Applying Multiframe Reconstruction to Pose Estimation

J. Inigo Thomas
Allen Hanson

CMPSCI TR93-11

February 1993

Appling Multiframe Reconstructions to Pose Estimation

J. Inigo Thomas
Allen Hanson *
Computer Science Department
University of Massachusetts at Amherst
Amherst, MA 01003
email: jthomas@cs.umass.edu

1 Introduction

Traditionally, a moving robot monitors its position in an environment by using a 3D model of the scene and a 2D view (image) of the scene from the current position of the robot. Though several techniques have been developed that determine the robot's position (or *exterior orientation* [11], *camera location determination* [15] [16]) these techniques require either fairly accurate 3D scene models or noisy 3D models with a reliable estimate of the model noise. Since it has not been possible either to automatically obtain accurate 3D models or to reliably estimate the error in noisy models, the 3D

*This work was supported by DARPA (via TACOM), contract number DAAE07-91-C-R035.

models have been constructed manually in previous work. In this paper we propose automating the model generation step by using a robust reconstruction algorithm developed by Thomas and Oliensis [27] [19] [26]. Apart from automatically obtaining fairly accurate 3D scene models, their technique also provides the error in the model; this paper shows that the error estimate is reliable enough for a position estimation algorithm, such as that of Kumar [13]. The main contribution of this paper is a coupling of an automatic 3D model acquisition algorithm with a position estimation algorithm, resulting in an Automatic Position Estimation algorithm. The second important contribution is an analysis of the current 3D model acquisition algorithms in order to select the best algorithm to be coupled to the position estimation algorithm.

The Automatic Position Estimation algorithm developed in this paper consists of a bootstrapping stage for model acquisition followed by a stage for the estimation of the robot position. Sections 2 and 3 discuss the two stages of the algorithm and Section 4 shows empirically how well this algorithm works.

2 Bootstrapping Stage

In the bootstrapping stage the robot is moved over a short distance, viewing the scene from several positions in order to reconstruct the scene. Traditionally, just two images have been used to reconstruct 3D scenes (*two-frame*

structure from motion). 3D points are reconstructed by triangulation, using the 2D image measurements from the two images. However, due to inherent problems with two–frame structure from motion the 3D reconstructions are erroneous [5]. As pointed out Thomas and Oliensis [25] the main problem is that the computation of the scene structure from two image measurements involves non–linear optimization, which is not robust when the image measurements are noisy. A natural approach for refining two–frame reconstructions is to use several frames. However, using multiple image frames gives rise to a different set of problems as analysed in the following section.

2.1 Multi–frame Model Acquisition Algorithms

Multi–frame model acquisition algorithms can be divided into two categories – *batch* methods and *recursive* methods. Batch methods attempt to reconstruct the 3D scene assuming that *all* images from every robot position are available before processing is begun. Unlike batch methods, recursive methods assumes that all past images are *not* available; rather, they assume that only the current (best) reconstruction of the scene is available along with an estimate of the error in this reconstruction.

2.1.1 Problems with Batch Methods

If the scene is reconstructed based on $m + 1$ pictures (from m robot moves) and represented using n features (say, n 3D points), the the most general

batch method involves $6m - 1 + 3n$ variables [1]. The number of points, n, determines the granularity of the scene reconstruction. If this granularity is fixed (i.e., n is fixed), then the batch methods have to deal with $6m - 1$ variables in the most general case. This is a very large number of variables (59 for 10 movements) for any non-linear optimization technique.

Almost all batch algorithms have reduced the number of variables representing the robot motion by restricting the kinds of allowable motion, hoping that any deviations from these constrained motions in realistic situations can somehow be dealt with as noise in the system. Problems arise because the restrictions on allowable motions are arbitrary, and/or because the nature of deviation in the actual robot motion is systematic and cannot be accounted for simply as noise. In the following we analyse the batch methods reported in the literature based on the number of parameters used to capture the underlying motion. Table 1 (at the end) provides the number of parameters for each algorithm.

In the simplest case, Sawhney, Oliensis and Hanson [21] restrict the robot camera motion to rotation about a fixed axis, which involves only 3 variables to represent the camera motion. In the two reported experiments the camera's motion is made to adhere to the motion restriction; the average accuracy of the reconstruction is 2.4% (using 25 frames) in one experiment and is 0.9% (using 20 frames) in the other. It is clear from the nature of the formulation

[1] Each robot motion involves 6 variables (3 for translation and 3 for rotation) and each 3D point involves 3 variables (x,y,z). The scale ambiguity [30] decreases the total number of unknown variables by 1.

that this technique cannot be generalized to arbitrary robot motion.

In a more general type of allowable motion, the objects in the scene are assumed to turn and move at a constant rate. [2] This reduces the number of motion variables from $6m - 1$ to 7. Broida & Chellappa [3] get a reconstruction of the scene within 2.5% accuracy in a 12 image sequence when the object in the scene moves and turns at a constant rate. However, in a second sequence, in which the object does not turn and move at a constant rate, the error is as high as 40.3% over 16 images. Kumar, Tirumalai and Jain [14] also restrain their object motion to constant rate of rotation and translation. They do not report quantitative results for real image sequences. Franzen [7] allows a slightly more general motion of objects in the scene, called chronogenous motion, which allows constant acceleration of the object. His approach requires only 14 variables to represent the motion. Reasonably good reconstructions within an accuracy of about 7.4% is reported for 8 images of the UMASS Rocket–Field Sequence (part of a standardized database).

Instead of constraining the object motion, Taylor, Kriegman and Anandan [24] require that the robot camera move on a plane. With this constraint on the camera motion they cut down the number of motion variables from $6m - 1$ motion variables to $3m - 1$. No real image experiments are reported yet.

[2] The situation in which the objects move rigidly and the robot is stationary is equivalent to the situation in which the robot moves and the objects are stationary.

Of all the batch methods, the work by Tomasi [29] involves the most unrestricted method, without any constraints on the robot camera motion. However, the problem with this approach is that it assumes that the 2D picture is an orthographic projection, i.e. the light rays striking the camera image are parallel. This assumption restricts the usability of the method to situations involving very far away objects, such as objects viewed from an airplane. The reconstruction is as accurate as 2.4% in a 150 image sequence that is reported [29].

To conclude, due to the large number of motion variables involved, which result in high computational time and unrobust optimal solutions, batch methods are useless for realistic, robotic applications where a model of the 3D scene has to be acquired in unmodeled, arbitrary environments.

2.1.2 Problems with Recursive Methods

Unlike batch MFSFM methods, recursive methods assume that all past images are *not* available; rather, they assume that only a reconstruction of the scene is available along with an estimate of the error of this reconstruction. Recursive methods try to refine the reconstruction by incorporating information from the new image (or new pair of images) obtained from a new robot position. However, this refinement is only possible if the *estimate* of the reconstruction error reflects the *actual* underlying error in the reconstruction.

One representation of the reconstruction error is a complete covariance matrix. That is, if the scene is reconstructed by n 3D points, then the re-

construction error is represented by a covariance matrix of size $9n^2$. This covariance matrix is difficult to compute, expensive to store, and computationally complex to manipulate. However, it is argued by Thomas [28] that every entry of the covariance matrix is meaningful; neglecting any entry amounts to a wrong approximation of the actual reconstruction error (for general robot motion). A simplistic explanation is as follows. The source of error in all structure from motion algorithms is the error in the estimated robot motion. The motion error affects all the 3D coordinates of the reconstruction in a systematic way i.e., the errors in all the 3D coordinates are correlated. Since every element of the covariance matrix represents the correlation of the error between pairs of 3D points, neglecting any non–zero element in the covariance matrix has dire consequences.

In previously reported work, the $9n^2$ elements of the covariance matrix are approximated by much fewer than $9n^2$ elements. Table 2 provides the number of elements for the various algorithms. The simplest class of approximations involve using only n elements to represent the reconstruction error. Each of the n elements approximates the expected error of distance of one 3D point from the robot camera. Matthies et. al. [18], [17] use such an approximation of the error, but constrain the camera to move exactly parallel to a fixed line, and obtain a reconstruction of 0.5% error using 11 images. Heel [9] [10] also approximates the error with n elements, and reports reasonably good *qualitative* results for cases when the camera moves exactly in a straight

line. In these two cases the approximation of the covariance matrix by n elements seems to be accurate enough to obtain reasonable reconstructions, but this performance is most likely due the highly restrained motions. This observation agrees with the case reported by Shigang, Tsuji, and Imai [22], who also use only n terms to approximate their error, but consider more general motions than Matthies et. al. and Heel. When they allow the camera to move freely in a plane, their reconstruction error is 15% even with as many as 40 images. Ando [1] also uses n elements but only simulation experiments are reported.

The next category of approximations involve using $9n$ elements to approximate the $9n^2$ covariance matrix. Stephens et al. [23] report reconstructions within 1% error *for 1 point* after 50 frames in the case of motion straight ahead. Cui, Weng and Cohen [4] also use $9n$ elements to approximate the full covariance matrix but allow for erroneous robot motion, unlike Stephens et al. The reported accuracy of the reconstruction (from a real image sequence) fluctuates randomly.

The algorithm developed by Thomas and Oliensis [26] [19] [27] uses the full covariance matrix. Highly accurate reconstructions (as accurate as the ground truth in 7 frames) have already been reported by Thomas and Oliensis [27] for image sequences without assuming any constraints on the robot camera motion. In section 4 of the present paper, two experiments are reported with errors of 1.8% and 2.1%.

In conclusion, of the two reported multi–frame structure from motion algorithms for general robot motion [27] [4], the algorithm developed by Thomas and Oliensis [27] produces the most accurate 3D scene reconstructions. Using this algorithm the output of the bootstrapping stage is a 3D scene model and an estimate of the error in the model. The algorithm attributes the error in the 3D model to the incorrect estimation of the camera motion and captures this error as cross-correlation terms of a covariance matrix.

3 Robot Position Estimation Phase

Once a stable 3D model has been obtained from the bootstrapping stage, this 3D model is given as input to the position estimation stage. The algorithm uses this 3D model together with the 2D image from the current robot position in order to determine the robot's position. The algorithm used in this work is a robust position estimation algorithm developed by Kumar, [13] which can handle noise in the image as well as in the 3D model. However, this algorithm can effectively account for noise in the 3D model only if a reliable estimate of the error in the 3D model is known (as provided by the bootstrapping stage).

The position estimation algorithm determines the robot's rotation and translation that has moved it from the origin, $(0, 0, 0)$, of the 3D model's coordinate system. In order to determine the robot motion, the 3D model

and the 2D image measurements are related through an unknown coordinate transform (of rotation and translation) and a known perspective, projection transform (of the camera). The problem of determining the robot motion is then cast as an optimization problem in terms of the the 3D model coordinates and the 2D image coordinates and solutions involving robust optimization techniques. [3] One such robust optimization technique is Maximum–likelihood estimation [12] as applied to the position determination problem by Haralick and Joo [8]. Another robust optimization method has been based on minimizing Least Median Squares as used by Rosseuw and Leroy [20], Fischler and Bolles [6] and Kumar [13]. However, the only reported work that allows for error in the 3D model is that of Kumar [13]; this makes it the only applicable algorithm for the position estimation phase for the Automatic Position Estimation algorithm presented here.

4 Experiments

In the following, two experiments are reported to provide empirical evidence for the plausibility of the Automatic Position Estimation algorithm. The first experiment is a simulation and the second experiment involves real imagery obtained from a moving robot. Building (or refining) the 3D model in the bootstrapping stage is of time complexity $O(n^3$ if the model is made up of n 3D points. This translates to approximately 4 minutes (for each refinement)

[3] For details refer to a detailed review by Kumar [13].

on a TI explorer for a model with 30 points.

4.1 Experiment 1: Simulated Robot Motion

The first experiment is a synthetic experiment that involves determining the position of the robot outside the Computer Science building at UMASS. In the bootstrapping stage of this experiment the robot is moved along the perimeter of a circle, 4 ft in diameter, with the robot's camera aimed approximately at the building over 130 ft away (Figure 1 shows the building). The camera parameters were: 70° FOV, 512 × 512 image size, focal length 15.24 mm and an image noise σ of 0.45 pixels (corresponding to a flow error σ of 0.64 pixels). Figure 4 shows the error in the 3D model acquired by the bootstrapping stage over time; the 3D model improves and finally after 25 moves of the robot, the model is accurate on an average to within 1.85% (i.e. 3.95 ft for 30 points between 132 and 275 ft from the robot). Figure 2 shows the 3D model acquired by the bootstrapping stage after 25 movements. The 3D model consists of the 3D coordinates of 30 points which are the corners of the faces[4] shown in Figure 2. The 3D model and the error in this model were used by the pose determination algorithm of Kumar [13]. The robot was now moved 30 ft towards the building in steps of 3 ft along with rotations of $[-1.0°, 1.0°]$ around the vertical axis. The robot's position was estimated to within an accuracy of 10.0 inches through all the robot's movement. Figure

[4]The points have been manually connected into faces in order to display the accuracy of the reconstruction.

3 shows the actual robot path and the recovered robot path. After 30 feet of motion the robot's final position was estimated with an error of 5.4 inches.

4.2 Experiment 2: Real Robot Motion

The second experiment involves a real image sequence of 10 images obtained from a robot moving in the lobby of the Computer Science Department at UMASS. The robot moved nine times straight ahead, about 1.4 ft/move (with small, yet problematic, rotations). The camera parameters were: $(29.3°, 22.9°)$ FOV, 256×242 image size and focal length 16 mm. In this experiment the 3D model was obtained by using the sequence in reverse (frame 10 to 1) in the bootstrapping stage. In the second phase the 3D model and the images from the same sequence (frames 1 to 10) were used to determine the position of the robot at each step[5]. Figure 5 shows how the 3D model improves, with a final accuracy of 2.1% (i.e. 0.75 ft for 31 points between 25 ft and 42 ft). Figure 6 shows the true reconstruction (obtained by hand) and the reconstruction that was automatically acquired. The reconstruction consists of the 3D coordinates of 31 points. Finally, Figure 7 shows the actual and recovered robot positions. In the 12.8 ft path of the robot, the error in each recovered robot position lies between 3.2 inches and 4.7 inches.

[5]Due to practical considerations, the same sequence was used for both stages of the algorithm. However, whether the same or a different sequence is used does not affect the performance of the algorithm, as shown by the results of the previous experiment.

5 Conclusion

The technique described here provides a feasible way of automating robot position estimation. The algorithm involves two stages : a bootstrapping stage which provides a 3D model and a position estimation stage which determines the motion of the robot. Of the existing algorithms for acquiring 3D models the algorithm of Thomas and Oliensis [27] is the most accurate one for general robot motion. Of the existing position estimation algorithms the only one that can use automatically acquired (i.e. noisy) models is that of Kumar [13]. The empirical results presented here show that coupling these two algorithms results in a promising Automatic Position Estimation algorithm.

In all previous work accurate 3D models had to be built by hand before the robot position estimation algorithm could be applied, severely restricting the physical environment of the robot. The present work shows that this restriction can be removed to allow for monitoring robot position even in unmodeled environments – a step that is essential for autonomous navigation.

Acknowledgements: We thank Teddy Kumar and Harpreet Sawhney for the lobby image sequence and the ground truth. We also thank Teddy Kumar for his pose determination software.

References

[1] H.Ando, "Dynamic Reconstruction of 3D Surfaces and 3D Motion," *Proc. IEEE workshop on visual motion*, Princeton, NJ, pp. 101-110, 1991.

[2] T.J.Broida and R.Chellappa, "Performance bounds for estimating three-dimensional motion parameters from a sequence of noisy images," *Journal of the Optical Society of America A*, 1989, pp. 879-889.

[3] T. J. Broida and R. Chellappa, "Estimating the Kinematics and Structure of a Rigid Object from a Sequence of Monocular Images", *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 13, no. 6, pp. 497–513, 1991.

[4] N. Cui, J. Weng and P. Cohen, "Extended Structure and Motion Analysis from Monocular Image Sequences," *Proceedings 3rd IEEE International Conference on Computer Vision,* Osaka, Japan, 1990, pp. 222-229.

[5] R. Dutta and M. Snyder, "Robustness of Correspondence-Based Structure from Motion," *Proceedings 3rd IEEE International Conference on Computer Vision,* Osaka, Japan, Dec. 1990.

[6] M.A.Fischler and R.C.Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Communications of the the ACM,* Vol. 24, pp. 381–395, 1981.

[7] W.O. Franzen, "Structure and Motion from Uniform 3-D Acceleration," *Proc. IEEE workshop on visual motion,* Princeton, NJ, pp. 14-20, 1991.

[8] R.M.Haralick, H.Joo, "2D-3D Pose Estimation," *CVPR,* Rome, Italy, pp. 385–391, 1988.

[9] J. Heel, 'Dynamic Motion Vision," *Image Understanding Workshop,* Palo Alto, CA, pp. 702-713, 1989.

[10] J.Heel, "Temporal Surface Reconstruction," *CVPR,* Hawaii, June, 1991, pp. 607-612.

[11] B.K.P.Horn, *Robot Vision,* MIT Press, Cambridge MA, 1986, pp.279-281.

[12] P.J.Huber, *Robust Statistics,* John Wiley and Sons, NY, 1981.

[13] R.Kumar, Doctoral dissertation, Dept. of Computer Science, Univ. of Massachusetts, Amherst, 1992.

[14] R. V. R. Kumar, A. Tirumalai and R.C. Jain, "A Nonlinear Optimization Algorithm for the Estimation of Structure and Motion Parameters," *CVPR*, San Diego, CA, pp. 136-143, 1989.

[15] Y. Liu, T.S.Huang and O.D. Faugeras, "Determination of Camera Location from 2D to 3D line and point correspondences," *Proceedings IEEE Conference on Computer Vision and Pattern Recognition,* pp 82-88, 1988.

[16] D.G.Lowe, "Fitting Parameterized Three-dimensional Models to Images," *Proceedings IEEE 3rd International Conference on Computer Vision,* Osaka, Japan, Dec 1990.

[17] L. Matthies, T. Kanade, and R. Szeliski, "Kalman Filter-Based Algorithms for Estimating Depth from Image Sequences," *International Journal of Computer Vision*, vol 3, pp. 209-236, 1989.

[18] L. Matthies, R. Szeliski, and T. Kanade, "Incremental Estimation of Dense Depth Maps from Image Sequences," *CVPR*, Ann Arbor, Michigan, pp. 366-374, 1988.

[19] J. Oliensis and J. I. Thomas, "Incorporating Motion Error in Multi-frame Structure from Motion," *Proceedings IEEE Workshop on Visual Motion,* Princeton, pp 8-13, 1991.

[20] P.J.Rousseeuw and A.M.Leroy, *Robust Regression and Outlier Detection,*, John Wiley and Sons, NY, 1987.

[21] H. S. Sawhney, J. Oliensis, and A. R. Hanson, "Description and Reconstruction from Image Trajectories of Rotational Motion", in *ICCV*, Osaka, Japan, December, 1990, pp. 494-498.

[22] L.Shigang, S.Tsuji and M. Imai, "Determining of Camera Rotation from Vanishing Points of Lines on Horizontal Planes," *Proceedings 3rd IEEE International Conference on Computer Vision,* Osaka, Japan, 1990, pp. 499-502.

[23] M.J.Stephens, R.J.Blissett, D.Charnley, E.P.Sparks and J.M.Pike, "Outdoor Vehicle Navigation Using Passive 3D Vision," *CVPR*, San Diego, CA, pp. 556-562, 1989.

[24] C.J.Taylor, D.J.Kreigman, and P.Anandan, "Structure and Motion in Two Dimensions from Multiple Images: A Least Squares Approach,"

Proc. IEEE workshop on visual motion, Princeton, NJ, pp. 242-248, 1991.

[25] J. I. Thomas and J. Oliensis, "Fusing Structure by Kalman Filtering," TR 90-93, COINS, UMASS, May 1990.

[26] J. I. Thomas and J. Oliensis, "Incorporating Motion Error in Multi-frame Structure from Motion," *7th Scandinavian Conference on Image Analysis*, Denmark, pp. 950-957, 1991.

[27] J. I. Thomas and J. Oliensis, "Recursive Structure from Multi-frame Motion,", *Proc. Darpa Image Understanding workshop*, Los Angeles, CA, 1992.

[28] J. I. Thomas, *(in progress)* Doctoral dissertation, Dept. of Computer Science, Univ. of Massachusetts, Amherst.

[29] C. Tomasi and T. Kanade, "Factoring Image Sequences into Shape and Motion," *Proc. IEEE workshop on visual motion*, Princeton, NJ, pp. 21-28, 1991.

[30] R.Y.Tsai and T.S.Huang, "Uniqueness and estimation of 3-D motion parameters and surface structures of rigid objects," pp. 135-171.

ALGORITHM	NUMBER OF VARIABLES	RESULTS ON REAL IMAGE SEQUENCES	
		ACCURACY	COMMENTS
Sawhney et. al. [21]	3	0.9% (25 images)	Camera moves according to constraints
Broida et. al. [3]	8	2.5% (12 images)	Object moves according to constraints
		40.3% (16 images)	Object moves violating constraints
Kumar et. al [14]	8	-	Only qualitative results
Franzen [7]	14	> 7.4% (8 images)	From IEEE motion database
Taylor [24]	$3m-1$	-	Only synthetic results
Tomasi [29]	$3m-1$	2.4% (150 images)	Object far away

Table 1: Batch model acquisition algorithms showing the number of motion variables and accuracy.

ALGORITHM	NUMBER OF ELEMENTS	RESULTS ON REAL IMAGE SEQUENCES	
		ACCURACY	COMMENTS
Heel [9]	n	-	Only qualitative results
Matthies et. al [17]	n	0.5% (11 images)	Only pure translation
Ando [1]	n	-	Only simulations
Shigang et. al. [22]	n	15% (40 images)	Only planar motion
Stephens et. al. [23]	$9n$	1% (50 images)	Results for only 1 point
Cui et. al. [4]	$9n$	random fluctuation	Ground truth unknown: no %s
Thomas et. al. [27]	$9n^2$	0.25% (7 images)	Large rotations
		5.8% (11 images)	From IEEE motion database

Table 2: Recursive model acquisition algorithms showing the number of elements used to model the reconstruction *error*.

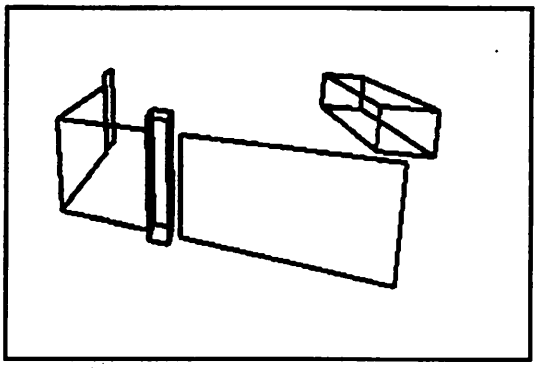

Figure 1. True 3D model of building

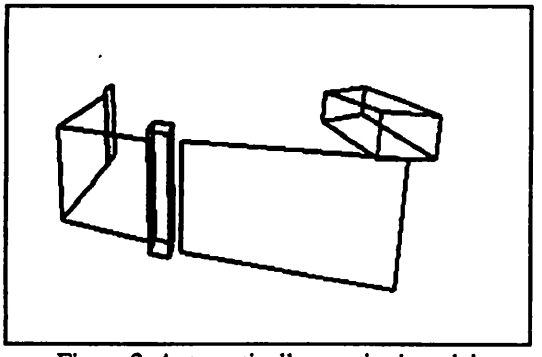

Figure 2. Automatically acquired model

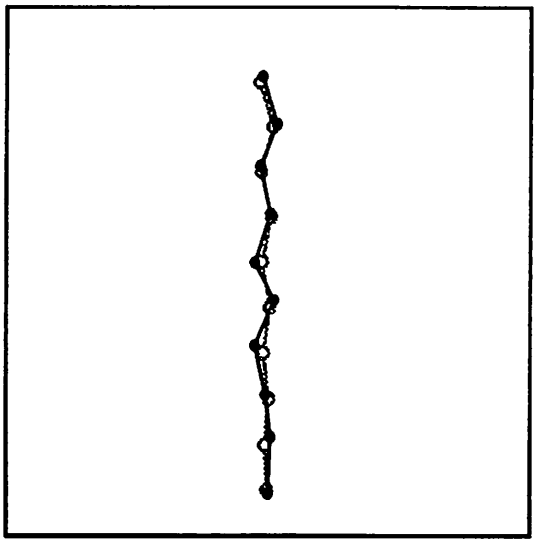

Figure 3. Robot path: truth (gray) and recovered (black)

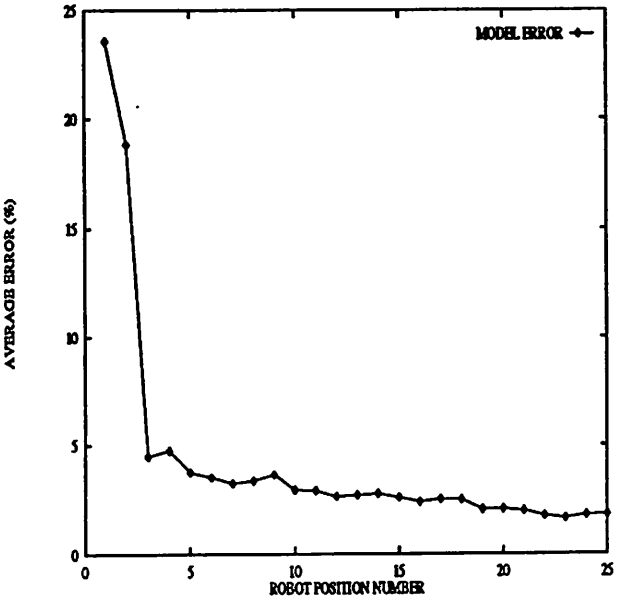

Figure 4. Error in the acquired 3D model of the CS department building

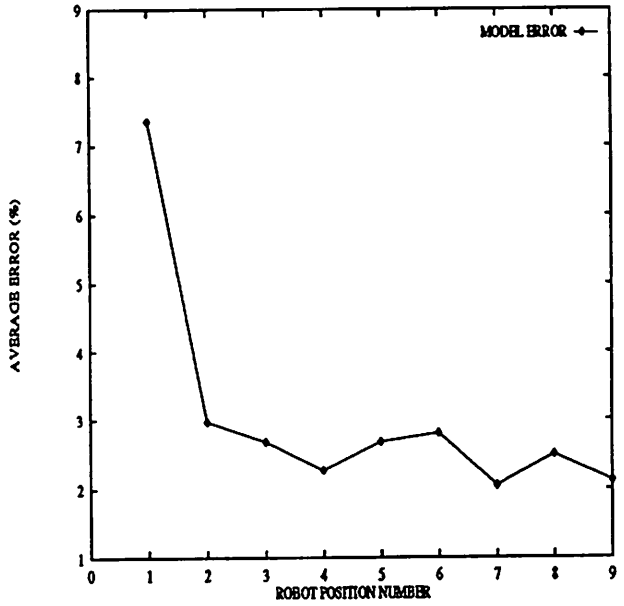

Figure 5. Error in the acquired 3D model of the CS department lobby

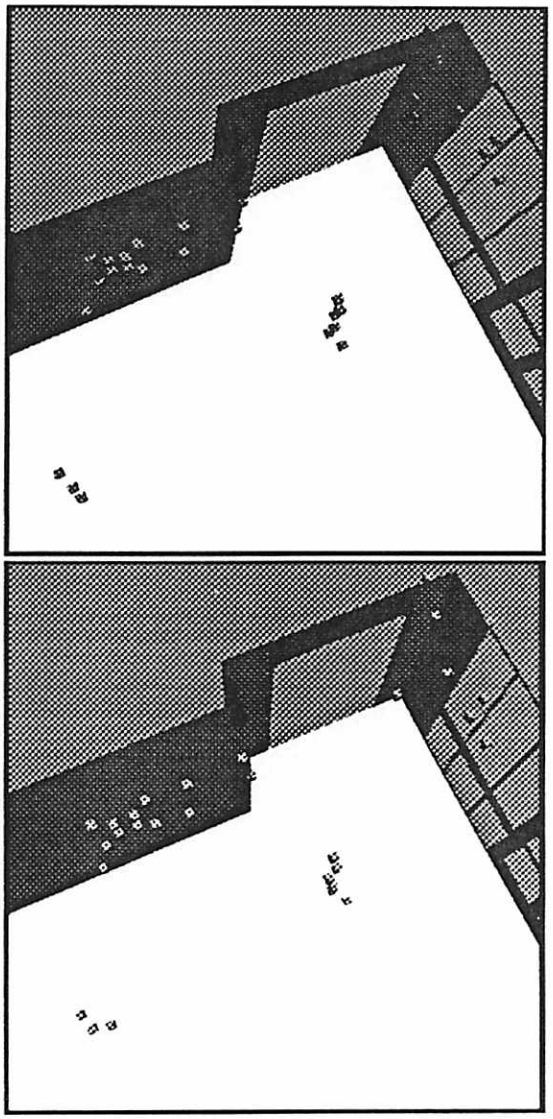

Figure 6. True (top) and automatically acquired models of 31 points

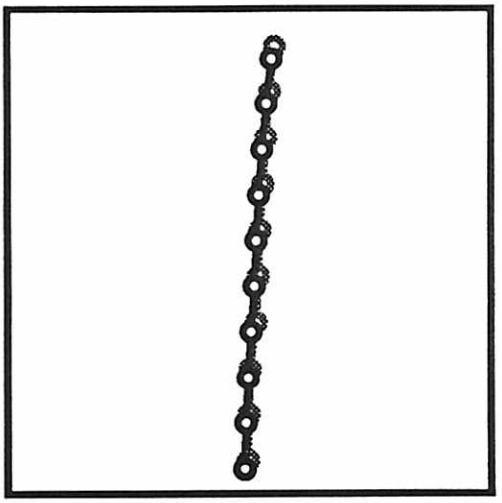

Figure 7. Robot path: truth (gray) and recovered (black)