Proceedings of the 2005 IEEE
International Conference on Robotics and Automation
Barcelona, Spain, April 2005

Vision SLAM in 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 describe an approach to feature representation for simultaneous localization and mapping, SLAM. It is a general representation for features that addresses symmetries and constraints in the feature coordinates. Furthermore, the representation allows for the features to be added to the map with partial initialization. This is an important property when using oriented vision features where angle information can be used before their full pose is known. The number of the dimensions for a feature can grow with time as more information is acquired. At the same time as the special properties of each type of feature are accounted for, the commonalities of all map features are also exploited to allow SLAM algorithms to be interchanged as well as choice of sensors and features. In other words the SLAM implementation need not be changed at all when changing sensors and features and vice versa. Experimental results both with vision and range data and combinations thereof are presented.**

***Index Terms*— Vision SLAM, Representation, Features, Symmetries, Constraints**

I. Introduction

Building maps as a robot moves through an environment at the same time as using this partially built map to maintain localized is known as as simultaneous localization and mapping, SLAM. The SLAM problem is central to autonomous mobile robotics. The representation of the maps can take on various forms, e.g. occupancy grids [1], raw sensor data [2] and features [3]. We are interested in the representation of features in terms of their position, orientation size and so on.

Some requirements of a good representation are that: i) all the information from the measurements of the features can be used to improve the state of the feature while preserving invariants, ii) can represent both size and position information, iii) can represent connections between different features such as walls that share a corner point, iv) handles any kind of sensor and any kind of feature.

The SLAM area has progressed fast over the last few years. Most of the work have been focused on finding methods that address the issue of computational complexity of SLAM. Examples are CEKF [4], FastSLAM [5] and SEIF [6].

The laser scanners have become the standard sensors in the field. They are ideally suited to making measurements in SLAM. However, they can be less useful in highly cluttered environments and they are both expensive, heavy and draw much current. For these reasons, and others, doing SLAM with a camera is attractive. A SLAM algorithm using a simple webcam would widen the range of applications that could use SLAM significantly compared to when a laser scanner has to be used.

Single monocular camera images provide bearing only information and this type of information is more difficult to use than the bearing range combination provided by a laser scanner. On the other hand using a camera one can find unique characteristics of features based on the pixels in and around the features. This might lead to more reliable feature matching than is possible with a laser scanner. There has been some SLAM work done with cameras. In [7] 'landmarks' which are not features in our sense are used. These landmarks are rather tracked pixel regions. A permanent map is not built, instead new features are created upon returning to the same region. In [8] SIFT features are used to do SLAM. Each image typically contains hundreds of features that have to be matched to the map also containing a large number of features.

We would like a to have features that correspond more closely to objects in the environment.

II. The Feature Model

As our model is similar to the SP-model [9], we will try to make the notation similar as well to bring out the parallels between the models. The main difference is the choice of parameterization of the features in terms of sets of points rather than a single transformation. This allows for more general features.

A map feature is parameterized by a set of coordinates. There are three different kinds of coordinates, 3-dimensional, $\{\mathbf{x}_f^{3D}\}$, 2-dimensional, $\{\mathbf{x}_f^{2D}\}$ and scalar coordinates, $\{\mathbf{x}_f^{S}\}$. There can be any number of coordinate vectors depending on the type of feature. Each of the three kinds of coordinates has its own rules for how it transforms under translation and rotation of the observation frame.

Let s denote the sensor frame, r the robot frame and m the map (or global) frame. A feature coordinate in the map frame then becomes $\mathbf{x}_{m,f}$. For brevity we will sometimes drop the m and also denote a feature measurement in the sensor frame with o (observation), i.e. $\mathbf{x}_o = \mathbf{x}_{s,f}$. We can write the transformation rules:

$$\mathbf{x}_o^{3D} = R_{m,s}^{3D}(\mathbf{x}_{m,f}^{3D} - \mathbf{x}_{m,s}^{3D}), \quad (1)$$

$$\mathbf{x}_o^{2D} = R_{m,s}^{2D}(\mathbf{x}_{m,f}^{2D} - \mathbf{x}_{m,s}^{2D}), \quad (2)$$

$$\mathbf{x}_o^{S} = \mathbf{x}_{m,f}^{S}, \quad (3)$$

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

where $R_{m,s}$ is the rotation matrix from the map frame to the sensor frame. Note that the 2D rotation matrix is not a normal rotation matrix as we define the 2D points as extending to plus/minus infinity in the z direction. See Appendix for more details.

We are now ready to introduce the Measurement Subspace, *M-space*, coordinates, $\mathbf{x}_p$. The M-space is an abstraction of the measured subspace of the feature space that reflects the symmetries and constraints. Let $\mathbf{x}_p$ denote the M-space coordinates corresponding to the map feature coordinates $\mathbf{x}_f$. Changes in $\mathbf{x}_p$ cause changes to the feature coordinates, $\mathbf{x}_f$. The actual values of the $\mathbf{x}_p$ are never known. What is known is the linear projection from small changes $\delta\mathbf{x}_f$ to $\delta\mathbf{x}_p$. The projection matrix is denoted by $B(\mathbf{x}_f)$ and the *dual* of it by $\widetilde{B}(\mathbf{x}_f)$. They are non-linear functions of $\mathbf{x}_f$.

$$\delta\mathbf{x}_p = B(\mathbf{x}_f)\delta\mathbf{x}_f \tag{4}$$

$$\delta\mathbf{x_f} = \widetilde{B}(\mathbf{x}_f)\delta\mathbf{x_p} \tag{5}$$

$$I_{pp} = B(\mathbf{x}_f)\widetilde{B}(\mathbf{x}_f) \tag{6}$$

The M-space can be of lower dimension than the space of $\mathbf{x}_f$ so that some parts of $\mathbf{x}_f$ are set via some other means. **Ex:** A map consisting of line segments. The distance to and direction of the lines are typically measured and not the tangential position of the line as it is difficult to measure reliably. The line segment might be represented as two 2D points, i.e. four dimensions in total. The $\mathbf{x}_p$ would have two dimensions though, corresponding to the distance and angle dimensions.

We allow the M-space to grow dimensions as more information is acquired about the features. The M-space will start out with zero dimensions and thus no coordinates. As information is gathered about the feature, we call this *dense information*, the $\mathbf{x}_p$ will gradually grow in dimensions. An example, looking once again at a map of line segments, will make this clearer. With a laser scanner as sensor, the *dense information* would be the scan points. Upon the first observation there is no prior information and thus the line segment will have a P-dimension of 0. The observation will add some dense information about the feature but will, in general, not be sufficient to initialize the wall in the M-space. Another reason for not initializing a features at the first observation is to be able to reject false measurements. The thresholds for adding dimensions can be based on the covariance of the point cloud in those directions, the number of points and the length of the observed wall. Eventually enough dense information will be collected to set the angle and distance from the origin of the wall but typically not the exact position of the end points along the wall tangent vector. This will cause the dimension of $\mathbf{x}_p$ to grow to 2. The B matrix is then given by:

$$B_{wall} = \begin{pmatrix} \frac{\cos\gamma}{L} & \frac{\sin\gamma}{L} & \frac{-\cos\gamma}{L} & \frac{-\sin\gamma}{L} \\ \cos\gamma & \sin\gamma & \cos\gamma & \sin\gamma \end{pmatrix} / \sqrt{2}, \tag{7}$$

where L is the length and γ is the angle between the wall normal and the x-axis (in the m-frame). Note how the first row of the B matrix corresponds to a rotating motion around the center of the line, whereas the second row corresponds to motion orthogonal to the line. As more dense information is collected the number of rows in B might increase to 3 or 4, adding tangent components for the endpoints.

We project the statistical properties of the measurements onto the M-space so that all probabilities are expressed in terms of changes to the $\mathbf{x}_p$. Therefore, the world based coordinates have no special significance to us. In contrast to the SP-model, where re-centering is done to keep the statistics in the $\mathbf{x}_p$ consistent with the world based coordinates.

III. Measurements

This leads us into a discussion of the measurements which we denote by $\mathbf{v}$. These measurements along with the predicted feature coordinates in the sensor frame, $\hat{\mathbf{x}}_o$, are used to calculate an innovation $\eta(\hat{\mathbf{x}}_o, \mathbf{v})$, with expected value of $\mathbf{0}$. Near the prediction and measurement we have,

$$\delta\eta = J_{\eta v}(\hat{\mathbf{x}}_o, \mathbf{v})\delta\mathbf{v} + J_{\eta o}(\hat{\mathbf{x}}_o, \mathbf{v})\delta\hat{\mathbf{x}}_o. \tag{8}$$

where $J_{\eta v}$ and $J_{\eta o}$ are Jacobians.

We now have all we need to calculate the effect of our measurement on the predicted feature coordinates in the sensor frame, $\hat{\mathbf{x}}_o$. These depend on the sensor pose, $\mathbf{x}_s$ and the feature coordinates in the map frame, $\mathbf{x}_f$ in a known way. Similarly $\mathbf{x}_s$ depends on the robot pose and robot to sensor transformation and $\delta\mathbf{x}_f$ depends on $\delta\mathbf{x}_p$ in known ways. Thus we can calculate the effect on η of changing any of these coordinates. Using a first order approximation we can write:

$$\delta\hat{\mathbf{x}}_o = \begin{pmatrix} \underbrace{J_{os}J_{sr}}_{robot\ pose} & \underbrace{J_{os}J_{ss}}_{sensor\ pose} & \underbrace{J_{of}\widetilde{B}_f}_{features} \end{pmatrix} \begin{pmatrix} \delta\mathbf{x}_{m,r} \\ \delta\mathbf{x}_{r,s} \\ \delta\mathbf{x_p} \end{pmatrix}. \tag{9}$$

Here J_{sr} and J_{ss} are the Jacobians from the transformation $T_s = T_{m,s} = T_{m,r} \oplus T_{r,s}$. That is to say T_s has 6 coordinates that describe the transformation to the sensor frame, $x_{m,s}$. These can be calculated from the coordinates of $T_{m,r}$ and $T_{r,s}$. So,

$$J_{sr} = \frac{\partial\mathbf{x}_{m,s}}{\partial\mathbf{x}_{m,r}}, \quad J_{ss} = \frac{\partial\mathbf{x}_{m,s}}{\partial\mathbf{x}_{r,s}}.$$

These calculations depend only on the transformations to the sensor frame and are independent of the feature. In general they are 6x6 matrices. One will normally be interested in changes to a smaller number of the coordinates. For instance, the sensor may be fixed relative to the robot or only have pan and tilt movements. Only those components need be calculated here. For example, assuming a fixed transformation from robot to sensor $\delta\mathbf{x}_{r,s} = 0$ and (9) would simplify.

J_{os} and J_{of} are the Jacobians from the transformation $T_o = T_{s,f} = \ominus T_{m,s} \oplus T_{m,f}$. This is a generic calculation depending only on the 3 sets of coordinates, (i.e. 3D, 2D or 0D) and the sensor transformation. It has the same form for all types of features and measurements.

To summarize, it is only the definition of η, $J_{\eta v}$, $J_{\eta o}$ and B_f that depend on the type of feature. The rest of the SLAM problem looks the same. This can be exploited to write SLAM code that is separated from the details of type of feature.

IV. The Four Phases of SLAM

We now outline the four phases of a generic SLAM algorithm using the M-space. The first two steps might look slightly different depending on the specific SLAM algorithm being used.

Prediction: Predict the sensor pose $\mathbf{x}_{m,s}$ and match all measurements to the map based on some criteria such as Mahalanobis distance. For unmatched measurements, create new features with M-space dimension 0.

Update: Calculate the innovation and the Jacobians needed to perform an update as described above. These can be used by the SLAM algorithm to calculate a change in $\mathbf{x}_p$, as well as to the estimated robot pose and if needed the covariance[1]. Apply the change in $\mathbf{x}_p$ to the feature by (5). This causes the B matrix to change but that has no effect on the $\mathbf{x}_p$ covariance.

Add Information: For each feature add the new dense information. This information could be in the form of an occupancy grid, a cloud of laser scan points or a list of bearing only information. The dense information will be used to adjust the feature's coordinates orthogonal to the M-space. Thus a wall, for example, might have its endpoints slide tangential to the wall. These changes do not affect the M-space, but we see from (7) that the B matrix will change.

Extend: Try to extend the M-space if enough new dense information was collected during the add information phase.

Notice that part of $\mathbf{x}_f$ can be shared between two different features. Thus, a wall described by two 2D endpoints could share a corner point with another wall. This then will force the corner to always be consistent with measurements of both walls. The B matrices for this point on both features would then be the identity matrix.

The issue of feature initialization is important. Initialization can be handled, for example, by forming a pseudo-measurement in the EKF case. Other SLAM methods may not require any initialization. Alternatively, one could use a delay in the filter between the add information and the predict-update-extend phases. This will allow more dense information to be collected from later measurements which increases the chance of being able to extend the M-space before the update.

The movements of the feature coordinates orthogonal to the M-space need to be tracked. The endpoints of walls and lines are extended to points consistent with the longest observations of the features. For vision features we move the feature to a position consistent with the last observation of it and the M-space coordinates. As long as the bearing to the feature does not change too much we can find it in its predicted position in the next image. This is the only visual tracking that was used in our experiments. All this is part of the routine for adding information on a feature. When enough dense information is collected in the form, for instance, of a point cloud, the covariance of the cloud is use to estimate the uncertainty of the new M-space coordinate dimensions.

To avoid accumulating too much dense information on a certain feature a mechanism for removing old information is needed. Older information needs to be removed since it cannot be combined reliably with newer information. In this paper we attach a measure of the 'distance' along the path to each piece of dense information. Our 'distance' metric reflects the fact that errors in rotation are both more serious and larger than the errors in robot translation. In practice, when the feature is a good one, the information is used rather quickly to extend the M-space and so this mechanism is not a sensitive part of the method.

For the example of EKF SLAM the predict step is as usual except that the change in $\mathbf{x}_f$ uses (5). The update step is also more or less as usual:

$$\delta\mathbf{x} = W\delta\eta, \tag{10}$$

$$W = CJ^T(JCJ^T + R)^{-1}, \tag{11}$$

$$\delta C = -WJC, \tag{12}$$

$$R = J_{\eta v}C_{vv}J_{\eta v}^T, \tag{13}$$

where C is the covariance, $\delta\mathbf{x}$ includes the pose, parameters[2], and $\mathbf{x}_f$ of each feature. J is (aside from its null columns):

$$J = J_{\eta o}\begin{pmatrix} J_{os}\begin{pmatrix} J_{sr} & J_{ss}\end{pmatrix} & J_{of}\widetilde{B}_f\end{pmatrix}. \tag{14}$$

We emphasis that B_f is recalculated every time that $\mathbf{x}_f$ changes. The true mapping is non-linear so the linearization must be done around the current state.

Expressing other probabilistic SLAM algorithms in M-space is also straight forward. One might not have explicit predict and update steps but one can formulate a similar alternating sequence of adding information/extending with estimation/matching.

V. An Example: Horizontal Line Features with a Camera

As an example of the utility of this approach we will show how camera images of lines that are known to be horizontal, (such as lines on the ceiling), can be used to form map features with a P-dimension of 0, 1 or 3. The feature is parameterized by two 3D points, its endpoints. This example illustrates the strength of the M-space; it can handle constraints, the lines are horizontal, and symmetries, the endpoints can slide tangent to the lines with no change to $\mathbf{x}_p$ and can use partial information, tangent direction initialized but position not sufficiently well known. With one observation of a horizontal line

[1]Some SLAM algorithms, such as Graphical SLAM, do not calculate the covariance, [10].

[2]Parameters that can be estimated includes for example, sensor poses, wheel parameters and camera parameters

its position is constrained to be in the plane defined by the camera and the vectors from the camera to the line end points. Given the that the line is horizontal we get an estimate of its tangential direction. The position of the line, i.e. position orthogonal to the tangent direction in the xy-plane and the height of it line is not known. As long as the camera moves in the same plane the position of the line remains unknown. By moving sufficiently towards or away from the line its position can be determined by triangulation.

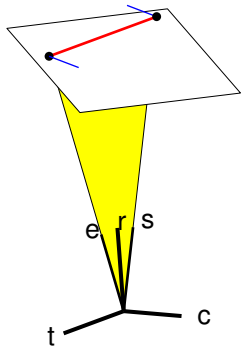

Fig. 1. Image measurement of the ceiling where a horizontal line segment has been detected. The vectors $\mathbf{s}$ and $\mathbf{e}$ point out the end points of the line. The entities $\mathbf{c}$, $\mathbf{r}$ and $\mathbf{t}$ are derived from $\mathbf{s}$ and $\mathbf{e}$ and are used to define the innovations. The two short lines attached to the end of the horizontal line give the directions corresponding to the B-matrix for a line with 1D M-space.

The situation where we know the tangent but not the position is particularly interesting. This leads to a B matrix that looks like:

$$B = \frac{\begin{pmatrix} \cos\gamma & \sin\gamma & 0 & -\cos\gamma & -\sin\gamma & 0 \end{pmatrix}}{L\sqrt{2}}. \quad (15)$$

Here γ is the angle between the projection of the normal to line into the xy plane and the x-axis. This B matrix, and thus the M-space, corresponds to a rotation of the line in the xy plane around its center. The measurements consist of two pixels in the image corresponding to the line end points and the focal length of the camera. These measurements give rise to two vectors in the camera frame which point to the end points of the detected line segments. Call these vectors $\mathbf{s}$ and $\mathbf{e}$ (see Fig. 1).

$$\mathbf{s} = (-v_0, v_4, -v_1), \text{ and } \mathbf{e} = (-v_2, v_4, -v_3). \quad (16)$$

Here the camera axis is along the y direction and (v_0, v_1) and (v_2, v_3) are the start and end point in image plane and v_4 is the focal length. Using the following entities (see Fig. 1),

$$\mathbf{c} = \frac{\mathbf{s}\times\mathbf{e}}{|\mathbf{s}\times\mathbf{e}|}, \ \mathbf{r} = \frac{\mathbf{e}+\mathbf{s}}{|\mathbf{e}+\mathbf{s}|}, \text{ and } \mathbf{t} = \frac{\mathbf{e}-\mathbf{s}}{|\mathbf{e}-\mathbf{s}|}, \quad (17)$$

we can define an innovation as,

$$\eta = \mathbf{c}\cdot\hat{\mathbf{t}}. \quad (18)$$

Here $\hat{\mathbf{t}}$ denotes the prediction of $\mathbf{t}$ calculated using predictions $\hat{\mathbf{s}}$ and $\hat{\mathbf{e}}$ of $\mathbf{s}$ and $\mathbf{e}$.

We can then calculate $J_{\eta o}(\hat{\mathbf{x}}_o, \mathbf{v})$ and $J_{\eta v}(\hat{\mathbf{x}}_o, \mathbf{v})$ by differentiating this innovation.

When the robot has moved enough sideways with respect to the line, the position of the line can be initialized from the resulting dense information using triangulation. This leads to the B matrix,

$$B = \begin{pmatrix} \frac{\cos\gamma}{L} & \frac{\sin\gamma}{L} & 0 & \frac{-\cos\gamma}{L} & \frac{-\sin\gamma}{L} & 0 \\ \cos\gamma & \sin\gamma & 0 & \cos\gamma & \sin\gamma & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} /\sqrt{2}. \quad (19)$$

where the second row corresponds to motion in the xy orthogonal to the line and the third row corresponds to changing the height of the line[3].

Now the innovation can be two dimensional and for the second component we take:

$$\eta_2 = \mathbf{c}\cdot\hat{\mathbf{r}}, \quad (20)$$

To summarize, given the assumption of horizontally we are able to quickly use the tangent direction to correct the robot orientation. As soon as the robot gets motion normal to the line we can extend the M-space to 3D and fix the line in space.

VI. Experimental Results

We mounted a simple low cost web camera pointing straight up at the ceiling on robot already equipped with a SICK laser scanner.

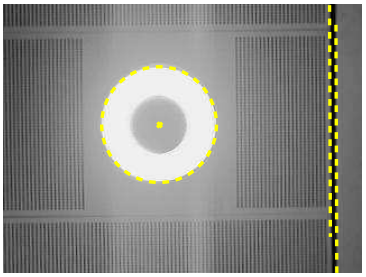

Fig. 2. Example image from the webcam with two line features and a lamp feature detected.

The robot was driven through several rooms ending up at our starting position. We collected odometry, laser and image data. The images had size 320x240 and were collected at 10Hz. The images were processed using the OpenCV library. As a first step the radial and tangential distortion were compensated for. The camera was calibrated using standard camera calibration software and was found to have a focal length of about 508 pixels. Two types of features were extracted, lines and points. Lines were extracted using the Hough Transform. For point features we used the center of lamps of circular shape that were detected based on their intensity. Figure 2 shows an example image from

[3]Note how the third row constrains the vertical motion of the line such that its stays horizontal

the webcam with two lines on the right side and a lamp feature extracted. The height of the linear structures above the camera varied between 1.5 and 2.5 meters. Figure 3 shows image snapshots from the environment. Especially in the corridor where the ceiling has a fine grid there are a number of false line readings that have to be rejected by the matching method.

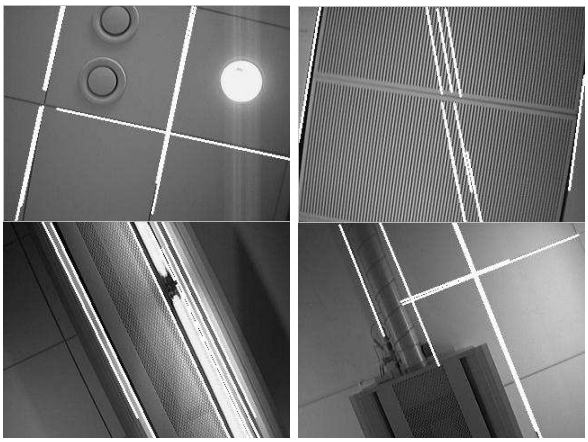

Fig. 3. Snapshots of the ceiling along the path showing the line detection output.

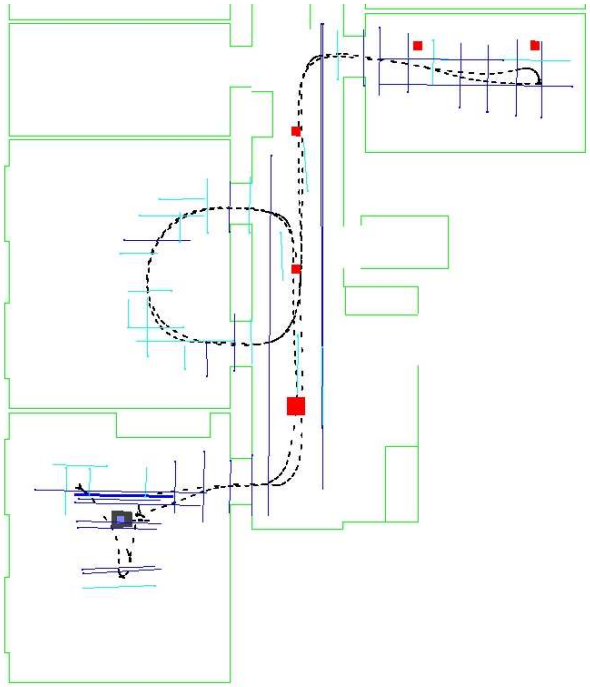

Fig. 4. The map obtained using only the camera images consists of lines and lamps on the ceiling. The true map of the walls in the lab is also shown for reference. One can see that the robot passes through the center of the doorways and that the rectangular nature of the ceiling lines is maintained indicating good accuracy. The lighter lines have M-space dimension of 1 (direction only) while the darker lines have 3. The squares are the ceiling lamps which were also used for doing SLAM.

The dense information was gathered by the map features as lists of bearings for the image features. For each item on the list there is a pair of bearing vectors in the map frame, the xyz position of the camera along with a 'distance' along the path of the robot.

Since we assume the ceiling lines to be horizontal we are able to initialize the tangent direction after only a few observations. Initialization of the position of the line requires that the robot move toward or away from the line in order to get a triangulated fix on the line. We found that for lines parallel to the corridor, this type of movement often occurred only after traveling a considerable distance down the corridor. It was important therefore that we could still use the lines to adjust the robot's heading for which the M-space provides the means.

Besides the visual features, lines were also extracted from the SICK data using the Range Weighted Hough Transform.

We demonstrated the principles of our feature representation using a standard Extended Kalman Filter, EKF. The results for using ceiling lines only were very good. We were easily able to find the same lines upon returning to the rooms a second time and the SLAM algorithm managed to correct the angular error in the odometry. We ended up with less than 0.5 degree error upon returning and less than 0.5 cm in y and x. The trajectory followed by the robot can be seen in Figure 4. The robot started/ended in the room in the lower left corner.

We also detected the ceiling lamps as 3-D point features. These features could then be combined with the lines giving valuable information about movements down the corridor where all the lines were in the same direction. The lamp features were widely spaced and reliably detected, making matching easy. The lamp features could be given to the SLAM program as M-space features treated exactly the same as the lines were. The results are shown in Figure 4 and Figure 5

Using only the 5 lamps we were able to reduce the odometry error upon return by one order of magnitude. When combined with the lines they gave some improvement in the distances along the corridor.

Using the SICK scanner was complicated by the glass doors of our lab and a large curved wall in a critical position. We were nevertheless able to get results very consistent with the camera. The final position was in this case also correct to less than 0.5 degree and less than 0.5 cm in all directions.

Several of the walls were able to give full endpoint information, P-dimension grew to 3 and in some cases 4. This was when the laser scan could clearly see the end of the wall. In addition, in two instances we were able to constrain the endpoints from different walls to be the same point, a corner. The criteria for merging the endpoints was simply close proximity of the estimated points.

We were able to build consistent and correct maps using all combinations of these three features with no need for retuning the parameters between runs.

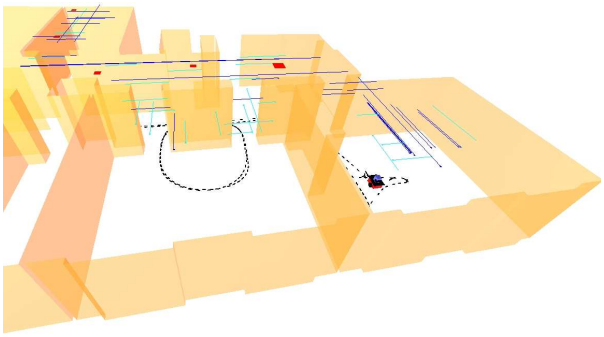

Fig. 5. The same map from another viewing angle. One can see that the features are also at the correct height. The ceiling height is higher in the rooms than in the hallway.

VII. Discussion

The results of the experiments confirm the validity of our approach. We were able to have features that only gave valuable heading corrections to the robot while others were able to also give distance along one axis. Some wall features were able to reach their full M-space dimension of 4 and corner constraints became implicit.

The EKF program and most parts of the matching were identical for the three kinds of features. Only a small amount of code was written to calculate the innovations for each feature type and to collect and analyze the dense information. Of course, one still has to extract the feature measurements with special code for each feature but the map representation and SLAM is standardized. One can thus use the same SLAM code in various environments with different sensors and features.

For our lab we found the horizontal line feature to be very practical. These horizontal lines are rather novel features due to the existence of both symmetries and constraints. This then required a more sophisticated feature representation than simpler features. Having features that could use partial information and then expand in dimension when more data was available also proved to be very valuable.

We were also be able to combine laser and camera data to make a fused map. Using more than one type of sensor for feature detection will give higher reliability to SLAM or simple localization. With this M-space approach adding features and sensors is straight forward. It is able to handle a much wider range of features than other representations. All invariances are maintained automatically by the B matrices.

In future work we plan to show that this representation can be the basis for a valid comparison of different SLAM approaches. So that the same data, same matching and mostly the same code will be used to compare the methods. We also would like to use more information from the camera images, recording visual characteristics of the features would help in matching the features to the measurements.

VIII. Conclusions

We were able to demonstrate our general feature representation in the context of a robot navigation problem in an office setting using a camera as our main sensor. This particular situation required a representation for the features that could grow as information was added to the features. As well as having constraints and symmetries maintained during the update of features.

The M-space feature representation has the advantage of being able to represent any kind of feature in a standardized way. Thus most of the calculations of a SLAM algorithm will be identical. This creates a separation of the SLAM implementation from the details of the feature representation, matching and extraction. It breaks the SLAM problem into four independent parts one of which is solved by the M-space representation itself.

Appendix

We define the 2D points as having an x and y but extending to plus/minus infinity in the z direction. This then implies that the rotation to a general frame will produce a line in the new frame. We use the intersection of that line with the transformed xy plane as the transformed coordinates of the 2D point. This may sound strange, but it is what is needed for the important case of a wall or vertical pole being observed by a 2D laser scanner. If the sensor is rotated it will still see a line on the wall but the end points might be at different heights. $R^{2D}_{m,s}$ can then be written terms of the Euler angles θ, ϕ and ψ as,

$$R^{2D}_s = \begin{pmatrix} \frac{\cos\theta + \sin\theta \sin\phi \tan\psi}{\cos\phi} & \frac{\sin\theta - \cos\theta \sin\phi \tan\psi}{\cos\phi} \\ \frac{-\sin\theta}{\cos\psi} & \frac{\cos\theta}{\cos\psi} \end{pmatrix}$$

References

[1] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[2] 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.

[3] J. J. Leonard and H. F. Durrant-Whyte, *Directed Sonar Sensing for Mobile Robot Navigation*. Boston: Kluwer Academic Publisher, 1992.

[4] 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.

[5] 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.

[6] 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.

[7] D. Burschka and G. D. Hager, "V-gps(slam): Vision-based inertial system for mobile robots," pp. 409–415, 2004.

[8] S. Se, D. G. Lowe, and J. Little, "Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks," *International Journal of Robotics Research*, vol. 21, no. 8, pp. 735–58, 2002.

[9] 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.

[10] 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.