spmap: a probabilistic framework for simultaneous localization and map building," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 5, pp. 948–952, Oct. 1999.

[12] J. Folkesson, P. Jensfelt, and H. I. Christensen, "Vision slam in the measurement subspace," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA05)*, 2005.

[13] J. Folkesson and H. I. Christensen, "Graphical slam - a self-correcting map," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA04)*, vol. 1, 2004.