Proceedings of the 2003 IEEE
International Conference on Robotics & Automation
Taipei, Taiwan, September 14-19, 2003

Using Visual Features to Build Topological Maps of Indoor Environments

Paul E. Rybski, Franziska Zacharias*, Jean-François Lett,
Osama Masoud, Maria Gini, and Nikolaos Papanikolopoulos
Center for Distributed Robotics
Department of Computer Science and Engineering
University of Minnesota, Minneapolis, U.S.A.
{rybski, franz, jlett, masoud, gini, npapas}@cs.umn.edu

***Abstract*— This paper addresses the problem of localization and map construction by a mobile robot in an indoor environment. Instead of trying to build high-fidelity geometric maps, we focus on constructing topological maps as they are less sensitive to poor odometry estimates and position errors. We propose a method for incrementally building topological maps for a robot which uses a panoramic camera to obtain images at various locations along its path and uses the features it tracks in the images to update the topological map. The method is very general and does not require the environment to have uniquely distinctive features.**

I. Introduction

We are interested in building maps using robots that are very small and have limited sensing. Since the robot must physically carry any sensors that it will use, laser range finders or stereo camera systems are generally too large for small robot systems. Miniature robots typically have extremely poor odometry. Slight differences in the speeds of the wheels and small debris or irregularities on the ground will greatly degrade the performance of any dead-reckoning position estimate. This makes accurate localization or mapping very difficult.

Any method for map construction and/or localization must take into account the large amount of error in the robot's sensing and odometric capabilities. We propose the construction of a topological map where each node represents a location the robot visited and took a sensor reading of its surroundings. Initially, the map will contain a node for each sensor snapshot that the robot acquired. Thus, if the robot has traversed the same location more than once, there will be multiple nodes in the map for a single location. To identify such nodes, Markov localization [4] is used to determine the probability of the robot's position at each timestep. These nodes must be combined in order to generate a map which correctly matches the topology of the environment.

As individual nodes are merged, the structure of the map will change and the relative distances and headings between each of the nodes will be affected. When a pair of nodes is merged, the map must find a stable energy configuration so that each of the local displacements between the nodes is maintained properly. A useful analogy to this problem is a physics-based model mass and spring system. Linear distances between each of the nodes can be represented as linear springs while rotational differences between nodes can be represented as torsional springs. The spring constants capture the certainty in the odometry estimates. Stiff springs represent high measurement certainty while loose springs represent low certainty.

Sensor data are obtained from monocular panoramic images of the robot's surroundings. The *Kanade-Lucas-Tomasi* (KLT) feature tracking algorithm [11], [14] is used to extract and match visual features from the images.

II. Related Work

Physics-based models that involve spring dynamics have been used quite effectively to find minimum energy states [3], [6]. The work most similar to ours is by Andrew Howard *et al.* [7]. They use spring models to localize mobile robots equipped with laser range finders. All of the landmarks used in their work are unique, and precise distances to objects are identified using the range finders. In contrast, we only assume we have bearing readings to landmarks and that the landmarks may not be distinguishable.

In [16], a map is learned ahead of time by representing each image by its principle components (using PCA). Kröse *et al.* [10] built a probabilistic model for appearance-based robot localization using features obtained by Principal Component Analysis. In [15], a series of images from an omnicamera is used to construct a topological map of an environment. A color "signature" of the environment is calculated using color histograms.

We use the KLT algorithm to identify and track features. Lucas and Kanade [11] proposed a registration algorithm that makes it possible to find the best match between two images. Tomasi and Kanade [14] proposed a feature selection rule which is optimum for the associated tracker under pure translation between subsequent images. We use an implementation of this feature selection and tracking algorithm to detect features in the environment [9].

*Work done at the University of Minnesota while visiting from the Universität Karlsruhe, Germany

0-7803-7736-2/03/$17.00 ©2003 IEEE

III. Localization and Map Construction

We are interested in constructing a spatial representation from a set of observations that is topologically consistent with the positions in the environment where those observations were made. The goal is to reduce the number of nodes in the map such that only one node exists for each location the robot visited and where it took an image.

More formally, let D be the set of all unique locations (d_i) the robot visited. Let S be the set of all sensor readings that are obtained by the robot at those position. Each $s_i^t \in S$ represents a single sensor reading taken at a particular location d_i at time t. If the robot never traveled to the same location twice, then $|D| = |S|$ (the cardinality of the sets is the same). However, if the robot visits a particular location d_i more than once, then $|D| < |S|$ because multiple sensor readings ($s_i^{t_m}$,$s_i^{t_n}$,...) were taken at that location. The problem then is to determine from the sensor readings and the sense of self-motion which locations in D are the same. Once identified, these locations are merged in order to create a more accurate map.

When using small, resource-limited miniature robots, there are several assumptions about the hardware and the environment that must be made. First, we assume that the robot will operate in an indoor environment where it only has to keep track of its 2D position and orientation. This is primarily a time-saving assumption which is valid because (for the most part) very small robots can only be used on flat surfaces.

We also assume that the robot is capable of sensing the bearings of landmarks around it. This is a valid assumption for small robots (on the order of $5\,\mathrm{cm}$ on a side) because the cameras and omnidirectional mirrors can be made quite small [2]. Finally, we assume that the robot has no initial map of its environment and that the mechanism by which it explores its environment is irrelevant (it might be randomly wandering in an autonomous fashion, or it might be completely teleoperated).

As the robot moves, it keeps track of its rotational and translational displacements. The assumption is that the robot moves in a simplified "radial" [5] fashion where pure rotations are followed by straight-line translations. This is not an accurate representation of the robot's motion because the robot will encounter rotational motion while translating, however in practice we have found that we can discount this for small linear motions.

A. Spring-Based Modeling of Robot Motion

Following each motion, a reading from the robots sensors is obtained. This sequence of motions and sensor observations can be represented as a graph where each node initially has at most two edges attached to it, forming a single chain (or a tree with no branches). The edges represent the translational and the rotational displacement. This can be visualized using the analogy of a physics-based model consisting of masses and springs. In this model, translational displacements in the robot's position can be represented as linear springs and rotational displacements can be represented as torsional springs. The uncertainty in the robot's positional measurements can be represented as the spring constants. For example, if the robot were equipped with high precision odometry sensors, the stiffness in the springs would be very high.

By representing the locations as masses and the distances between those locations as springs, a formulation for how well the model corresponds to the data can be expressed as the potential energy of the system. The Maximum-Likelihood Estimate (MLE) of the set of all sensor readings S given the model of the environment M can be expressed as $P(S|M) = \prod_{s \in S} P(s|M)$.

By taking the negative log likelihood of the measurements, the problem goes from trying to maximize a function to minimizing one. Additionally, by expressing the allowable compressions of the spring as a normal probability distribution (i.e., the probability is maximized when the spring is at its resting state), the log likelihood of the analytical expression for a Gaussian distribution is the same as the potential energy equation for a spring, or $-log(P(s|M)) = \frac{1}{2}(e - \hat{e})^2 k$.

In this formulation, e is the current elongation of the spring, $\hat{e}$ is the relaxation length of the spring and k is the spring constant. In order to minimize the energy in the system, direct numerical simulation based on the equations of motion can be employed. Figure 1 shows a simple example of how the linear and torsional springs are used to represent the difference between the current model and the robot's sensor measurements.

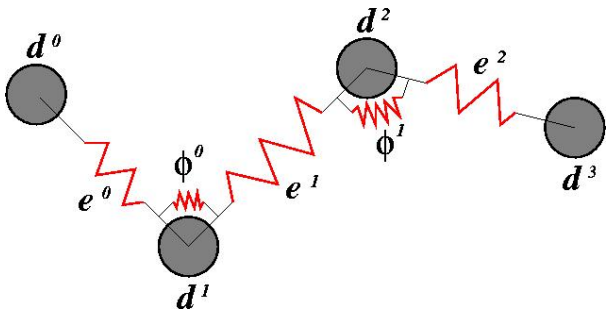


Fig. 1. *Examples of relative poses of the robot connected by linear and torsional springs. Locations of sensor readings, lengths of linear robot translation and angles of robot rotation are represented as d^i, e^j, and ϕ^k, respectively.*

When the sensor readings of two nodes are similar enough to be classified as a single node, the algorithm will attempt to merge them into a single location. This will increase the complexity of the graph by increasing the number of edges attached to each node. This will also apply additional tension to all of the other springs and the structure will converge to a new equilibrium point.

If the landmarks observed at each location are unique, such as in the work of Howard *et al.*, then the task of matching two nodes which represent the same locations is fairly straightforward. However, in real world situations and environments, this is extremely unlikely to occur.

Without pre-marking the environment and/or without extremely good *a priori* information, the robots cannot assume to be able to uniquely identify each location.

B. Map Construction

Markov Localization will compute, for each timestep, a distribution which shows the probability of the robot's position across all nodes at a particular time. In cases where the probability distribution is multi-modal, or where it is nearly equally likely that the robot was in more than one node at a time, there exists a good chance that those nodes are actually a single node that the robot has visited multiple times. The hypothesis with the highest probability of match from all of the timesteps is selected and those nodes are merged. Merging nodes distorts the model and increases the potential energy of the system. The system then attempts to relax to a new state of minimum energy. If this new state's potential energy value is too high, then the likelihood that the hypothesis was a correct one is very low and must be discarded. This process runs through several iterations until it convergences on the most topologically-consistent map of the environment. This iterative process is similar in spirit to the algorithm proposed by Thrun *et al.* [13]. Since this algorithm relies on local search to find nodes to merge, there is no guarantee that the map constructed from this algorithm will be optimal.

C. Sensor and Motion Models

The robot's sensor model can be described as $P\left(s^t|L^t, M\right)$. This is an expression for the probability that at time t, the robot's sensors obtain the reading s^t given that the estimate for the robot's position is given by the probability distribution L^t. One way to generate this distribution is through a non-parametric method such as Parzen windows (a similar approach is used by [10]). Following the definition of conditional probabilities, the equation for the sensor model can be described as:

$$\begin{aligned} P\left(s^t|L^t, M\right) =& \frac{P\left(s^t, L^t, M\right)}{P\left(L^t, M\right)} \\ =& \frac{\frac{1}{N}\sum_{n=1}^{N} g_s\left(s^t - s_n^t\right) g_d\left(d^t - d_n^t\right)}{\frac{1}{N}\sum_{n=1}^{N} g_d(d^t - d_n^t)} \end{aligned}$$

where g_s and g_d are Gaussian kernels. The value $(s^t - s_n^t)$ represents the difference between two sensor snapshots and is described in Section IV-A below. The value $(d^t - d_n^t)$ represents the shortest path between two nodes.

Similarly, the robot's motion model can be expressed as $P\left(L^{(t+1)}|s^{(t)}, L^{(t)}\right)$, which represents the probability that the robot is in location $L^{(t+1)}$ at time $t+1$ given that its odometry registered reading $s^{(t)}$ after moving from location $L^{(t)}$ at time t. This is represented as:

$$P\left(L^{(t+1)}|s^{(t)}, L^{(t)}\right) = g_e(e - \hat{e}) g_\phi(\phi - \hat{\phi})$$

where e and ϕ represent the linear and torsional components of the robot's motion in the current map and $\hat{e}$ and $\hat{\phi}$ represent the originally measured values.

IV. Real-World Validation

In order to determine the effectiveness of the KLT algorithm for localizing the robot, two separate experiments were performed. The first was a localization-only experiment where the KLT algorithm was used in two different ways. The second combined the KLT algorithm with the spring system to test the ability of the MLE algorithm to converge to a topologically-consistent map.

A. Visual Features

Localization is done by matching features extracted from visual images. To compare images, two different metrics were tried: (1) static feature matching and (2) feature tracking.

In the *feature matching* approach, features are selected in each histogram normalized image using the KLT algorithm. The *Undirected Hausdorff metric H(A,B)* [8] was used to compute the difference between the two sets. Since this metric is sensitive to outliers, we used the generalized undirected Hausdorff metric and looked for the k-th best match (rather than just the overall best match), where k was set to 12. This is defined as:

$$\begin{aligned} H(A, B) &= \max_{kth}(h(A, B), h(B, A)) \\ h(A, B) &= \max_{a \in A} \min_{b \in B} \parallel a_i - b_j \parallel \end{aligned}$$

where $A = \{a_1, a_2, ..., a_m\}$ and $B = \{b_1, b_2, ..., b_n\}$, are two feature sets. Each feature corresponds to a 7x7 pixel window (the size of which was recommended in [14]) and $\parallel a_i - b_j \parallel$ corresponds to the sum of the pixel differences.

In the *feature tracking* approach, KLT features are selected from each of the images and are tracked from one image to the next taking into account a small amount of translation. The degree of match is the number of features successfully tracked from one image to the next. Each approach has different advantages and disadvantages. Extracting features using the KLT algorithm but not accounting for the translation of the feature from one image to the next has the advantage of being faster and requiring less memory than using the associated tracker. However, it is likely to be less precise due to the fact that there is no model for how the features move in the images.

A total of 15 features are selected from each image and used for comparison. To take into account the possibility that two panoramic images might correspond to the same location but differ in rotation, the test image was rotated to eight different angles to find the best match.

B. Image-Based Localization Experiments

A set 26 of panoramic images were obtained in an office environment. shown in Figure 2. The dotted lines show the outline of the office and the furniture within it while the solid lines show the path along which the images were taken. Images were taken at $1.07\,\mathrm{m}$ increments by a panoramic camera mounted on the back of a Pioneer 2 [1] mobile robot. The KLT feature matcher was used to extract features from panoramic images. Figure 3 shows a set of

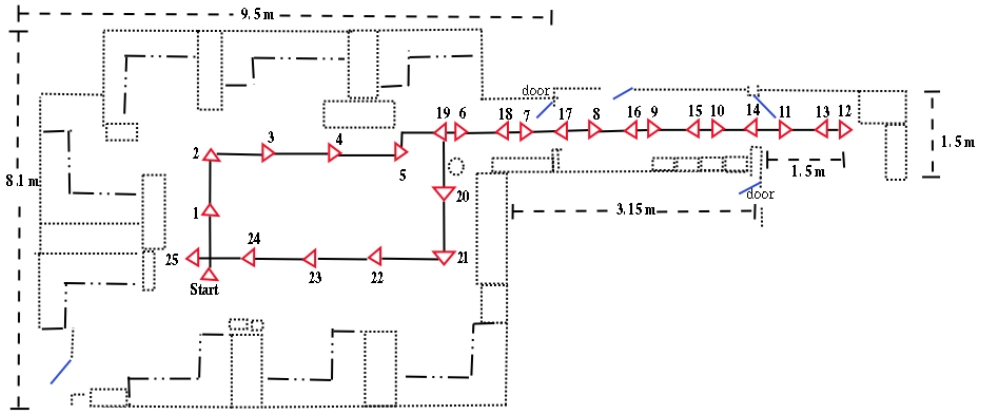


Fig. 2. *Map of the office environment where our tests were conducted. The nodes of the robot's training path are shown with triangles.*

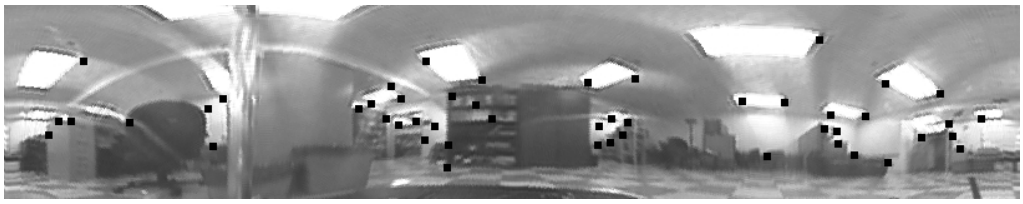


Fig. 3. *The 50 best features selected with the KLT feature selector on a panoramic image. In our experiments, only the 15 best features were used.*

features obtained by applying the feature matcher to a panoramic image. As can be seen, features corresponding to corners and prominent edges are selected.

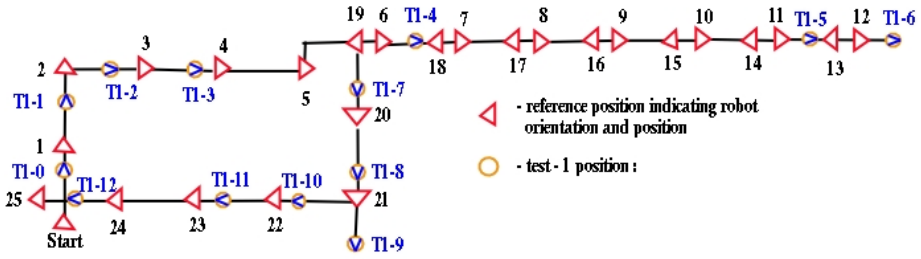
Two sets of test images were acquired along the paths shown in Figure 4(a) and Figure 4(b). Triangles show the positions of the original test set of images. Circled arrows show the positions of the images taken for the test sets. The images in the first test set were mostly taken along the original path from which the training set was obtained. The images in the second set were taken in a zig-zag pattern that moved mostly perpendicular to the path of the training set. Table I illustrates the performance of the two vision algorithms on the different sets of data. The average distance error is the average Euclidean distance between the correct position and the reported position. The second metric is the number of position matches that reported multiple possible positions of the robot with equal certainty (caused by perceptual aliasing). The correct position to be attributed to a test position is assumed to be the nearest position (by Euclidean distance) of the reference path. When multiple position estimates are available, the worst possible position is used. The reason that the tracker had multiple position estimates when the matcher did not was due to the scale difference in the error metrics. The tracker computed the number of features that matched between images which could range between $[0-15]$. The feature matcher compared the difference in pixel image intensity which could range between $[0-12495]$. The tracker was thus far more likely to have cases where multiple locations had the same matching score.

Error metric	Test set 1		Test set 2	
	Feature matcher	Feature tracker	Feature matcher	Feature tracker
Ave. distance in meters	1.58	0.51	2.97	1.34
Number of multiple position estimates	0	1	0	6

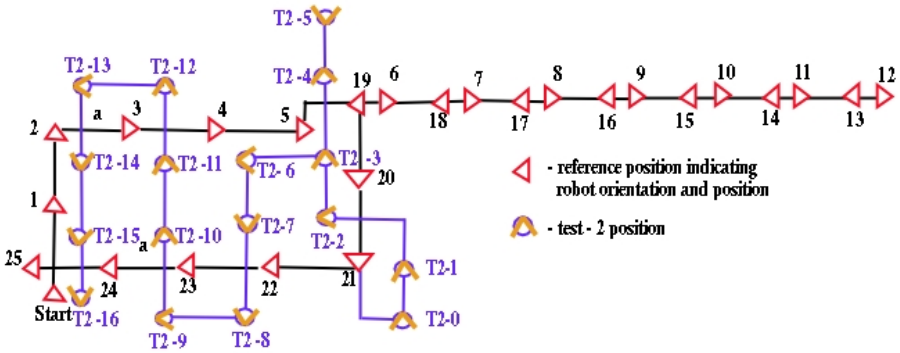
TABLE I

Average errors for the tests.

As can be seen from the results, the static KLT feature matching algorithm was worse at finding the best match between an image in the test set and the image in the training set. When the training and test images are nearly identical (taken from virtually the same location in space), the static feature matcher was very good at finding the correct match. However, as the spatial difference between the images increases, the resulting match rapidly degrades. The feature tracking algorithm did a much better job of matching images in the test set to the training set. This algorithm was much better at handling changes in feature position caused by the motion of the robot since it takes into account the translational motion of the features in the image. Unfortunately, the KLT feature tracking algorithm is much more complex in terms of computing time and memory/storage requirements.



(a) *Images for test 1 set were taken in between reference positions along the path.*



(b) *Images for test 2 were taken on a zigzag path across the training path.*

Fig. 4. *Paths in the environment where our tests were conducted. The training positions are labeled with triangles and the test positions are labeled with circled wedges. The heading of the robot at each node is shown by the direction of the triangle or wedge.*

C. Mapping Experiment

The set of training images taken in the previous experiments were used to test the MLE map construction algorithm. Noisy odometry estimates were assigned to each of the paths between images in the training set. The KLT feature tracking algorithm was used to compare features in pairs of images and only the training set of images was used. This corresponds to the case where a robot explored an unknown environment. As the robot explores, it attempts to find the most likely structure by merging nodes from its map which appear to correspond to the same sensor data.

Figure 5 illustrates the process of how the algorithm works. The original data reflects the errors in the odometric readings of the robot. In Step 1, Markov localization identifies a high probability of the robot's position in nodes at timestep 6 and 19. These two are merged and the spring model is allowed to relax. In Step 2, Markov localization is run again on the map and nodes 11 and 14 are merged. By this point, the map has obtained a shape that better matches the topology of the environment. Each possible merge candidate is evaluated by how the merge affects the entropy of the pose distribution. Bad merges will create inconsistent topological structures and have a tendency to increase the robot's pose entropy. This means that it is less sure of its position in the environment.

V. Future Work

We ultimately plan to use this technique to localize miniature robots, called Scouts, using images taken from their cameras. The Scout robot, developed at the Center for Distributed Robotics of the University of Minnesota [12], is a differentially-driven cylindrical robots ($11\,\mathrm{cm}$ in length and $4\,\mathrm{cm}$ in diameter) equipped with a single monocular camera. Their small size restricts them to off-board processing of their video signals.

The mapping algorithm has been found to be very sensitive to certain parameters. The spring and dampening constants used by the spring convergence step must be selected carefully to ensure convergence. To address this,

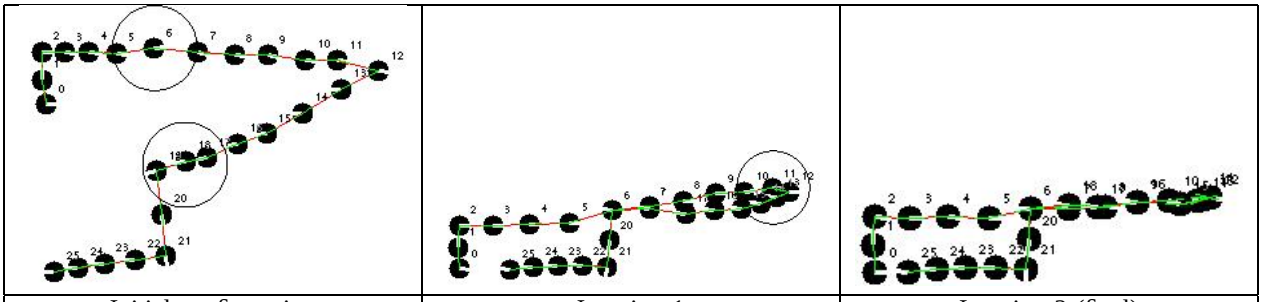


Fig. 5. *Several iterations of the convergence algorithm. Circled nodes are to be merged in the next iteration. Only the accepted node merge candidates are shown in this example. Node merge candidates that increased the entropy of the pose distribution (and thus were rejected) are not shown. After iteration 2, all other node merge pairs were rejected.*

other methods being examined include weighted least squares and the Kalman filter. Another parameter that could affect the performance of the localization algorithm are the widths of the Gaussian distributions used in the Parzen windows. Empirical studies are being done to determine good values for these parameters. Finally, the entropy of the pose distribution is used as a method for tracking the progress of the algorithm. More robust methods, such as stochastic sampling, are under development.

VI. Acknowledgements

The authors would like to thank Andrew Howard at the University of Southern California for his thoughtful comments and insight into the MLE spring model formalism.

This material based in part upon work supported by the National Science Foundation through grant #EIA-0224363, Microsoft Inc., and the Defense Advanced Research Projects Agency, Microsystems Technology Office (Distributed Robotics), ARPA Order No. G155, Program Code No. 8H20, Issued by DARPA/under Contract #MDA972-98-C-0008.

VII. References

[1] ActivMedia Robotics, LLC, 44 Concord Street, Peterborough, NH, 03458. *Pioneer 2 Operation Manual v6*, 2000.

[2] S. Derrien and K. Konolige. Approximating a single viewpoint in panoramic imaging devices. In *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, pages 3932–3939, 2000.

[3] T. Duckett, S. Marsland, and J. Shapiro. Learning globally consistent maps by relaxation. In *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, volume 4, pages 3841–3846, 2000.

[4] D. Fox. *Markov Localization: A Probabilistic Framework for Mobile Robot Localization and Navigation*. PhD thesis, Institute of Computer Science III, University of Bonn, Germany, December 1998.

[5] R. Grabowski, L. E. Navarro-Serment, C. J. J. Paredis, and P. Khosla. Heterogeneous teams of modular robots for mapping and exploration. *Autonomous Robots*, 8(3):293–308, 2000.

[6] V. V. Hafner. Cognitive maps for navigation in open environments. In *Proceedings of the 6th International Conference on Intelligent Autonomous Systems (IAS-6)*, pages 801–808, Venice, Italy, 2000.

[7] A. Howard, M. Matarić, and G. Sukhatme. Localization for mobile robot teams using maximum likelihood estimation. In *Proc. of the IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, EPFL Switzerland, Sept. 2002.

[8] D. Huttenlocher, D. Klanderman, and A. Rucklige. Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863, September 1993.

[9] KLT: An implementation of the Kanade-Lucas-Tomasi feature tracker. http://vision.stanford.edu/~birch/klt/.

[10] B. Kröse, N. Vlassis, R. Bunschoten, and Y. Motomura. A probabilistic model for appearance-based robot localization. In *Image and Vision Computing*, volume 19, pages 381–391, 2001.

[11] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. *IJCAI*, pages 674–679, 1981.

[12] P. E. Rybski, S. A. Stoeter, M. Gini, D. F. Hougen, and N. Papanikolopoulos. Performance of a distributed robotic system using shared communications channels. *IEEE Trans. on Robotics and Automation*, 22(5):713–727, Oct. 2002.

[13] S. Thrun, W. Burgard, and D. Fox. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, 31:29–53, 1998.

[14] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical report, School of Computer Science, Carnegie Mellon University, April 1991.

[15] I. Ulrich and I. Nourbakhsh. Appearance-based place recognition for topological localization. In *Proc. of the IEEE Int'l Conf. on Robotics and Automation*, pages 1023–1029, San Francisco, CA, April 2000.

[16] N. Winters and J. Santos-Victor. Omni-directional visual navigation. In *Proc. of the 7th Int. Symp. on Intelligent Robotic Systems (SIRS 99)*, pages 109–118, Coimbra, Portugal, July 1999.