July 1990

Computational Methods for Task-Directed Sensor Data Fusion and Sensor Planning

Greg Hager
University of Pennsylvania

Max L. Mintz
University of Pennsylvania, mintz@cis.upenn.edu

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-90-38.

Computational Methods for Task-Directed Sensor Data Fusion and Sensor Planning

Abstract

In this paper, we consider the problem of task-directed information gathering. We first develop a decision-theoretic model of task-directed sensing in which sensors are modeled as noise-contaminated, uncertain measurement systems and sensing tasks are modeled by a transformation describing the type of information required by the task, a utility function describing sensitivity to error, and a cost function describing time or resource constraints on the system.

This description allows us to develop a standard conditional Bayes decision-making model where the value of information, or payoff, of an estimate is defined as the average utility (the expected value of some function of decision or estimation error) relative to the current probability distribution and the best estimate is that which maximizes payoff. The optimal sensor viewing strategy is that which maximizes the net payoff (decision value minus observation costs) of the final estimate. The advantage of this solution is generality--it does not assume a particular sensing modality or sensing task. However, solutions to this updating problem do not exist in closed-form. This, motivates the development of an approximation to the optimal solution based on a grid-based implementation of Bayes' theorem.

We describe this algorithm, analyze its error properties, and indicate how it can be made robust to errors in the description of sensors and discrepancies between geometric models and sensed objects. We also present the results of this fusion technique applied to several different information gathering tasks in simulated situations and in a distributed sensing system we have constructed.

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-90-38.

Computational Methods For Task-Directed Sensor Data Fusion and Sensor Planning

MS-CIS-90-38
GRASP LAB 218

Greg Hager
Max Mintz

Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104-6389

July 1990

Computational Methods for Task-directed Sensor Data Fusion and Sensor Planning

Greg Hager
Max Mintz
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, Pennsylvania 19104-6389

Abstract

In this article we consider the problem of task-directed information gathering. We first develop a decision-theoretic model of task-directed sensing in which sensors are modeled as noise-contaminated, uncertain measurement systems, and sensing tasks are modeled by a transformation describing the type of information required by the task, a utility function describing sensitivity to error, and a cost function describing time or resource constraints on the system.

This description allows us to develop a standard conditional Bayes decision-making model where the value of information, or payoff, *of an estimate is defined as the average utility (the expected value of some function of decision or estimation error) relative to the current probability distribution and the best* estimate *is that which maximizes payoff. The optimal sensor viewing strategy is that which maximizes the net payoff (decision value minus observation costs) of the final estimate. The advantage of this solution is generality—it does not assume a particular sensing modality or sensing task. However, solutions to this updating problem do not exist in closed form. This motivates the development of an approximation to the optimal solution based on a grid-based implementation of Bayes' theorem.*

We describe this algorithm, analyze its error properties, and indicate how it can be made robust to errors in the description of sensors and discrepancies between geometric models and sensed objects. We also present the results of this fusion technique applied to several different information gathering tasks in simulated situations and in a distributed sensing system we have constructed.

1. Introduction

As sensor-based robotics systems are employed in increasingly complex, real-world situations, the volume and complexity of information required for adequate performance will increase substantially. To effectively gather and use sensor information, robotic systems will need the capability of making intelligent choices about the deployment of sensing capabilities and computational resources. Developing algorithms flexible enough to handle a wide variety of sensing problems and situations is a crucial step in the process of building intelligent systems.

The purpose of this article is to present a mathematical framework for describing geometric sensing problems and develop methods for computing the solutions to these problems. Our solutions include both the strategy or plan for gathering sensor information and the integration of sensor observations into a consistent geometric description. Furthermore, by explicitly representing the costs of information gathering and the effect of decision errors, we are able to determine how much information to gather. Throughout this article, the emphasis is on techniques that are independent of a particular sensing modality or a particular sensing problem. The former implies that, by definition, we solve the multisensor fusion problem. The latter implies that the same techniques can be employed in a diverse set of applications and therefore provide a unified treatment of many different sensor-based systems.

We define geometric sensing problems as those requiring a description of the shape, extent (size), and position (the last is taken to include both translational and rotational position components) of objects. An unrestricted environment containing several objects will require several distinct geometric descriptions. Furthermore, there may be constraints among the descriptions, (e.g., the position of an object on a table depends on the position of the table, or the shapes of two interlocking pieces must fit together). A consistent geometric description is one that can account for all observed sensor data and geometric constraints by one or more assignments of shape, extent, and position. The data fusion problem is to construct a consistent geometric description from sensor data.

The International Journal of Robotics Research,
Vol. 10, No. 4, August 1991,

Examples of geometric sensor fusion problems abound in the robotics literature. For example, computing a consistent map of the environment based on sensor observations is a core problem in mobile robotics (Brooks 1985; Durrant-Whyte 1988; Giralt et al. 1984; Moravec 1988). These maps usually include an explicit representation of position and sometimes use a description of shape or extent. Many recognition systems use very refined descriptions of shape and extent to classify observed objects relative to a database of models (Allen 1988; Brooks 1981). Task level programming systems (Lozano-Pérez 1985) use more complex geometric specifications for planning and performing grasping and manipulation. In particular, computing the stability of a grasp or the initial lift vector for an object requires quantities such as a centroid or weight (Trinkle 1987)—quantities that, given appropriate prior information about density, can be computed from geometric descriptions.

As this last example illustrates, one geometric form may be suitable for describing the data, but the application requires information in another form. However, the information needed by the application can be often expressed as a function of the geometric parameterization. We note that these functional descriptions can describe *qualitative* (propositional) properties about the environment and thereby facilitate applications that manipulate representations syntactically (Brooks 1981; Stansfield 1987). That is, a proposition can be represented by an *indicator function* mapping the parameters of a geometric description into truth values. Similarly, parameter-based classification can be represented by describing the function mapping parameter values to object classifications.

The choice of what is observed and how it is observed clearly affects the efficiency of fusion with respect to a particular task. Furthermore, sensor applications vary in their sensitivity to decision error, which in turn affects the number of observations needed to obtain an adequate geometric description. The purpose of sensor planning is to enhance the performance of sensor fusion by tailoring the choice and number of observations to the given task and current operating conditions. However, fusion and planning must be tempered by the *cost* of gathering and fusing information. In some cases, it is better to allow the possibility of an incorrect decision or action than to spend the additional resources needed to improve the quality or accuracy of a decision. An optimal sensor strategy is one that has the maximum *net* value.

The idea of using active probing and adaptation is not new in the robotics area (Aloimonos 1987; Bajcsy 1985; 1988). For example, Allen (1988) used a tactile probe to gather visually occluded surface information for the purposes of object recognition. Stansfield (1987) extended this paradigm by considering categorical models. Grimson (1986) and Hutchinson et al. (1988) consider the problem of determining the optimal sensor placement for disambiguating the pose of polygonal objects. Cameron (1989) describes a system that uses decision-theoretic principles to compute a plan of observation for determining the type and pose of objects from tactile probe data.

Many sensor data fusion and sensor planning systems have the common characteristic that they were designed to work efficiently for specific applications (typically recognition) using specific sensors. However, the information required by even simple tasks can be highly varied and ranges from very simple measurements by simple sensors to the determination of relatively complex quantities using multiple information sources. The goal of our work is to build flexible systems that can work with several sensors and sensing tasks based on a description of both sensor and sensing task in a suitable language. In this article we first focus on describing a general framework for describing sensors, models, and tasks. We then use decision-theoretic principles to define optimal solutions to the sensor planning and fusion problems and, finally we develop computational algorithms that approximate these optimal solutions.

The next section presents a mathematical framework for describing geometric models, sensor models, and task models and illustrates its use with some simple examples. Section 3 briefly introduces the decision-theoretic principles we use and describes the decision-theoretic interpretation of sensor models and task models. Following that, section 4 discusses how these decision-theoretic methods can be implemented using a grid-based representation of probability densities. Section 5 is a mathematical and simulation-based analysis of approximation error and robustness of the methods. In section 6 we present some experimental results and close with a discussion of the limitations and open problems of this methodology.

2. Describing Sensors and Sensing Tasks

In overview, we describe sensing tasks by first defining one or more parametric, geometric representations for observed objects. We then describe how the available sensors image those objects and

how tasks make use of information contained in a representation. The advantage of this organization is that it separates the description of the sensor from the sensing task and thereby enhances the modularity of the system. That is, it allows (1) the addition or deletion of sensors observing an object independent of sensing task as long as the available set of sensors can supply the required information and (2) the addition or deletion of tasks using the information stored in a model independent of how the information was obtained.

The effectiveness of this framework depends heavily on the choice of a parametric representation. The use of a particular parametric model fixes the "vocabulary" of data modeling and hence is highly application dependent. The complexity of a parametric model should reflect the question that we seek to answer with the model: a model with only a few degrees of freedom provides significantly more data compression and is generally faster to compute than a more flexible model, but the flexible model is able to fit a wider variety of observations and may bring out important aspects of the data that a simple model cannot express. Thus an important issue is to find a concise, computationally efficient model that adequately describes the data for a given application.

For example, when manipulating and positioning integrated circuits (IC), a parametric model of polygons consisting of a position in space and three size parameters is probably adequate. A single mobile camera can observe corners and lines, and these features can be used to determine the size and position of the IC. Solina (1990) considers the problem of postal sorting and manipulation. This domain is more complicated and requires a more flexible model and a richer source of sensor information. Consequently, he used a superellipsoid representation augmented with bending and twisting and recovered model parameters based on laser-range data of exposed object surfaces. In both cases, we have a parametric model (polygons or superellipsoids in space) and observable features (corners and lines or surface points) that can be used to determine the model parameters.

Sensor tasks should describe the relevant aspects of the relationship between the model and the application using sensor information. This information is used to determine the way the sensors should observe an object. For example, classifying an object as large, small, round, or square is independent of location. Hence a classification task can be thought of as focusing on the subset of the model parameters describing shape and size, and the optimal sensing strategies concentrate on refining an estimate of those parameters to the precision required to distinguish object types. Conversely, manipulating the object requires good location information so that a gripper can safely grasp the object. In this case the sensors must focus on the location of the object instead of (or in addition to) its shape, and they will probably have to acquire more and/or different information to obtain a description with the required accuracy.

In the remainder of this section we describe a mathematical form for geometric, sensor, and task models and provide some concrete examples to illustrate their use. Durrant-Whyte (1988) and Richardson and Marsh (1987) provide a more extended discussion of geometric models and statistically-based sensor models. Berger (1985) is an excellent reference for the underpinnings of the statistical decision models on which our task models are based.

2.1. Geometric Models

Our basic geometric modeling primitives are parametric, geometric surface descriptions of the following form:

$$g(l;\ s,\ x) = g(p,\ x) = 0,\ p \in \mathcal{P},\ x \in \mathcal{X},\quad g \in \mathcal{G}. \tag{1}$$

In this description, p is a vector of parameters that describes the essential structure of the system and x is a vector of observable characteristics or features of the object. In the case of geometry, p can be decomposed into a vector representing location, l, and a vector describing size and shape, s. Thus the function $g(\cdot,\cdot)$ is a description of the relationship between the parameterization of a physical or geometric structure in euclidean three-space and its observable characteristics.

The function g is itself taken from a set $\mathcal{G}$. The intent is that $\mathcal{G}$ contains a family of geometric surfaces that have (dimensionally) the same parameterization and are essentially a deviation from a given "ideal" type. That is, it is unreasonable to expect geometric idealizations to agree with real surfaces. Normally, each set of observed features would determine a slightly different value for the describing parameters. We refer to such a family as an *envelope* of models. The definition and extent of an envelope defines what constitutes an acceptable model variation. In the simplest case (and the one considered in this article) we simply describe the deviation required to fit the model to the data. However, more complex schemes are certainly possible. For example, Leyton (1988) has developed an exten-

sive theory of continuous deformation processes for describing model variation.

In this article, we require that (1) can be rewritten in the following explicit form:

$$g'(\boldsymbol{p}, \boldsymbol{c}) = \boldsymbol{x},\ \boldsymbol{p} \in \mathcal{P},\ \boldsymbol{c} \in \mathcal{C},\ \boldsymbol{x} \in \mathcal{X},\ g' \in \mathcal{G}'. \quad (2)$$

In general, the relationship between parameters requires the introduction of "helper" parameters, $\boldsymbol{c}$, for explicit solution. We hereafter refer to the parameters in $\boldsymbol{c}$ as *correspondence* parameters, because, by fixing their value, we fix the "correspondence" between observed features and unknown parameters. In those cases where there is already a unique relationship between parameters and observables, the vector $\boldsymbol{c}$ is of dimension zero.

Example 1 The location of the object restricted to a plane can be expressed relative to an arbitrary base coordinate system using homogeneous transforms (Paul 1981) as:

$${}^{b}\boldsymbol{T}_0(\boldsymbol{l}) = {}^{b}\boldsymbol{T}_0(x_0; y_0; \alpha_0) = \text{trans}\,(x_0, y_0, 0)\ \text{rot}\,(z_0, \alpha_0).$$

The simplest parameterization of rectangular 3D box is to describe the relative positions of the corners:

$$M(\boldsymbol{a}) = R(a_1, a_2, a_3) = \begin{bmatrix} 0 & 0 & a_1 & a_1 & 0 & 0 & a_1 & a_1 \\ 0 & a_2 & a_2 & 0 & 0 & a_2 & a_2 & 0 \\ 0 & 0 & 0 & 0 & a_3 & a_3 & a_3 & a_3 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

The full geometric description of an arbitrarily sized rectangle can now be expressed as:

$$\mathrm{B}(\boldsymbol{l}, \boldsymbol{a}) = {}^{b}\boldsymbol{T}_0(\boldsymbol{l})\mathrm{M}(\boldsymbol{a}).$$

In order to focus on a single feature, we add an index as a correspondence parameter and define a new function as:

$$g_b(\boldsymbol{l}, \boldsymbol{a}, c) = \mathrm{B}(\boldsymbol{l}, \boldsymbol{a})[c], \quad c \in \{1, 2, \ldots, 8\}. \quad (3)$$

This model can, in principle, be used to describe any sort of object that is topologically equivalent to a box provided some model deviation is allowed for. For example, Figure 1 illustrates a (planar) nonrectangular object described (within ϵ) by a box located at (x_0, y_0), rotated α_0, of size a_1 by a_2. In this example, observation of the horizontally aligned corners determine one description, and observations of the vertically aligned corners determine a second (smaller) description. Any combination of three corners reveals the discrepancy and forces some type of model deformation.

It is important to note that because (3) only refers to corners, the geometry of the model is only restricted at the corners and says effectively nothing about the surfaces and lines between them. Continuous correspondence parameters are generally required in order to ensure model fit at all boundary points.

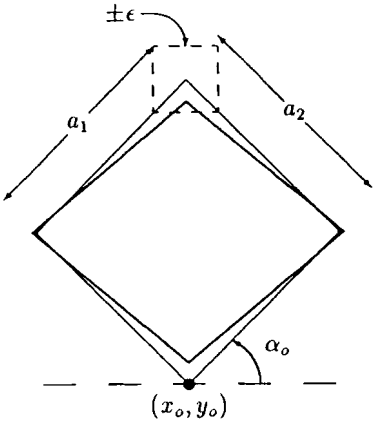

Fig. 1. A rectangular box that has been fit to a nonrectangular object.

Example 2 Pentland (1986) introduced *superquadrics* as a modeling primitive, and Solina (1990) developed a least squares algorithm for recovering superellipsoids (convex superquadrics) from range data. Superellipsoids are described by a parametric equation of the form:

$$S(\boldsymbol{a}, \boldsymbol{\gamma}, \eta, \omega) = \begin{bmatrix} a_1 C_\eta^{\gamma_1} C_\omega^{\gamma_2} \\ a_2 C_\eta^{\gamma_1} S_\omega^{\gamma_2} \\ a_3 S_\eta^{\gamma_1} \\ 1 \end{bmatrix},$$

$$0 \le \eta \le \frac{\pi}{2}, \quad 0 \le \omega \le \frac{\pi}{2}, \quad 0 < \gamma_i \le 1,$$

where $C_x = \cos(x)$, $S_x = \sin(x)$, and an enclosed volume is described by reflecting this surface into the other seven octants. The vector $\boldsymbol{a} = [a_1, a_2, a_3]$ can be interpreted as the size of the superellipsoid, the vector $\boldsymbol{\gamma} = [\gamma_1, \gamma_2]$ governs the shape of the superellipsoid, and the angles η and ω are correspondence parameters.

The full transformation of an arbitrary superquadric can be expressed as:

$$g_s(\boldsymbol{l}, \boldsymbol{a}, \boldsymbol{c}) = {}^{b}\boldsymbol{T}_0(\boldsymbol{l})S(\boldsymbol{s}, \boldsymbol{c})$$
$$\text{where}\quad \boldsymbol{s} = [\boldsymbol{a}; \boldsymbol{\gamma}]^T, \quad \boldsymbol{c} = [\eta, \omega]^T.$$

By sweeping over η and ω, we describe the entire object surface and consequently enforce model constraints at all surface points.

There is another fundamental difference between

the correspondence parameter in example 1 and the correspondence parameters in this example. In the case of a rectangle we can, by suitable bookkeeping, determine the proper value of the correspondence parameter for each observation; that is, we can usually determine which corner we are looking at up to an arbitrary symmetry. In the case of superellipsoids, the parameter is continuous and, depending on the sensor and its imaging geometry, may have to be considered as an unknown along with the other system parameters. However, this additional parameter carries no information about the observed structure itself and changes from observation to observation. Consequently, in the process of inverting the object/sensor relationship we must somehow account for these additional degrees of freedom.

2.2. *Sensor Models*

In our formulation, a sensor is considered to be both the hardware and software used to extract specific properties or features of observed surfaces. These sensors generally lack perfect resolution in the following two senses:

1. *Statistical noise*. The physical design of the transducer and its attendant elements lead to corruption of the sensor signal that can be modeled using probability measures.
2. *Quantization and model uncertainty*. The design of the sensor and the associated algorithms have a limited resolution, mechanical backlash, or other uncertainties that may not be well modeled using statistical methods.

A *complete* model of a sensor would include a description of the effect of all influences on the output of the sensor. What constitutes an *adquate* sensor model depends largely on how it will be applied (Hager and Mintz 1989a). In this article, we employ a sensor model of the form:

$$z_i = H(x_i, w_i, e) + V_i(x_i, w_i, e), \quad x_i \in \mathcal{X}, \quad w_i \in \mathcal{W}, \quad H \in \mathcal{H}, \quad V_i \in \mathcal{V}. \tag{4}$$

The intent here is that H describes the *ideal* relationship between observed features and sensor observations. The behavior of H depends on the world geometry (through the features, x_i) and the choice of sensor control parameters w_i. There may be additional calibration parameters, e, influencing the imaging properties of the sensor.

In practice, H is almost never known with complete certainty. Slight variations in the actual behavior of the sensor cause it to depart from the idealized model in unpredictable ways. These variations arise from modeling (systematic) errors, mechanical backlash, quantization, and communication delay, to name a few sources. Most previous work in fusion has assumed that the idealized model is good enough—that the variations are small enough not to warrant an explicit accounting. However, Hager and Mintz (1989b) demonstrated that, in some circumstances, even small model variations can cause unpredictable system performance and consequently must be accounted for. Hence we explicitly allow H to vary within an envelope $\mathcal{H}$ and require fusion methods to tolerate such variations.

Observations may be corrupted by additive noise with properties that also vary with both the observed parameters and the control parameters. Again, instead of assuming a single description for V_i, we take the view that $V_i \in \mathcal{V}$ where $\mathcal{V}$ is a specified class of random variables. The intent is that we usually are not in a position to state a single model of statistical noise, though we can usually place bounds on the form of its distribution. In this article we assume that V_i is *bounded*—its probability density does not have tails extending to infinity (note that this assumption excludes Gaussian noise models), and V_i is independent of V_j for $i \neq j$.

Example 3 The description of a monocular vision sensor observing the outlines of surfaces is easily described using projective geometry. That is, suppose the object is described by M_b or M_s, as given in examples 1 and 2. A sensor above a table with motion in x, y, height, and rotation is described by the transform

$$^bT_s(w) = {}^bT_s([x_c, y_c, h_c, \alpha_c]^T) = \text{trans}(x_c, y_c, h_c)\ \text{rot}(z, \alpha_c).$$

The effects of perspective can be modeled by a function of the form:

$$P([x, y, z, 1]^T) \rightarrow \begin{bmatrix} x/z \\ y/z \end{bmatrix}.$$

These can be combined to give a nominal sensor model of the form:

$$H(p, w, c) = P({}^bT_s(w)^{-1} g_s(p, c)) \quad \text{where} \quad x \in \{b, s\}.$$

The statistical characteristics of sensor observations can be modeled using standard techniques (Box and Jenkins 1976) and the set of sampling distributions described by suitable means, e.g. two bounding histograms. The variability in the model can described by two tolerance parameters, ϵ_x and

ϵ_y, describing the deviation in sensor outputs from the nominal model. Because V is bounded, these parameters can be discovered over a series of test runs of the system.

Notation: In the sequel, we will use the shorthand notation $H(\boldsymbol{p}, \boldsymbol{w}, \boldsymbol{e})$ for $H(g(\boldsymbol{p}), \boldsymbol{w}, \boldsymbol{e})$ in those cases where the distinction between H and g is not crucial to the development. Similarly, we will often suppress the parameters $\boldsymbol{w}$ and $\boldsymbol{e}$ when we are only concerned with H as a function of $\boldsymbol{p}$.

2.3. *Task Models*

Information gathering and fusion, within our geometric framework, consists of choosing a parametric representation and determining the values of the unknown model parameters. As stated at the outset, our work fundamentally rests on the tenet that this is a purposeful, *directed* activity—the priorities of the current goal should influence the information-gathering process. This can be viewed as a way of optimizing the use of limited computational resources. Instead of gathering all possible information about the environment, the system should concentrate on those geometric aspects that are the most relevant or have the highest value for the current application.

This may be an open-ended interaction: attempting to gather information may depend on further information-gathering tasks. The dynamics of this process is be governed by *what* information we are seeking, the *value* we place on that information, and the *costs* associated with the search. This point of view naturally suggests a decision-theoretic approach (Berger 1985). We use *utilities* to reflect the value of information, quantify the costs of information processing, and consider the problem of maximizing the *net* gain of information. We note that this is, in essence, the basis for the study of experimental design (Fedorov 1972; Mendenhall 1968; Silvey 1980).

Geometric Transformations

Robotic tasks often use information in a form different from or independent of a given geometric parameterization. For example, as noted in the introduction, when lifting an object, an estimate of center gravity may be needed to compute the initial lift vector. Under appropriate assumptions about density, the center of gravity can be computed from the object shape and size. Therefore for this task the sensor system should concentrate on obtaining a good estimate of size and shape parameters so as to produce a good estimate of center of gravity.

To express these relationships, we introduce some ancillary transformations, $I(\boldsymbol{p})$, indicating how requested information is related to model geometry. For example, if the requested information is volume and we are using a rectangular representation, we can relate the description of a rectangle (see example 1) to its area by:

$$I(\boldsymbol{p}) = I(l; \boldsymbol{a}) = a_1 a_2 a_3.$$

We note the following two special cases of I as being of particular interest:

1. *The projection function.* In this case I restricts attention to a subset of the parameter space. For example, we may only be interested in the shape description of an object, even though the geometric model includes position information.
2. *The indicator function.* In this case, I encodes a proposition. It is then possible to formulate the problem so that the result of estimation is an indication of whether that proposition is true, false, or not completely decidable (true with probability t and false with probability $1 - t$) based on the available sensor information.

The latter form is of particular interest for those who model information using logic or similar qualitative descriptions. It implies that we can use the same framework to determine quantitative (point-based) quantities and qualitative (propositional) quantities.

Utilities

Any application using sensor information must confront the fact that error-free point estimates are not possible. Sensor uncertainty, sensor resolution, and bounded computational resources limit the accuracy of any sensor-based judgment. Sensor-based systems must therefore be able to tolerate some error. The types of errors that can be tolerated may vary considerably between applications and may have substantial effect on the information-gathering process. For example, gripping an object 2 cm wide using a parallel gripper with an opening of 5 cm implies that relative position accuracy within ± 1.5 cm will ensure a successful grasp. It would be a waste of time and effort to refine a position estimate past this level. By the same token, a peg-in-hole method using compliance may be characterized by a *bias* toward one-sided errors, and a smoother, more

graceful performance degradation as the correct estimate of relative position varies from its true value.

Consequently, an important element of an information request is some quantification of the effect of errors on task performance. Therefore we require that an information request make reference to a function $u(p, \hat{p})$, where $\hat{p}$ is an estimate of the unknown parameters, and interpret this function as a decision-theoretic *utility*. We note that u can be extended to a function of the form $u(p, a)$, where a represents a generic action from a set of possible actions $\mathcal{A}$. A variety of utility/cost formulations have appeared in the literature. The most common utility formulation is the *quadratic* utility, though others such as the one-zero utility have also been considered.

The following example illustrates three different tasks, all using the same basic representation but focusing on different parts of the parameter vector with different accuracy requirements.

Example 4 Consider a parallel gripper with jaw travel between 2 cm (minimum closing distance) and 4 cm (maximal opening distance) manipulating boxes on a table. The geometric representation is defined in example 1, where positions are restricted to the plane of the table, and the sensor description is given by example 3. To manipulate the object, the system must make three decisions:

1. Is the box of a manipulable size?
2. What is an approach vector that will place the gripper around the box?
3. Given that the box is in the gripper, what is a reasonable lifting force for moving the box?

For the first question, we define a mapping that determines what values of length and width parameters represent manipulable objects:

$$I(p) \mapsto \begin{cases} \text{yes} & \text{if } a_1 \text{ or } a_2 \text{ is between 2 and 4 cm,} \\ \text{no} & \text{otherwise.} \end{cases}$$

Then, if we assume that the consequences of both types of wrong decisions (trying to manipulate an unmanipulable object or deciding not to attempt an object that is in fact manipulable) are equal, we can define a utility as:

$$u(I(p), a) = \begin{cases} 1; & I(p) = a, \\ 0; & \text{otherwise,} \end{cases} \quad p \in \mathcal{P}, \quad a \in \{\text{yes, no}\}.$$

In some circumstances, the effects of one error may be more detrimental than the other. For example, the time lost trying to manipulate something that is not manipulable may be more costly than just leaving it undisturbed and looking for a more suitable object. In this case, we would adjust the weights so that the case ⟨no, yes⟩ has a value between 0 and 1.

In the case of determining an approach vector, we introduce a set of possible approach vectors, $\mathcal{V}$, and describe the problem choosing a suitable vector. Then, assuming a suitable collision detection algorithm is available, we can describe the problem using just a utility as:

$$u(p, v) = \begin{cases} 1; & \text{if gripper would encompass,} \\ 0; & \text{otherwise,} \end{cases}$$

$$p \in \mathcal{P}, \quad v \in \mathcal{V}.$$

Both of the above examples have a geometric constraint that leads to a 0-1 type of formulation. That is, either the constraint is satisfied and the action succeeds, or the constraint is not satisfied and the task fails.

Computing a weight to facilitate picking up the object is a task that is more tolerant of small errors. It suffices to have a "close" estimate of weight and to assume that the control algorithms will adapt on-line. Thus we describe this task as a transformation from object descriptions with density d to weight given by:

$$I(p) = I([x, y, \alpha, a_1, a_2, a_3]) = d\, a_1\, a_2\, a_3$$

and a *quadratic* utility:

$$u(I(p), I(\hat{p})) = -(I(p) - I(\hat{p}))^2.$$

The effect of the quadratic utility should be contrasted with the 0-1 utility. Namely, the 0-1 expresses a tolerance interval within which the task succeeds and variations have no effect on performance, and outside this interval the task simply fails. The quadratic utility express a performance *degradation* with no notion of success or failure.

The Cost of Gathering Information

An estimate can nearly always be refined by using more observations and more computation. Therefore the *value* of an estimate (in terms of its utility) must be weighed against the *cost* of gathering the information needed to make the estimate. What factors constitute costs and the trade-off between those factors can be a complex and involved problem in its own right (Keeney and Raiffa 1976). In our work, we concentrate on *time* costs. The time costs involved in the process of gathering and aggregating information are:

1. Time to select a control sequence.

2. Time to move to the specified configuration.
3. Time to gather and integrate new information.

These costs may depend on many factors, including the choice of sensor control parameters, the values of the unknown parameters, the organization of the sensor system, and the external constraints imposed by the geometry of the current situation. Typically, costs have been taken to be a linear function of sample size. These formulations have given rise to a number of results in linear-quadratic-Gaussian and linear regression experimental designs (Fedorov 1972).

In the most general setting, we denote the time to change from a sensor configuration w_n to w_{n+1} by

$$c_s(w_n, w_{n+1}, p).$$

To simplify the notation, we assume the current position is known and use the simpler form $c(w_{n+1}, p)$ to represent the cost of taking another observation. We note that cost formulations in the literature do not usually depend on p, the unknown parameter. In fact, the effect of p on the cost of executing an action may yield information about its value. For example, the amount of time it takes to move to the other side of an object yields information on its size.

Example 5 A natural model for time costs is a deadline model. In this case, we specify a nominal maximum time and also how important it is to meet that deadline. One possible deadline description is:

$$c(w, p) = \left(\frac{t_e + t(w, p)}{t_p}\right)^h, \quad h \geq 1 \qquad (5)$$

where t_p is the deadline for the sensing task, t_e is the current elapsed time, $t(\cdot,\cdot)$ is the time taken to execute w when the unknown parameters are p, and h is a factor governing how "hard" the deadline is. For large h, the deadline acts as a barrier, whereas for $h = 1$ the cost growth is strictly linear.

3. Review of Bayesian Techniques for Data Fusion and Experimental Design

In this section we summarize the basic principles of decision theory and illustrate decision-theoretic solutions using the examples of the previous section. For a more complete reference see Berger (1985). For other applications of decision theory to robotics problems, we refer the reader to Cameron (1989), Coles et al. (1975), Durrant-Whyte (1988), and Jacobs and Kiefer (1973).

3.1. Data Fusion

A standard Bayesian decision-making framework takes the following general form (Berger 1985): For a fixed sensor model (i.e., the uncertainty envelope contains only one geometric model and one sampling density), the sensor model gives rise to a conditional probability distribution:

$$f_z(z \mid p, w, e) = f_V(z - H(p, w, e)). \qquad (6)$$

Assume w and e are known (for simplicity we no longer explicitly indicate them). Given a prior density, π, over unknown model parameters, *Bayes' theorem* describes how to compute the new probability density over the unknown parameters:

$$\pi(p \mid z) = \frac{f_z(z \mid p)\pi(p)}{\int_{\mathcal{P}} f_z(z \mid p)\pi(p)\, dp}. \qquad (7)$$

This updating process can be iterated over time using independent observations, over sensor configuration by adjusting w, and over sensors by substituting different sensor descriptions into (6). Consequently, the basic representation of parameter uncertainty is the probability density of the parameters. We note that in the case of bounded sampling densities, this update can also include the *elimination* of portions of the parameter space. Hence this expression includes the incorporation of *error bound* information, as well as probabilistic information.

3.2. Decision Making

In Bayesian decision theory, decisions are made by finding that action or estimate that has the maximal *expected* payoff relative to the current parameter uncertainty. In other words, given a density, π, on $\mathcal{P}$ and a utility, u, we can compute the *expected payoff* of a decision $\hat{p}$ as:

$$\rho(\pi, \hat{p}) = E^{\pi}[u(p, \hat{p})] = \int_{\mathcal{P}} u(p, \hat{p})\, d\pi(p).$$

The optimal decision is that having the maximum expected payoff:

$$p^* = \arg\max_{\hat{p}} \rho(\pi, \hat{p}).$$

Alternately, in the case of a nontrivial transformation, I, we have:

$$\rho(\pi, I(\hat{p})) = E^{\pi}[u(I(p), I(\hat{p}))].$$

The optimal decision is then $I(p^*)$. For convenience we define, for a fixed task, the following two functions:

$$\delta(\pi) \mapsto I(p^*),$$

$$r(\pi) = \rho(\pi, \delta(\pi)).$$

The first is simply a decision rule mapping probability distribution representations to decisions, and the second is the payoff of a decision with respect to a given distribution.

Example 4 (cont.) The decision rule for the first task would be whichever of yes or no has higher probability of being correct, and the payoff is the probability of being correct. If the weights are changed, then the payoff becomes weighted probability, and the optimal decision is the choice with highest weighted probability. Note that this decision only requires knowledge of either length or width to an accuracy of 2 cm—relatively little information.

The decision rule for the second task is the vector with highest probability of succeeding, and the payoff is that probability. This requires knowledge of one size parameter and object location. Because observation errors are bounded, it is possible to determine a vector of probability one, in which case there is no point in processing more observations (for this task).

The decision rule for the final task is the average (conditional mean) weight, and the payoff is the negated variance of the estimation error. In this case, the task requires information on all three size parameters, and in most cases, more observations result in a better (lower mean-square error) estimate.

As this example illustrates, the task descriptions of the last section indicate the appropriate decision space, provide the means for making a decision when uncertainty exists, and describe the value of processing additional observations.

3.3. Choosing Viewpoints and Features

We consider the problem of choosing sensor control parameters in terms of the theory of *experimental design* (Fedorov 1972; Mendenhall 1968). Experimental design is concerned with the problem of maximizing the information gained from an experiment under cost constraints. We assume we have some set of experimental actions, $\mathcal{A}$, and some decision rule δ. We attempt to find the action or sequence of actions that maximizes the *net payoff* (average utility minus experimental costs) of a decision made by δ.

There are two different perspectives on solving an experimental design problem. The first perspective corresponds to off-line planning. That is, before any data is taken, we select both the optimal number of samples and the optimal sensing strategy. This, of course, has the disadvantage that sensor behavior is not tailored toward individual circumstances. Instead, the optimal strategies are those that, when averaged over all anticipated situations, result in the best (in the sense of net payoff) final decision.

It is important to note that such strategies (1) depend critically on prior information (that is, if any prior assumptions were incorrect, then the resulting sensing strategies are nonoptimal); and (2) depend critically on the type of sensing task (that is, we would need to compile lists of sensing strategies indexed by the type of information to be gathered, the utility function, the cost, and the prior information).

These points suggest that batch rules are most appropriate when sensor models and prior distributions are well known, there are a small number of possible sensing tasks, or when the net payoff of a decision is essentially independent of observations. A well-studied example of the last case is optimizing experimental parameters in the context of linear regression under Gaussian noise (Fedorov 1972). Several different optimization criteria, including the determinant, trace, and maximum eigenvalues of the variance-covariance matrix, have been documented (Silvey 1980). Within the control literature, Müller and Weber (1972) consider the problem of finding the measurement system design maximizing a suitable norm of the observability or controllability of a system linear in both state and control. The norms they discuss are the trace, determinant, and maximum eigenvalue of the observability matrix. Mehra (1974) combines and extends these results to include time-varying systems and randomized designs.

The *sequential* experimental design problem is to incrementally choose the sequence of measurements maximizing the net value of the final decision online. Sequential procedures are appropriate when the range of situations faced by the system is large, the unknowns and control parameters are coupled, and there is large variation in the effect of observations on unknown parameters. For example, in estimating rotations it often turns out that some viewing positions immediately constrain angle, whereas others give very little rotation information. Similarly, when estimating size with a monocular camera, the information gained about size depends on knowledge of the viewing distance (to fix the aspect ratio) and knowledge about rotations (to determine the effect

of foreshortening). If the prior information about position and rotation is poor, it is difficult to anticipate which points of view and selection of features will yield the best estimate of rotation and/or position. In some cases, the first measurement may suffice. In others, three or four measurements may be required. Hence for the class of general geometric sensing tasks that we have outlined, we advocate online sequential procedures for choosing viewpoints and sample size.

The difficulty is that for general sampling densities and payoff functions, the optimal strategy is highly dependent on the number of observations (look-ahead) the system uses. For example, when the relationship between unknown parameters is highly coupled, a one-step look-ahead is sometimes not enough for adequate system performance; there may be no *single* observation that has positive net value, but there may be a sequence of two or more that do (the example of a monocular camera estimating distance is a case in point). In general, the optimal procedure may use a number of samples, N, which is a random variable that cannot be bounded. A significant amount of theoretical work in experimental design has been devoted to the study of finite horizon approximations and their relationship to the optimal procedure. Because we are working in a time-constrained application, we use a fixed sample size n-step look ahead approximation (Berger 1985).

The Sensor Action Space

Within the above paradigm, the simplest approach to sensor observation planning is to identify the set of available sensing actions, $\mathcal{A}$, with the a priori supplied control space $\mathcal{W}$. Recall that the latter set represents all information-gathering alternatives available to the system, and accordingly may describe a large variety of sensing alternatives. In general, these actions correspond to (1) the selection of processing parameters (e.g., thresholds), (2) the selection of sensor position or configuration, and (3) the selection of features to observe. Exactly what actions are available depends on the details of the sensor, the geometry of the situation, and the predictability of observation. In general, the constraints imposed by the structure of the sensor and its interaction with geometry must be treated individually for each sensor. For example, Cowen (1988) details the computation of feasible actions for a vision sensor for objects in a known position. Hutchinson et al. (1988) discuss similar computations for a multisensor system consisting of laser range sensors and vision sensors.

The effect of uncertainty is to decrease the predictability of the effects of action, which may, in turn, lead us to alter the size or structure of the action set. For example, given an object in a known pose and known position, the features observable from any point of view are predictable, and viewpoint selection can be done in an object-centered coordinate system. This means that the space of actions can be identified with the set of viewable features, and viewpoint can be coupled to feature selection. Conversely, if the object is in an unknown pose, then the outcomes of actions (observed features) are no longer predictable because of limited sensor scope. That is, not only is the type of information that will be observed unpredictable, but we are no longer guaranteed to observe *anything*—sensor control is *open-loop*.

However, if the sensor has detected the object, then a *closed-loop* control model can be used. For example, if an object is in an unknown position but the sensor has observed some feature known to lie on the object, we can couple the sensor position to observations and again work in an object-centered coordinate system. Moreover, if we have some information about the topology of the viewed object, we can navigate over the surface of the object. This increases the reliability of feature detection and also increases knowledge about interfeature correspondence (for example, it allows us to solve for the correspondence parameter in example 1). In the remainder of this article we focus on the closed-loop model and refer to Hager and Mintz (1987) where we discuss the open-loop model.

To employ a closed-loop model, we must describe the relationship between the full control vector of the system, $\boldsymbol{w}$, the unknown parameters, $\boldsymbol{p}$, and a reduced control space, $\mathcal{A}$. In this article, we assumed that this relationship can be expressed in the form:

$$L(\boldsymbol{p}, a) = \boldsymbol{w}, \quad \boldsymbol{p} \in \mathcal{P}, \quad \boldsymbol{w} \in \mathcal{W}, \quad a \in \mathcal{A}$$

where a is a sensor action and $\boldsymbol{w}$ is the sensor configuration that would result by taking sensing action a when the state of the world is $\boldsymbol{p}$.

Example 6 For example, we can express the restriction of sensor motion in a plane (parameters x_c, y_c, α_c) to a planar, object-center, polar coordinate system with parameters (α, r) by:

$$\begin{bmatrix} x_0 + r\cos(\alpha) \\ y_0 + r\sin(\alpha) \\ \alpha_c \end{bmatrix} = \begin{bmatrix} x_c \\ y_c \\ \alpha_c \end{bmatrix}.$$

This form is used later in this article to describe a

camera coupled into an object-centered coordinate system using image feedback.

In this context, we note that it is important to differentiate between an *external* control loop, which maintains a particular sensor/world relationship, and the *model* of that relationship, which is expressed by L and used for sensor planning.

In a complex system, several such constraints could be available to the system. For example, one such constraint might implement tactile compliance, another may implement curvature-based exploration, and another may implement visual tracking. By intelligent choice of these feedback constraints, the complexity of the search process may be significantly reduced.

Formulating the Sensor Control Problem

The decisions of *how much* information to gather and *how* to gather it are based on the expected gain in payoff from an observation relative to the cost of gathering and processing that observation. This trade-off is expressed by:

$$n(\pi, a) = E^{\pi}[E_z[(r(\pi_{p|z}) - r(\pi)) - c(w, p) \mid w = L(p, a), p]], \quad a \in \mathcal{A}. \tag{8}$$

Note this covers the case where there is no control constraint by identifying $\mathcal{A}$ and $\mathcal{W}$ and defining $L(p, a) = a$.

This quantity is the expected *net* gain from an observation (averaged over current parameter uncertainty and sensor observation uncertainty) minus the expected cost of processing the next observation. The best choice of a is that maximizing this quantity (which depends on w through f_z as derived from the sensor model). If the resulting net gain is negative, then cost of gathering and processing an observation is larger than the gain in information, and the system should stop taking observations and make a final decision.

For a given cost formulation $c(\cdot)$, an optimal sampling plan relative to a prior π is a vector of actions a that satisfies:

$$n(\pi, a^*) = \max_a n(\pi, a). \tag{9}$$

Example 4 (cont.) Based on the previous discussion, we can qualitatively describe the sensing strategies for each of the three example tasks.

The sensing actions for the first task concentrate on localizing either length or width. So, for example, observing the corner located at the origin yields no information—the expected marginal gain is 0. The best sensing strategies are those that measure the length of one side. Depending on the type of sensor, it is entirely possible that the location parameters are left untouched.

The second task requires location, some orientation information, and at least one of width or height. The last will have been determined by the last task, but location and orientation may not have been. If not, the obvious strategy is to localize the corner at the origin of the coordinate system, as it gives direct location information.

The third task is again independent of location but requires the height, length, and width. Thus the expected marginal payoff of the corner at the origin is 0, and the gain of observing a corner rises depending on the number of size parameters it determines. Consequently, given that location was established in the previous example, the best corner to observe is clearly that with object position $[a_1, a_2, a_3]$, as it depends on all three required parameters.

3.4. Discussion

These methods have the intuitive appeal of mathematical simplicity, clarity and generality. In essence, by describing the sensor, the geometric representation, and the task, we determine the solution to a problem. However, this philosophy has the following drawbacks:

1. The computation of Bayes' theorem requires a representation for probability distributions that can adequately represent updates from nonlinear, coupled, non-Gaussian sensors and is also computationally tractable.
2. The computation of a decision and its payoff requires the evaluation of an integral, as well as a maximization.
3. Computation of optimal sensor control values requires two additional integral evaluations and a maximization.
4. Bayes' theorem is formulated for known sensor models and so must be modified to account for model uncertainty.

One way out of these difficulties is to restrict attention to those cases where the updating procedure is effectively calculable, and to approximate problems with no effectively computable solution by those that do. This is, in effect, the route taken by those who use the extended Kalman filter (EKF) to implement sensor data fusion, for example Ayache and Faugeras (1988), and Durrant-Whyte (1988), to list just two. However, as demonstrated in Hager

(1988), the EKF only suffices as an approximate solution in a restricted range of cases—roughly those where prior uncertainty and model uncertainty are small.

In the next section we develop an approximation method that is appropriate for a wider variety of cases. This method is based on approximating the prior probability distribution using a grid-based representation and formulating Bayes' theorem for this representation.

4. Grid-Based Conditional Bayes Analysis

The generality and computability of the methods described in the last section depend largely on the representation of probability distribution functions. The problem is to find a class of probability distributions that is flexible enough to describe both prior and posterior distributions after updating with a nonlinear, coupled sensor description; can be easily transformed and integrated to accommodate a variety of task descriptions; and is still computationally tractable. We adopt, for our implementation, the class of probability distributions that can be described by *piecewise-constant* density functions. Intuitively, such densities are defined by choosing a partition of the parameter space and defining a probability associated with each set.

Densities that are not inherently piecewise-constant are approximated by a piecewise-constant density. For example, Figure 2 illustrates the approximation of three different densities by a piecewise-constant density function. From this we see that piecewise-constant densities can represent skewed, multimodal, and bounded distributions. Furthermore, Bayes' theorem, estimate calculation, and payoff calculation for this class of distributions are all relatively simple. Before proceeding to the general case, we illustrate the basic steps with the following example.

Referring to Figure 3, we consider a prior density

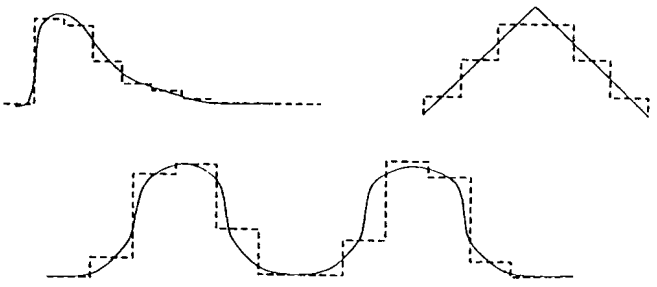

Fig. 2. Approximation with piecewise-constant prior densities.

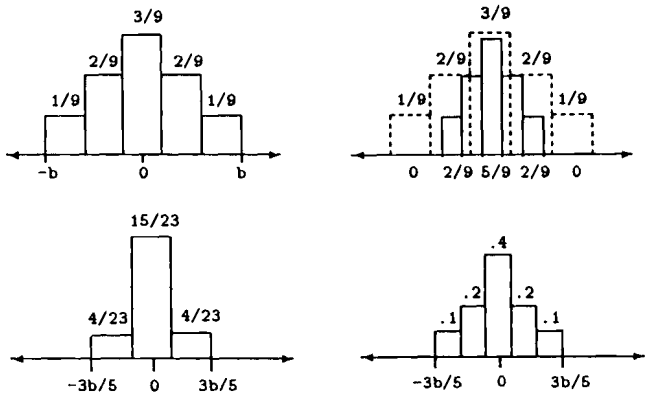

Fig. 3. An example of a scalar update.

with support on the interval from $-b$ to b, as shown on the upper left. Given an observation $z = 0.0$ with uncertainty described by a scaled ($-b/2$ to $b/2$) version of the prior density, we compute the posterior density by:

1. Computing $\int_{\Omega_i} f(z \mid p)\, dp$ for each element Ω_i of a partitioning of the parameter space. These values are written under the density on the upper right.
2. Multiplying the prior value for each partition element by the value calculated in the last step and normalizing the result, giving the density on the lower left.
3. Repartitioning and interpolating from the old partition to the new partition. We note that the values in the figure are approximate. The true values are 12/115, 1/5, 9/23, 1/5, 12/115.

Referring to Figure 4, we show how the process changes for the scalar system $z = p^3 + V$. Namely, we first project the density on p through the system description p^3 to compute a probability description on the range space of h. We then apply the process described in the previous example to the transformed density and reflect the computed probabilities back onto the original partition. So, if we were to update as described in the previous example, we would compute a distribution with smaller support if

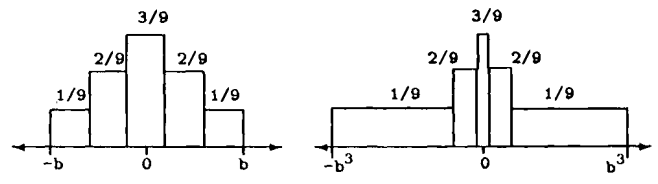

Fig. 4. The projection of a density with respect to the function $h(p) = p^3$.

$b > 1$ and larger support if $b < 1$. If $b = 1$, only the calculated probabilities change.

4.1. General Grid-based Probability Density Updating

In generalizing to n dimensions, the basic process remains unchanged, though the management of the partition and the computation of the projection become more difficult. As notation, we adopt the convention that *time* is indicated by a superscript, and partition element indices are indicated by a subscript. The only exception is observations, where time indices continue to be shown as subscripts. Ω_α, $\alpha = 1, 2, \ldots, n$, is a given finite partition of $\mathcal{P}$, $\boldsymbol{\lambda}^k$ is the vector of probability values for each set, and the two together define the piecewise-constant density π^k.

Given an observation z_{k+1} and making use of the conditional independence of the z_i given $\boldsymbol{p}$, we can apply Bayes' theorem (7) to π^k:

$$\lambda_i^{k+1} = \pi^{k+1}(\Omega_i) = \frac{\int_{\Omega_i} f(z_{k+1} \mid p)\, d\pi^k(p)}{\int_{\mathcal{P}} f(z_{k+1} \mid p)\, d\pi^k(p)} = \frac{\int_{\Omega_i} f(z_{k+1} \mid p)\, d\pi^k(p)}{\sum_{i=1}^{n} \int_{\Omega_i} f(z_{k+1} \mid p)\, d\pi^k(p)}. \tag{10}$$

Because $\pi^k(\cdot)$ is a piecewise-constant function, the following equality is now valid:

$$\int_{\Omega_i} f(z_{k+1} \mid p) d\pi^k(p) = a_i^k \int_{\Omega_i} f(z_{k+1} \mid p) dp \tag{11}$$

where $a_i^k = \lambda_i^k / \mu(\Omega_i)$.[1] Let $\cdot$ denote inner product. We can now rewrite (10) in an iterative form:

$$\lambda_i^0 = \pi^0(\Omega_i),$$

$$x_i^k = \int_{\Omega_i} f(z_{k+1} \mid p) dp, \tag{12}$$

$$a_i^k = \lambda_i^k / \mu(\Omega_i), \tag{13}$$

$$\lambda_i^{k+1} = \frac{a_i^k x_i^k}{\mathbf{a} \cdot \mathbf{x}}. \tag{14}$$

Note that the actual computation of Bayes' theorem, (14), only requires a parallel multiply, a sum of vector elements, and a vector multiplication by a scalar. In terms of the previous example, $\boldsymbol{\lambda}^0$ is the vector of initial probabilities shown on the upper right of Figure 3, $\mathbf{x}^0$ are the values written under the histogram on the upper left, and $\boldsymbol{\lambda}^1$ are the values given on the histogram on the lower left.

When the observation system is not described by a simple identity, we must compute the value of an integral expression depending on the function describing imaging geometry:

$$x_i^k = \int_{\Omega_k} f(z_{i+1} - H(p))\, dp.$$

We approximate this expression by linearizing H for each grid element and computing this integral piecewise for each grid element. By a change of variable and defining $|L|$ to be the Jacobian of the function H, which must now be fully determined and evaluated at the center point of the grid element, we derive the following approximation:

$$x_i^k = \frac{F_V(\Omega_k')}{|L|} \quad \text{where } \Omega_k' = \{z_{i+1} - H(p) \mid p \in \Omega_k\}.$$

However, the Jacobian term is the ratio of the area of differential elements before and after the mapping H, which in turn is approximately the ratio $\mu(\Omega_i')/\mu(\Omega_i)$. Substituting this into (13) and adjusting (12) accordingly yields the following modified forms:

$$x_i^k = F_V(\Omega_k')$$

$$\text{where } \Omega_k' = \{z_{i+1} - H(p) \mid p \in \Omega_k\}, \tag{12a}$$

$$a_i^k = \lambda_i^k / \mu(\Omega_i'). \tag{13a}$$

Expression (12a) requires the computation and representation of the sets $H(\Omega_i)$, $i = 1, 2, \ldots n$. We refer to this collection as the *range grid*. In general, the exact form of a projected set is difficult to represent, so in practice we approximate the projection using a rectangular representation. In this case, the value of the integral can be determined though simple table lookup. In general, the choice of what to use as an approximation and how to compute it is governed by the ease of computing that particular representation, the accuracy of the representation, and the ease with which (12a) can be computed. Because these properties change from application to application, the behavior of a particular approximation to (12a) must be carefully understood.

Example 7 We construct the matrix

$$H = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix},$$

which can be interpreted as a rotation of the parameter space through an angle θ about the origin. The observation system is described by

$$z = Hp + V,$$

1. The function μ is interpreted as a generalized volume measure, also known as Lebesgue measure.

where H is as given, and p and V are vectors of mutually independent, bounded random variables. Referring to Figure 5, on the upper left we see a domain grid of rectangular elements. Each element Ω_i has an associated probability $\lambda_i = P(p \in \Omega_i)$ computed from a given prior probability distribution. These values, together with the values $\mu(\Omega_i)$, define the vector $\mathbf{a}^\lambda$ given by (13a).

When an observation is made, the domain grid is projected through the sensor description, H. The form of $\mathbf{H}$ above leads to a projected grid of the form depicted on the upper right of Figure 5. One method for computing this projection is to evaluate a point on the middle of each border of a domain element and construct the bounding box of these points. The grid in the lower left is the resulting range grid. Note that because of representation errors, there are gaps in the range grid. Another possibility would be to project the corners of the grid elements in which case the range grid elements overlap each other.

Given an observation, z_k, we compute the probability of the intersection between each range grid element and the same space of the observation (expression (12a)). Because we are using a rectangular representation for projected grid elements, this value is now easy to compute using a lookup table of probabilities. The result is the vector $\mathbf{x}^\lambda$.

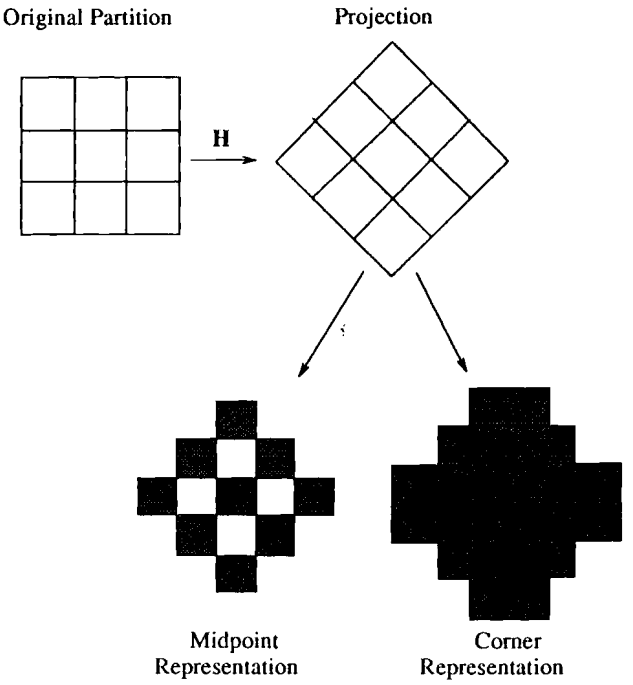

Fig. 5. An illustration of two different methods for computing the projection of elements of the parameter space grid.

These vectors can now be combined using (14) to produce the new vector of probabilities on the original domain grid, and the process repeats.

In the above example, we noted it is possible to obtain a range grid that has "gaps" or "overlaps" as a result of projection errors. Numerically the effect of such projection errors depends on the magnitude of the errors in relation to the size of the support set of the sampling distribution. Very small projection errors change the value of (12a) and (13a) slightly but do not affect the final outcome in a significant way. However, a large gap could lead to a situation where the observation falls on a gap, and the sampling distribution does not have large enough support to intersect a range grid element. In this case (12a) will yield a vector of zeros, and the update will fail. More generally, we see that if there is any gap in the representation, it is possible that the true parameter value will be excluded from the support of the conditional density.

Hence, in order for the method to function correctly, it must be the case that an approximate projection, Ω_i', of a grid element i *contains* the true projection:

$$H(\Omega_i) \subseteq \Omega_i' \quad \text{where } H(\Omega_i) = \{H(p) \mid p \in \Omega_i\}.$$

In this case we can guarantee that any parameter vector that is a possible candidate for having generated the data will not be excluded. We have recently discovered a principled way of ensuring this condition holds using interval analysis (Moore 1966). For example, when using the midpoint projection, we *scale* the partition elements (making it a finite *covering*) until the criterion is satisfied. The updating rule remains unmodified, though (12a) is computed using the covering rather than the partition. In the case above, for example, a rotation of 45° requires an expansion of grid elements by a factor of 0.5 to account for projection errors. The resulting range grid is equivalent to the results of projecting the corners of the grid elements.

Finally, when the output vector of the system is of lower dimensionality than the input vector, we use exactly the same process, though these expressions are only approximately correct for this case.

4.2. Adding Robustness

We now consider the effects of variation in the distribution of V as a result of the influences of the parameters we are estimating, the choice of control, and unmodeled variations. In the statistical literature, this problem is called *model robustness*; Huber

(1981) presents a number of results related to this issue.

First we need a definition and a simple related result:

DEFINITION 1: Given two unimodal measures π_1 and π_2 centered about the origin, we say π_2 is *at least as peaked* as π_1 (Sherman 1955), denoted $\pi_1 \overset{p}{\leq} \pi_2$, if, for all $A \in \mathcal{A}_n$, ($\mathcal{A}_n$ the class of compact, convex, symmetric sets in $\mathfrak{R}^n$ centered about the origin),

$$\pi_1(A) \leq \pi_2(A).$$

For example, a Gaussian distribution, F_1, on $\mathfrak{R}$ is at least as peaked as another Gaussian, F_2, if F_1 has a smaller variance.

THEOREM 1: Let u be a quasiconcave function bounded from above and symmetric about the origin, and π_1 and π_2 be two continuous distribution functions unimodal about the origin such that $\pi_1 \overset{p}{\leq} \pi_2$. Then $E^{\pi_1}[u(x)] \leq E^{\pi_2}[u(x)]$.

The implication is that any approximation errors should lead to distributions *less* peaked than the exact result. Similarly, a robust algorithm should yield a result no more peaked than the modeling uncertainties warrant.

Classes of Probability Distributions

We first consider the effects of parameter dependence and variation of the density describing the random variable V. We first note that, by considering the possible range of sampling distributions and their associated uncertainty over the parameter space, we can construct the set of all possible sampling densities. Based on the above theorem, the worst case sampling distribution is that one that leads to the least peaked posterior distribution. The general problem of isolating worst case distributions is unsolved, but several special cases can be cited. For example, in the case of a Gaussian prior and a class of Gaussian sampling distributions, the worst case distribution is the least peaked member. Similar results hold for uniform distributions. Zeytinoglu and Mintz (1988) have shown that, for the case of a 0-1 loss under suitable restrictions, the minimax solution maximizing over the unknown parameters and a class of sampling densities while minimizing over the class of monotone decision rules uses the *upper envelope* of the class of sampling distributions. The upper envelope of a family of distributions is no more peaked than any member of the family. These observations suggest that a reasonable approach to model robustness for quasiconcave utilities is to choose a distribution no more peaked than any member of the class of possible sampling distributions.

If there is a large variation of sampling densities over the range of unknown parameters, this approach could lead to a significant performance degradation by not incorporating the distributional information over unknown parameters. However, we have not found this to be the case in practice.

Accommodating Model Variation

We now turn to the problem of making the updating method robust to modeling errors in the measurement system model H or geometric model g. We can consider two types of errors: systematic and nonsystematic. Systematic errors are unknowns that remain constant over the course of taking data. This type of error can be best handled by augmenting the parameter vector with the systematic error parameters and performing estimation in the larger space. That is, from a theoretical viewpoint, there is no difference between parameters of interest and systematic error parameters.

Nonsystematic variation may arise from two sources: variations in the sensor system itself or discrepancies between the subject and the geometric model. This distinction is important: the former is an error that must be tolerated by fusion, whereas the latter may be an important source of information about the suitability of the model. When updating, the effect of both of these variations is to increase the size of a range grid element. That is, for a "perfect" sensor model, the observations expected for a set of parameter values Ω_i is $H(g(\Omega_i), w, e)$. However, if the sensor model or geometric model has some variation in addition to the model parameters, the set of possible observations is enlarged by this uncertainty, and consequently the following relation must hold between a parameter space subset, Ω_i, and its approximate projection Ω_i':

$$\Omega_i' \supseteq \bigcup_{H \in \mathcal{H}, g \in \mathcal{G}} H(g(\Omega_i), w, e) \quad i = 1 \ldots n.$$

To keep the computation of this expression simple, we approximate the enlargement of range grid elements with a vector of tolerance parameters, ε. So, for example, if we are representing range grid elements by bounding boxes parameterized by a vector of minimal elements l and a vector of maximal elements u, then the enlarged range grid element is defined by $l - \varepsilon$ and $u + \varepsilon$.

To estimate model variation, we decompose ε into the tolerance component resulting from the sensor model variation (which we assume is known) and the component resulting from geometric model variation: $\varepsilon = \varepsilon_s + \varepsilon_m$. We then note that for any given ε_m, there is an associated posterior probability. That is, ε_m parameterizes a *class* of posteriors: $\pi(\Omega_i; \varepsilon_m)$. Observe that there is a minimal value of ε_m, ε_l, that is either 0 or a positive number such that choosing a component of ε_m to be smaller than the corresponding component of ε_l causes the posterior distribution to become inconsistent (it places 0 mass everywhere). Given that F_V is a distribution taking values in $[-d, d]$ and the sensor tolerance is ε_s, it can be shown that there is an upper bound on the observed tolerance given by $\varepsilon_u = \varepsilon_l + 2d + 2\varepsilon_s$. We further observe that larger values of ε lead to less peaked distributions. Therefore $\pi(\Omega_i; \varepsilon_l)$ provides an "upper bound" on the true probability distribution, and $\pi(\Omega_i; \varepsilon_u)$ provides a "lower bound." In those cases where $\varepsilon_u - \varepsilon_l$ is small, using ε_u provides a reasonable (pessimistic) estimate of ε_m.

4.3. *Estimation and Payoff Computation*

The transformation functions I are divided into three types: reductions of the parameter space, transformations to a discrete space, and transformations to a continuous space. Reductions of the parameter space require integrating out over the unwanted dimensions. This is easily done by summing the elements of the grid along these dimensions and placing the results in the lower dimensional grid. For discrete transformations, the probability of each of the discrete alternatives is tabulated over the grid. In most cases, this is simply summing the probability contained in the inverse projection of each element. Continuous transformations require a projection similar to that used for (12a). The resulting grid is used as a representation of the transformed density function.

The best estimate of geometric parameters is that which maximizes the expected payoff. Computing such an estimate directly—that is, by maximizing payoff over all values of the parameter space—is generally too complex to perform quickly. In some cases the optimal estimate can be solved for directly; for example, in the case of a quadratic loss, the conditional mean is known to be the optimal estimate of parameters. In those cases where optimal estimate or decision is difficult to express in closed form, we take the approach of approximating the optimal estimate with some combination of relatively simple statistics such as mean, mode, median, or higher moments of the transformed distribution.

Payoffs are computed by integrating the task utility or loss over the transformed grid. This is usually a relatively simple operation as, because of the nature of the grid, the integral becomes a weighted sum of integrals of the utility or loss over a grid element. That is:

$$\int_{\Omega} u(p, \hat{p})\, d\pi(p) = \sum_{i=1}^{n} a_i \int_{\Omega_i} u(p, \hat{p})\, dp.$$

These integrals usually have relatively simple closed-form solutions, and so payoff computation is inexpensive.

4.4. *Implementing Sensor Search*

From a computational standpoint, (8) is expensive to compute. It requires two integrations of a computed function with respect to (possibly vector) variables. In the grid-based method, integration is carried out by evaluating the integral on each grid element and multiplying by the probability mass associated with that element. Therefore on a scalar machine, this has *superexponential* complexity; on a parallel machine it would be exponential. Furthermore, we would eventually like to carry out planning and fusion on different machines. If these machines are connected via a network, the communication of a complete grid carries a substantial communication overhead. Therefore we would like to reduce the size of the representation as much as possible.

In most robotic applications, the effects of sensor observation uncertainty are relatively small compared with the accuracy generally needed for effective task performance. What is more relevant, at least initially, is obtaining sensor observations that *overdetermine* the underlying model parameters. Stated another way, the most important aspect of an observation is its effectiveness at reducing gross geometric uncertainty, rather than its statistical effect on the posterior distribution. Moreover, in the grid-based method we can determine when this is true by the following simple rule: when the sample space corresponding to an observation is smaller than the smallest range grid element, the statistical properties of observations will have almost no effect on the updated distribution.

Example 8 To illustrate this point, consider the simple scalar example of a sensor system described by

$$z = p + v \quad -d \le v \le d.$$

Furthermore, let $p \in [-10d, 10d]$. We note that, in this case, the domain grid and the range grid are identical, because H is the identity function.

If the number of elements in the domain grid is $n \leq 10$, then any sensor observation will eliminate at least $n - 2$ grid elements. In this case, the value of the parameter space reduction through elimination of grid elements is generally more important than the final probabilities of the remaining elements. In fact, if $n - 1$ elements are eliminated, then any probabilistic information is below the resolution of the grid anyway.

In cases where geometric uncertainty is large relative to sensor uncertainty (i.e., the above rule holds), we simplify (8) by removing the inner expectation and computing:

$$n(\pi, w) = E^{\pi}[(r(\pi_{p|z}) - r(\pi)) - c(p, w) \mid z = H(p, w, e)]. \quad (8a)$$

That is, we do not average over the random variable V of the sensor model.

On the other hand, when the system has a set of observations that overdetermine the underlying geometry, the effect of parameter variation becomes small. In this case, we can fix p at a value $\hat{p}$, remove the outer expectation of (8), and consider only the effects of sensor noise and modeling error:

$$n(\pi, w) = E_z[(r(\pi_{p|z}) - r(\pi)) - c(p, w) \mid w, \hat{p}]. \quad (8b)$$

The amount of computation required for the remaining expression may still be prohibitive. In the case of large geometric uncertainty, we can further simplify the computation by disregarding the fine structure of the remaining integrals and restricting our attention to actions with relatively large net gains. That is, instead of evaluating the integrand at each grid element, we pick some subset of the parameter space, $\mathcal{B} \subseteq \mathcal{P}$, and find the average value for those points. Of course, the effectiveness of this procedure depends on a good choice of elements in $\mathcal{B}$ and insensitivity to minor variations in payoff. In a sense, this approximation can be viewed as a hypothesize-and-test approach. The value of a hypothesis generator is a trade-off between the cost of generating and evaluating the points in $\mathcal{B}$ and the quality of those points.

Choosing the maximal information viewpoint or description vector is a process of evaluating possibilities and choosing the point with the maximum net information gain. There are two types of actions to be considered: discrete and continuous. Discrete spaces must be dealt with in an intelligent combinatoric fashion. Continuous spaces can either be discretized or handled through a continuous minimization procedure.

For the problems we have considered, we maximize (8) over the allowed set viewpoints for each feature and choose the feature/viewpoint pair with the highest rating overall. The maximization method we use is a variation on well-known golden section search algorithms (Press et al. 1986). As such, these techniques are very weak—they use no information about the sensing system other than the evaluation of the current points in the action set and an initial bound on the maximum. This is an advantage from the point of view of generality but a disadvantage from the point of view of efficiency. As suggested in earlier works (Hager 1987; 1988), sensor control should properly be placed *in the sensor*, and more sensor-specific information should be used to enhance the control process of each sensor.

5. Analysis of Approximation Errors

In this section, we first present some basic, qualitative mathematical analysis of the behavior of the grid-based method, based on the notion of peakedness presented in the last section. These ideas will be used to evaluate the error characteristics of the method, its sensitivity to prior assumptions, and its ability to deal with envelopes of models. We then present some Monte Carlo simulation results for several example problems in order to verify the performance quantitatively.

5.1. Mathematical Error Analysis

Because the sampling distribution is bounded, one of the effects of Bayes' theorem is to eliminate portions of the parameter space that are incompatible with sensor observations. Consequently, the updating algorithm acts as a "root finder" until the remaining parameters are compatible with the observations up to observation uncertainty. After a few more observations, the grid does not contract with any great frequency, and most of the change in the posterior is the result of conditioning effects. We refer to this state as the *steady state* of the system and analyze the errors when it is in steady state. For the most part, this analysis is independent of particular choices of observation systems and so is qualitative rather than quantitative.

Single-Step Updating Errors

The error in updating is attributable to the approximation

$$\int_{\Omega_i} f(z_{k+1} \mid p)\, d\pi(p) \approx a_i \int_{\Omega_i} f(z_{k+1} \mid p)\, dp.$$

The denominator of the updating rule serves as a scaling factor. We assume that the difference between the true denominator and the approximated one is small. This leads to an approximation error (up to scaling) of the form:

$$e_i^{k+1} = \int_{\Omega_i} f(z_{k+1} \mid p)\, d\pi(p) - a_i^k \int_{\Omega_i} f(z_{k+1} \mid p)\, dp$$

$$= \int_{\Omega_i} f(z_{k+1} \mid p)(f_\pi(p) - a_i^k)\, dp. \tag{15}$$

In order to be concrete, we consider the magnitude of first-order errors in a scalar system (all of the results can be generalized to nonscalar systems). Partition elements are parameterized as $\Omega_i = [m_i - d, m_i + d]$, and we assume $f_\pi(\cdot)$ is a piecewise-linear function of the form $f_\pi(p) = a_i + b_i(p - m_i)$. Substituting into (15) and simplifying, we get:

$$e_i^{k+1} = b_i^k \int_{m_i - d}^{m_i + d} f(z_{k+1} \mid p)(p - m_i)\, dp.$$

Expanding $f(z_{k+1} \mid p)$, adopting the change of variable $p = p - m_i$, and defining $v = z - m_i$ leads to:

$$e_i^{k+1} = b_i^k \int_{-d}^{d} f_V(v - p)p\, dp.$$

We can further simplify by exploiting the symmetry of the integral and write:

$$e_i^{k+1} = b_i^k \int_0^d (f_V(v - p) - f_V(v + p))p\, dp. \tag{16}$$

This representation makes it clear that the magnitude of the error is related to (1) the local slope of the prior density function (b_i^k); (2) the local asymmetry of the sampling distribution (the effect of the difference in (16)); and (3) the size of intervals (d).

This suggests that, as is expected, a finer grid reduces error, and less peaked prior distributions lead to smaller errors. Thus a good gridding scheme attempts to grid finely in areas where the prior density changes rapidly, and coarsely in other areas. This keeps the error magnitude relatively constant throughout the grid.

Example 9 To give a graphic illustration of the sign and magnitude of errors, consider the case where f_V and π are described by symmetric triangle distributions parameterized by the width w as:

$$t(p, w) = 1/w - |p/w^2| \quad -w \le p \le w.$$

We fix $\pi = t(\cdot, 1)$, partition the parameter space into four equal regions numbered (left to right) from 1 to 4, and vary $f_V = t(\cdot, w)$ for values of w between 0.5 and 0.7. Figure 6 shows e_i, $i = 3, 4$ for three values of w while varying z. This shows the immediate effects of updating errors. Note that for z near 0, the error for element 3 is positive, while the error for element 4 is negative, indicating that the true distribution is more peaked than the approximation. As z moves to the right, the trend reverses. However, the peak of the posterior distribution is also moving so that the approximated final distribution is again less peaked than the true final distribution.

Figure 7 shows the expected error averaging over z while varying p. Again the result is that the approximated distribution is less peaked than the final distribution for almost all values of p. In particular, if we average these curves with respect to F_p, we get a positive value for element 3 and a negative value for 4.

This example illustrates another very important property of this method: for unimodal prior and sampling distributions, the expected error is positive

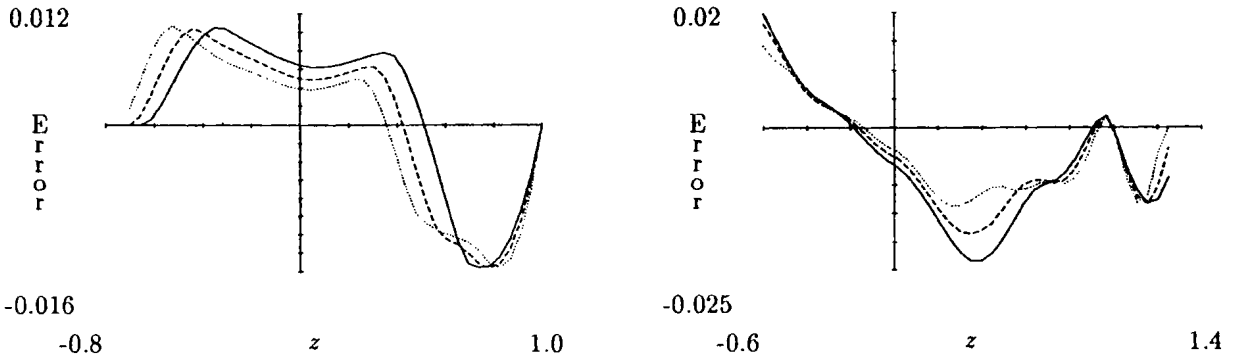

Fig. 6. Updating errors as a function of z *for element 3* (left) *and element 4* (right) *with values of* w = *0.7* (solid), w = *0.6* (dashed) *and* w = *0.5* (dotted).

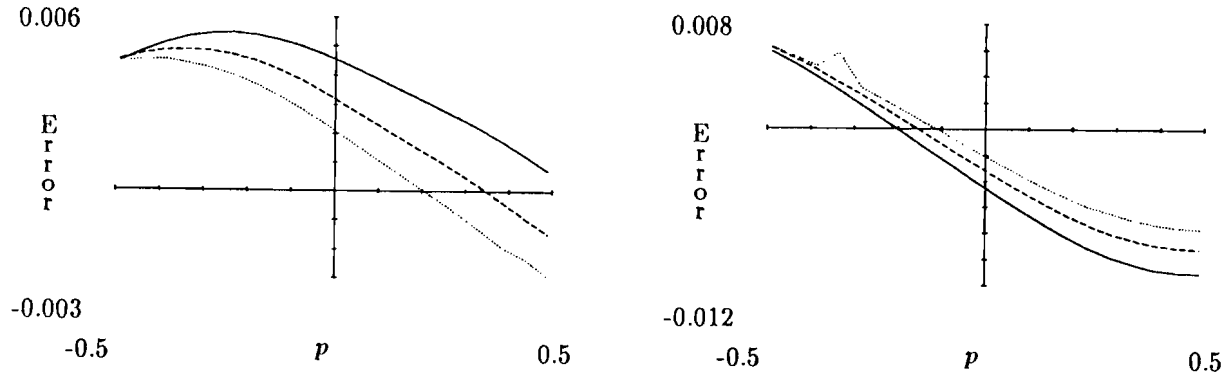

Fig. 7. Updating errors as a function of p *for element 3* (left) *and element 4* (right) *with values of* w = *0.7* (solid), w = *0.6* (dashed) *and* w = *0.5* (dotted).

near the center and negative near the tails. This implies that, on the average, the approximated posterior is *no more peaked* than the true posterior. This result is not surprising, as a histogram representation of a unimodal distribution tends to be less peaked than the original distribution. Nonetheless, this is an extremely important property; it implies that the method has some *built-in* robustness to modeling assumptions.

Error Propagation

For simplicity assume the elements of the domain grid are uniform size so that the factors $\mu(\Omega_i)$ drop out (i.e., $a_i^k = \lambda_i^k$). Let π be the *true* kth stage (updated) prior, and $\eta_i^k = \pi(\Omega_i)$. This is, η is the correct probability associated with grid element i. We consider errors of the form:

$$c_i^{k+1} = \eta_i^{k+1} - \lambda_i^{k+1} = \eta_i^{k+1} - \frac{\lambda_i^k \int_{\Omega_i} f(z_{k+1} \mid p)\, dp}{\sum \lambda_i^k \int_{\Omega_i} f(z_{k+1} \mid p)\, dp}.$$

We can rewrite the final term as a combination of the correct probability η_i^k and the effect of previous errors c_i^k:

$$c_i^{k+1} = \eta_i^{k+1} - \left(\frac{\eta_i^k \int_{\Omega_i} f(z_{k+1} \mid p)\, dp}{\sum \eta_i^k \int_{\Omega_i} f(z_{k+1} \mid p)\, dp} + \frac{c_i^k \int_{\Omega_i} f(z_{k+1} \mid p)\, dp}{\sum \lambda_i^k \int_{\Omega_i} f(z_{k+1} \mid p)\, dp} \right).$$

We again assume that the difference between the denominators is not substantial. Now, by gathering the first two terms together into the single-stage probability error e_i^{k+1} and multiplying the top and bottom of the final term by λ_i^k, we get:

$$c_i^{k+1} = \frac{e_i^{k+1}}{\sum \eta_i^k \int_{\Omega_i} f(z_{k+1} \mid p)\, dp} + c_i^k \left(\frac{\lambda_i^{k+1}}{\lambda_i^k} \right) \quad \text{with } c_i^0 = 0. \tag{17}$$

This is a nonlinear, stochastic, difference equation with the following qualitative behavior: the term e_i^k tends to be positive near the center and negative near the tails, so the cumulative errors tend to flatten the distribution. Furthermore, in areas of increasing mass $[(\lambda_i^{k+1}/\lambda_i^k) > 1]$, so previous errors have an increasing weight—effectively "damping" the rapid update and adding robustness.

Other Sources of Error

Interpolation is a source of error both before the updating algorithm reaches steady state and, to a lesser degree, when it is in steady state. However, this error is generally inconsequential. Moreover any errors that are introduced make the interpolated distribution less peaked than the ideal distribution.

Another source of error is the imperfect representation of the range grid pointed out earlier. That is, we approximate the elements of the range grid, which leads to overlap among the elements. However, the enlargement of grid elements to account for representation error acts, in a sense, as an added model uncertainty and increases the tendency of updating to flatten the posterior. In other words, in cases where this error is large, the procedure is also very robust to error. We note that the propagation of all of these errors follows (17).

5.2. *Simulation Evaluation of Sensor Data Fusion*

We have implemented this method on scalar processors using a regular rectangular gridding of the initial parameter space and a rectangular bounding box representation of the range grid. The construction of the range grid uses the midpoint projection heuristic (described in example 7) with a scaling parameter indicating the fraction of a domain grid element (e.g., a factor of 0.2 indicates that the grid element should be enlarged to 1.2 times its original size and then projected). Modeling error is handled through additive fitting parameters as discussed previously. For a more concise description of the algorithms and data structure manipulation, we refer the reader to Hager (1988).

In the remainder of this section, we present a number of problems and tests of the algorithms on simulated problems. The emphasis of these tests is to evaluate the types of approximation errors incurred in typical problems.

Comparison with the Optimal Estimator

Here we compare the behavior of the grid-based method to the known optimal solution to the linear-quadratic-Gaussian estimation problem. The observation system is that described in example 7. We use a mean square error performance criterion:

$$u(\hat{p}, p) = -E[\,\|\hat{p} - p\|^2\,].$$

It is well known that the optimal estimate, in this case, is the conditional mean. When the observation system is linear and the prior and sampling densities are independent and Gaussian, the mean square error is independent of the values of observations.

Figure 8 shows the theoretically expected value of the estimation error,[2] and three simulations using grid resolutions of five, 10, and 15 elements per dimension. These data illustrate the convergence of the technique to the optimal solution and verify the error analysis, which predicts the method will increasingly overestimate errors with coarser grids.

Because H is orthonormal, the error is also independent of the choice of the rotation angle, θ. However, we use the representation scheme presented in Figure 5 of the previous section, so we should use an expansion factor that depends on the angle of rotation. We tested the estimation performance for values of the scale factor from 0 and 0.5. The performance was nearly identical to that shown above. Hence in this case, statistical updates are relatively insensitive to the value of this parameter. More generally, as long as the projection errors are small relative to the sampling density, the resulting updating errors are inconsequential.

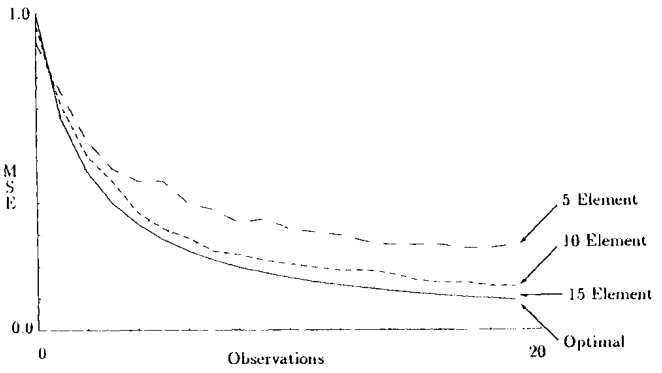

Fig. 8. The observed mean square error for grids of 5, 10, and 15 elements and the optimal expected error.

Nonlinearities and Updating Errors

At this time, the implementation can only estimate model parameters or subsets of the model parameters by reducing the parameter space. General transformation of parameters is not yet implemented. Consequently we cannot examine the behavior of all of the example problems directly, but we can test the ability of the method at localizing all or some of the parameters of the rectangular model (see example 1). To do this, we simulated taking monocular camera observations of individual corners of the block. At each iteration, we moved the sensor 30° clockwise about the object and observed the next corner. In this way we obtain a mix of corners and sensor observation positions. The sampling density is a triangle sampling density with width of one pixel. The prior distribution is uniform, and the evaluation function is the 1-0 utility. This utility leads to a payoff that is the probability of capturing the unknown parameters within an interval. The estimate is taken as the distribution mode.

Figure 9 shows the performance of the estimator for estimating the 2D position and orientation of a block of known size. The tolerance intervals in the 1-0 utility are 2 mm on position and 2° of angle; the left graph is the performance of a five-element,[3] grid and the right is the performance of a seven-element

2. We note that the actual sampling and prior distributions for the simulation have been clipped at ±4.0. However, the difference in mean square error between the clipped and unclipped distribution is less than 0.01%.

3. When we say "n-element grid," we mean n grid elements *per dimension*.

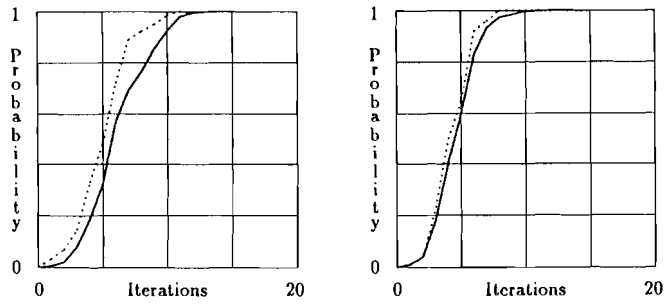

Fig. 9. Rotation and position of a known-size block for resolutions of five and seven elements. The dotted curve is the actual frequency of the correct answer over many simulated trials, and the solid line is the expected probability calculated by the estimator.

grid. What is important to note is that the calculated payoff (probability) is below the actual frequency of capturing the parameters as predicted from the error analysis. Naturally, the seven-element grid has somewhat better performance than the five-element grid.

The left side of Figure 10 shows the curve for estimating an unknown-position, unknown-size block using a four-element grid. We see that convergence is slowed slightly because of the coarser grid, but that the additional size parameters do not have more than a minor effect on convergence. The right side of Figure 10 shows the performance on the problem of determining three rotations and two translations using a stereo camera. The model of error in image location is the same as the above simulations, and the distribution over distance is of the same form, but spread over a range of ± 10 mm. Again, for both of these cases, the payoff estimates are conservative but are reasonably close to the observed values, in spite of the coarse grid. We note that each of the angles was originally constrained to lie in a 60° range—far larger than the range that most linear techniques can effectively handle (Hager 1988).

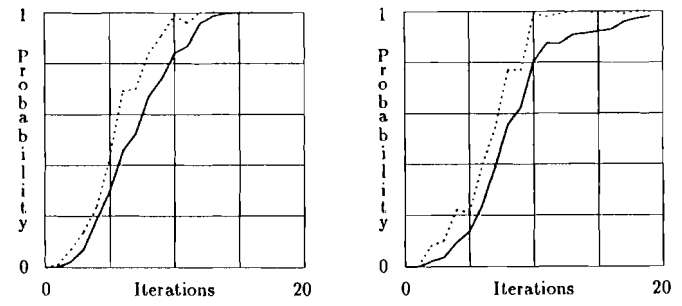

Fig. 10. The probability curves for an unknown position, unknown-size block (left) *and unknown 3D-position block* (right) *using a four-element grid. The dotted curve is the actual frequency of the correct answer over many simulated trials, and the solid line is the expected probability calculated by the estimator.*

Effects of Grid Resolution on Bias

Naturally there is a relationship between the accuracy of estimates and the resolution of the grid. In the case of a 1-0 utility, if the tolerance interval is smaller than a grid element, there are a number of estimates with the same payoff. That is, if the width of a grid element, w is larger than $2d$, then the payoff of an estimate $\hat{p}$ is constant in an interval of length $w - 2d$. Figure 11 shows the bias[4] of estimates of a 0.2-unit confidence estimate after three observations for grids of resolution 3, 5, and 7. The prior is uniform, and the sampling noise is Gaussian. Note that there is an obvious bias for the three-element grid. The five-element grid displays almost no bias, and the seven-element grid has none. For this problem, a five-element grid is probably sufficient.

Finally, Figure 12 compares the bias of estimates after three observations and 30 observations. The lack of bias in the latter is a result of the effects of grid contraction. The width of grid elements becomes smaller than the width of the estimate interval, and the accuracy improves. This suggests that the best grid size is one that, on the average, has an end resolution at least as fine as the requested tolerance interval. Similar statements hold for the mean as an estimator, though the mean tends to be less sensitive to grid quantization.

Evaluation of Robustness

Thus far we have not given any quantitative indication of how our implementation of model robustness behaves. One method of evalating robustness is to consider the variability of the sensor model as an additional contamination and then determine what distributions for this parameter can be tolerated. That is, we now consider the model:

$$z = H(p, w, e) + W + V(p, w, e)$$

and attempt to determine acceptable distributions for the random variable W modeling uncertainty in H.

Analysis of this model indicates that there is no consistent interpretation of our robustness method directly as an independent random variable. However, through simulation analysis, we have been able to determine what types of distributions can be tol-

4. We define bias as $b(p) = E[\delta(z) \mid p] - p$.

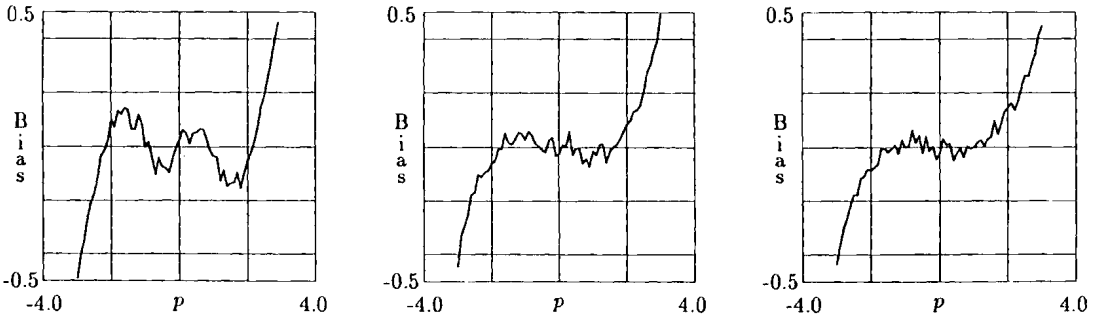

Fig. 11. Estimation bias for grids of three, five, and seven elements.

erated. In all cases we have examined, these distributions are *less peaked* than a uniform.

To be more precise, we have the following results: consider dividing the (scalar) interval $[-2, 2]$ into five equal subintervals. We assign a probability to each subinterval; this yields a histogram representation of a sampling density. We then set a value of ϵ (the parameter describing model tolerance) and determine what distributions can be tolerated in the sense that calculated payoff is lower than true payoff. We represent this contamination using a five-element histogram over the interval $[-\epsilon, \epsilon]$. Table 1 lists the values for the sampling and contaminating distributions for three cases we have examined.

The values parameterizing the least-peaked contamination are not unique, but they serve to illustrate the general trend: as the sampling density becomes more peaked, the maximum contaminating density that can be tolerated becomes less peaked, but all are less peaked than a uniform distribution. Furthermore, in the limit (as grid elements become small), the distribution for W becomes uniform. These two results suggest that the procedure is very robust to modeling error, tolerating distributions less peaked than a uniform, and the uniform distribution is the limiting case.

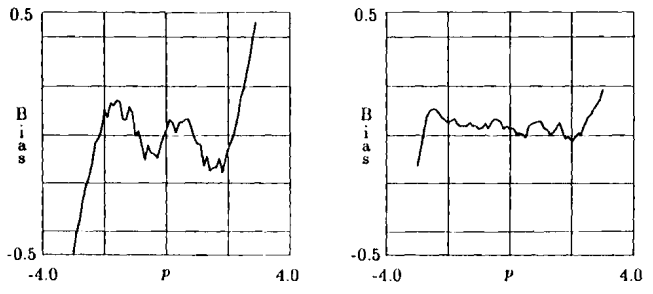

Fig. 12. Left, the estimation bias after three observations; right, the estimation bias after 30 observations.

5.3. *Simulation Analysis of Sensor Planning*

We have tested several heuristics for choosing the points in $\mathcal{B}$, the approximation set used in evaluating sensor viewpoints. In this section, we demonstrate the behavior of three different approximations. The simplest heuristic is to evaluate marginal gains at the current estimate

$$\mathcal{B} = \{\hat{p}\}$$

and assume that this represents a reasonable approximation to the true marginal gain.

This expression is adequate to pick up large uncertainties but tends to fail to fully evaluate the effectiveness of a view; instead it picks a view that is optimal for a very specific object-sensor relationship. A more computationally expensive method, but one that we have found to yield better results, is to choose points on or near the border of the grid, as well as the current best estimate. For example,

$$\mathcal{B} = \{\hat{p}\} \cup \{p \mid \mathbf{p} \text{ is a point near the border of } \mathcal{P}\}.$$

This heuristic has the effect of finding viewpoints leading to gross uncertainty reductions and also more accurately evaluates the change in those reductions over different possible object-sensor configurations.

Finally, an even more computationally intensive approach is to choose a set $\mathcal{B}$ that contains at least

Table 1. A Comparison of Sampling Density With the Least Peaked Density to Which it is Robust

Sampling Distribution					Contaminating Distribution				
0.05	0.1	0.7	0.1	0.05	0.25	0.2	0.1	0.2	0.25
0.1	0.2	0.4	0.2	0.1	0.3	0.15	0.1	0.15	0.3
0.2	0.2	0.2	0.2	0.2	0.35	0.12	0.06	0.12	0.35

one point for each element of the domain grid. This corresponds to averaging over the entire parameter space.

Effectiveness at Representing Viewpoint Payoff

Figures 14, 15, and 16 show how the three methods—using the current estimate, using a subgrid consisting of the best estimate and the grid corners, and the full grid integration—compared over three sample situations. The geometric model is an unknown position block, and the sensor model is a monocular camera (see example 3) observing corners under perspective with observation noise in the range of one pixel. To simplify the presentation, we have set $a_3 = 0$ so that there are only four corners to consider, and occlusion is not an issue. (To picture the situation, imagine observing an envelope laying in a shelf just below eye level.) By convention, we fix the coordinate system at corner 0 and number the remaining corners counterclockwise (Fig. 13). The cost function is zero so that we can clearly see the calculated payoff values.

We assume the camera maintains a fixed distance from the object and compute the payoff for each of the four corners as the camera rotates through 90°. The solid lines represent the single-point, best-estimate approximation; dashed lines indicate the subgrid approximation; and the dotted line is the full integration over the grid. The location of the ordinate axis is the current sensor location; negative angles go to the left and positive to the right. The

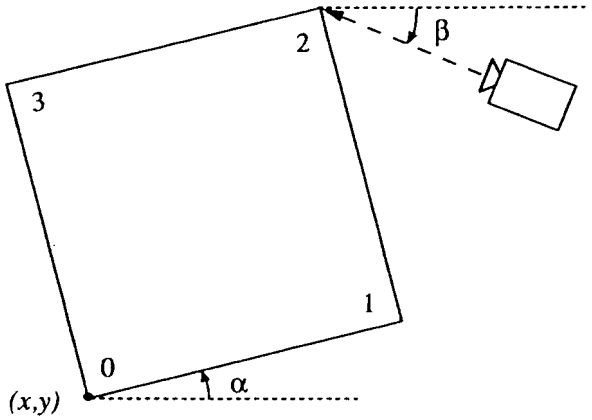

Fig. 13. An illustration of the simulation geometry. A camera at orientation β observes corner 2 of a rectangle located at point (x, y) with orientation α. As the simulation proceeds, the camera observes other corners from different positions and orientations.

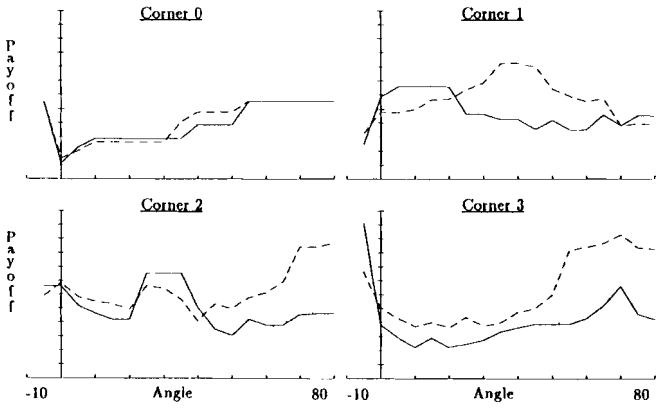

Fig. 14. The payoff of different combinations of viewpoints and corners after single view of corner 0 along the x *axis.*

graphs correspond to corners 0, 1, 2, and 3 from upper left to lower right. The abscissa corresponds to (relative) viewing angle, and the ordinate is the expected payoff of an observation from that angle.

We first note that, as expected, the widest variation is in Figure 14, where only one view has been taken, and large uncertainty still exists. However, even in this case the approximation curves generally follow the shape of the true curve, and most importantly, the current observation point has the lowest payoff value. This indicates that the next observation would be taken from another perspective as we would hope.

Figure 15 shows the payoffs after two orthogonal views of corner 0. In this case, position is well established. Thus we see that corner 0 has a flat payoff; this is expected, as it does not yield any

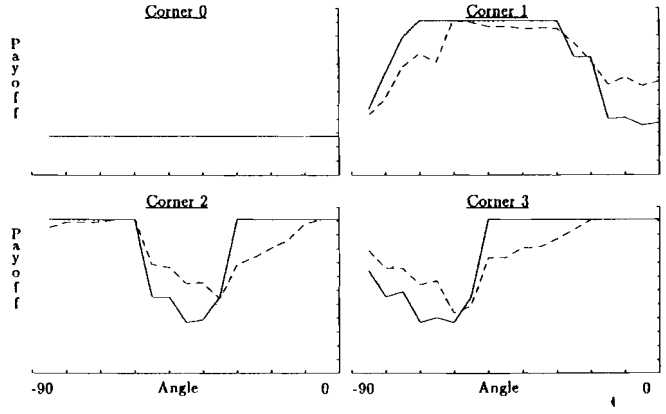

Fig. 15. The payoff of different combinations of viewpoints and corners after views of corner 0 along the x*-axis and* y*-axis.*

information on rotations. We also note that each corner has its "optimal" viewpoint. This viewpoint corresponds to the viewing angle where the (monocular) observation of corner position is most sensitive to rotation. We observe that the subgrid approximation is clearly superior to the single estimate approximation. This is because the single point approximation assumes the object is in a specific orientation and optimizes a plan *for that orientation*.

Finally, Figure 16 shows the payoffs after orthogonal views taken of corner 0 and corner 3. In this case there is partial information on both rotations and translations. Again, the most important point is that both approximations do very well qualitatively, though the subgrid approximation clearly outperforms the single point approximation. These observations suggest the subgrid approximation is generally adequate for this case.

Selection of Sample Size

The sampling procedure stops when there is no viewpoint with positive expected marginal gain. Thus another important evaluation criteria of a heuristic is its ability to accurately approximate expected marginal gain. Figure 17 shows the expected marginal gain curve for the subgrid heuristic and true marginal gain curve averaged over 100 runs for the system described above. We note that the heuristic consistently underestimates the true gain until the very end, where the approximation error goes to zero. In practice, the heuristic is "noisier" than the full integration.

These two observations led us to use a stopping criterion that tends to sample past the projected peak of the payoff curve. We implement this by

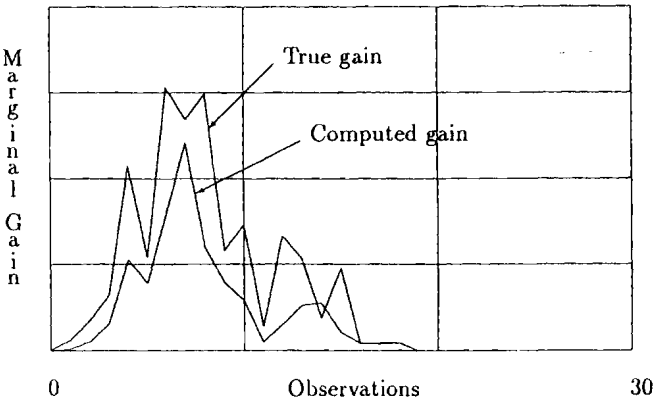

Fig. 17. The true average marginal gain curve (top curve) *and the projected average marginal gain* (bottom curve).

stopping after *two consecutive* projections of negative marginal gain. Consider the utility/cost formulation given in example 4. Figure 18 shows the stopping performance when the exponent h is 1, the estimate payoff is the probability of a correct answer, the cost of an observation is the CPU time taken to process it, and t_d is fixed at the time when, on the average, the payoff curve reaches 80%. The upper curve is the averaged subgrid payoff curve, and the lower curve is the percentage of runs that stopped at that point. We see that the stopping rate peaks just past the top of the payoff curve as expected. This indicates that, on the average, the sampling procedure stops taking data when the (true) marginal gain becomes negative.

The stopping behavior is, of course, affected by how costs are weighed against gains as governed by the model given by (5). We illustrate this point in Tables 2 and 3. In Table 2, we computed the average CPU time and final probability for a unit priority

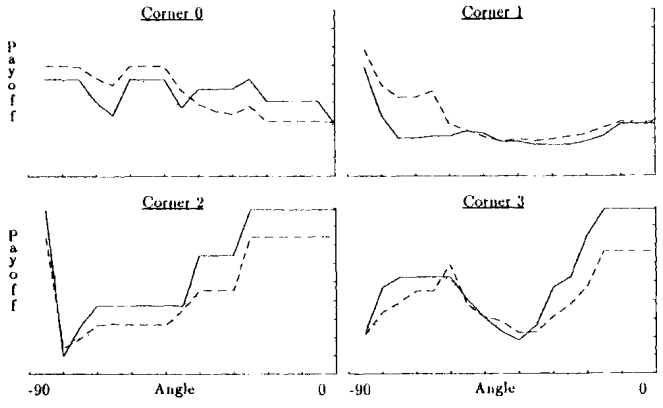

Fig. 16. The payoff of different combinations of viewpoints and corners after a view of corner 0 along the x*-axis and corner 3 along the* y*-axis, respectively.*

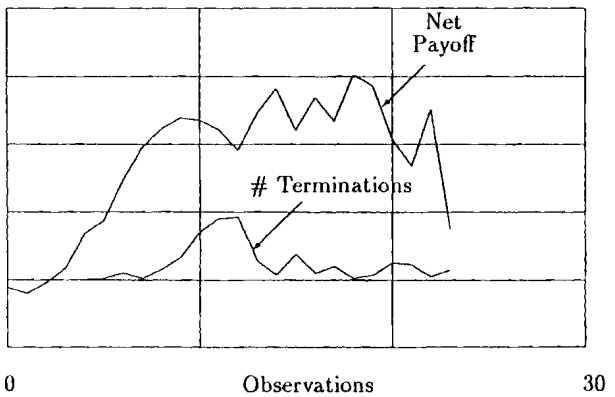

Fig. 18. The stopping rule behavior against a linear CPU time–based cost function.

Table 2. Stopping Rule Performance for a Unit Priority

	Deadline (CPU seconds)					
	1		2		10	
h	CPU	Prob.	CPU	Prob.	CPU	Prob.
1	0.93	0.26	1.08	0.33	1.60	0.45
3	0.96	0.25	1.28	0.37	2.37	0.60
8	1.11	0.28	1.52	0.42	3.39	0.76

Table 3. Stopping Rule Performance for a Priority of Five

	Deadline (CPU seconds)					
	1		2		10	
h	CPU	Prob.	CPU	Prob.	CPU	Prob.
1	1.29	0.37	1.78	0.48	1.97	0.53
3	1.14	0.32	1.65	0.45	2.85	0.67
8	1.16	0.32	1.74	0.47	3.64	0.79

($w = 1$ in the $0 - w$ utility). In this case we see that the effect of increasing the deadlines is to increase probabilities, and the effect of increasing h is to allow the estimation to proceed closer to the deadline.[5] In Table 3, we have increased the priority to $w = 5$. The increased value on information allows the estimator to sample past the 1-second deadline. Thus the effect of increasing h is now to decrease the probability as the estimator becomes more deadline oriented for that value of t_d.

6. Performance of the Method in a Real System

In this section we describe the results of applying the grid-based methods to several sensing problems. The system is based on a real-time image processing component that can follow and track brightness contours. In the first experiments, we work with a static camera and illustrate the behavior of the method when sensor and model uncertainty must be taken into account. The grid-based fusion algorithm and sensor planning methods have also been incorporated into a distributed sensor system described in Hager (1988) and Lee et al. (1989). The camera is mounted on a robot controlled by a processor that uses visual feedback to provide an object-centered polar coordinate system. (Examples 1, 3, and 6 describe the geometry of sensing and control for this system.) A third processor performs fusion and sensor planning. We use this system to test the sensor planning methods.

This section is intended as a summary of results; the interested reader will find a more detailed list of the experimental results in Hager (1988). Unless otherwise noted, all distance units in this section are in millimeters, and all angular measurements are in degrees.

6.1. Calibration

The accuracy of any result depends on how accurately the focal length can be determined. We first calibrated the system by placing a known-size subject at a known distance from the camera, estimating the focal length of the lens, as well as the position of the lower left corner of the subject. The object was to obtain a probability one bracketing of focal length within the tolerance given in column 3 of Table 4. The scaling factor on the domain grid was set to 0.5, and the uncertainty factor was 0.01. Table 4 presents the results of estimation. The primary observation is that, as expected, the smaller tolerance interval required significantly more observations. We also see the effects of grid quantization as the final results take on one of only two values.

Table 4. The Results of Calibrating the Camera Focal Length

X Pos.	Y Pos.	Focal Length	FL Tolerance	Iterations
−40.08	135.67	12.87	±0.2	18
−39.57	135.57	12.95	±0.2	7
−39.58	135.57	12.95	±0.2	7
−40.08	135.85	12.87	±0.2	14
−40.13	135.91	12.87	±0.05	46
−39.54	134.99	12.95	±0.05	20

5. The decision to continue is made based on *current* usage; thus the estimator tends to sample one time *past* the deadline given. Hence the seeming paradox in the lower left corner of Table 2, where the deadline-oriented estimator passed the deadline.

Set 1

Having determined these calibration parameters, we then had the estimation and information-gathering apparatus determine the left-to-right position and size of a book. In this case, the tolerance was set at ± 3 mm, and the scaling factor was set to 0, even though there is coupling between x size and x position. The results are presented in Table 5. These results correspond, up to measurement error, with the true parameters.

Next, we tipped the book, creating a gross model discrepancy, and ran the system. It exhibited one of two behaviors. If the only corners sensed were 0 and 2, then the system quickly returned a probability 1 estimate of the wrong size. This is to be expected, as these corners yield orthogonal information and do not indicate the height discrepancy of corner 3 or the position discrepancy between corners 1 and 0. On the other hand, if the system takes information at either 1 and 0 or 3, it immediately "softens" the observation model to account for the modeling discrepancy. The model tolerance grew to 0.64 mm (about 30 pixels), and at that point, there was essentially no information to be gained from more observations relative to the requested estimate tolerance.

We then accounted for this discrepancy by allowing rotations about z axis (which points directly out of the camera). The system parameters are the same as the previous experiment, except we added a tolerance of $\pm 1°$ on rotations, and the scaling factor was 0.2. The results are presented in Table 6. We note that, on the first trial, the system increased the fitting tolerance to 0.02. Also, the number of observations required more than doubles with the addition of this parameter. Part of this comes from the additional complexity of the system, and part from the effects of grid quantization.

Set 2

The object of this set of experiments is to demonstrate some of the effects of determinedness and model error on estimation performance. In these runs, we attached a fixed cost to each observation (actually derived from the CPU time consumed by the estimator) so that it stopped making observations when the expected gain in probability fell below the cost of observation (a linear cost model). We refer the reader to Hager (1988) for a more detailed explanation of this strategy.

Again, consider estimating the position and size of the object at a given distance. In Table 7, we show

Table 5. The Results Estimating the Size and Position of an Object

X Pos.	X Size	Y Size	Initial Corner	Iterations
−41.07	163.35	241.69	0	4
−41.00	163.10	241.69	2	7
−40.47	162.60	241.03	1	4
−40.45	162.56	241.08	3	9

the estimates, the uncertainty factor, and the final probabilities. In particular, note that the final results are with probability one, except in those cases where the fitting tolerance moved up. In these cases, the estimator "stalls" and returns results that are lower than probability one.

To demonstrate the effects of determinedness on estimation performance, we fixed the height and size parameters (as determined from the previous run) and estimated the x position, distance, and rotation of the object about the y (vertical) axis. The determination of rotation comes from perspective. Therefore when the book is perpendicular to the camera, there is no perspective information and rotation is poorly determined. As rotations increase from this zero point, the system becomes more determined. Table 8 gives the experimental results. We note that the convergence figures correspond with the above argument and that the effects of increased fitting tolerance are seen on three of the runs.

6.2. *Mobile Camera*

The mobile camera system was tested with variations on the example problems used throughout this article. Namely, we used monocular cues (corners and lines) to compute the position and size of polygonal (and superellipsoidal) objects. This forced the system to choose viewpoints and features so that triangulation and perspective combine to constrain the

Table 6. Estimating the Size, Position and Rotation of a Rectangular Object

X Pos.	X Size	Y Size	Rotation	Iterations
−42.7	164.7	241.7	9.7	12
−42.2	162.5	242.2	9.7	19
−42.3	163.7	241.8	9.6	24
−42.2	161.9	242.3	9.7	13

Table 7. Estimation Results for a More Complex Positioning Problem

X Pos.	Y Pos.	X Size	Y Size	Tolerance	Probability
−128.66	133.74	238.22	166.22	0.02	0.854
−128.66	133.74	238.82	165.91	0.01	1.00
−128.66	133.44	239.37	165.53	0.01	1.00
−129.39	133.10	239.52	166.81	0.02	0.714
−128.05	133.36	239.11	166.00	0.01	1.00

geometry of the object. All experiments were carried out using a single-step look-ahead.

The experimental results indicated that the observation selection algorithms found nearly optimal strategies for simple problems. For example, for simple triangulation problems the solution was to use views with the widest possible separation angle until the required estimate accuracy was reached. For more complex problems, such as finding all six parameters of a rectangle on a table, the strategies were nonoptimal but still served to quickly constrain the estimate down to the level of sensor observation uncertainty. The nonoptimality was not a result of the approximations used in computing the strategies but was simply due to the horizon effects of a one-step look-ahead. In general, if n views would be needed to solve for the unknown parameters in the ideal (no noise or model uncertainty) case, the system used approximately $2n$ views to reduce the bounds of an estimate to the level of sensor noise and model uncertainty. Part of this behavior is also a result of the finite resolution of the grid.

Table 8. Estimator Performance on a Series of Rotations

X Pos.	Z Pos.	Rotation	Probability	Tolerance
−134.7	792.5	0.0	50	0.01
−134.7	791.4	0.0	50	0.01
−134.7	792.5	0.0	50	0.01
−125.5	765.7	11.8	35	0.02
−126.5	766.8	13.2	53	0.01
−125.5	766.6	11.2	72	0.01
−126.8	766.8	11.0	99	0.01
−122.7	759.0	−22.1	39	0.02
−122.5	761.5	−23.2	63	0.01
−122.5	761.5	−23.4	60	0.01
−122.8	755.4	−23.8	20	0.02
−127.2	817.7	−25.56	99	0.01
−127.5	816.4	−25.58	98	0.01
−127.2	818.2	−25.52	99	0.01

We also observed that fitting tolerance had a substantial effect on the performance of sensor search. Namely, in those cases where the fitting tolerance was quite high, the performance of the search procedures degraded. This appears to be caused by the fact that high tolerances decrease the discriminating ability of the sensor and therefore make it more difficult to determine which observations will yield information relevant to the current task.

7. Discussion

We believe the process of information gathering will play a central role in the development of intelligent autonomous systems. Conceptually, information gathering requires a representation for information with uncertainty, a method for describing sensors and fusing sensor information into the representation, a method for deciding what type of and how much sensor information is most fruitful to pursue, and a method for delivering a final decision based on the resulting observations.

From a practical perspective, the approach of solving problems with specific sensors, models, and methods has the advantage of allowing relatively complete solutions to complex problems. However, we argue that the information needed by robotic systems is highly varied, and the only efficient method for gathering this information is to make the system task directed. Therefore we believe the first step in the realization of information gathering is to build a system that can handle a general class of information gathering problems in a goal-directed fashion.

To this end, we have presented a decision-theoretic framework for describing geometric sensing tasks. The advantage of this framework is its ability to accommodate the many different representations, sensors, and sensing tasks encountered in robotic applications. In particular, this framework incorporates the notions of *accuracy* or *value* of information, the *cost* of information, and the *trade-off* between these quantities.

The ability to efficiently and accurately manipulate probability representations is central to the realization of this framework. The grid-based techniques we have presented have the advantage of extreme flexibility, as well as reasonable qualitative and quantitative approximation characteristics. By suita-

ble application of these methods, it is possible to implement a wide variety of problems directly from the framework as presented. We also showed how this method is extended to uncertain sensor models and discussed its robustness. We have implemented this technique and demonstrated mathematically and through simulation that it has stable and predictable error properties for a wide range of problems.

The simulations and experiments we have carried out indicated that the two fundamental concepts in applying these methods are the method of gridding and the type of modeling error allowed for. This is particularly true in those cases where the statistical noise level is fairly low, in which case sensor model error can easily force the system into an inconsistent situation, and poor grid representation can significantly inhibit convergence. To date, most of the practical limitations we have encountered are of these two types. The gridding technique described in this article is relatively rigid and works best for those situations where the parameter vector is well-determined by sensor observations. Similarly, the additive method of accounting for modeling error behaves poorly when modeling error is nonlinearly related to the parameter vector.

These problems are the focus of our current research. The current rigid gridding scheme makes poor use of grid elements and requires global grid reorganizations. These properties also make it unsuitable for parallel implementation. We are not experimenting with methods for *locally* reorganizing the grid elements. This has the advantage of increasing the independence of grid elements and, when done properly, increases the speed of convergence. However, it introduces new problems in grid management that will need to be understood. With this more flexible implementation, it will also be possible to use a secondary gridding over arbitrary model tolerance parameters. We also hope to prove some general convergence properties and thereby classify more precisely the types of problems to which this method is applicable.

The methods used to search for sensor plans are essentially brute force, and in order to make them practical, we use the approximations described earlier. By knowing more about the measurement system description, there may be ways of using more high-level information about the geometry of sensing to both speed up the process of predicting the results of a sensor observation and reduce the size of the search space. In particular, we are interested in the possibility of *learning* strategies over time and essentially implementing parts of the search process using what amounts to a table lookup.

We have recently defined an interface to the implementation that insulates the user from the details of grid manipulations (Hager 1990). The interface is for the C language (Kernighan and Ritchie 1978), and the style resembles that of the RCCL system (Hayward and Lloyd 1984). The interface facilitates a "task-oriented" programming style supported by precompiled libraries of sensor descriptions, parametric models, and task descriptions. In the near future, we expect to modify the implementation to conform to this interface and test the system in interaction with task-level robot programming.

Acknowledgments

Support for G. Hager was provided in part by an IBM Graduate Fellowship in Manufacturing Automation and a National Science Foundation Graduate Fellowship. Any opinions, findings, and conclusions or recommendations expressed in this article are those of the author and do not necessarily reflect the views of the National Science Foundation.

Support for both authors was provided in part by NSF DCR-8410771, NSF DMC-8411879, and DMC-12838, Air Force F496 0-85-K-0018, F33615-83-C-3000, F33615-86-C-3610, DARPA/ONR grants NOO14-85-K-0807, NOOO14-85-K 0018, ARMY/DAA29-84-K-0061, DAA29-84-9-0027, NSF-CER MCS-8219196, DCR82-19196 A02, NIH NS-10 39-11 as part of the Cerebrovascular Research Center, by DEC Corp., and the LORD Corp.

References

Allen, P. 1988. *Robotic Object Recognition Using Vision and Touch*. Boston: Kluwer.

Aloimonos, J. 1987. Active vision. *Proceedings of the First International Conference on Computer Vision*, pp. 35–54.

Ayache, N., and Faugeras, O. D. 1988. Building, registrating, and fusing noisy visual maps. *Int. J. Robot. Res.* 7(6):45–65.

Bajcsy, R. 1985. Active perception vs. passive perception. *The Third IEEE Workshop on Computer Vision*, pp. 55–59.

Bajcsy, R. 1988. Active perception. *Proc. IEEE*, 76(8):996–1005.

Berger, J. O. 1985. *Statistical Decision Theory and Bayesian Analysis*, ed. 2. New York: Springer-Verlag.

Box, G., and Jenkins, G. 1976. *Time Series Analysis: Forecasting and Control*. San Francisco: Holden-Day.

Brooks, R. 1981. Symbolic reasoning among 3-D models and 2-D images. *Artif. Intell.* 17:285–348.

Brooks, R. 1985. Visual map making for a mobile robot. *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, pp. 824–829.

Cameron, A. 1989. Optimal tactile sensor placement. *Pro-*

ceedings of the 1989 IEEE International Conference on Robotics and Automation, pp. 308–313.

Coles, L., Robb, A., Sinclair, P., Smith, M., and Sobek, R. 1975. Decision analysis for an experimental robot with unreliable sensors, 749–757. Morgan Kaufmann Publishers Inc. Los Altos, CA: *Proceedings of IJCAI-75*.

Cowan, C. K. 1988. Model-based synthesis of sensor location. *Proceedings of the 1988 IEEE International Conference on Robotics and Automation*, pp. 900–905.

Durrant-Whyte, H. 1988. *Integration, Coordination, and Control of Multi-Sensor Systems*. Boston: Kluwer.

Fedorov, V. 1972. *Theory of Optimal Experiments*. New York: Academic Press.

Giralt, G., Chatila, R., and Vaisset, M., 1984. An integrated navigation and motion control system for autonomous multisensory mobile robots. In Brady, M, Paul, R. (eds.): *Robotics Research, The First International Symposium*, Cambridge, Mass.: pp. 191–214.

Grimson, W. 1986. Sensing strategies for disambiguating among multiple objects in known poses. *IEEE J. Robot. Automat.* 2:196–213.

Hager, G. 1988. Active reduction of uncertainty in multisensor systems. Ph.D. thesis, University of Pennsylvania, Philadelphia. Available as Dept. of Computer Science report MS-CIS-88-47.

Hager, G. 1987. An agent specification language. Dept. of Computer Science report MS-CIS-87-08, University of Pennsylvania, Philadelphia.

Hager, G. 1990. *Computational Methods for Sensor Data Fusion and Sensor Planning*. Kluwer: Boston.

Hager, G., and Mintz, M. 1987. Searching for information. *Proceedings of the First Workshop on Spatial Reasoning and Multi-Sensor Fusion*.

Hager, G., and Mintz, M. 1989a. Estimation procedures for robust sensor control. In Kanal, L., Levitt, T., Lemmer, J. (eds.): pp. 285–301. *Uncertainty in Artificial Intelligence 3*. New York: North-Holland.

Hager, G., and Mintz, M. 1989b. Sensor modeling and robust sensor fusion. *Proceedings of the Fifth International Symposium on Robotics Research*. Cambridge, Mass.: MIT Press.

Hayward, V., and Lloyd, J., 1984. *RCCL User's Guide*. Montreal: McGill University.

Huber, P. 1981. *Robust Statistics*. New York: John Wiley & Sons.

Hutchinson, S., Cromwell, R., and Kak, A. C. 1988. Planning sensing strategies in a robot work cell with multisensor capabilities. *Proceedings of the 1988 IEEE Conference on Robotics and Automation*, pp. 1068–1075.

Jacobs, W., and Kiefer, M. 1973. Robot decisions based on maximizing utility. *Proceedings of IJCAI-73*, pp. 402–411.

Keeney, R. L., and Raiffa, H. 1976. *Decisions with Multiple Objectives*. New York: John Wiley & Sons.

Kernighan, B. W., and Ritchie, D. M. 1978. *The C Programming Language*. Englewood Cliffs, N.J.: Prentice-Hall.

Lee, I., King, R. B., and Paul, R. P. 1989. A predictable real-time kernel for distributed multi-sensor systems. *IEEE Computer* 22(6):78–83.

Leyton, M. 1988. A process-grammar for shape. *Artif. Intell.* 34:213–248.

Lozano-Pérez, T. 1985. An approach to automatic robot programming. A.I. lab memo 812. Cambridge, Mass.: MIT.

Mehra, R. K. 1974. Optimal measurement policies and sensor designs for state and parameter estimation. Presented at The Milwaukee Symposium on Automatic Controls.

Mendenhall, W. 1968. *Introduction to Linear Models and The Design and Analysis of Experiments*. Belmont, California: Wadsworth.

Moravec, H. P. 1988. Sensor fusion in certainty grids for mobile robots. *AI Magazine* 9(2):61–74.

Müller, P., and Weber, H. 1972. Analysis and optimization of certain qualities of controllability and observability for linear dynamical systems. *Automatica* 8:237–246.

Paul, R. 1981. *Robot Manipulators*. Cambridge, Mass.: MIT Press.

Pentland, A. 1986. Perceptual organization and the representation of natural form. *Artif. Intell.* 28(3):293–332.

Press, W., Flannery, B., Teukolsky, S., and Vetterling, W. 1986. *Numerical Recipes*. New York: Cambridge University Press.

Richardson, J., and Marsh, K. 1987. Fusion of multisensor data. *Int. J. Robot. Res.* 7(8):78–96.

Sherman, S. 1955. A theorem about convex sets with applications. *Ann. Math. Statistics* 26:763–767.

Silvey, S. 1980. *Optimal Design*. New York: Chapman and Hall.

Solina, F., and Bajcsy, R. 1990. Recovery of parametric models from range images: the case for superquadrics with global deformations. *IEEE* Trans. on pattern analysis and machine intelligence, 12(2):131–147.

Stansfield, S. 1987. Visually-guided haptic object recognition. Ph.D. thesis, University of Pennsylvania, Philadelphia. Available as Dept. of Computer Science report MS-CIS-87-111.

Trinkle, J. 1987. The mechanics and planning of enveloping grasps. Ph.D. thesis, University of Pennsylvania, Philadelphia. Available as Dept. of Computer Science report MS-CIS-87-46.

Zeytinoglu, M., and Mintz, M. 1988. Robust fixed size confidence procedures for a restricted parameter space. *Ann. Statistics* 16(3):945–957.