Model-Based Sonar Localisation for Mobile Robots

Bill Triggs

Oxford University Robotics Group,
19 Parks Rd, Oxford OX1 3PJ, U.K.
bill@robots.oxford.ac.uk

[0]This paper originally appeared in Proceedings of the International Workshop on Intelligent Robotic Systems, Zakopane, Poland, July '93.

Abstract

We describe a sonar localisation system for autonomous mobile robot navigation in a known environment, which tries to extract as much information as possible from the sensors by building a detailed probabilistic model of each sonar event. It takes account of multiple hypotheses about the source of each signal and uses a probabilistic sensor fusion technique to merge the results into a single location update. The system is designed to run under our decentralised, highly parallel vehicle architecture, and we discuss some of the implementation techniques required to achieve this. The results of some initial simulations are presented.

KEYWORDS: mobile robots, sonar sensing, Kalman filtering, probabilistic data fusion.

Figure 1: Oxford's 'Turtle' AGV.

1 Introduction

Several members of our team in the Robotics Group at Oxford are working on equipping the 'Turtle' — a scaled-down version of a commercially available robot truck donated by GEC-Fast — with sufficient sensing and planning capability to allow it to function autonomously in a typical factory.

The vehicle (fig. 1) came supplied with a rotating infrared scanner which allows it to localise relative to wall-mounted bar-code reflectors. We have added 12 Polaroid sonars, as many as 3 CCD cameras, and an infrared depth scanner developed at Oxford [3, 22]. Several other sensors are under development. Most of the sensor processing is Transputer based, with Datacube hardware for low-level vision. The original vehicle control system is currently being replaced with a distributed, message-passing network of Transputer (TRAM) modules [13].

To complement the sensing, there is a variety of reactive planning [14], path planning [11], task planning [21], and world modelling [24] work. The overall approach of the project is eclectic, but with a definite bias towards mathematical, model-based methods.

Ultimately, we want to build robots capable of doing useful work in busy office and factory environments. They will have to manoeuvre through aisles cluttered with obstacles and rooms filled with complex pieces of machinery, and recognise and avoid moving people and other robots. In general they will have to handle complex tasks, environments and situations gracefully, without human intervention.

Centrally controlled drones following predefined paths would not get far in such complex, uncertain, rapidly changing environments. On-board sensing is essential to detect the various situations in which the robot may find itself, appropriate responses must be planned taking into account all of the relevant constraints and interactions, and there must be a controller capable of executing the planned actions safely while handling any sudden alarms.

It is important that our robots be cheap, reliable, and robust enough to do practical work, but they need not be particularly fast or intelligent. Their primary task will simply be moving from place to place in the world without getting lost or stuck or running into things. Unfortunately, even this ‘primitive’ skill turns out to be surprisingly complex and is well beyond the scope of most current mobiles. It requires the coordination of a wide range of sensing, planning and control strategies at several levels.

The lowest levels provide basic sensor-motor control loops for trajectory following and local obstacle detection and avoidance capabilities. These layers can be ‘short-sighted’ and largely reactive, but at higher levels a more detached global perspective is essential to support overall navigation and planning.

On a large enough scale, man-made environments are typically static, structured and fairly well known. Since they are likely to be traversed repeatedly it is reasonable to build some sort of *a priori* map or model as a guide to navigation. Although such maps can be an invaluable aid, they are no substitute for on-line sensing in an uncertain environment: both sensing and map reading are essential for effective navigation.

Clearly, a map is useless unless you can locate yourself in it. This requires the identification and tracking of mapped sensor ‘landmarks’. The current paper describes a location-tracking system for mobile vehicles based on detailed modelling of the sonar environment. At best this is only a small part of a complete navigation system: there must be an obstacle avoidance layer beneath it and map updating levels above it, and in reality several sensor modalities will probably be used simultaneously.

Although sonar gives quite accurate range measurements it typically has a very poor angular resolution. This makes it hard to determine which feature of the environment is being observed. The heart of our method is a thoroughgoing *model-based, probabilistic* approach to this fundamental correspondence problem:

- Build a detailed physical model of each sensor event and extract *a priori* probabilities for the various possible event-source attributions from it.
- Combine these with the observed readings to generate *a posteriori* probabilities for the event-source attributions.
- Merge the results into a single location estimate using a probabilistic data fusion technique such as likelihood weighting.

This approach has several salient advantages:

- It has a sound mathematical and physical foundation.
- It fits very well with probabilistic sensor noise uncertainty models. These are *de rigueur* for any modern sensor system.
- It produces ‘hard’ probability estimates for the modelled possibilities. By monitoring these it should be possible to detect when the map is *not* valid and instantiate a map-correction process. Indeed, we feel that the ability to predict sensor readings from a *given* map is an essential prerequisite to any form of map-building.

The main disadvantage of this method is simply the amount of work involved in building detailed models of sensor events. In fact we have deliberately tried to push the physical, geometrical, and statistical aspects of the event model as far as possible in this work, to see what could reasonably be achieved.

The main source of inspiration and immediate precursor of our system is the work of Leonard [17], which in turn draws on the sensor modelling program of Kuc [16, 15] and the sonar map building work of Crowley [7, 8]. Contrast these approaches with the certainty-grid systems of Moravec and Elfes [9, 20]. These build probabilistic grid-maps of unstructured environments, using physically-motivated heuristic sensor models and Bayesian 'one-grid-cell-at-a-time' update rules. Although the resulting maps look promising, it is not clear how far they can be trusted: despite their formal trappings, certainty-grids are essentially a heuristic mechanism. For example, since there is no detailed event model, a deep multiply-reflected sonar reading might well cause the update rule to 'blast' a hole through a previously mapped wall. Moreover, the difficult problem of self-localisation in the grid is largely ignored, although any significant localisation error is likely to seriously 'defocus' the map.

2 Design Considerations

Our vehicle architecture is based on a particular point of view which is worth outlining here as it has deeply affected almost every aspect of the system. We can not claim that this approach is ideal, optimal, or particularly original: at best it is intended to be a first approximation to 'good engineering practice' for autonomous robot design. It shares some common ground with Brooks' subsumption program [4, 5, 6], but is more directly inspired by Hoare's theory of Communicating Sequential Processes (CSP) [12] and Minsky's notion of 'Societies of Mind' [19].

Firstly, real-time performance clearly requires a substantial amount of processing power and a great deal of software. Organising these is a major challenge.

Sensor interpretation, response planning, and actuator control may each involve a number of complex, computationally-intensive processes, with the output from each stage feeding into the input of the next. For good real-time responsiveness it is essential to keep the total delay from sensing to action as short as possible. Typically there will be several levels of sensor-motor control, with slower, higher level outer loops fine-tuning the behaviour of faster, coarser inner loops.

It is most natural to express these stages in a process-oriented 'data-flow' framework, and implement them as a decentralised network of specialised data processors linked by dedicated high-bandwidth data pathways. Each node should perform a single relatively simple data transformation with the minimum possible delay and (ideally) be implemented as a single dedicated processor.

The resulting architecture closely implements Hoare's Communicating Sequential Process model of concurrency [12]. The parallelism and decentralism arise naturally from the problem. Von Neumann machines, shared memory devices and synchronised array processors are not really appropriate here because of the need for concurrent, directed, heterogeneous, high-bandwidth data flows, although they may be useful *within* specific nodes.

In fact, we expect very heterogeneous designs to be useful. The nodes may be anything from simple finite state machines to pipelined image processors, as appropriate to the particular task. The key requirements are that they must be able to communicate with one another and they must not require access to global state. (Global state can be represented using a client/server model, but this increases the overall coupling and may produce bottlenecks).

Secondly, it is very expensive to rebuild an entire system every time a small change is made to some component. Ideally, we would like to be able to 'bolt together' a special-purpose robot from a kitset of standard hardware and software components.

This makes sense for commercial robots since there are many potential applications for autonomous mobile robots — each requiring a slightly different configuration of sensors, actuators and software — for which a purpose-built robot would not be economical.

It also makes sense for a research project. At the current state of the art we simply do not know which techniques will prove successful, so it is essential to be able to try out new ideas quickly and to avoid becoming locked in to bad design decisions. This is particularly true of architectures for autonomous robots: currently it is much easier to produce good components than good systems.

But if such reusable components are to be viable they must perform obviously useful tasks with minimal support from the surrounding system, and they must be sufficiently reconfigurable to integrate well into a variety of settings. This suggests that they should probably be self-contained modules containing dedicated hardware and software, with well-defined outputs and some degree of user-programmability. A 'smart sensor' combining a physical device with some filtering and preprocessing would be an example of such a component. Even pure software such as planning algorithms could usefully be bundled with dedicated hardware, since processing requirements will vary widely.

This kitset approach is particularly suited to our decentralised architecture. Once again, the main potential barrier to implementing it is the need to cleanly decouple the various components.

Finally, we must decide what sort of algorithms are appropriate to our decentralised setting. Reaction time is very important so it is essential to keep things fairly simple, and we want to avoid global state as much as possible. Robustness is also very important.

Any algorithm must inevitably make assumptions about its input. Sensor data is meaningless without a model of the sensing process; planning requires a knowledge of the capabilities of the vehicle and the relevant features of the situation; and some idea of the parameters of the robot is essential for effective control. In fact, all systems embody a collection of explicit or implicit *domain models*.

It is essential to keep the role of these models clear. It would be futile to try to create a single, perfect, all-encompassing model of the entire environment, which can replace the external world in all of our calculations. Such a model would be almost as complex as the real world, and extremely difficult to maintain.

Rather, each model should be a lightweight, situated, highly task-specific abstraction, which captures only the key aspects of the appropriate domain. In fact, the most effective models tend to be rather crude: simplicity, robustness, and aptness count for more than elegance or completeness. Different tasks will require different models, so we aim to build a loosely-coupled 'toolkit' of small, specialised models from which effective actions can be synthesised, rather than a single, monolithic world-model.

We take the view that since all of these models can not be avoided they should be made explicit and developed as clearly as possible, both for ease of understanding and implementation, and also so that they can be replaced if they prove inadequate. Hence, we take a firmly model-based approach, but with the caveat that we are aiming for a loose alliance of simple, robust, well-situated models, rather than a single global world-model.

To summarise the key points of our approach:

- We use a decentralised parallel network of 'data-transformers' linked by hard-wired data paths.

- The nodes come from a 'kitset' of programmable self-contained components. 'Smart' sensors and actuators come packaged with dedicated processors and software.

- Data transformations are based on small, relatively simple, purpose-built models, designed to capture the key aspects of their situation. Models are made explicit and replaced if they prove inadequate.

- The emphasis is on building complete, responsive systems: particular theories, models, architectures or components are useful only to the extent that they further this goal.

At present, only a few parts of our software conform to the above paradigm (some aspects of which only became clear in retrospect), but we are working to remedy this. Some issues are still unresolved. In particular, it is difficult to see how to deal with a world model or sensor map in this framework. If the map is held in a central server there may be access bottlenecks, while if it is distributed there are problems of consistency and update propagation. These are currently key issues in database research. At present we just keep a central map, but ultimately maps for different sensor modalities should probably be separated.

3 Sonar

Sonar has unique properties which make it very useful for robot vehicle work. The transducers are cheap, reliable, and physically robust, and they can 'see' a wide variety of objects since most solids are good sonar reflectors. Most significantly, accurate, robust range estimates are easy to extract from sonar data. These can be used for obstacle avoidance and are often difficult to obtain by other means.

Many robot vehicles have a peripheral ring of sonars to allow them to detect nearby obstacles. Such 'sonar bumpers' are easy to construct and reasonably effective in practice, but they typically use only *qualitative* range data. One would expect that cheap *quantitative* location estimates could be extracted from the readings, by tracking the signals from known sonar 'landmarks'. It might even be possible to drive a local sonar-map-building process from the data, but in any case the first stage must be localisation in a *known* environment: without this we can not hope to deal with an unknown one.

Sonar localisation turns out to be surprisingly difficult. The main source of problems is the relatively long wavelength of sound in the usable frequency range ($\sim$7mm at $\sim$50kHz). This forces any transducer of a reasonable size (a few cm) to be diffraction-limited to an extremely poor angular resolution, and also tends to produce beam patterns with significant side-lobes. For example the popular Polaroid transducer has a diameter of 2 cm and an angular resolution of about 40 degrees full width for the central lobe, or 75 if the side-lobes are included (as they should be, since they frequently produce detectable signals). This lack of resolution makes it very difficult to reliably attribute sonar readings to objects in the environment.

The long wavelength also means that any surface which is smooth on the centimetre scale behaves as a highly specular 'sonar mirror'. Such surfaces may reflect the sonar pulse away from the detector entirely, or bounce it back from an unexpected direction after further reflections. Hence, planar surfaces at oblique angles to the beam are often invisible and deep, localised, 'spurious' readings caused by multiple reflections are common, especially in man-made environments.

These features combine to make the interpretation of sonar readings very difficult, while the low speed of sound forces sampling rates to be kept quite low ($\sim$10-20 Hz for ranges up to 10 metres) and puts a premium on effective data interpretation.

A high-resolution sonar scan of an empty room is shown in fig. 2 (from [17], fig.3.2). This was taken by firing a rotating Polaroid sensor at 1 degree intervals and plotting apparent range against sensor-axis angle. Note the following typical features:

- There are many directions with large readings which do not correspond to obvious sources in the environment: these are the result of multiple reflections.

- When the sensor is pointing directly at a wall there is a clear, accurate reading which is visible

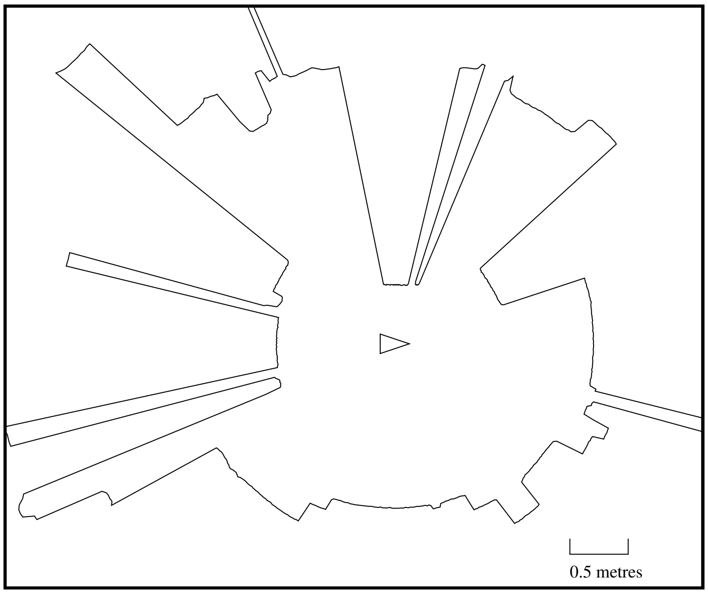

Figure 2: A typical high-resolution sonar scan (from [17]).

over a fairly well-defined range of angles. This is known as a 'region of constant depth' [17]. Both inside and outside corners produce similar regions.

- The signals from the left and top walls show the lobed structure of the beam: there is a wide interval of strong reflections when the central lobe is normal to the surface, with narrower intervals of weaker reflections on either side in which the side lobes are normally incident.
- Small features of the environment often produce strong sonar signals. For example the almost-invisible door jamb towards the right end of the top wall produces a very clear signal over a wide range of angles. Electrical sockets, mouldings, wiring conduits, and even joints between wall panels can all produce significant readings [17]. This makes the *a priori* modelling of sonar environments rather difficult. Visual intuition is a poor guide to the sonar environment.
- The above scan represents at least 20 seconds worth of data. For real-time response on a moving robot it is not possible to sample this finely. In fact we will be lucky to have even one sample from any given region of constant depth, so we must be prepared to work with isolated samples rather than coherent blocks of readings. This makes the task of interpretation even more difficult. (One promising solution to this is to use stereo pairs of transducers which fixate and actively track the most salient landmarks [18, 2]).

In fact navigating with sonar is a bit like trying to find your way around a darkened hall-of-mirrors with a wide-beam torch and a light meter. Bats certainly manage it, but their sonar processing capabilities are comparable in complexity to mammalian vision [23] and far better than any current artificial system.

4 Sonar Localisation

To simplify the interpretation problem it is sensible to do sonar localisation incrementally, making small updates to an existing position estimate. Localisation can be then thought of as a parameter-tracking problem to which standard Kalman filter techniques can be applied [10, 1]. This approach is reasonably effective, but it relies heavily on having a good initial location estimate to prime the process and it does make the system susceptible to getting lost through mistracking.

The most difficult part of incremental sonar localisation is identifying the source of an observed signal. Once this has been done it is fairly simple to compare observed and predicted ranges and derive a location update. The problem is the poor angular resolution of sonar: we would often be unsure of the true source of an event even if we knew the external world and the sensor location and calibration exactly.

Unfortunately, location estimates are typically very sensitive to the event-source attribution and a few misattributions are usually enough confuse a localisation system. This is particularly true of Kalman filter based systems. These are very effective at tracking moving parameters so long as their hypotheses are met, but they are extremely susceptible to outliers and systematic model deviations because the Kalman update rule depends critically on the assumed (known, independent, Gaussian) error distributions.

Robust localisation therefore requires very careful moderation of the event-attribution mechanism. We claim that the best way to do this is by thorough statistical modelling of the physical situation and the sensing process, allowing multiple hypotheses about the source of the observed signal and with an eye to capturing the most likely sources of error and uncertainty. The aim is to derive 'hard' probabilities for each hypothetical attribution, and to merge these into a single 'best' location update using a probabilistic data fusion technique.

This approach has the attraction of a sound mathematical foundation, and luckily it turns out to be practically feasible for typical sonar data. Moreover, since each sonar event is modelled in detail, unexpected, missing, and deviant readings (indicating objects added to, removed from, or moved in the environment) can be caught and sent to a map-updating process.

In outline, the strategy is to build a list of the things a sensor *might* see given the current world-model, vehicle state, and sensor configuration. This is refined into a probabilistic model of what the sensor *should* see by detailed geometric and statistical modelling of the sensing event (fig. 3). The result is a list of predicted sensor readings annotated with *a priori* (model-based) probabilities for the readings to occur. These predictions are compared with what the sensor *does* see to build a list of location update terms annotated with estimated *a posteriori* probabilities for them to be applicable (ie, for the object-signal matching to be correct). The corrections are weighted by their posterior probabilities, merged into a single location update using 'probabilistic data association' [1], and applied to the vehicle state estimate through a Kalman filter.

5 The Implementation

In more detail, the principal stages in the processing of a sonar event are as follows (fig. 4):

Sensor preprocessing: This converts the raw sensor output to an estimate of apparent range and its expected uncertainty, using sensor calibration and noise models.

Probabilistic modelling of sonar event: A detailed *a priori* model of the sonar event is built using the current world model, vehicle state estimate and sonar sensor model, and prior probabilities for the various possible occurrences are extracted from it. This process is logically independent of the sensor observation, however for efficiency the observed range is used for coarse search pruning. The stages of modelling are as follows:

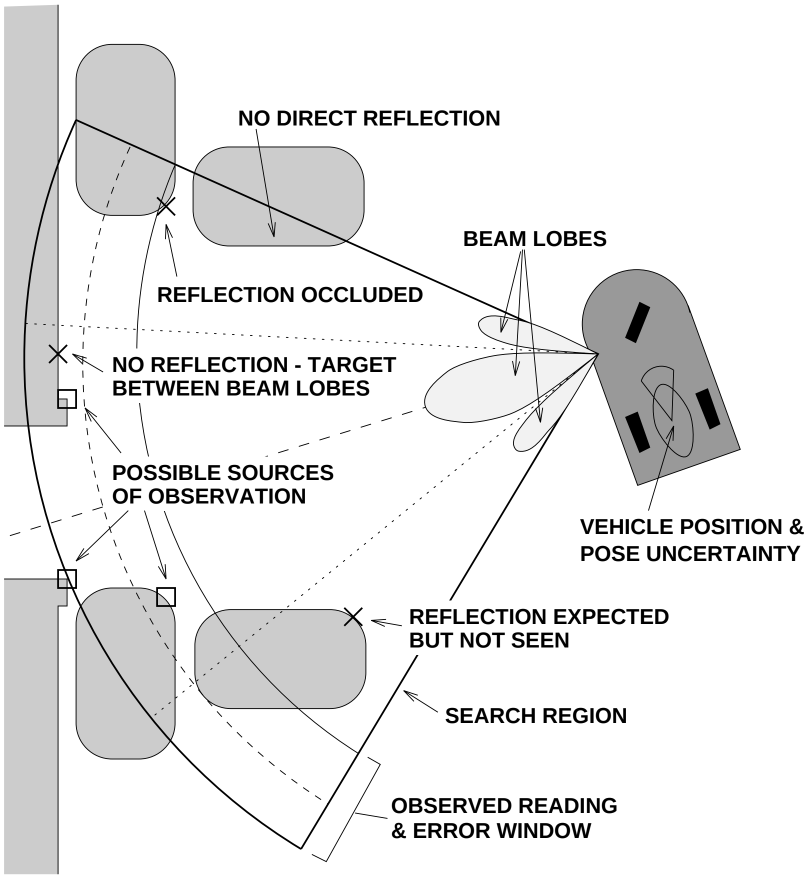

Figure 3: Modelling the possible sources of a sonar event.

- **Beam geometry:** Using the sonar sensor and vehicle state models and the observed reading a sector-shaped 'illuminated' region is found which safely encompasses all sonar reflectors which might plausibly have affected or contributed to the observed event.

- **Illuminated object search:** The environment model is searched for reflectors ('targets') in the illuminated region, and these are classified according to their geometry relative to the sensor. Illuminated targets need not form a specular 'image' visible to the detector, and those that do need not be at a suitable range to cause the observed signal. However *all* illuminated targets are remembered for later use.

- **Detectability of visible images:** The expected reflection strength from each visible target is estimated using beam-intensity and target-reflectivity models, and the probability that it causes a *detectable* reflection is evaluated. Weak reflections and reflections which would fall in the dead spot between the beam-lobes are eliminated.

- **Geometric occlusion test:** Images which are likely to be occluded are eliminated by searching the list of illuminated targets for objects lying between the image and the sensor.

- **Statistical model of detection:** The above estimates of signal strength and range are collated to predict the *a priori* detector response. This depends on internal details of the sensor. For example the Polaroid detector 'fires' on the first above-threshold signal to arrive and ignores all later signals, so the output is a list of *a priori* probabilities for the various sources to have fired the detector.

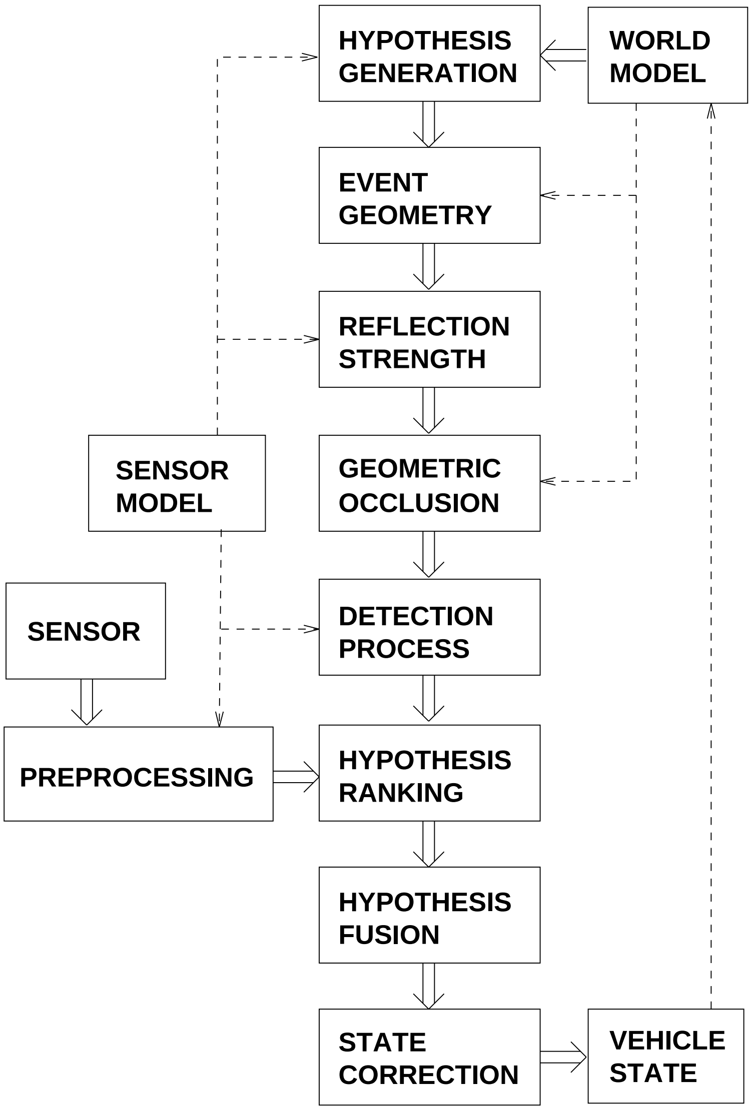

Figure 4: Stages of sonar processing.

Data fusion: This compares the observed sonar data with the model-based predictions to produce a probabilistic state update, as follows:

- **Probabilistic signal-source assignment:** The posterior probability for each target to be the source of the observed signals is evaluated, using Bayes' rule to combine the model-based prior with the conditional probability for the observations to occur given that the target was their cause. The conditionals are estimated from the observed readings, the sensor location estimate, and a sensor noise model.

- **Merging of event-source hypotheses:** Location corrections are derived for each of the hypothesised target-event matchings, weighted by their posterior probabilities, and merged into a single correction as in the 'probabilistic data association' method [1].

- **Vehicle state update:** The vehicle's dynamic state is currently modelled by position, orientation, speed and path curvature. State uncertainty is described by the corresponding covariance matrix and state corrections are applied through a standard Kalman filter. Sonar information feeds mostly into the position and orientation components, while odometry and steering data dominate the speed and path curvature estimates.

The observed reading is used to guide the prediction process, but this is solely for convenient search pruning and is not an essential part of the technique: the final result would not be significantly different if the search region were infinite. Contrast this with 'gating' methods which aggressively prune the search to targets likely to be the direct cause of the observed signal: essential context is lost, so that effects such as occlusion, multiple reflection, and the likelihood of a closer object firing the detector can not be modelled.

Some important simplifying assumptions are made:

Each sonar event is processed independently. In fact, we make a blanket assumption of independence and ignore all statistical correlations unless they are easily modelled and seem especially likely to be important.

Several heuristic parameters are introduced, describing beam geometry, detector thresholds, and cut-offs for unlikely and over-complex events.

Sonar events are modelled in two-and-a-half dimensions. 2D modelling is not sufficient for accurate localisation because the sonar beam has a considerable vertical spread and 3D distances can differ significantly from 2D ones. We assume that sensor axes will be horizontal and model objects as vertically-extruded prisms with planar or cylindrical-section sides. For objects above or below the detector axis, the range is assumed to be the distance to the nearest point on the object at the 2D image position. This may seem a crude approximation but it simplifies the geometry enormously, makes model-specification much easier, and still covers the majority of practically common cases.

We do not attempt to model secondary and higher order reflections, except for inside-right-angle-corners which have a characteristic and reliable signature. Such reflections are typically very localised and sensitive to model details, so they would be impossible to predict reliably. However they are a ubiquitous feature of the sonar landscape, so we include several heuristics to lessen their impact:

- Regions of 'sonar clutter' may be designated. If one of these is illuminated a reliable event-source attribution is assumed to be impossible and the sonar event is discarded.

- Event probabilities can be depreciated proportionally to the number of 'illuminated' objects at a shorter range. This ensures that when secondary reflections do occur they have a reduced weight.

- A uniform spatial distribution of unmodelled 'spurious' reflectors is hypothesised and probabilities are depreciated proportionally to the likelihood that the signal is spurious (which grows as the illuminated volume, ie $(\text{range})^3$). Thus, distant signals are discounted quite rapidly.

The software is written in C and is carefully modularised so that detector, environment, and vehicle models can easily be altered. As would be expected the process is rather floating-point intensive for a sonar algorithm, although not prohibitively so. Processing a single event in a fairly cluttered environment can take as much as 5-10ms (perhaps $10^3 - 10^4$ floating point operations) on a 12 MIPS Sparc 1 workstation. This may improve somewhat when the code is optimised.

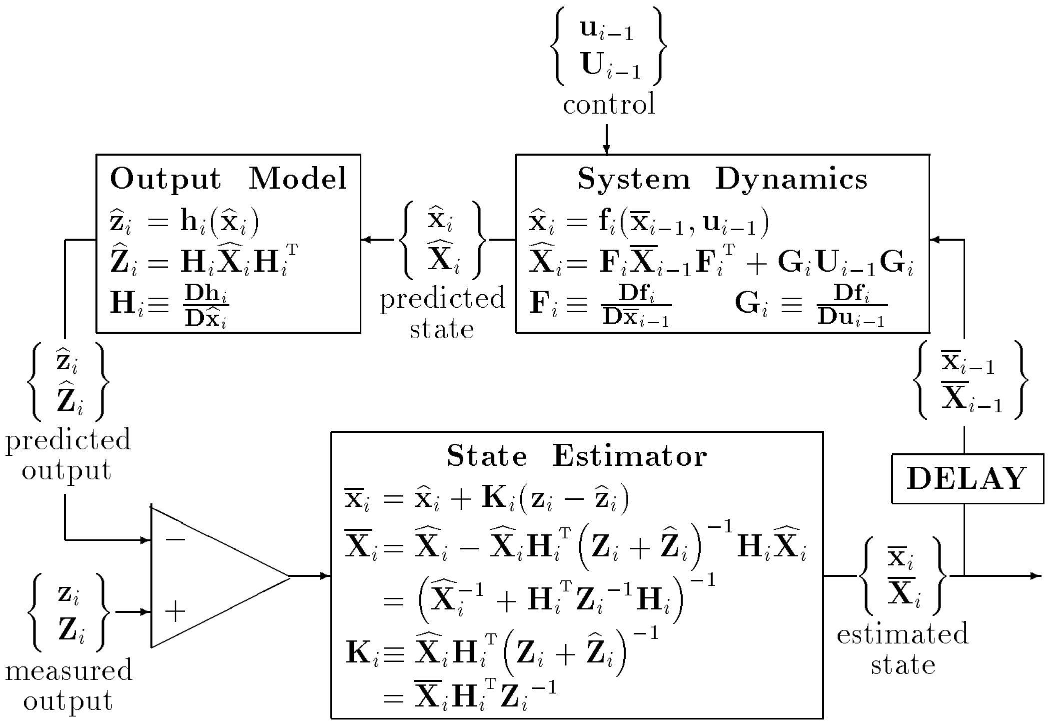

Figure 5: The Extended Kalman Filter equations.

6 Decentralised Sensor Fusion

In this section we outline the mapping of the vehicle state Kalman filter into our decentralised architecture and give details of the 'Probabilistic Data Association' technique used to fuse alternative position corrections into a single state update.

We will assume familiarity with the basic Extended Kalman Filter (EKF) loop summarised in fig 5. Briefly, the system state is predicted from a previous state estimate using the dynamical equations for the system. From this the expected output (observation) is predicted and compared to the measured output. The difference signal is used to refine the state estimate using a linearised optimal (maximum likelihood) estimator. In our notation $\mathbf{u}$, $\mathbf{x}$, and $\mathbf{z}$ stand for control-, state-, and observation-space variables; $\mathbf{U}$, $\mathbf{X}$, $\mathbf{Z}$, are the corresponding covariance matrices; subscript i indicates the time t_i at which the value applies; a bar $\overline{\mathbf{x}}_i$ indicates an estimated value at t_i (ie, conditioned on all measurements up to and including t_i); and a circumflex $\widehat{\mathbf{x}}_i$ indicates an *a priori* value predicted from the previous state estimate. We have given both 'covariance' and 'information' forms for the state-covariance-update ($\overline{\mathbf{X}}_i$) and Kalman-gain ($\mathbf{K}_i$) equations. For mechanical systems it is often reasonable to assume that dynamical noise feeds into the state through the controlled variables (as in fig 5), but more general noise models are possible.

The main problems with a monolithic implementation of the EKF for sensor fusion are that it has to know too much and do too much. Details of the system state and dynamics, the sensor model, and the update equations are all mixed up together, and each stage of the process is typically rather computationally intensive. We would like to hide details of the system state model from the sensor code and vice versa, so that we can change either without affecting the other, and also spread the computational load. Separating the components in this way produces a cleaner, more flexible, more scalable system.

In practice the separation is straightforward. A given sensor typically uses only part of the state, for example sonar readings depend only on the sensor's position, not on its speed or acceleration or on any non-dynamical state variables. In fact, the output model $\widehat{\mathbf{z}}_i = \mathbf{h}_i(\widehat{\mathbf{x}}_i)$ can usually be split into a chain of projections and coordinate transformations ($\mathbf{h}_i = \mathbf{h}_i^k \circ \ldots \circ \mathbf{h}_i^0$, $\mathbf{H}_i = \mathbf{H}_i^k \cdot \ldots \cdot \mathbf{H}_i^0$). For sonar, we project the state to a vehicle position estimate, use this to predict the sensor location, and finally estimate the expected sensor readings from this and the sonar map. The results are compared to the observed reading and must then be pushed back through the chain of transformations (along with the associated covariance matrices) to produce a state correction.

In our current design, sensor readings are made asynchronously and merged into the state as they occur. A 'dynamics server' provides a new state prediction each time a reading occurs, and this is sent through the chain of output transformations to observation-predicting and sensor-location-correcting nodes associated with each sensor. The resulting corrections are sent back down the chain to a central 'state updater', which outputs the improved state estimate. The sensors essentially provide corrections to their own positions. They use the given sonar map for predictions, but they do not know that they are mounted on a vehicle or have any details of the vehicle state model. The dynamics server knows only about the vehicle state and dynamics, and the state updater simply accepts anonymous state corrections and merges them with the current state: neither of these know about the world or sensor models.

Actually, our system is slightly more complicated than this because we use the Probabilistic Data Association (PDA) method to allow for and merge alternative interpretations of the sensor reading. PDA works as follows:

When a sensor returns a reading $\mathbf{z}_i$ we make an estimate $\mathbf{Z}_i$ of the possible error from a sensor noise model. From our current model of the situation we extract a list of *a priori* possible causes of sensor signals $\{\widehat{\mathbf{z}}_i^\alpha, \widehat{\mathbf{Z}}_i^\alpha, p_\alpha\}$. Here, α labels the sources that *may* produce a signal, p_α is the *a priori* probability that a source *will* produce a signal (given everything we know about the situation and the sensor dynamics, but *not* the observed sensor reading), and $\widehat{\mathbf{z}}_i^\alpha$ and $\widehat{\mathbf{Z}}_i^\alpha$ are the expected reading and covariance assuming that the signal *does* come from α.

From the list of priors we can use Bayes' rule to calculate the *a posteriori* probability for each α to have been the source of the observation given the observed reading:

$$\mathrm{P}[\alpha|\mathbf{z}_i] = \text{Normalisation} \cdot \mathrm{P}[\mathbf{z}_i|\alpha]\ \mathrm{P}[\alpha]$$

Here, $\mathrm{P}[\alpha]$ is the model-based prior p_α and $\mathrm{P}[\mathbf{z}_i|\alpha]$ is the likelihood of the observed sensor reading given that it came from α, ie a Gaussian$(\widehat{\mathbf{z}}_i^\alpha,\ \widehat{\mathbf{Z}}_i^\alpha + \mathbf{Z}_i)$ distribution sampled at $\mathbf{z}_i$.

If we were certain that the observed signal came from some source α, we could just use the usual EKF equations to merge the observation into the state estimate. Instead, we can try to hedge over the various alternative interpretations α by taking a weighted sum of the possible state updates. The weighting factors can be chosen to be the posterior probabilities $P_\alpha \equiv \mathrm{P}[\alpha|\mathbf{z}_i]$, so that more probable interpretations have a higher weight. This convenient heuristic is known as Probabilistic Data Association [1]. A more 'correct' approach to event-source uncertainty might be to postpone the source attribution and maintain the alternatives as separate hypotheses, but this leads to an exponentially growing hypothesis tree. Such 'Multiple Hypothesis Tracking' is very laborious and to some extent it only serves to postpone the attribution decision, so we have preferred the simpler PDA method. Ultimately the test of any such technique is how well it performs in practice.

In more detail, under PDA we produce a list of alternative updated state values $\{\overline{\mathbf{x}}_i^\alpha, \overline{\mathbf{X}}_i^\alpha, P_\alpha\}$, where $\overline{\mathbf{x}}_i^\alpha$ is the output of the standard EKF algorithm with the event-source attribution α and P_α is the posterior probability of the attribution α being correct. We form a single random variable $\boldsymbol{x}$

whose distribution is the sum of the $\overline{\mathbf{x}}_i^\alpha$ distributions $\mathrm{P}[\boldsymbol{x}] = \sum_\alpha P_\alpha$ Gaussian$(\overline{\mathbf{x}}_i^\alpha,\ \overline{\mathbf{X}}_i^\alpha)(x)$. The state estimates follow from the linearity of expectation and $\langle \boldsymbol{x}^2 \rangle = \langle \boldsymbol{x} \rangle^2 +$ Covariance$(\boldsymbol{x})$:

$$
\begin{aligned}
\overline{\mathbf{x}}_i &\equiv \langle \boldsymbol{x} \rangle \;=\; \sum_\alpha P_\alpha \overline{\mathbf{x}}_i^\alpha \\
\overline{\mathbf{X}}_i &\equiv \langle \boldsymbol{x}^2 \rangle - \langle \boldsymbol{x} \rangle^2 \\
&= \sum_\alpha P_\alpha \langle \boldsymbol{x}^2 \rangle_\alpha - \langle \boldsymbol{x} \rangle^2 \\
&= \sum_\alpha P_\alpha \left(\overline{\mathbf{X}}_i^\alpha + \overline{\mathbf{x}}_i^{\alpha 2} \right) - \overline{\mathbf{x}}_i^2
\end{aligned}
$$

It is also wise to include an ‘escape clause’ — a broad hypothesis of ‘unexplained sensor reading’ — in the list of possible explanations for a sensor event. This may be given a low prior probability, but unexplainedness is likely to become the dominant hypothesis if the world model is incorrect or there is a sensor failure. This prevents too much trust being put in a single, possibly erroneous reading, and hence increases the overall robustness of the system. The net effect of this hypothesis is simply to add a term involving the raw state prediction $\{\widehat{\mathbf{x}}_i, \widehat{\mathbf{X}}_i\}$ into the PDA sum, which effectively dilutes the sensor-based updates by some reliability factor. Of course, it is also important to monitor such unexplained readings as they give us important information for map-building.

As they stand, the PDA equations require the calculation of several alternative updates to the state before the hypotheses can be merged. This is computationally expensive and requires the state estimation software to know about all of the alternatives. It would be more efficient to merge the hypotheses nearer to the sensor. This is straightforward using the covariance form of the update equations.

As far as the PDA equations are concerned, the predicted state and covariance are just constants (independent of the source-attribution hypothesis), and some part of the output chain $\mathbf{H}_i^k \cdots \mathbf{H}_i^0$ is typically constant too. For sonar, the mappings from vehicle state to vehicle location to sensor location are independent of the object seen, although the predicted range does of course depend on the target as well as the sensor location.

Hence, we can use the linearity of expectation values to push the PDA calculation back through the output chain to the sensor location stage, but no further. This yields a sensor which can combine multiple hypotheses about its readings into a single update of its own location $\mathbf{v}$, and then send a single combined correction back to the vehicle state updater. (If sonar readings provided useful angular information, the sonar state $\mathbf{v}$ would also have to include the sensor orientation).

It is simpler to work in the sensor's ‘information space’ $\overline{\mathbf{y}}_i \equiv \overline{\mathbf{V}}_i^{-1} \overline{\mathbf{v}}_i$ rather than its state space. Decomposing the output mapping into vehicle-state to sensor-state ($\mathbf{h}_i^0$) and hypothesis-specific ($\mathbf{h}_i^\alpha$) components gives:

$$
\begin{aligned}
\widehat{\mathbf{v}}_i &= \mathbf{h}_i^0(\widehat{\mathbf{x}}_i) & \qquad \widehat{\mathbf{V}}_i &= \mathbf{H}_i^0\, \widehat{\mathbf{X}}_i\, \mathbf{H}_i^{0\mathrm{T}} \\
\widehat{\mathbf{z}}_i^\alpha &= \mathbf{h}_i^\alpha(\widehat{\mathbf{v}}_i) & \qquad \widehat{\mathbf{Z}}_i^\alpha &= \mathbf{H}_i^\alpha\, \widehat{\mathbf{V}}_i\, \mathbf{H}_i^{\alpha\mathrm{T}}
\end{aligned}
$$

We then accumulate the following correction terms:

$$
\begin{aligned}
\triangle\overline{\mathbf{y}}_i^\alpha &\equiv \mathbf{H}_i^{\alpha\mathrm{T}} \Big(\mathbf{Z}_i + \widehat{\mathbf{Z}}_i^\alpha \Big)^{-1} (\mathbf{z}_i - \widehat{\mathbf{z}}_i^\alpha) & \qquad \triangle\overline{\mathbf{Y}}_i^\alpha &\equiv \mathbf{H}_i^{\alpha\mathrm{T}} \Big(\mathbf{Z}_i + \widehat{\mathbf{Z}}_i^\alpha \Big)^{-1} \mathbf{H}_i^\alpha \\
\triangle\overline{\mathbf{y}}_i &\equiv \textstyle\sum_\alpha P_\alpha \triangle\overline{\mathbf{y}}_i^\alpha & \qquad \triangle\overline{\mathbf{Y}}_i &\equiv \textstyle\sum_\alpha P_\alpha \left(\triangle\overline{\mathbf{Y}}_i^\alpha - (\triangle\overline{\mathbf{y}}_i^\alpha)^2 \right) + (\triangle\overline{\mathbf{y}}_i)^2
\end{aligned}
$$

The accumulated location updates can then be applied directly to the sensor-state, or forwarded to the vehicle state updater:

$$\begin{aligned} \overline{\mathbf{v}}_i &= \widehat{\mathbf{v}}_i + \widehat{\mathbf{V}}_i \,\triangle\overline{\mathbf{y}}_i & \qquad \overline{\mathbf{V}}_i &= \widehat{\mathbf{V}}_i - \widehat{\mathbf{V}}_i \,\triangle\overline{\mathbf{Y}}_i \,\widehat{\mathbf{V}}_i \\ \overline{\mathbf{x}}_i &= \widehat{\mathbf{x}}_i + \widehat{\mathbf{X}}_i \,\mathbf{H}_i^{0^{\mathrm{T}}} \triangle\overline{\mathbf{y}}_i & \qquad \overline{\mathbf{X}}_i &= \widehat{\mathbf{X}}_i - \widehat{\mathbf{X}}_i \,\mathbf{H}_i^{0^{\mathrm{T}}} \triangle\overline{\mathbf{Y}}_i \,\mathbf{H}_i^0 \,\widehat{\mathbf{X}}_i \end{aligned}$$

The result of all this is identical to the conventional PDA algorithm applied to the sensor or vehicle states.

7 Experiments and Simulations

The localisation software has been tested extensively in simulation. Live experiments using our GEC vehicle were in progress, but have been delayed by a series of wiring and hardware problems, followed by a decision to rewire the entire vehicle. We hope to have the vehicle back on the road within a couple of months.

The simulations use a 'two model' system: two software models are run side by side, one representing the *physical* robot and environment, the other the robot's current *model* of its state and environment. The 'model' software has to track and control the badly behaved 'physical' robot, using only its own imperfect estimate of the physical robot's dynamics and environment. The 'physical' robot responds to external control signals imperfectly according to its (noisy) dynamical model, and generates noisy synthetic sensor readings by backward-engineering its state, sensor calibration and noise models. These readings are fed into the 'model' robot's software and processed as usual, the 'model' state estimates are updated, and control signals are issued which are fed back to the 'physical' robot.

This process is clearly no substitute for real-world experiments, but it does provide an effective, controlled way of studying mismatch between the physical world and the model. In particular it allows us to investigate the causes of mistracking in the localisation system and the effects of poor model-parameter estimation. Although quite a lot of code is currently shared between the models, the various parameters and the environment models usually differ.

Fig. 6 shows part of a typical run of our sonar simulation software in a model of the vehicle's real environment. The sources of accepted readings are marked, and the physical vehicle position (cross) and model vehicle position (outline of vehicle) and uncertainty (error ellipse) are plotted. The empty closed polygons represent regions of clutter: the system ignores any reading which may have come from these. The robot starts at the left with a slightly incorrect estimate of its position. Because of the clutter it receives no usable readings until it 'locks on' to the first pillar, but from then on it locates itself successfully and makes continual control adjustments to keep itself on its predefined path around the pillar, despite dynamical noise and occasional spurious sensor readings.

Overall the software performs well in simulation, although there are still several outstanding problems:

- The most serious omission is the lack of a model building capability. Repeated observations of unmodelled objects produce correlated errors which can confuse the current system. This should be less of a problem when our model-update-hypothesiser is in place.
- As with all incremental localisation systems a reasonable initial state estimate is required to prime the system and mistracking is possible, especially in a cluttered environment. It would be useful to add a 'location guessing' mode for use when the system was lost [17]. This might

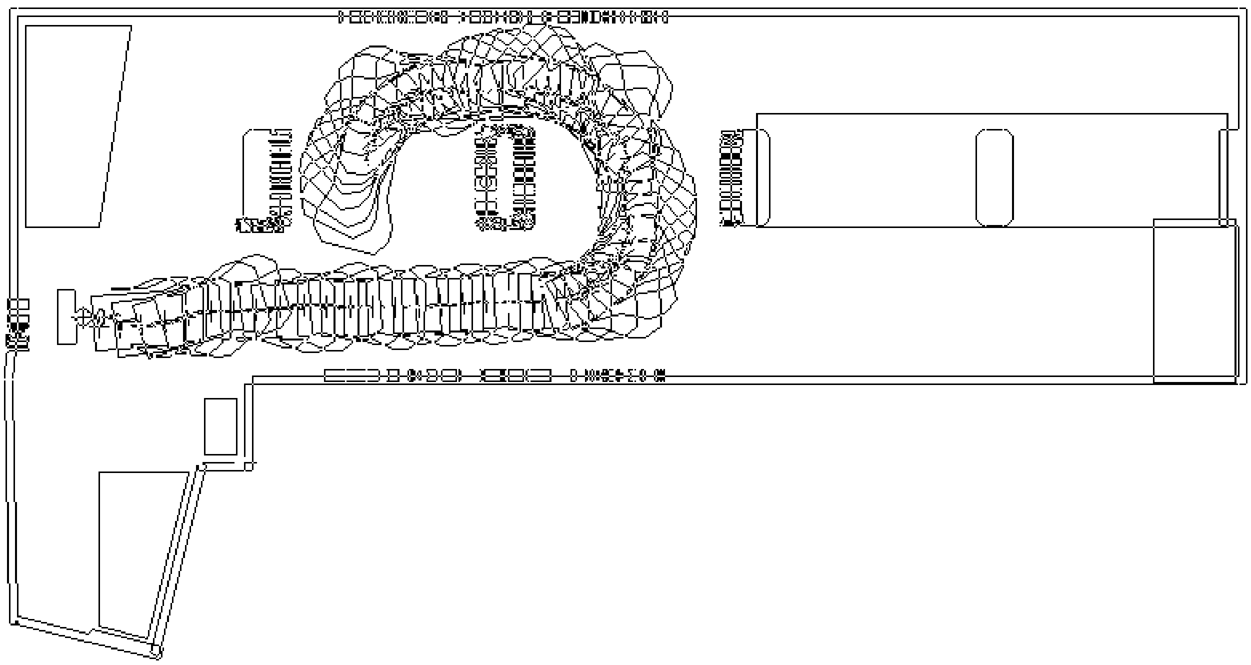

Figure 6: A simulated sonar run.

make a fine sonar scan, extract parameters such as apparent inter-wall spacing, index into the world model using these, and finally verify the resulting location hypotheses by more detailed model matching.

- The performance is still too sensitive to mismatch between model and physical parameters. We are considering using adaptive parameter estimation to overcome this.

- We do not currently allow for uncertainties in the positions of external objects. These would be fairly easy to incorporate into the localisation system, but it is less clear how to represent uncertainty effectively in a full-scale world model. Realistic uncertain models tend to have many local *constraints* which must be maintained across updates, and location updates typically produce *correlations* between the various model parameters because they are defined relative to other model features. Both of these interactions are highly significant locally, but if unchecked they tend to propagate and the model becomes an intractable tangle of inter-relations.

8 Summary

Free-ranging robot vehicles have great potential as tools for intelligent automation, but they need to become considerably more flexible if they are to see wider use. In particular, they must learn to find their way around a busy workplace environment without getting lost or running into things. We have described progress towards a vehicle localisation system based on detailed physical and statistical modelling of the sonar sensing process, which promises to provide one substantial piece of this capability.

Acknowledgements

This work was supported by the SERC ACME Directorate, under grant GR/E62776, and is based on a vehicle donated by GEC-FAST. The author is currently supported by the E.C. Esprit Basic Research Action SECOND.

References

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

[2] B. Barshan and R. Kuc. ROBAT: A sonar-based mobile robot for bat-like prey capture. In *IEEE Int. Conf. Robotics & Automation*, pages 274–279, 1992.

[3] J M Brady and H Wang. Vision for mobile robots. *Proc. Roy. Soc. B*, 1992.

[4] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE J. Robotics and Automation*, RA-2:14–23, April 1986.

[5] R. A. Brooks. Challenges for complete creature architectures. In *From Animals to Animats: First Int. Conf. Simulation of Adaptive Behaviour*, pages 434–443. MIT Press, 1990.

[6] R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–160, 1991.

[7] J. L. Crowley. Navigation for an intelligent mobile robot. *IEEE J. Robotics and Automation*, pages 31–41, 1985.

[8] J. L. Crowley. World modelling and position estimation for a mobile robot using ultrasonic ranging. In *IEEE Int. Conf. Robotics & Automation*, pages 674–80, 1989.

[9] A. Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, RA-3(3):249–265, June 1987.

[10] A. Gelb. *Applied Optimal Estimation*. MIT Press, 1974.

[11] T. Hague and S. Cameron. Motion planning for non-holonomic industrial robot vehicles. In *IEEE Int. Workshop Intelligent Robots and Systems*, Osaka, Japan, 1991.

[12] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, London, 1985.

[13] Huosheng Hu, Michael Brady, and Penny Probert. Coping with uncertainty in control and planning for a mobile robot. In *IEEE Int. Workshop Intelligent Robots and Systems*, Osaka, Japan, 1991.

[14] Huosheng Hu, Michael Brady, and Penny Probert. Navigation and control of a mobile robot among moving obstacles. In *IEEE Int. Conf. Decision and Control*, Brighton, U.K., 1991.

[15] R. Kuc. A spatial sampling criterion for sonar obstacle detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-12(7):686–690, July 1990.

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

[17] John J. Leonard. *Directed Sonar Sensing for Mobile Robot Navigation*. PhD thesis, University of Oxford, 1990. Published by Kluwer Academic Press, Boston, MA, 1992.

[18] J.M. Manyika, I.M. Treherne, and H.F. Durrant-Whyte. A modular architecture for decentralised sensor data fusion. In *2nd Int. Advanced Robotics Programme Workshop on Sensor Fusion and Environmental Monitoring*, 1991.

[19] Marvin Minsky. *Society of Mind*. Simon and Schuster, New York, 1986.

[20] H. Moravec. Sensor fusion in certainty grids for mobile robots. In *NATO ASI Sensor Devices and Systems for Robotics*, pages 253–276. Springer-Verlag, 1989.

[21] Jian Peng. *Rule-Based Spatial Reasoning for Robot Planning*. PhD thesis, Programming Research Group, Oxford University, 1993.

[22] Ian Reid, Michael Brady, Alan McIvor, S. Marshall, I. B. Knight, and R. C. Rixon. Range vision and the Oxford AGV. In *Image Processing 89*, pages 51–72, Wembley, 1989. Online publications.

[23] J. Simmons. A view of the world through the bat's ear: The formation of acoustic images in echolocation. *Cognition*, 33:155–199, 1989.

[24] Bill Triggs and Stephen Cameron. The Oxford Robot World Model. In *NATO ASI Expert Systems and Robotics*, volume F71, pages 275–284, Corfu, 1990. Springer Verag.

Biography

Bill Triggs was born in sunny New Zealand and currently lives in rainy England. He has worked as a nuclear physicist, a general relativist and a dishwasher, but is now a peon of Oxford University Robotics Group. His research concentrates on sensor-based reasoning and planning under uncertainty, and his hobbies include staying up late, walking, cycling and chasing frisbees.