Coping with Uncertainty in a Control System for Navigation and Exploration

Thomas Dean* **Kenneth Basye** **Robert Chekaluk**
Seungseok Hyun **Moises Lejter** **Margaret Randazza**
Department of Computer Science
Brown University, Box 1910, Providence, RI 02912

Abstract

A significant problem in designing mobile robot control systems involves coping with the uncertainty that arises in moving about in an unknown or partially unknown environment and relying on noisy or ambiguous sensor data to acquire knowledge about that environment. We describe a control system that chooses what activity to engage in next on the basis of expectations about how the information returned as a result of a given activity will improve its knowledge about the spatial layout of its environment. Certain of the higher-level components of the control system are specified in terms of probabilistic decision models whose output is used to mediate the behavior of lower-level control components responsible for movement and sensing. The control system is capable of directing the behavior of the robot in the exploration and mapping of its environment, while attending to the real-time requirements of navigation and obstacle avoidance.

Exploration and Navigation

We are interested in building systems that construct and maintain representations of their environment for tasks involving navigation. Such systems should expend effort on the construction and maintenance of these representations commensurate with expectations about their value for immediate and anticipated tasks. Such systems should employ expectations about the information returned from sensors to assist in choosing activities that are most likely to improve the accuracy of its representations. Finally, in addition to reasoning about the future consequences of acting, such systems must attend to the immediate consequences of acting in a changing environment: consequences that generally cannot be anticipated and hence require some amount of continuous attention and commitment in terms of computational resources.

We start with the premise that having a map of your environment is generally a good thing if you need to move between specific places whose locations are clearly indicated on that map. The more frequent your need to move between locations, the more useful you will probably find a good map. If you are not supplied with a map and you find yourself spending an inordinate amount of time blundering about, it might occur to you to build one, but the amount of time you spend in building a map will probably depend upon how much you anticipate using it. Once you have decided to build a map, you will have to decide when and exactly how to go about building it. Suppose that you are on an errand to deliver a package and you know of two possible routes, one of which is guaranteed to take you to your destination and a second which is not. By trying the second route, you may learn something new about your environment that may turn out to be useful later, but you may also delay the completion of your errand.

Huey, the robot used in our experiments, is built almost entirely from off-the-shelf components: a 12-inch diameter synchro-drive base from Real World Interface (Dublin, New Hampshire), a sonar ring subsystem from Denning Mobile Robotics (Wilmington, Massachusetts) equipped with 8 Polaroid ultrasonic sensors, and an 80286-based IBM/AT compatible computer with 2M of memory, a 3-inch floppy, and a serial interface card, running the QNX operating system from Quantum Software (Kanata, Ontario). The various microprocessors on Huey communicate through serial lines. Huey can operate autonomously using on-board power and computing, or tethered to a Unix workstation.

Huey's ultrasonic sensors provide it with information about the distance to nearby objects. With a little care, Huey can detect the presence of a variety of geometric features using these sensors. In gathering information about the office environment, Huey will drive up to a surface to be investigated, align one of the sensors to the right or to the left of its direction of travel along the surface, and then move parallel to that surface looking

*This work was supported in part by a National Science Foundation Presidential Young Investigator Award IRI-8957601 with matching funds from IBM, and by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract No. F49620-88-C-0132.

for abrupt changes in the information returned by the aligned sensor that would indicate some geometric feature such as a 90° corner. In doing this, Huey keeps track of the accumulated error in its movement and the variation in its sensor data to assign a probability to whether or not a feature is present.

Huey has strategies for checking out many simple geometric features found in typical office environments; we refer to these strategies as *feature detectors*. The complete set of feature detectors used by Huey and the details concerning their implementation are described in [Randazza, 1989]. Each feature detector is realized as a control process that directs the robot's movement and sensing. On the basis of the data gathered during the execution of a given feature detector, a probability distribution is determined for the random variable corresponding to the proposition that the feature is present at a specific location.

Huey is designed to explore its environment in order to build up a representation of that environment suitable for route planning. In the course of exploration, Huey induces a graph that captures certain qualitative features of its environment [Kuipers and Byun, 1988, Levitt *et al.*, 1987, Basye *et al.*, 1989]. In addition to detecting geometric features like corners and door jambs, Huey is able to classify locations. In particular, Huey is able to distinguish between corridors and places where corridors meet or are punctuated by doors leading to offices, labs, and storerooms. A corridor is defined as a piece of rectangular space bounded on two sides by uninterrupted parallel surfaces 1.5 to 2 meters apart and bounded on the other two sides by *ports* indicated by abrupt changes in one of the two parallel surfaces. The ports signal *locally distinctive places* (LDPs) (after [Kuipers and Byun, 1988]) which generally correspond to hallway junctions. Uninterrupted corridors are represented as arcs in the induced graph while junctions are represented as vertices. Junctions are further partitioned into classes of junctions (*e.g.*, L-shaped junctions where two corridors meet at right angles, or T-shaped junctions where one corridor is interrupted by a second perpendicular corridor). Huey is given a set of junction classes that it uses to classify and the label the locations encountered during exploration.

In the following sections, we consider two of the main decision processes that comprise Huey's control system, but first we consider briefly the overall architecture in which these decision processes are embedded.

Planning and Control

Huey's control system is composed of a set of decision processes running concurrently under a multi-tasking prioritized operating system. There is no shared state information; all communication is handled by interprocess message passing. Run-time process arbitration is handled by dynamically altering the process priorities. Coordination among processes is achieved through a set of message-passing protocols.

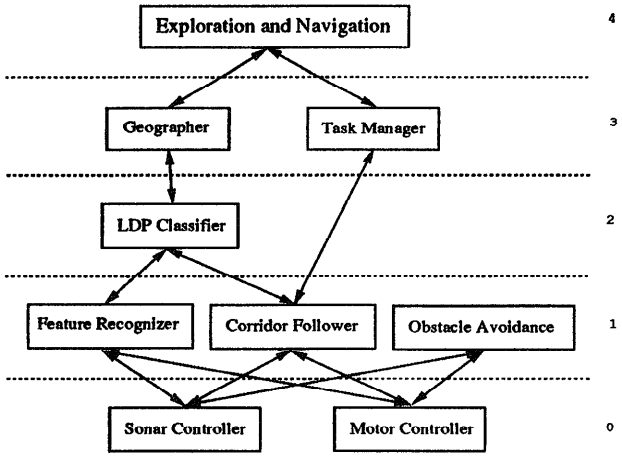

Figure 1: Huey's Control Processes

The different processes that make up Huey's controller are partitioned into levels (see Figure 1). For each level, there is a corresponding arbitrator designed to coordinate the different processes located at that level. At Level 0, we find the processes responsible for control of the different sensor/effector systems on board the mobile base. Each Level 0 process is completely independent of the other processes, so no arbitration is needed. At Level 1, we find the processes responsible for the low-level control of Huey. Level 1 processes are coordinated using a simple priority scheme: the obstacle avoidance process always takes priority over the other Level 1 processes. The activities of the feature recognition and corridor following processes are coordinated by higher-level processes.

Currently, Huey has only one Level 2 process, the LDP classifier, but, as we increase Huey's capabilities, we anticipate several additional processes on this level. At Level 3, we find the two processes responsible for Huey's higher-level behaviors: the task manager in charge of running user-specified errands, and the geographer in charge of exploration and map building. The geographer (roughly) implements the algorithms in [Basye *et al.*, 1989]. The task manager is a very simple route planner. The activities of these two processes are coordinated by a Level 4 decision process that takes into account the possible costs and benefits to be derived from different strategies for mixing exploration and errand running. To get a better idea of how Huey handles some of its higher-level decision making, we now describe the decision processes at Levels 2 and 4.

Classifying Locally Distinctive Places

Upon exiting a corridor through a port, Huey will want to determine what sort of LDP it has entered. If Huey is in a well-explored portion of its environment, this

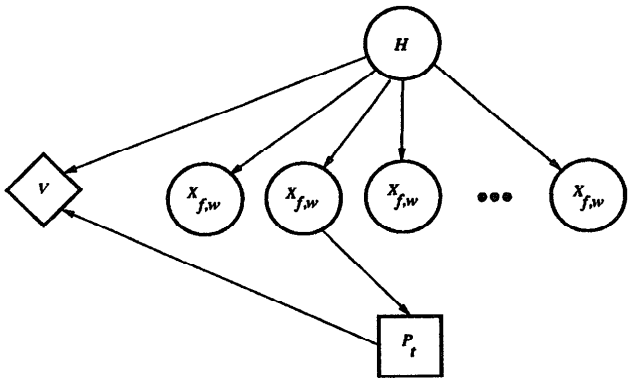

Figure 2: LDP-classification module's influence diagram

determination should match Huey's expectations as indicated in its map. If, on the other hand, Huey is in some unknown or only partially-explored area, this determination will be used to extend the map, possibly adding new vertices or identifying the current LDP with existing vertices. In this section, we describe how Huey classifies the LDPs encountered during exploration.

Let L be the set of all locally distinctive places in the robot's environment, $C = \{C_1, C_2, \ldots, C_n\}$ be a set of equivalence classes that partitions L, and F be a set of primitive geometric features (*e.g.*, convex and concave corners, flat walls). Each class in C can be characterized as a set of features in F that stand in some spatial relationship to one another. As Huey exits a port, a local coordinate system is set up with its origin on the imaginary line defined by the exit port and centered in the corridor. The space about the origin enclosing the LDP is divided into a set of equi-angular wedges W. For each feature/wedge pair (f, w) in $F \times W$, we define a specialized feature detector $d_{f,w}$ that is used to determine if the current LDP satisfies the feature f at location w in the coordinate system established upon entering the LDP. Let D be the set of all such feature detectors plus *no_op*, a pseudo-detector that results in no new information and takes no time or effort to execute.

Huey's LDP-classification module maintains a a probabilistic assessment of the hypotheses concerning the class of the current LDP given the evidence acquired thus far. At any given time, Huey will have tried some number of feature detectors. Let P_t be the pool of detectors available for use at time t; P_t is just D less the set of detectors executed up until t in classifying the current LDP. The LDP-classification module is responsible for choosing the next feature detector to invoke from the set P_t. It does so using a decision model cast in terms of an *influence diagram* [Howard and Matheson, 1984]. The details are described in [Chekaluk, 1989]; in the following, we highlight the main points.

The LDP-classification module's influence diagram includes a set of *chance* nodes corresponding to random variables, a *decision* node corresponding to actions that the robot might take, and a *value* node representing the expected utility of invoking the different feature detectors in various circumstances. The chance nodes include a hypothesis variable, H, that can take on values from C, and a set of boolean variables of the form, $X_{f,w}$, used to represent whether or not the feature f is present at location w. Each $X_{f,w}$ is conditioned on the hypothesis H according to the distribution $\Pr(X_{f,w}|C_i)$ determined by whether or not the class requires the feature at the specified location. The decision node, P_t, indicates the feature detectors available for use at time t, and the value node, V, represents the utility of invoking each feature detector. V is dependent only upon the hypothesis and decision nodes. The predecessors of P_t are just the feature detectors invoked so far, thereby indicating temporal precedence and informational dependence. A graphical representation of the influence diagram is shown in Figure 2.

The utility of invoking each detector is based on (i) the ability of the detector to discriminate among the hypotheses, (ii) the cost of deploying the detector, (iii) the probability that the current best hypothesis is correct, and (iv) the cost of misidentifying the LDP. The first two are used to select from among $D - \{no\_op\}$ and the last two are used to choose between the best detector from $D - \{no\_op\}$ and *no_op*. The LDP-classification module selects from $D - \{no\_op\}$, using the function, $\mu : P_t \times H \rightarrow \Re$, defined by $\mu(d_{f,w}, h) =$

$$\kappa_1 \text{Discrim}(d_{f,w}) - \kappa_2 \text{Cost}(d_{f,w}, h),$$

where κ_1 and κ_2 are constants used for scaling, $\text{Cost}(d_{f,w}, h)$ is a function of the expected time spent in executing $d_{f,w}$ for an LDP of a given class, and $\text{Discrim}(d_{f,w})$ is the discrimination function of [Cameron and Durrant-Whyte, 1988] adapted for our application, and defined by

$$\sum_{i=1}^{n} \Pr(C_i) \sum_{v \in \{0,1\}} |\Pr(d_{f,w} = v|C_i) - \Pr(d_{f,w} = v)|,$$

where $d_{f,w} = v$ is meant to represent the proposition that the detector $d_{f,w}$ returns the value v. The terms in the above formula are easily obtained. $\Pr(d_{f,w} = v|C_i)$ is the distribution associated with the corresponding $X_{f,w}$ node, and $\Pr(d_{f,w} = v)$ can be calculated using

$$\Pr(d_{f,w} = v) = \sum_{i=1}^{n} \Pr(d_{f,w} = v|C_i) \Pr(C_i)$$

The LDP-classification module evaluates the influence diagram using Agogino and Ramamurthi's [1988] algorithm to obtain a decision policy and an expected value function for choosing from among $D - \{no\_op\}$. The LDP-classification module can also choose to do nothing by selecting *no_op*, thereby committing to the class C_i with the highest posterior probability given the information returned by the feature detectors invoked thus

far. The actual decision model used by Huey is somewhat more complicated than the one described here; in particular, Huey has an additional set of chance nodes corresponding to micro features, the set of feature detectors is more extensive than indicated here, and the current system allows for a feature detector to be invoked multiple times.

Expected Value of Exploration

We have experimented with several decision models for reasoning about the expected value of exploration. In the simple model presented in this section, we assume that the system of junctions and corridors that make up Huey's environment can be registered on a grid so that every corridor is aligned with a grid line and every junction is coincident with the intersection of two grid lines. In the following, the set of junction types, J, corresponds to all possible configurations of corridors incident on the intersection of two grid lines. Intersections with at least one incident corridor correspond to LDPs. Since we also assume that Huey knows the dimensions of the grid (*i.e.*, the number of x and y grid lines), Huey can enumerate the set of possible maps $M = \{M_1, M_2, \ldots, M_m\}$, where a map corresponds to an assignment of a junction type to each intersection of grid lines. For most purposes, we can think of a map as a labeled graph.

We restrict M by making a number of assumptions about office buildings of the sort that Huey will find itself in (*e.g.*, all LDPs are connected). To further restrict M, Huey engages in an initial phase of task-driven exploration. Each task specifies a destination location in x, y grid coordinates. Huey computes the shortest path assuming that all intersections have as many coincident corridors as is consistent with what is known about the intersection and its adjacent intersections. Huey then follows this path, acquiring additional information as it moves through unknown intersections until it either finds its path blocked, in which case it recomputes the shortest path to the goal taking into account its new knowledge, or it reaches the goal.

Huey continues in this task-driven exploration phase until it is likely—based on the spatial distribution of known locations—that all locations have been visited at least once. From this point on, given a task to move to specific location, it is likely that Huey will be able to compute a path through known territory. Huey now faces the decision whether to take the known path or to try an alternative path through unknown territory. In the model considered here, Huey has to choose between taking the shortest path through known territory, and trying the shortest path consistent with what is known. In the latter case, Huey will learn something new, but it may end up taking longer to complete its task.

Let H be a random variable corresponding to the actual configuration of the environment; H takes on values from M. Let $J_{x,y}$ be a random variable corresponding to the junction type of the intersection at the coordinates, $\langle x, y \rangle$, in the grid; $J_{x,y}$ can take on values from the set C defined previously. Let $X_{f,w}$ be as previously defined, a boolean variable corresponding the presence of a feature at a particular position. Let $S_{x,y}$ be a random variable corresponding to a possible sensing action taken at the coordinates, $\langle x, y \rangle$, in the grid. Let $\mathcal{E}$ correspond to the set of sensing actions taken thus far. The complete probabilistic model is shown in Figure 3.

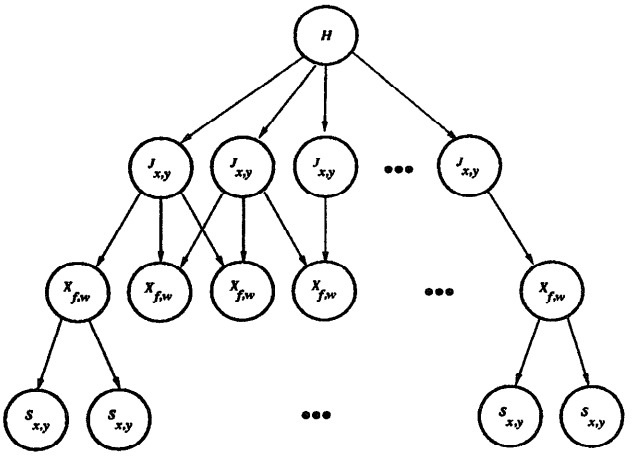

Figure 3: The probabilistic model for map building

In our simple model, Huey has to decide between the two alternatives, P_K and P_U, corresponding to paths through known and unknown territory. To compute $\Pr(H|\mathcal{E})$, $\Pr(H)$ is assumed to be uniform, $\Pr(J_{x,y}|H)$ and $\Pr(X_{f,w}|J_{x,y})$ are determined by the geometry, and $\Pr(S_{x,y}|X_{f,w})$ is determined experimentally. Let $T = \{T_1, T_2, \ldots, T_r\}$ denote the set of all tasks corresponding to point-to-point traversals, and $\mathrm{E}(|T_i|)$ denote the expected number of tasks of type T_i. Let $\mathrm{Cost}(T_i, M_j, M_k)$ be the time required for the task T_i using the map M_j, given that the actual configuration of the environment is M_k; if M_j is a subgraph of M_k, then $\mathrm{Cost}(T_i, M_j, M_k)$ is just the length of the shortest path in M_j. Let T^* denote Huey's current task. For evaluation purposes, we assume that Huey will take at most one additional exploratory step.

To complete the decision model, we need a means of computing the expected value of P_K and P_U. In general, the value of a given action is the sum of the immediate costs related to T^* and the costs for expected future tasks. Let

$$\mathrm{Futures}(M_i, I) = \sum_{j=1}^{r} \mathrm{E}(|T_j|)\,\mathrm{Cost}(T_j, M_I^*, M_i),$$

where $M_I^* = M_{\arg\max_j \Pr(M_j|I)}$.

If classification is perfect, Huey correctly classifies any location it passes through, and $M_{\mathcal{E}}^*$ is the minimal

assignment consistent with what it has classified so far. In this case, the expected value of P_K is

$$\text{Cost}(T^*, M^*, \_) + \text{Futures}(\_, \mathcal{E}).$$

If classification is imperfect, the expected value of P_K is

$$\sum_{j=1}^{m} \Pr(M_j|\mathcal{E}) \left[\text{Cost}(T^*, M^*, M_j) + \text{Futures}(M_j, \mathcal{E})\right].$$

Handling P_U is just a bit more complicated. Suppose that Huey is contemplating exactly one sensing action that will result in one of several possible observations $O_1, \ldots, O_n$, then the expected value of P_U is

$$\sum_{j=1}^{m} \Pr(M_j|\mathcal{E}) \, \text{Cost}(T^{*'}, M^*, M_j) +$$
$$\sum_{i=1}^{n} \Pr(O_i) \sum_{j=1}^{m} \Pr(M_j|O_i, \mathcal{E}) \text{Futures}(M_j, [O_i, \mathcal{E}])$$

where $T^{*'}$ is a modification of T^* that accounts for the proposed exploratory sensing action.

We use Jensen's [1989] variation on Lauritzen and Spiegelhalter's [1988] algorithm to evaluate the network shown in Figure 3. The time required for evaluation is determined by the size of the sample spaces for the individual random variables and the connectivity of the network used to specify the decision model. In the case of a singly-connected[1] network, the cost of computation is polynomial in the number of nodes and the size of the largest sample space—generally the space of possible maps. The network shown in Figure 3 would be singly-connected if each feature, $X_{f,w}$, had at most one parent corresponding to a junction, $J_{x,y}$; a network of this form with 100 possible maps can be evaluated in about 10 seconds, assuming an 8×8 grid.

In the case of a multiply-connected network, the cost of computation is a function of the product of the sizes of the sample spaces for the nodes in the largest clique of the graph formed by triangulating the DAG corresponding to the original network. By making use of the information gathered in the initial exploratory phase, Huey is able to reduce the connectivity of the network used to encode the decision model. The multiply-connected networks that Huey currently uses have around 50 possible maps, and require on the order of a few minutes to evaluate.

The space of possible maps chosen may not include the map corresponding to the actual configuration of the environment. To handle such possible omissions, we add a special value, $\perp$, to the sample space for H, and make all of the $\Pr(J_{x,y}|\perp)$ entries in the conditional probability tables $1/s$ where s is the number of junction types. If Huey ever detects that $M^*_{\mathcal{E}} = \perp$, then it assumes that it has excluded the real map, and dynamically adjusts its decision model by computing a new sample space for H guided by the results of the exploratory actions taken thus far.

[1] A network is said to be singly-connected if there is at most one directed path between any two nodes; otherwise, it is said to be multiply connected [Pearl, 1988].

Designing Robot Control Systems

Our approach to designing Huey's control system is outlined as follows. We begin by considering Huey's overall decision problem, determining an optimal decision procedure according to a precisely stated decision-theoretic criteria, neglecting computational costs. We use an influence diagram to represent the underlying decision model and define the optimal procedure in terms of evaluating this model.

Huey's overall decision problem involves several component problems associated with specific classes of events occurring in the environment. These component decision problems include what action to take when approached by an unexpected object in a corridor, what sensor action to take next when classifying a junction, and what path to take in combining exploration and task execution. Each of these problems is recurrent.

Problems involving what sensor action to take in classification or what path to take in navigation are predictably recurrent. For instance, during classification each sensor action takes about thirty seconds to a minute, so the robot has that amount of time to decide what the next action should be if it wishes to avoid standing idle lost in computation. The frequency with which choices concerning what path to take occur is dependent on how long Huey takes to traverse the corridor on route to the next LDP. With the current mobile platform operating in the halls of the computer science department, moving between two consecutive LDPs takes about four minutes. The problem of deciding what to do when approached by an unexpected object occurs unpredictably, and the time between when the approaching object is detected and when the robot must react to avoid a collision is on the order of a few seconds.

By making various (in)dependence assumptions and eliminating noncritical variables from the overall complex decision problem, we are able to decompose the globally optimal decision problem into sets of simpler component decision problems. Each of the sets of component problems are solved by a separate module. The computations carried out by these modules are optimized using a variety of techniques to take advantage of the expected time available for decision making [Kanazawa and Dean, 1989]. The different decision procedures communicate by passing probability distributions back and forth. For instance, the module responsible for making decisions regarding exploration and the module responsible for classifying LDPs pass back and forth distributions regarding the junction types of LDPs.

Conclusions and Related Work

The original designs for Huey's control system were influenced by the design of the Hilare robot [Chatila and Laumond, 1985]. The lower-levels of the control system rely little upon the existence of a global clock and adhere for the most part to the specifications of Brooks' subsumption architecture [Brooks, 1986]. Our use of influence diagrams and Bayesian decision theory was inspired by recent work on decision-theoretic control for visual interpretation and sensor placement [Cameron and Durrant-Whyte, 1988, Hager, 1988, Levitt *et al.*, 1988]. The design of the geographer module was based on the work of Kuipers [Kuipers and Byun, 1988] and Levitt [Levitt *et al.*, 1987] on learning maps of large-scale space, and our own extensions to handle uncertainty [Basye *et al.*, 1989]. The design of the module responsible for coordinating exploration and errand running was based on an application of information value theory [Howard, 1966].

Huey's control system combines high-level decision making with low-level control and sensor interpretation to provide for navigation, real-time obstacle avoidance, and exploration in an unfamiliar environment. The basic controller handles multiple asynchronous processes communicating via simple message-passing protocols. The architecture supports a variety of arbitration schemes from fixed-priority processor scheduling to decision-theoretic control. This paper emphasizes two decision processes: one responsible for reasoning about the uncertainty inherent in dealing with noisy and ambiguous sensor data, and a second responsible for assessing the expected value of various exploratory actions. Our basic approach to designing robot control systems involves constructing a decision model for the overall problem and then decomposing it into component models guided by the time criticality of the associated decision problems.

References

[Agogino and Ramamurthi, 1988] A. M. Agogino and K. Ramamurthi. Real-time influence diagrams for monitoring and controlling mechanical systems. Technical Report Technical Report, Department of Mechanical Engineering, University of California, Berkeley, 1988.

[Basye *et al.*, 1989] Kenneth Basye, Thomas Dean, and Jeffrey Scott Vitter. Coping with uncertainty in map learning. In *Proceedings IJCAI 11*, pages 663–668. IJCAI, 1989.

[Brooks, 1986] Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2:14–23, 1986.

[Cameron and Durrant-Whyte, 1988] Alec Cameron and Hugh F. Durrant-Whyte. A bayesian approach to optimal sensor placement. Technical report, Oxford University Robotics Research Group, 1988.

[Chatila and Laumond, 1985] R. Chatila and J.-P. Laumond. Position referencing and consistent world modeling on mobile robots. In *IEEE International Conference on Robotics and Automation*, pages 138–145, March 1985.

[Chekaluk, 1989] Robert A. Chekaluk. Using influence diagrams in recognizing locally-distinctive places. M.Sc. Thesis, Brown University, 1989.

[Hager, 1988] Gregory D. Hager. Active reduction of uncertainty in multi-sensor systems. Ph.D. Thesis, University of Pennsylvania, Department of Computer and Information Science, 1988.

[Howard and Matheson, 1984] Ronald A. Howard and James E. Matheson. Influence diagrams. In Ronald A. Howard and James E. Matheson, editors, *The Principles and Applications of Decision Analysis*. Strategic Decisions Group, Menlo Park, CA 94025, 1984.

[Howard, 1966] Ronald A. Howard. Information value theory. *IEEE Transactions on Systems Science and Cybernetics*, 2(1):22–26, 1966.

[Jensen, 1989] Finn V. Jensen. Bayesian updating in recursive graphical models by local computations. Technical Report R-89-15, Institute for Electronic Systems, Department of Mathematics and Computer Science, University of Aalborg, 1989.

[Kanazawa and Dean, 1989] Keiji Kanazawa and Thomas Dean. A model for projection and action. In *Proceedings IJCAI 11*, pages 985–990. IJCAI, 1989.

[Kuipers and Byun, 1988] Benjamin J. Kuipers and Yung-Tai Byun. A robust, qualitative method for robot spatial reasoning. In *Proceedings AAAI-88*, pages 774–779. AAAI, 1988.

[Lauritzen and Spiegelhalter, 1988] Stephen L. Lauritzen and David J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society*, 50(2):157–194, 1988.

[Levitt *et al.*, 1987] Tod S. Levitt, Daryl T. Lawton, David M. Chelberg, and Philip C. Nelson. Qualitative landmark-based path planning and following. In *Proceedings AAAI-87*, pages 689–694. AAAI, 1987.

[Levitt *et al.*, 1988] Tod Levitt, Thomas Binford, Gil Ettinger, and Patrice Gelband. Utility-based control for computer vision. In *Proceedings of the 1988 Workshop on Uncertainty in Artificial Intelligence*, 1988.

[Pearl, 1988] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan-Kaufman, Los Altos, California, 1988.

[Randazza, 1989] Margaret J. Randazza. The feature recognition module of the ldp system for the robot huey. M.Sc. Thesis, Brown University, 1989.