Markov Localization using Correlation

Content Areas: robotics, perception
Tracking Number: A595

Abstract

Localization is one of the most important capabilities for autonomous mobile agents. Markov Localization (ML), applied to dense range images, has proven to be an effective technique. But its computational and storage requirements put a large burden on robot systems, and make it difficult to update the map dynamically. In this paper we introduce a new technique, based on *correlation* of a sensor scan with the map, that is several orders of magnitude more efficient than ML. CBML (correlation-based ML) permits video-rate localization using dense range scans, dynamic map updates, and a more precise error model than ML. In this paper we present the basic method of CBML, and validate its efficiency and correctness in a series of experiments on an implemented mobile robot base.

1 Introduction

Localization is the process by which a mobile robot or other physical agent keeps track of its position as it moves around an environment. It is an essential capability for autonomous mobile robots if they are to perform tasks in an efficient way: a robot that gets lost in performing a delivery is useless.

The problem of localization is made difficult because, in general, we are trying to construct robots that can act intelligently in environments that are imperfectly known, and for which their sensors give only uncertain information. This naturally leads to the consideration of probabilistic methods, in which the spatial state of the robot is represented as a probability distribution over the space of possible robot *poses* (location and direction). The problem of localization is then the problem of updating the distribution, based on robot motion and sensing, given a map of the environment that may be imperfect.

A recent approach to this problem called *Markov localization* (ML) has proven to be both robust and accurate. ML makes the choice of an explicit, discreet representation for the prior probability, using a grid or topological graph to cover the space of robot poses, and keeping a probability for each element of this space. The key idea of ML is to compute a discrete approximation of a probability distribution over all possible poses.

Different variants of ML have been developed [Burgard *et al*. 1998, Kaelbling *et al.* 1996; they can be distinguished according to the type of sensor readings they work with. Here we are concerned with *dense methods*, in which a all the information in a range scan is used to match against a model of the environment [Burgard *et al.* 1997].

Dense methods have proven particularly successful in robot localization tasks, including several real-world robot installations as tour-guides in museums [Burgard *et al.* 1998, Thrun *et al.* 1998]. Still, dense methods are subject to several daunting representational and computational challenges that make them difficult to use. The basic problem is that the probability update step of ML involves comparing actual sensor readings to readings that would have resulted from having the robot at every possible pose in a given map. This update can be enormously expensive, generating billions of sensor comparisons for even moderately sized maps, if done in a straightforward way. But through clever approximations, and especially by pre-computing expected sensor readings from all map poses, researchers have made it possible to apply dense ML in practical systems [Burgard *et al.* 1998]. Yet there remain significant problems, some brought on by the approximations.

- Dense ML remains computationally difficult. For example, globalization in moderately-sized environments (50x50 m^2) can take many minutes for integration of a set of sensor readings.
- Environment maps are fixed and cannot be easily updated to account for dynamic objects or structural changes such as open doorways, because the expected sensor readings must be re-computed.
- Precise localization with fine spatial grids is difficult, because it presents increased computational and storage demands, especially for the pre-computed expected readings.
- Error models for objects in the map are poor approximations to actual errors, because they must be

represented by a single number for each pose, the expected distance to the object.

In this paper we present a new method for computing Markov localization, which we call *correlation-based Markov localization* (CBML), that addresses these problems. Instead of pre-computing expected distances to map objects, we use techniques from image processing to correlate a sensor scan with an existing map. Because these methods are regular, that is, they are applied uniformly over the map, we can take advantage of SIMD (single-instruction, multiple data) instructions in modern processors to yield extremely efficient implementations. In comparisons with the best existing ML methods on similar processors, CBML is two orders of magnitude faster.

Besides being efficient, CBML does not have to precompute expected object distances in the map. Thus, it tolerates changes to the map on-the-fly, allowing dynamic objects or structural information to be incorporated. An additional benefit is that the grid size of the map need not be fixed ahead of time, so that the grid resolution can be increased or decreased as necessary: a coarse grid for initial globalization, and a fine grid for precise maneuvering. A more realistic uncertainty model can be introduced into the map, increasing the fidelity of the probabilistic updating process. Finally, correlation opens the door for some techniques that are not possible with standard ML, for example, the use of fused sensor readings for extended matching.

In the rest of this paper, we develop the mathematics of CBML, and show how it corresponds to the equations of ML. Then, we explore some of the performance issues associated with CBML, and show how it can be implemented in an efficient way. We present experimental results of the method using a Pioneer II platform equipped with a laser range finder. Finally, we discuss some implications of correlation on future capabilities of ML.

2 Markov Localization as a Correlation Operation

2.1 Markov Localization

ML, like other filtering operations designed to estimate the state of a system, consists of two steps: predicting the next state of the system based on state-space motion, and updating the predicated state based on sensor readings. For mobile robots, the prediction step uses dead-reckoning from internal encoders, and we won't consider it further in this paper. The update step changes the predicted probabilities according to the formula:

$$p(l \mid s) = k \int p(s \mid l, m)\, p(l)\, p(m) dm, \qquad (1a)$$

where l is the robot pose, s is the sensor reading, m is the probability that a particular map location is occupied by a surface seen by the sensor, and k is a normalization factor. Given a new sensor reading s, the probability that the robot is at location l is a function of the sensor model $p(s \mid l, m)$, the prior probability of the robot being at pose l, and the prior probability of the map location m being occupied. The sensor model represents the probability of getting a reading s, given the robot is at pose l and the nearest map object is at m. Note that (1) explicitly takes into account both sensor error and map error. The Markov assumption is that the updated probability depends only on the robot's expected position and current sensor readings, and not their history.

Multiple sensor readings taken at different angles from the same location are assumed to be independent, and the update function is:

$$p(l \mid \vec{s}) = k \prod_i \int p(s_i \mid l, m)\, p(l)\, p(m) dm\ . \qquad (1b)$$

In practical systems, uncertainty in the map position is eliminated by adding more noise into the sensor model, so (1) becomes

$$p(l \mid s) = k \cdot p(s \mid l, m)\, p(l)\ . \qquad (2)$$

Here it is assumed that the location of the map object is fixed and known. The evaluation of (2) involves computing the distance to the nearest map object in the direction of the sensor, which is expensive. To compute $p(s \mid l, m)$ quickly, the distance is pre-computed and stored for every possible sensor angle and position, leading to a very large data structure for the map. Any changes to the map mean that the expensive precomputation must be repeated.

2.2 Correlation and ML

To obviate the need for pre-computation of object distances, we introduce a new method for determining the probability $p(s \mid l, m)$. This method is based on the concept of *correlation* between the map and the sensor scan. In correlation, a sensor scan is converted to a small discretized area patch, and matched against the larger discretized map. A high value for correlation indicates a good match between sensor readings and the map. Figure 1 shows the basic idea: a sensor indicates an uncertain range to an object, as shown by the discretized sensor patch on the left. A discretized map contains uncertain information about the location of objects. The correlation operation places the sensor patch on the map, multiplies the corresponding values, and sums them to give the response of the sensor at the origin of the sensor. Moving the sensor patch over the whole map computes the complete sensor response.

Given a sensor patch P and a map M, we must define a correlation operation that computes (1). For each cell c_i in P, let

$$c_i = p'(s \mid l, m_i)\ , \qquad (3)$$

where $p'(s \mid l, m_i)$ is the probability of getting a sensor reading s, given the robot is at location l and there is a map object at the cell location. Then, the correlation is computed as

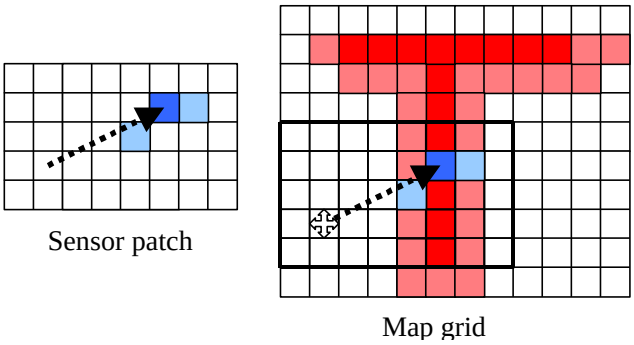

Figure 1 A sensor patch correlated with a map grid.

$$Corr(s,l) = \sum_i c_i \, p(m_i) \,, \tag{4}$$

and the updated probability is

$$p(l \mid s) = k \cdot Corr\,(s,l)\, p(l) \,. \tag{5}$$

Equation (5) is only an approximation to (1), since we are using an approximate sensor model $p'(s \mid l, m_i)$. The difference is that $p'(s \mid l, m_i)$ does not take into account *visibility constraints*, that is, objects closer than m_i are not considered to interfere with the sensor. For example, in Figure 2 correlation shows a strong response, but ML would not, because the closest object is not at the sensor range. Correlation violates the visibility constraint of the range sensor. In the next section we present experimental evidence that correlation is still a good approximation to ML updating.

Note that (4) deals with uncertainty in both the map and the sensor. In a manner similar to ML, we can fold the sensor error back into the map error, so that the sum in (4) reduces to a single value per sensor reading. This idea is the key to implementing an efficient version of correlation, since it allows many sensor readings to be processed using the same correlation patch. Let c_i be the cell where a sensor reading s_i falls. Then the sensor patch is defined as being 1 at all such cells c_i, and 0 elsewhere. Now multiple-sensor correlation becomes:

$$MCorr(\vec{s},l) = \prod_i c_i \, p(m_i) \,. \tag{6}$$

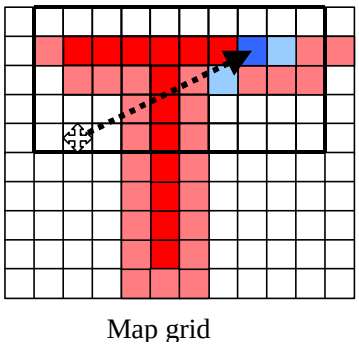

Figure 2 Correlation violating a visibility constraint for the sensor.

Using *Mcorr* in place of *Corr* in Equation (5) gives an approximation to Equation (2). It does not take into account visibility constraints, and assumes the sensor error has been accounted for in the map error.

The advantage of correlation is that it uses the map as is, without any additional pre-computed information. Storage requirements for the map are reduced by two orders of magnitude, and it is possible to dynamically update the map as new information is made available. Correlation using (6) is also computationally efficient, as we show in Section 3.

2.3 Experimental Validation

In this section we report results of numerical experiments to compare the performance of the CBML and ML methods. The focus of these experiments is to evaluate the effects resulting from conceptual and implementation differences between the two methods. For CBML, we use the variance of the sensor probability *p(s|l,m)* as the basis for an isotropic kernel to blur the map. This gives roughly equal error models for the two methods.

The numerical experiments were performed using a 100 by 100 position grid. Figure 1 displays the results of the ML (top) and CBML (bottom) methods applied to data corresponding to a 360 degree span, where range measurements are taken every 1 degree. The sensor patch was taken from the middle of the figure. The variance, as explained above, is set to 0 for the left figures, 4 pixels for the middle, and 8 for the right. The color images represent the values of log*(p(l|s))* in pseudo-color, with yellow and then red being the highest values. We use a log scale for the intensities in order to highlight the differences in the fine structure of the densities. Note that the CBML result shows additional lines not found in the ML case. These lines result from the violation of visibility constraints. These spurious features, however, are extremely low in probability; if plotted on a linear rather than a log scale, the results of the two methods are virtually identical.

For the nonzero variance cases, the results show a much greater spread in probability values, as expected. The CBML results, however, differ from those of the ML method in at least two ways in the vicinity of the peak: 1) the spreading seems larger for CBML, given identical sigma and 2) the spreading is less symmetrical in the case of CBML vs. ML. The first point indicates that differences in the implementation of the uncertainty models results in greater spreading of the map uncertainty for CBML. The second point provides evidence of the fact that the uncertainty model for CBML more faithfully reflects map uncertainty than ML does.

In order to characterize these differences on average, an experiment was performed using 10 different robot positions. In order to compare the posterior density *p(l|s)* for the two different methods the following difference measures were used.

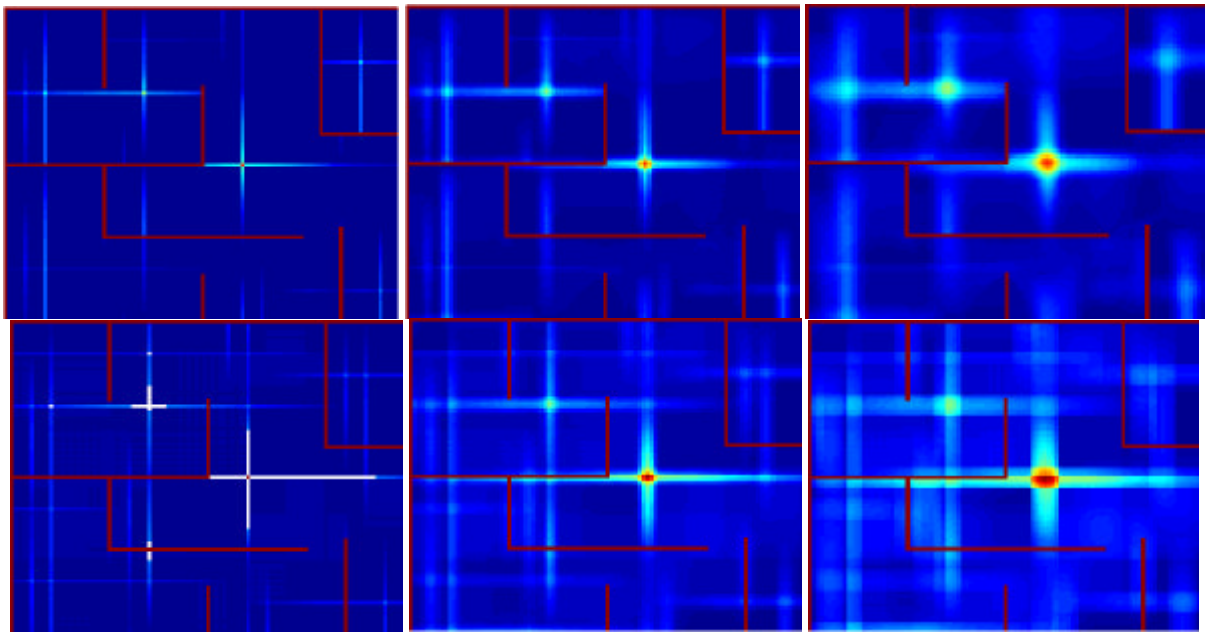

Figure 3 ML vs. CBML on a typical map, with different sensor variances. Top figures are ML, bottom CBML. Dark red lines are map segments; color indicates increasing probability from blue, white, yellow, and red.

- The *divergence* is an information-theoretic measure for the closeness of two distributions. It is defined as the difference in expected values of the log-likelihood ratio with respect to two densities: E1[*L*] – E2[*L*], where *L* = *log(p(l|s))* / *log(p(l|s))* and E1 is expectation with respect to p(l|s) using ML, while E2 is the expected value taken using CBML.
- *max(p(l|s))* gives a measure of the peakedness of the density. This measure provides a way of comparing the performance of the two methods in that a) the location of the peak indicates the estimated position and b) the value of *p(s|l,m)* at the peak is the probability that the position corresponds to the pose *l*.

The average statistics for the divergence measure and peak *p(l|s)* values are shown in Figure 4, The top graph shows the divergence of CBML and ML, divided by the divergence of ML and a uniform distribution. A value close to zero indicates that CBML tracks ML much better than chance. In fact, the divergence measure statistics show that *p(l|s)* for the two methods is virtually identical for small variance. As variance increases, the CBML results spread more than ML, and CBML at smaller spreads tracks ML more closely. As before, this effect is probably due to the difference in error models between the map and the sensor.

The peak value graph (bottom) shows the ratio of peak values for CBML and ML. Here, a result near 1 indicates they are similar; lower than this, CBML is less specific at localization than ML. Again, CBML tracks ML well at lower sensor/map variance, then tends to spread more (and have a lower peak) at higher variance. In part, this difference may be accounted for by the fact that at

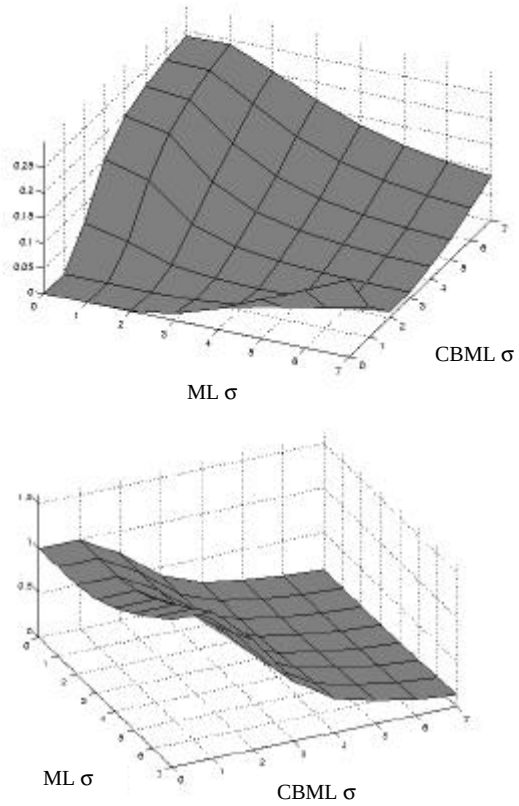

Figure 4 Divergence (top) and peak value measures.

higher variances, CBML has increased sensitivity to visibility constraint artifacts; in part, it arises because the map error model is different from sensor error model, and probably more appropriate (see below).

2.4 Sensor and Map Error Models

We return briefly to the subject of error models, to point out the differences between ML and CBML assumptions. In general, the sensor response to a map, $p(s \mid l, m)$, is a complicated function of the sensor characteristics and the environment. This is especially true for wide-angle, active sensors such as sonars or radars, which are subject to specular reflection, corner reflection, and similar phenomenon [Leonard *et al.* 1990]. Simplifying assumptions are needed to evaluate (1a) efficiently.

ML implementations typically assume that the distance to the nearest object, given a particular sensor pose, is the only map information that is needed in (1a). If the map is uncertain, then the sensor model is broadened by an appropriate amount, e.g., if the sensor variance is σ_1^2, and the map variance is σ_2^2, then the sensor variance is reset to $\sigma_1^2+\sigma_2^2$. This assumption fails badly, however, when there are depth discontinuities in the map. Consider Figure 5, in which two walls are offset in depth.

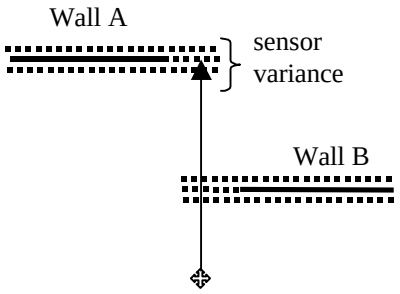

Figure 5 Sensor vs. map variance.

Wall variance is indicated by the dotted lines; sensor variance is along the sensor line of sight. I n this situation, we would expect a sensor reading at the depth of wall A or B. Relying on sensor variance alone, the best we can do is to pick a point midway between the walls, and extend the sensor variance to include both walls. This is not a very accurate approximation of the actual situation. On the other hand, folding the sensor variance into the map, by extending the wall variance slightly, gives a much more realistic error model.

2.5 Other Sensor Grid Matching Schemes

Techniques for matching evidence grids [Moravec and Elfes 1985] are similar to the sensor correlation method proposed here [Schiele and Crowley 1994]. Recent implementations of the method [Schultz and Adams 1998] have shown impressive results, and do allow the simultaneous integration of new information with the map. Our work is distinguished by being more efficient (by many order of magnitude; see below) and having a Bayesian theoretical justification.

3 Implementation and Results

To test the efficiency and practicality of the method, we integrated it into a robot localization system on our Pioneer II robot, equipped with a laser range finder (LRF) for sensing [ref omitted]. An LRF scan returns a hemicircle of 180 readings at 1 degree increments; the range error is 1 cm, and the maximum range is 8 m. Several techniques were used to make CBML efficient; we describe these methods, some experiments in global localization, and extensions to the method.

3.1 Log Grid Update

To implement (6) and (5), we use a log representation of the probabilities. Log probabilities have some advantages for computation:

- Multiplication of probabilities is addition of logarithms, so the correlation operation (6) uses addition.
- Normalization is accomplished by addition of a constant to all values; in practice we don't worry about normalization, just relative log probabilities.
- Small probabilities are represented more precisely.
- Small integers can be used for efficient storage.

Mcorr is implemented as a series of additions on small integers. If the robot poses are represented by a regular grid, we can take advantage of parallel instructions (SIMD) available on most processors to generate several results simultaneously.

To describe the computational properties of the CBML algorithm, or any other dense ML update process, we propose the following measure. In general, the amount of time taken by any method is proportional to the number of poses that must be updated, and the number of sensor readings that must be integrated. So the measure of power of an implementation is the amount of time spent per pose-reading. For our CBML implementation, on a 400 MHz PII we achieve a power rating of less than 1 ns per pose-reading. This compares with 75 ns per pose-reading using an optimized ML algorithm [D. Fox, personal communication].

The efficiency of CBML makes it possible to run local tracking, using a small grid, at video rates, even for the relatively large number of sensor readings in an LRF scan. Typically it takes only a few milliseconds to integrate these readings and update the pose.

Globalization is also efficient. Figure 6 shows an example in our office environment. The map is 38 by 30 meters, and the grid resolution is 10 cm and 2 degrees, for a total of 20×10^6 poses. Each sensor scan has 180 readings, yielding a total of 3.6×10^9 update operations per scan. The time for one scan update is less than a second on a 400 MHz PII (the lower than expected time is a result of the sensor patch discretization, which collapses several LRF readings into one grid space).

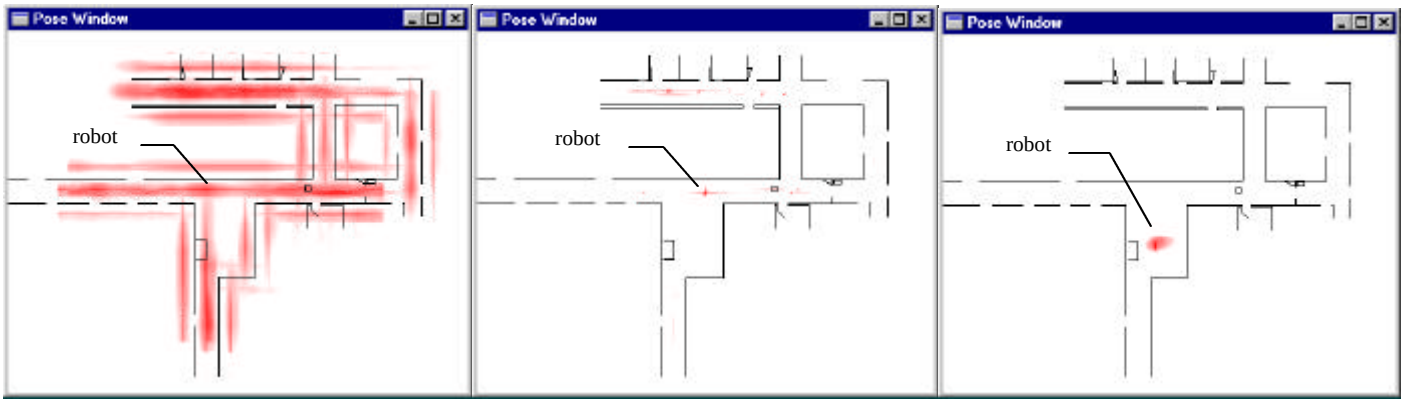

Figure 6 Global localization of the robot using simulated scans. See text for details.

Figure 6 shows a sample run of the robot. After several scans, its position is ambiguous (left). After 4 scans, the robot has seen enough of the environment to become uniquely localized (middle). Finally, its position is tracked through subsequent motion (right).

3.2 Structural Uncertainty in Maps

Because CBML does not adhere to the visibility constraint, it can represent map objects at different possible locations; we call this a *disjunctive map*. An example is the state of a door. Figure 7 shows a disjunctive map used to anticipate three possible door positions: closed, open by 30 degrees, and open by 90 degrees. The actual door position is in black, and the robot position is the cross. The ML response (middle) uses the default close-door position, while the CBML response (right) uses the disjunctive map. Note how much better localized the CBML response is.

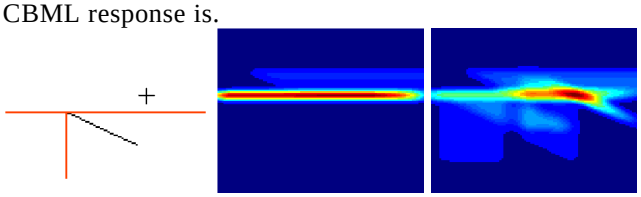

Figure 7 Map of doorway in several positions (left), and corresponding ML (middle) and CBML (right) response.

4 Conclusion

Localization based on correlation operations is an efficient alternative to standard ML updating. It is several orders of magnitude faster, and uses corresponding less storage in representing map objects. This efficiency opens CBML to applications that have been too expensive: realtime updates of robot position at video rates. It has other advantages over ML. Because it doesn't require any special pre-computations in the map, changes can be incorporated dynamically. The error model for CBML violates visibility constraints, but does allow uncertainty in the map, which is a more realistic model than typically used for ML. Finally, CBML can deal with structural uncertainty in the map, allowing disjunctive information about map objects.

CBML is a method for updating robot pose. Although our experiments have used a pose grid for representing robot locations, other more compact representations are also possible, e.g., a gaussian [Gutmann 1998, Leonard *et al.* 1990]. In this case, the probability is discretized for update, and then translated back to the gaussian form.

References

[Burgard *et al.* 1998] W. Burgard, A. Cremers, D. Fox, G. Lakemeyer, D. Hahnel, D. Schulz, W. Steiner, and S. Thrun. The interactive museum tour-guide robot. In *Proc. AAAI*, 1998.

[Burgard *et al.* 1997] W. Burgard, D. Fox, and S. Thrun. Active mobile robot localization. In *Proc. IJCAI97*.

[Gutmann *et al.* 1998] Gutmann, J-S, W.Burgard, D. Fox, and K. Konolige. Experimental comparison of localization methods. *Proc. IROS*, Victoria, B.C. 1998.

[Kaelbling *et al.* 1996] L. Kaelbling, A. Cassandra, and J. Kurien. Acting under uncertainty: Discrete bayesian models for mobile-robot navigation. In *Proc. IROS98*.

[Leonard *et al.* 1990] J. Leonard, H. Durrant-Whyte, and I. J. Cox. Dynamic map building for an autonomous mobile robot. In *IROS*, pages 89–95, 1990.

[Moravec and Elfes 1985] H. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Proc. ICRA*, pages 116–121, 1985.

[Schiele and Crowley 1994] B. Schiele and J. L. Crowley. A comparison of position estimation techniques using occupancy grids. In *Proc ICRA*, 1994.

[Schultz and Adams 1998] A. Schultz, and W. Adams. Continuous Localization using Evidence Grids. *Proc. ICRA*, Leuven, Belgium,1998.

[Thrun *et al.* 1998] S. Thrun, A. B¨ucken, W. Burgard, D. Fox, T. Fr¨ohlinghaus, D. Hennig, T. Hofmann, M. Krell, and T. Schimdt. Map learning and high-speed navigation in RHINO. In D. Kortenkamp, R. Bonasso, and R. Murphy, editors, *AI-based Mobile Robots: Case studies of successful robot systems*. MIT Press, Cambridge, MA, 1998.