IEEE International Workshop on Intelligent Robots and Systems
IROS '90

Dynamic Map Building for an Autonomous Mobile Robot*

John Leonard
Hugh Durrant-Whyte

Department of Engineering Science
University of Oxford
Parks Road, Oxford OX1 3PJ
ENGLAND

Ingemar J. Cox

NEC Research Institute
4 Independence Way
Princeton, NJ 08540
U.S.A.

Abstract

This paper presents an algorithm for autonomous map building and maintenance for a mobile robot. With each geometric target in the map we associate a *validation measure* to represent our belief in the validity of a target, in addition to the usual covariance matrix to represent spatial uncertainty. At each position update cycle, predicted features are generated for each target in the map and compared to features actually observed . Successful matches to targets with a high validation measure are used for localization. Unpredicted observations are used to initialize target tracks for new environment features, while unobserved predictions result in a target's validation measure being decreased. We describe experimental results obtained with the algorithm that demonstrate successful map-building using real sonar data.

1 Introduction

Navigation is a fundamental requirement of autonomous mobile robots. We understand navigation to be "the science of getting ships, aircraft, or spacecraft from place to place; *esp*: the method of determining position, course, and distance traveled"[1]; we do not interpret it to include motion planning. In this paper we examine the problem of constructing and maintaining a map of an autonomous vehicle's environment for the purpose of navigation. In previous papers[7][11], we examined the problem of mobile robot localization when an accurate map is provided *a priori*. We believe this problem to be well understood[8], particularly in the context of marine and aerospace navigation. However, for many autonomous vehicles, it is desirable to *automatically* build and maintain a map of the robot's environment. Maintenance of the map is, perhaps, the primary motivation because even if a map is provided *a priori*, few robot environments are completely static. Consequently, as the robot's environment alters, so must its model.

In this paper, we describe preliminary work intended to allow an autonomous robot vehicle to construct and maintain a map of its environment. One major hurdle to this goal is that the information received by the vehicle is unreliable because of the sensor's noise and physical operating characteristics. The following comments by Lozano-Pérez describe this problem [8]:

> The robot must be able to determine its relationship to the environment by sensing. There are a wide variety of sensing technologies possible: odometry, ultrasonic, infrared and laser range sensing, and monocular, binocular, and trinocular vision have all been explored. The difficulty is in interpreting this data, that is, deciding what the sensor signals tell us about the external world.
>
> The trend is to attack this problem by so-called sensor fusion, that is, by combining the outputs of multiple feature detectors possibly operating on a variety of sensors or simply multiple observations of the same object. Much of this work has focused on applications of Kalman filtering, which essentially provides a mechanism for weighting the various pieces of data based on estimates of their reliability (covariances when we assume the noise has a Gaussian distribution). This is reasonable as far as it goes. But, no amount of Kalman filtering will restore a missing feature or prevent a totally erroneous feature from having an impact on the estimate.
>
> I believe that part of the problem is that models that characterize a sensor or feature detector by a covariance matrix are too weak; they do not capture the relevant physics. The fact is that in some cases the detector will provide very good estimates in others it will be completely useless data. [18]

We broadly agree with T. Lozano-Pérez's comments. Automated cartography must deal with two distinctly different sensor problems

- the sensor noise and
- validation of sensor measurements.

The first problem is well understood. All sensor measurements have an associated accuracy that is fundamentally noise limited, e.g. thermal or shot noise at the receiver. We can usually handle noise by conventional techniques such as the Kalman filter, the corresponding uncertainty due to noise usually being represented by a covariance matrix.

The second item is often erroneously referred to as noise. However, a sensor's physical and environmental operating conditions are rarely random, but can result in missing, spurious or irrelevant information. For example, spurious information may be due to the specular reflection characteristics if ultrasonic sensing is used; irrelevant information might be caused by a person walking by the vehicle. In this case, a sensor may accurately identify the presence of an object, but the object may be of no relevance to the task at hand - for navigational cartography we are only interested in features that are unchanging.

Covariance matrices can not represent the uncertainty associated with the physical and environmental operating conditions of the sensor and the relevance of a particular sensor reading to a given problem. We are unaware of any general tool for handling such uncertainty. In fact, given that this uncertainty is very problem specific, e.g. sensing a person walking nearby is irrelevant for navigational purposes but critical to obstacle avoidance and motion planning, we think it is unlikely that a general tool can be found.

Clearly there are two uncertainties; (i) associated with the noise in the system and which is amenable to modeling using conventional statistical tools, e.g. covariances, and (ii) associated with the *validity* of a sensor measurement. These two uncertainties are independent of one another; it is entirely reasonable to have a small uncertainty due

*The authors would like to acknowledge the support of NATO collaborative research grant 0204/89. This work is also partially supported in part by ESPRIT 1560 (SKIDS) and SERC-ACME GRE/42419

to noise but a high degree of belief uncertainty — this is common for ultrasonic imaging. Thus the main issues facing us in building and maintaining a map of the environment are

1. accurately modeling the sensor noise and
2. accurately modeling the validation/belief uncertainty of a sensor reading.

A third issue is an efficient map representation, but this is not addressed here.

We refer to the task of determining the robot's position as the process of localization. Localization is a top-down, expectation-driven competence; it is a process of "looking" for and tracking *expected* events. In contrast, obstacle avoidance is a bottom-up, data-driven competence; it is a process of detecting and explaining *unexpected* events. An event is "expected" if it can be predicted from either an internal model of the environment or from previously observed events. It is possible to estimate the motion of a vehicle by observing the motion of these expected events [14]. Logically, an "unexpected" event is one that has not been predicted. Such events may arise as the result of spurious sensor readings or as the result of observing an unmodeled object. In a probabilistic sense, all sensor data is either expected or unexpected. Building maps of the environment from sensor information involves maintaining a coherent map of these expected events while explaining and incorporating new, unexpected, events.

Two important observations can be made about about this process:

- Tracking expected events and explaining unexpected events are two complimentary parts of the same problem. When an event or target is observed, predictions based on an environment model or previous observations can be used to suggest possible interpretations or explanations of the data. If a target can be explained through correspondence with prior predictions, then it may be tracked to provide positional information. The rejects from this process are those events that can not be explained. If an observation can not be explained, then a new target can be initiated and subsequently tracked to provide an explanation for the information.
- Map building is a dynamic process that involves providing an interpretation of observed sensor information in terms of physical features in the environment. The prediction of expected events relies on the fact that a correct interpretation has been found for targets that are repeatably observed. Further, tracks that are initiated to explain unexpected events are an attempt to develop such an interpretation over time. Thus the map is built up as new events are observed and explained, and is refined by reobserving events that have been correctly interpreted.

The starting point for formalizing the problem of unifying these different aspects of the navigation problem is to develop a good model of the relation between sensor observations and physical features in the environment. This sensor model is crucial to generating predictions of what can be observed from different sensor locations and correspondingly in providing an interpretation for observed events or targets. Without such a model, expectations and explanations become meaningless. The reader is directed to the appendix for a description of the feature extraction algorithms used in this paper.

Before further developing our algorithm, we briefly summarize previous work in map building. Next, Section 3 summarizes an algorithm for mobile robot localization, which utilizes the competence of tracking geometric beacons which naturally occur in the environment. This navigation algorithm is based around an extended Kalman filter which utilizes matches between observed geometric beacons and an *a priori* map of beacon locations to provide a reliable estimate of vehicle location. This approach is extended in Section 4 for situations in which the map is built up and dynamically maintained, allowing operation in unknown and changing environments. A crucial component of this extension is that we attach a measure of validation/belief uncertainty to each target in the map that is independent of the covariance matrix which represents the target's spatial uncertainty. Experimental results are presented in Section 5.

2 Previous Work in Map Building

Previous automated cartography for autonomous mobile robots has not discriminated between spatial uncertainty due to noise and validation uncertainty due to sensor and environmental conditions. Moreover, much earlier work has attempted to build maps for multiple purposes, e.g. navigation, motion planning, 3-D world modeling. Although a single map is obviously desirable, we believe that multiple maps, each containing only information relevant to a particular task, may be significantly easier to construct.

2.1 Geometric approaches

There are several research groups that have used Kalman filtering techniques to to build models of the environment. Impressive results are reported by Ayache and Faugeras[2] for a mobile vehicle using trinocular stereo to determine depth. Kriegman et al[15] also describe how a map can be built using binocular stereo. Crowley[9] describes a similar approach using ultrasonic sensing. Common to all these approaches is the use of the Kalman filter to model and propagate uncertainty in both the position of the robot and the geometric features using covariance matrices. This is a powerful tool for dealing with noise, but, as discussed earlier, is of little help in modeling the sensor/environment/task operating characteristics. The experimental results presented by these groups are all for static scenes, i.e. the environment is unchanging. It is unlikely that these approaches would be successful in a dynamic environment without extension, perhaps along the lines of the work described here.

2.2 Occupancy Grids

The occupancy grid representation developed by Moravec and Elfes [21, 12] does in fact combine a probability of occupancy (belief) with a spatial uncertainty. Occupancy grids represent space as a 2– or 3-D array of cells, each cell hold an estimate of the confidence that it is occupied. The uncertainty surrounding an objects position is then represented by a spatial distribution of these probabilities within the occupancy grid. The larger the spatial uncertainty, the greater the number of cells occupied by the feature. This representation allows a large neighborhood of cells (high spatial uncertainty) to have a high probability of occupancy (belief).

We feel the occupancy grid map representation is particularly useful for the task of obstacle avoidance[1], since there is an explicit representation of free space. Moreover, the occupancy grid representation provides a powerful tool for dealing with sensory data in which reliable feature extraction is not practical.

Ideally, position estimation should be a process of (i) determining the correspondence between observed sensory features and predicted map features and (ii) then computing an optimal estimate of the vehicle's relative position. The optimality criterion is almost always least mean square error and should weight each measurement by its spatial uncertainty. Grid-based position estimation[21] has been proposed based on the concept of matching occupancy grids via cross correlation. The accuracy of the position estimate must, however, be inferior to the feature based approach because (i) the correlation uses all points in the map, but some points will be spurious (this must reduce accuracy) and (ii) neighborhoods of cells with both a high probability of occupancy and high spatial uncertainty will have a disproportionate affect on the correlation compared with those neighborhoods that have a high probability of occupancy and small spatial uncertainty. Moreover, while the original $O(n^5)$ algorithm was sig-

[1] For example, Borenstein[5] has presented the combination of a grid-based representation with a potential field algorithm to achieve very fast obstacle avoidance.

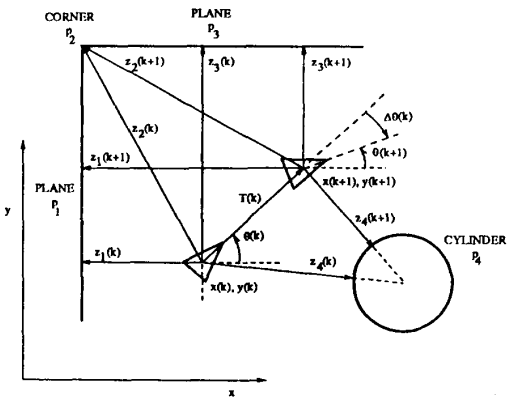

Figure 1: Localization by Tracking Geometric Beacons.

nificantly improved, the cost of matching is still slightly larger than $O(n)$, where n is the number of cells in the map. In contrast, the cost of a feature based approach should only be proportional to the number of features currently present in the map. Under normal conditions, this should be significantly less than the number of grid cells.

We believe the building of a navigational map is a distinctly different process from that used to construct occupancy grids. Map-building for navigation needs to confront the correspondence problem head-on. It is a process of making decisions: is this piece of data good or bad? does this piece of data correspond to this geometric beacon? Consequently, we believe that navigation requires a feature-based approach, in which a precise, concise map, is used to efficiently generate predictions of what the robot should "see" from a given location.

3 Localization by Tracking Geometric Beacons

This section summarizes our approach to the problem of model-based localization. We view model-based navigation as a process of tracking geometric targets. A *geometric beacon* is a special class of target which can be reliably observed in successive sensor measurements (a beacon), and which can be accurately described in terms of a concise geometric parameterization.

With reference to Figure 1, we denote the position and orientation of the vehicle at time step k by the state vector $\mathbf{x}(k) = [x(k), y(k), \theta(k)]^T$ comprising a cartesian location and a heading defined with respect to a global coordinate frame. At initialization the robot starts at a known location, and the robot has an *a priori* map of the locations of geometric beacons $\mathbf{p}_i$. At each time step, observations $\mathbf{z}_j(k)$ of these beacons are taken. Our goal in the cyclic process is to associate measurements $\mathbf{z}_j(k)$ with the correct beacon $\mathbf{p}_i$ to compute an updated estimate of vehicle position.

The Kalman filter relies on two models: a *plant* model and a *measurement model*. The plant model describes how the vehicle's position $\mathbf{x}(k)$ changes with time in response to a control input $\mathbf{u}(k)$ and a noise disturbance $\mathbf{v}(k)$

$$\mathbf{x}(k+1) = \mathbf{F}(\mathbf{x}(k), \mathbf{u}(k)) + \mathbf{v}(k), \quad \mathbf{v}(k) \sim \mathbf{N}(0, \mathbf{Q}(k)) \tag{1}$$

where $\mathbf{F}(\mathbf{x}(k), \mathbf{u}(k))$ is the (non-linear) state transition function. We use the notation $\mathbf{v}(k) \sim \mathbf{N}(0, \mathbf{Q}(k))$ to indicate that this noise source is assumed to be zero mean gaussian with variance $\mathbf{Q}(k)$ [13].

The measurement model expresses a sensor observation in terms of the vehicle position and the geometry of the beacon being observed, and has the form:

$$\mathbf{z}_j(k) = \mathbf{h}_i(\mathbf{p}_i, \mathbf{x}(k)) + \mathbf{w}_j(k), \quad \mathbf{w}(k) \sim \mathbf{N}(0, \mathbf{R}(k)) \tag{2}$$

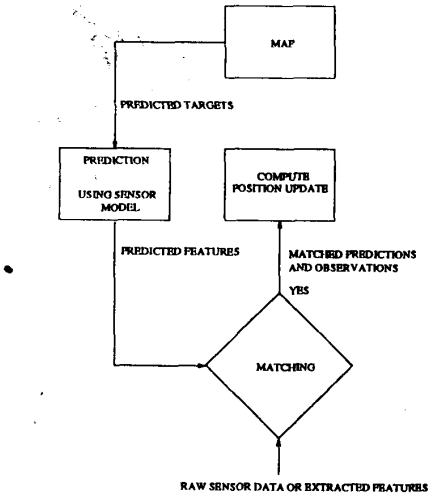

Figure 2: Model-based localization using an *a priori* map.

The observation function $\mathbf{h}_i(\mathbf{p}_i, \mathbf{x}(k))$ expresses an observed measurement $\mathbf{z}_j(k)$ as a function of the vehicle location $\mathbf{x}(k)$ and beacon location $\mathbf{p}_i$. This observation is assumed corrupted by a zero-mean, gaussian noise disturbance $\mathbf{w}_j(k)$ with variance $\mathbf{R}(k)$. The form of the observation function $\mathbf{h}_i(\cdot,\cdot)$ depends on the sensor employed *and* the type of beacon being observed. Functions for a sonar sensor observing corner, plane, and cylinder targets are described in Appendix A.

The goal of the cyclic computation is to produce an estimate of the location of the robot [2] $\hat{\mathbf{x}}(k+1 \mid k+1)$ at time step $k+1$ based on the estimate of the location $\hat{\mathbf{x}}(k \mid k)$ at time step k, the control input $\mathbf{u}(k)$ and the new beacon observations $\mathbf{z}_j(k+1)$. The algorithm employs the following steps: prediction, observation, matching, and estimation. Figure 2 presents an overview of this cyclic process. We state the Kalman Filter equations without derivation. (More detail can be found in [4], [11].)

Prediction

First, using the plant model and a knowledge of the control input $\mathbf{u}(k)$, we predict the robot's new location at time step $k+1$:

$$\hat{\mathbf{x}}(k+1 \mid k) = \mathbf{F}(\hat{\mathbf{x}}(k \mid k), \mathbf{u}(k)) \tag{3}$$

We next compute $\mathbf{P}(k+1 \mid k)$, the variance associated with this prediction:

$$\mathbf{P}(k+1 \mid k) = \nabla\mathbf{F}\ \mathbf{P}(k \mid k)\ \nabla\mathbf{F}^T + \mathbf{Q}(k) \tag{4}$$

where $\nabla\mathbf{F}$ is the Jacobian of $\mathbf{F}(\cdot,\cdot)$ obtained by linearizing about the updated state estimate $\hat{\mathbf{x}}(k \mid k)$. Next, we use this predicted robot location to generate predicted observations of each geometric beacon $\mathbf{p}_i$:

$$\hat{\mathbf{z}}_i(k+1) = \mathbf{h}_i(\mathbf{p}_i, \hat{\mathbf{x}}(k+1 \mid k)), \qquad i = 1, \cdots, N_k \tag{5}$$

Observation

The next step is to actually take a number of observations $\mathbf{z}_j(k+1)$ of these different beacons, and compare these with our predicted observations. The difference between a predicted beacon location $\hat{\mathbf{z}}_i(k+1)$ and an observation is written as

$$\begin{aligned} \nu_{ij}(k+1) &= [\mathbf{z}_j(k+1) - \hat{\mathbf{z}}_i(k+1)] \\ &= [\mathbf{z}_j(k+1) - \mathbf{h}_i(\mathbf{p}_i, \hat{\mathbf{x}}(k+1 \mid k))] \end{aligned} \tag{6}$$

[2] The term $\hat{\mathbf{x}}(i \mid j)$ should be read as "the estimate of the vector $\mathbf{x}$ at time step i given all observations up to time step j".

The vector $\nu_{ij}(k+1)$ is termed the innovation. The innovation covariance can be found by linearizing Equation 2 about the prediction, squaring, and taking expectations as

$$\begin{aligned} \mathbf{S}_{ij}(k+1) &\equiv \mathrm{E}\left[\nu_{ij}(k+1)\nu_{ij}^T(k+1)\right] \\ &= \nabla\mathbf{h}_i\ \mathbf{P}(k+1 \mid k)\ \nabla\mathbf{h}_i{}^T + \mathbf{R}_i(k+1) \end{aligned} \tag{7}$$

where the Jacobian $\nabla\mathbf{h}_i$ is evaluated at $\hat{\mathbf{x}}(k+1 \mid k)$ and $\mathbf{p}_i$.

Matching

Around each predicted measurement, we set up a validation gate in which we are prepared to accept beacon observations:

$$\nu_{ij}(k+1)\ \mathbf{S}_{ij}^{-1}(k+1)\ \nu_{ij}^T(k+1) = q_{ij} \tag{8}$$

This equation is used to test each sensor observation $\mathbf{z}_j(k+1)$ for membership in the validation gate for each predicted measurement. When a single observation falls in a validation gate, we get a successful match. Measurements which do not fall in any validation gate are simply ignored for localization. More complex data association scenarios can arise when a measurement falls in two validation regions, or when two or more measurements fall in a single validation region. At this stage, such measurements are simply ignored by the algorithm, as outlier rejection is vital for successful localization.

Estimation

The final step is to use successfully matched predictions and observations to compute $\hat{\mathbf{x}}(k+1 \mid k+1)$, the updated vehicle location estimate. We utilize the standard result [4] that the Kalman gain matrix associated with each successful match can be written as

$$\mathbf{W}_j(k+1) = \mathbf{P}(k+1 \mid k)\ \nabla\mathbf{h}_i^T\ \mathbf{S}_{ij}^{-1}(k+1), \tag{9}$$

Using all successfully matched prediction–observation pairs, we find the optimal (minimum variance) linear estimate for the state as

$$\hat{\mathbf{x}}(k+1 \mid k+1) = \hat{\mathbf{x}}(k+1 \mid k) + \sum_j \mathbf{W}_j(k+1)\nu_j(k+1) \tag{10}$$

with variance

$$\mathbf{P}(k+1 \mid k+1) = \mathbf{P}(k+1 \mid k) - \sum_j \mathbf{W}_j(k+1)\mathbf{S}_j(k+1)\mathbf{W}_j^T(k+1). \tag{11}$$

Equation 10 states that the estimated location is a linear weighted sum of the predicted location, and the difference between expected and matched observed beacon locations. The weighting depends on our relative confidence in prediction $\mathbf{P}(k+1 \mid k)$ and innovations $\mathbf{S}_j(k+1)$.

Note that this algorithm facilitates a *directed* sensing approach, in which we can use our expectations of where useful beacon information can be observed to control the sensors of the mobile robot to only "look" in these locations. By knowing where to "look", the speed and reliability of the sensing process can be greatly enhanced. From time-to-time, the localization process will predict observations which are not supported by measurements; there will also be occasions when sensor measurements are not previously predicted. In these circumstances it is desirable to update our map of the world.

4 Building and Maintaining Localization Maps

The buiding and maintaining of localization maps can be broadly considered a problem in machine learning, an area of research which has received a tremendous amount of interest. The reader is directed to [20, 10] for an overview of the subject. Broadly, learning can be divided into supervised and unsupervised learning. Supervised learning assumes an initial training set of correctly labeled examples is available. This is not true for the autonomous vehicle which must therefore learn its environment by some unsupervised means. Traditionally, unsupervised learning takes the form of a cluster analysis problem arranging features into clusters or classes based on a numerical measure of feature similarity. In our case, the features are geometric beacons. Clustering of the geometric beacons *in space* is unnecessary since the Kalman filter and validation gate methodologies have already determined which features correspond with each other.

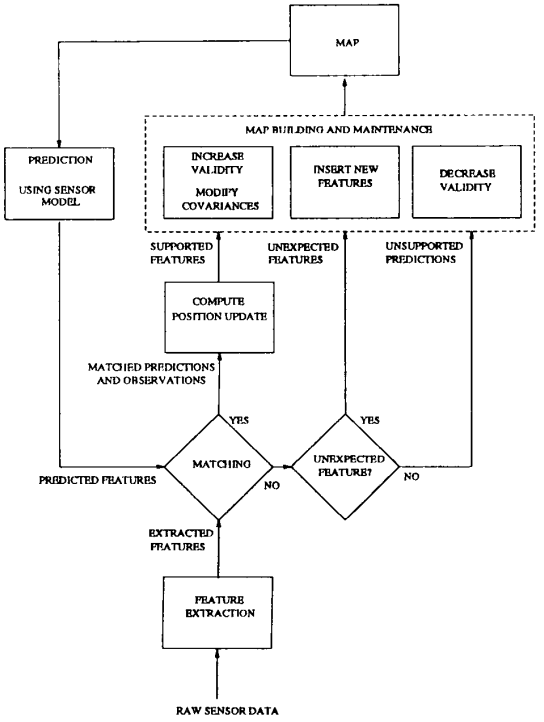

Figure 3: Model-based localization incorporating map building and maintenance.

Previous theoretical work on robot learning of an environment includes Lumelsky et al[19] and Rivest and Shapire[23]. The first is concerned with determining a path for the robot, the traversing of which would guarantee that the entire environment is viewed. Navigation is not an issue here, but rather, the development of a sensing strategy for "terrain acquisition". Rivest and Shapire examined the problem of unsupervised learning for deterministic environments. However, dynamic environments are non-deterministic.

We outline an algorithm to perform these learning functions.

4.1 Proposed Algorithm

Figure 3 shows the extension of the localization algorithm presented in section 3 to incorporate map-building and maintenance. Initialization no longer requires an *a priori* map, though initial target locations can be provided. In this way, the algorithm accommodates any level of prior information ranging from none to full. The Kalman filtering steps of **Prediction**, **Observation**, and **Matching** proceed just as before. Matching yields three sets: matched predictions and observations, unobserved predictions, and unpredicted observations. Note in the original algorithm only successful matches are processed, and the rest were considered "rejects."

Estimation is now performed only using matched predictions and observations which correspond to targets in the map which have

a high validation measure. In this way, we only use the data we have the most confidence in–data corresponding to beacons–to compute an accurate position estimate. Updates from just a few beacons can quickly provide an accurate, reliable position estimate. This should be contrasted with the position update method proposed for occupancy grids, as discussed above in Section 2.2, which utilizes all the data in the map in a cross-correlation computation to update position.

The primary modification to the algorithm is the addition of a Map Update step. Matched features, together with unexpected features and unsupported predictions, are all then sent to an explanation phase which must decide on how best to update the world model. The set of actions consists of inserting additional features, deleting existing features and increasing or decreasing the validity of existing features.

The problem of updating the navigation map can be formulated as: Given sets of predicted **P** and sensed **S** features and the intersection of the two sets **P∩S**, i.e. those features that were both predicted and sensed, how can we build and maintain a model of our environment? There are two error terms; the number of ***unsupported predictions***, $\| \mathbf{P} - \mathbf{P} \cap \mathbf{S} \|$, and the number of ***unexpected measurements***, $\| \mathbf{S} - \mathbf{P} \cap \mathbf{S} \|$. We should forget unsupported predictions and learn unexpected events, provided they persist in time. More precisely, belief in a prediction that is frequently unsupported should be reduced each time until forgotten; belief in an unexpected event should be increased if subsequent predictions of the event are regularly supported by measurements. Intuitively, in a highly dynamic environment in which there are many unmodeled, moving objects there will be many temporarily unsupported predictions and many transient unexpected events. Conversely, in an almost static environment in which there is very little change we will have few unsupported predictions and few unexpected measurements. One might expect that the rate of change of belief with respect to errors will be slower in the former case, faster in the latter case. However, if all errors are due to transient objects moving through the environment, e.g. people, we should update our belief at the same rate, the relative frequency of occurrence of these transient objects is not relevant. If, on the other hand, changes in the (almost) static environment usually represent more permanent change, e.g. the moving furniture, then the rate of changes of belief should be increased.

Precisely what the functional form of this updating should take remains unclear. A probabilistic Bayesian approach does not seem possible since no knowledge is available of the prior probabilities. Instead, we chose to increase our belief in a feature based on the number of times it is supported and decrease our belief based on the number of times it is unsupported. Obviously the asymptotic values for the belief are 0 and 1. Given a model containing a set of features, $\{f_i\}$, then if n_s is the number of supported sightings of f_i and n_u is the number of unsupported sightings of f_i and $p(f_i)$ is the validation probability/confidence of feature f_i then it seems reasonable to assume an update function of the form

$$p(f_i) = 1 - e^{-(n_s/\alpha - n_u/\beta)} \tag{12}$$

The arbitrary coefficients, α and β are dependent on the sensor and environmental operating conditions. If α and β are equal then we learn at the same rate we forget. However, it may be advantageous to have different rates for increasing and decreasing the belief in a feature. In particular, localization accuracy is unaffected by the number of unsupported predictions, they simply increase the computational requirements of the prediction and verification stages. Figure 4 summarizes the proposed algorithm.

5 Experimental Results

To test our map-building capabilities, a grid of sonar scans were taken from precisely known positions in an uncluttered office scene. Figure 5 shows triangles at each vehicle location for 28 sonar scans, and a hand-measured model of the room in which the scans were taken. The scans were processed off-line in a spiral sequence, starting from

```
for(all predicted and observed points)
{
    if observed = predicted
    {
        then increase confidence in model
    }
    else if predicted is not observed
    {
        then decrease confidence in predicted feature
        (model)
    }
    else if observed is not predicted
    {
        then insert new feature and begin
        increasing confidence
    }
}
```

Figure 4: Map building and maintenance algorithm

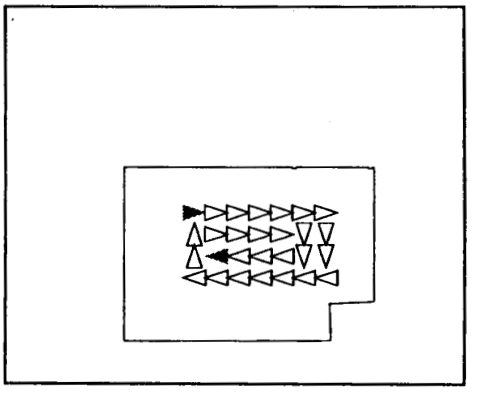

Figure 5: Hand-measured map of the room, with triangles at each of 28 locations where sonar scans were taken. The shaded triangles indicate the start and finish of the run. The room is 3 meters wide, with a closed door in the upper right hand region of the picture.

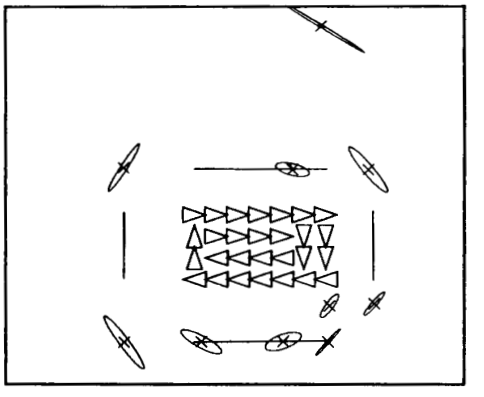

Figure 6: Localization map of the room produced by the algorithm. 8σ (magnified by 64) error ellipses are shown for point (corner) targets.

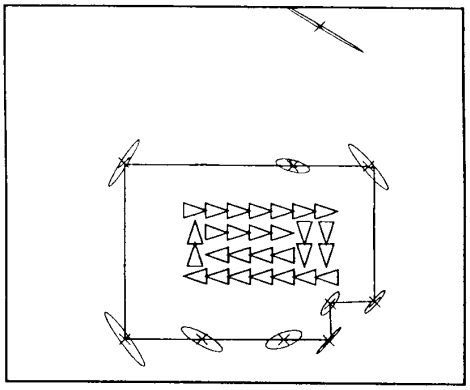

Figure 7: Learned map superimposed over room model.

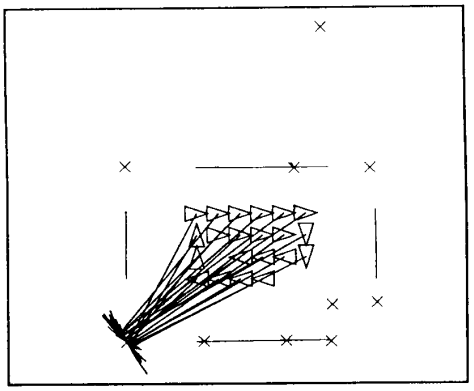

Figure 8: RCD's matched to a typical corner target.

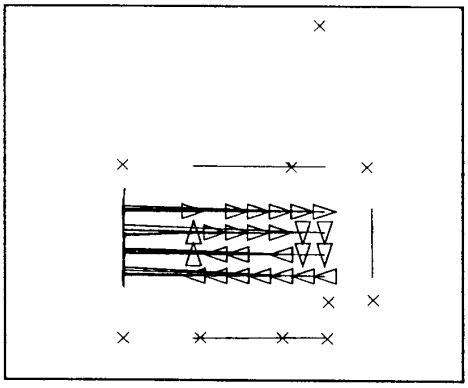

Figure 9: RCD's matched to a plane target. A triangle is shown at each location from which the target was observed. Note the absence of a triangle for scans 2 and 26–these were the scans in which a chair was inserted in the room to block this wall from view. A line is drawn from each vehicle location to the midpoint of each RCD.

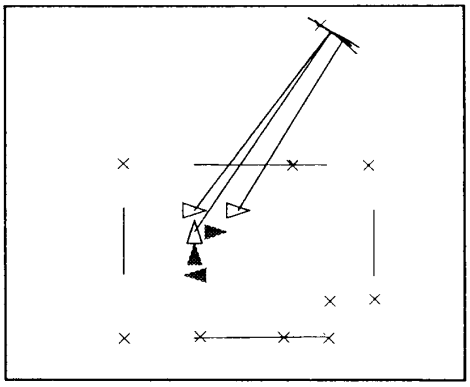

Figure 11: A corner track initiated by multiple reflection RCD's. The open triangles show locations from which the hypothesized target was observed. The shaded triangles show locations "between" the first two sightings from which the hypothesized target was not observed.

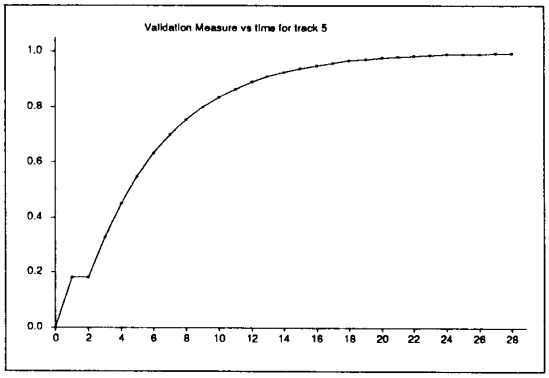

Figure 10: Validation measure vs. time for the line target shown in Figure 9. The validation probability rises exponentially as this target is repeatedly sighted, except for steps 2 and 26 when the environment was modified to occlude this target.

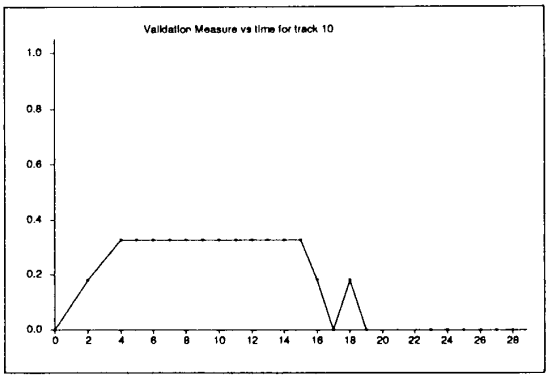

Figure 12: Validation measure vs. time for the target shown in Figure 11. The validation measure is increased for each observation (open triangles in Figure 11) and decreased for each unobserved prediction (shaded triangles in Figure 11).

the upper left of the figure. Scans 2 and 26 were taken with a changed environment, in which a chair was inserted in to the left part of the room, and a door leading into the upper right part of the room was opened, presenting two "transient" targets. For this run, α and β, the learning and forgetting factors for the validation update function (Equation 12), were both set to a value of 5.

A sensor model for the standard Polaroid ultrasonic ranging system [22] which we use in our experiments is presented in Appendix A. We wish to build a map consisting of the locations of walls and corners. The features these targets produce in sonar scans are RCD's. An RCD which cannot be matched to any beacons currently in the map causes a new track to be initiated. Because our sonar model tells us that a single RCD could be caused by either a corner, plane, cylinder, or false multiple reflection, tracks for several hypothesized geometries are initialized for a new, unexpected RCD. Through subsequent matches from different locations the unlikely hypotheses are eliminated to determine the true target geometry.

Figure 6 shows the map of line and corner beacon locations produced by the algorithm. 8σ (magnified by 64) error ellipses are displayed for each corner estimate. Figure 7 shows this map of beacon locations superimposed over the hand-measured model, revealing the close correspondence between the built-up map and the actual geometry of the room. Figures 8 to 12 show RCD's matched to a typical line and point beacons. Figure 9 shows a plot of validation probability vs. time for the target shown in figure 8. We can see that at time step 2 this target is not visible (the chair inserted into the room at step 2 occluded the left wall), and hence the validation measure is not increased at this time step.

With accurate position estimates, two or three RCD's can very accurately determine the spatial location of a target. However, this target could be caused by a multiple reflection. As discussed in Appendix A, the most crucial task in sonar data interpretation is to identify and eliminate these multiple reflections. Figure 13 shows a corner target track initiated by the algorithm for a false multiple reflection. The three open triangles show locations from which a hypothesized corner target was matched. Because of sonar's high range accuracy, this hypothesized target has a low spatial uncertainty (as shown by the magnified error ellipse in Figure 6 for this target.) However, the hypothesized target can be exposed as a false target because it can not be consistently observed from a wide range of vehicle locations. The three shaded triangles in the figure show sensing locations "between" the open triangles from which the hypothesized corner target was not observed. As a result, the validation measure was decreased at each of these time steps in accordance with Equation 12. This can be seen in Figure 14, which shows the validation probability for this target as a function of time.

This track is the only false track with three or more matches produced by the algorithm in the run shown. Thus, we feel that our sonar sensor model coupled with the validation measure presented here provides a way of overcoming false, multiple reflections to build precise maps bottom-up from sonar data.

Conclusion

We are interested in automatically building a map with which a vehicle may navigate, i.e. locate itself. Automatic cartography is important for autonomous mobile robots; costly off-line cartography is unneeded and potential obsolescence of the map is avoided for dynamic (changing) environments.

We believe such a map should be feature rather than grid based, since localization requires the prediction of expected features, a process which is ill-suited to grid-based representations. Since the map is for navigation, the stored features should be static and widely visible. We use these characteristics of geometric navigation beacons to reject those features that, for whatever reason (bad data, moving person), are spurious.

Each feature has an associated spatial uncertainty represented by a covariance matrix and an associated validation/belief uncertainty, represented as a probability. We increase the probability of a feature each time it is both predicted and seen. Features that are predicted but not sensed have their probabilities decreased. Finally, unexpected features, those that are seen but not predicted, are inserted into the map, initially with a low probability. These new features increase in probability if subsequent predictions are supported by measurements.

Our experimental results support this methodology. The results shown in section 5 present a limited form of the algorithm in which a precise knowledge of vehicle position is provided. This is easier than the general map building problem, which calls for the map to be used for localization while it is being constructed. The algorithm presented in section 4 is aimed at this general problem. None-the-less, bypassing the vehicle position update step allows us to demonstrate the map-building process.

As shown by Figure 7, very precise maps can be produced with sonar, provided one has a good sensor model. Our model tells us that sonar measurements take the form of Regions of Constant Depth (RCDs), each RCD corresponding to a different target in the environment. We group RCD's which correspond to the same target together to determine if the target is a corner, plane or higher-order multiple reflection. Observing a corner or plane target from many locations results in a high validation measure, while multiple reflections have a low validation measure, despite having a small amount of geometric uncertainty.

Currently, significant time is spent in the prediction phase, where, for an estimated vehicle position and current map, we must predict which map features should be visible. We are currently determining whether there are known algorithms for efficiently solving this problem. Such an algorithm would presumably define a representation for the map; intuitively, we would expect to do better than the present representation of unordered sets of features.

Acknowledgements

The authors thank Hans P. Moravec and Alberto Elfes for fruitful discussions.

References

[1] *Webster's Ninth New Collegiate Dictionary.* Merriam-Webster, 1985.

[2] N. Ayache and O. D. Faugeras. Maintaining representations of the environment of a mobile robot. *IEEE Trans. on Robotics and Automation*, 804–819, 1989.

[3] D. Ballard and C. Brown. *Computer Vision.* Prentice-Hall, 1982.

[4] Y. Bar-Shalom and T. E. Fortmann. *Tracking and Data Association.* Academic Press, 1988.

[5] J. Borenstein and Y. Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Trans. on Systems, Man and Cybernetics*, 19:1179–1189, September 1989.

[6] O. Bozma and R. Kuc. Building a sonar map in a specular environment using a single mobile transducer. *Trans. IEEE Pattern Analysis and Machine Intelligence*, 1989. In the Press.

[7] I. J. Cox. Blanche: position estimation for an autonomous robot vehicle. In *IEEE/RSJ Int. Workshop on Intelligent Robots and Systems*, pages 432–439, 1989.

[8] I. J. Cox and G. T. Wilfong, editors. *Autonomous Robot Vehicles.* Springer-Verlag, 1990.

[9] J. L. Crowley. World modeling and position estimation for a mobile robot using ultrasonic ranging. In *Int. Conf. Robotics and Automation*, pages 674–680, 1989.

[10] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* Wiley, 1973.

[11] H. F. Durrant-Whyte and J. J. Leonard. Navigation by correlating geometric sensor data. In *IEEE/RSJ Int. Workshop on Intelligent Robots and Systems*, 1989.

[12] A. Elfes. A tesselated probabilistic representation for spatial robot perception and navigation. In *Proc. NASA Conference on Space Telerobotics*, 1989.

[13] A. C. Gelb. *Applied Optimal Estimation.* The MIT Press, 1973.

[14] J. C. T. Hallam. *Intelligent Automatic Interpretation of Active Marine Sonar.* PhD thesis, University of Edinburgh, 1984.

[15] D. J. Kriegman, E. Triendl, and T. O. Binford. Stereo vision and navigation in building mobile robots. *IEEE Trans. on Robotics and Automation*, 5(6):792–803, 1989.

[16] R. Kuc and B. Barshan. Navigating vehicles through an unstructured environment with sonar. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1422–1426, May 1989.

[17] R. Kuc and M. W. Siegel. Physically based simulation model for acoustic sensor robot navigation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-9(6):766–778, November 1987.

[18] T. Lozano-Pérez. Foreword. In I. J. Cox and G. T. Wilfong, editors, *Autonomous Robot Vehicles*, Springer-Verlag, 1990.

[19] V. Lumelsky, S. Mukhopadhyay, and K. Sun. Sensor-based terrain acquisition: a "seed spreader" strategy. In *IEEE/RSJ Int. Workshop on Intelligent Robots and Systems*, pages 62–67, 1989.

[20] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors. *Machine Learning, An Artificial Intelligence Approach.* Tioga, 1983.

[21] H. P. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Int. Conf. Robotics and Automation*, pages 116–121, 1985.

[22] Polaroid Corporation, Commercial Battery Division. Ultrasonic ranging system. 1984.

[23] R. L. Rivest and R. E. Schapire. A new approach to unsupervised learning in deterministic environments. In *FOCS*, 1989.

A Reliable Ultrasonic Feature Extraction Using Regions of Constant Depth

Our model of sonar is inspired by the work of Kuc and Siegel [15]. Their work describes a physically-based model of sonar which considers the responses of corners, walls, and edges in a specular environment. One key conclusion from their work is that corners and walls produce responses that can not be distinguished from a single scan. The responses from these specular targets take the form of a sequence of headings over which the range value measured is *very accurate* (typically within 1 cm.) Figure 13 shows a typical, densely-sampled (612 range readings), sonar scan obtained in an uncluttered office scene. With this high sampling density, one can see that the scan is composed of sequences of headings at which the range value measured is essentially constant. We refer to such sequences as Regions of Constant Depth (RCD's).

Figure 13 shows that in a typical indoor scene, many "false" range readings are produced by the system when the beam is oriented at high angles of incidence to planar targets. At these high angles of incidence, the sound energy emitted from the side-lobes of the beam that strikes the wall perpendicularly is not of sufficient strength to exceed the threshold of the receiving circuit. As a result, the first echo detected by the system is a *multiple reflection* by some other part of the beam reflecting specularly off the wall to some other target and then back. These multiple reflections have been observed by many other researchers, and have commonly been referred to in the literature as "specularities." put a reference in here We feel this term is misleading, because Kuc's model shows that most accurate range measurements produced by planes and corners are in fact due to specular reflections.

To prevent this confusion we define the order of a range measurement as the number of surfaces the sound has reflected from before returning to the transducer. Orienting the transducer perpendicular to a planar surface such as a wall produces a 1st order range reading. Corners produce 2nd order range readings, because the sound has reflected specularly off two surfaces before returning back to the transducer. Multiple reflections will produce 3rd order and higher range readings. A crucial task in interpretation is to eliminate these higher-order reflections which, if naively taken to be the distance to the nearest object, yield false range readings.

Figure 14 shows the result of extracting RCD's visible over 10 degrees or more from the scan in Figure 13 superimposed on a line segment model of the room. Here we can see RCD's corresponding to walls, convex and concave corners, and higher-order targets.

To use 1st and 2nd order RCD's for navigation, we need to define which geometric targets in the environment produce them. 1st order RCD's principally arise from *planes* and *cylinders*. Most 2nd order RCD's arise from *corners*. Kuc and Siegel show that corners and walls will appear the same in a scan from a given location [15]. A cylinder is a 1st-order target that appears similar to a plane or corner from a single location. Higher-order multiple reflections also produce RCD's that if considered in isolation from a single location could be interpreted as any of these targets. Hence, to determine the target geometry which caused an RCD, we must observe the target from more than one location [5]. Figure 15 illustrates this process.

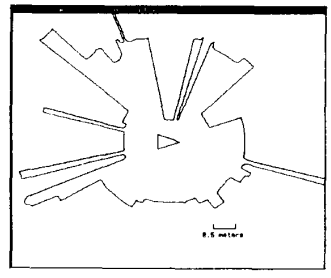

Figure 13: A typical sonar scan.

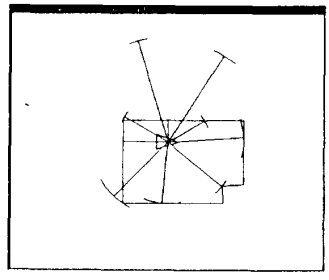

Figure 14: Regions of Constant Depth (RCD's) of width ≥ 10 degrees extracted from this sonar scan. A line is drawn from the vehicle position to the mid-point of each RCD. The order of each RCD has been written in by hand. We can see 4 1st-order RCD's from the four walls of the room, 3 2nd-order RCD's from corners, and 2 4th-order RCD's that result from reflections off the top wall into corners on the opposite side of the room. There is a single 0th-order RCD resulting from a diffuse reflection from the edge in the lower right-hand region of the room.

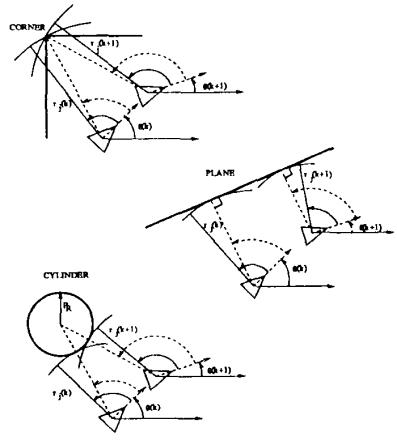

Figure 15: RCD's for corner, wall, and cylinder targets, as observed from two vehicle locations. RCD's which correspond to a plane (or cylinder) will all be tangent to the plane (or cylinder). RCD's which correspond to a corner will all intersect in a point, at the corner.