NORTH-HOLLAND

Informatics and Computer Science

A Probabilistic Spatial Data Model

YORAM KORNATZKY

and

SOLOMON EYAL SHIMONY

Department of Mathematics and Computer Science, Ben-Gurion University of the Negev, P.O.B. 653, Beer-Sheva 84105, Israel

ABSTRACT

Spatial information in autonomous robot tasks is uncertain due to measurement errors, the dynamic nature of the world, and an incompletely known environment. We present a probabilistic spatial data model capable of describing relevant spatial data, such as object location, shape, composition, and other parameters, in the presence of uncertainty. Uncertain spatial information is modeled through continuous probability distributions on values of attributes. The data model is designed to support our visual tracking and navigation prototype.

1. INTRODUCTION

Recent advances in technology enable construction of autonomous mobile robots with considerable on-board processing power. However, making use of such hardware in the real world requires complicated, as yet unavailable, data models, that facilitate the following tasks: the robot must perceive the environment, plan, and execute courses of action. Other related tasks are navigation, map learning, and tracking the location of objects in the environment. All these tasks require an explicit spatial model of the environment. Due to measurement errors, the uncertain dynamic nature of the world, and unknown terrain, an explicit way of dealing with uncertainty is necessary.

This paper begins to address this difficult problem by proposing a probabilistic spatial data model for representing such uncertain spatial

INFORMATION SCIENCES **90**, 51–74 (1996)

655 Avenue of the Americas, New York, NY 10010

0020-0255/96/$15.00
SSDI 0020-0255(95)00241-3

information used in robot tasks. The data model describes uncertain spatial information through continuous probability distributions of spatial attributes such as object location and parameters of transformations. In most practical cases, such distributions are finitely represented (e.g., by their expected value and covariance), and can be efficiently manipulated. The paper is a significant extension of previous work by the authors [21], adding methods and functions to data model in order to allow for the additional spatial aspects of the data model, as well as constraints and the continuous distributions used to model sensor and object location errors.

Before describing our data model in detail, we motivate its development by presenting our application, a visual tracking and navigation prototype.

1.1. A VISUAL TRACKING AND NAVIGATION PROTOTYPE

Our scenario is an office space, where most items can be approximated in a concise way with three-dimensional (3D) models. To navigate in the environment, the robot has to track the location of objects, which requires real-time efficient algorithms for calculating the pose of objects and the mapping of free-space (e.g., [11, 23]). Many existing algorithms benefit from prediction of the identity and locations of expected image features, available from models of the environment and previous images [18]. Prediction of features together with their expected quality is useful in two respects. First, it allows a priori weighing of the quality of the features and their correlation, for use in maximum-likelihood filtering schemes, such as Kalman filters (KF) or least-squares (LS) approximations. Second, by predicting and looking for good features first, we can save much computation time [18]: estimating the probability of detection of a feature is useful for deciding whether detecting the feature is worthwhile, and for weighting-in the positional error of the feature once it is detected. Prediction of the locations where "good" features can be expected, as well as locations where problems are to be expected, allows the system to discount the contribution of "bad" locations. It should be possible to compensate for some of the problems, such as expected texture (which could be recovered from previous image frames, as well as from the database). The higher-level task of depth map construction can also be aided by a priori high-level knowledge, for example, that a certain area in the image (or sequence of images) is a plane or an occlusion edge. In contrast to most previous research on model-based feature prediction [8, 19, 27], our study concentrates on the performance of visual navigation and tracking tasks employing feature prediction at run-time.

An explicit 3D model of the environment that allows for limited positional and model parameter error is used. We do not assume that all objects in the scene are planes or generalized cylinders, but that for many image locations we have some prior expectation of the object it contains, and that in turn may help us do better edge localization, or better 3D position and orientation estimation of features and objects. Features are predicted, ranked according to merit, and searched for in the image locations where they are most likely to appear. Model position and parameters are then updated according to a probabilistic maximum-likelihood scheme, such as Kalman filtering on feature location and model position and parameters, for the Gaussian case.

The size and complexity of the world model required to describe in detail a man-made environment (e.g., complete office environment), as well as outdoor environments, motivated us to develop a database (knowledge base) for representing the world model. The database serves as an object manager mediating and integrating the different parts of the sensing system, for example, in the case of visual sensing, managing the large amount of intermediate descriptions generated during image interpretation [5, 15, 17]. Model-based vision applications are based on models classifying the objects in the application into a collection of classes having various geometric attributes [28]. These classes are often related through an inheritance hierarchy which guides the model-based recognition algorithms [6]. Accordingly, our database supports an object-oriented data model.

1.2. THE PROBABILISTIC DATA MODEL

Most information incorporated in the data model is known only incompletely and is subject to imprecision. Thus, values of attributes such as object location in the world, or parameters of transformations, are only estimated subject to errors. This uncertainty is due to incompleteness and uncertainty in the world model with respect to the location of the robot within the world, the dynamic nature of the world, and imprecision of sensors employed, and uncertainty introduced during the interpretation and data fusion stages. This uncertainty translates into uncertainty with respect to the presence and location of features sensed by the robot's sensors. Thus, a spatial data model supporting our application has to incorporate mechanisms for representing uncertain spatial information. For example, the location attribute of an object, which is a 3D coordinate (i.e., a value in $\Re^3$), is estimated with a certain error, and hence should be

represented as a probabilistic distribution over $\Re^3$. More generally, one can observe that uncertain spatial information, such as parameters of scaling and distortion, is most often described as a probabilistic distribution over the values of simple attributes (e.g., real numbers) or vectors of such attributes. Accordingly, we introduce different kinds of probability distributions as classes in our object-oriented data model, and represent uncertain spatial information using probability distributions. More generally, we allow any kind of attribute to be uncertain, in which case its value is a probability distribution over the set of possible values [21].

While probability distributions describing spatial information are in principle infinite and continuous, they can be finitely represented in most practical cases. Thus, a joint Gaussian distribution is represented by its expected value and covariance matrix. Based on the finite representation of distribution classes, we demonstrate how useful spatial data manipulations, required in our application, can be implemented with the data model.

Existence of objects in a scene is also uncertain due to imprecision of the sensing system in identifying objects from the world model in sensor images, and ascertaining the existence of particular spatial relations between objects, e.g., occlusion of image features. Such uncertain information is modeled by special *existence* attributes giving the probability of their existence.

Though the data model was designed to support our visual tracking and navigation research (hence the multiple references to computer vision in the examples), the model has much broader applications. It is designed to support spatial information acquisition using any sensing system, be it passive sensing, active electromagnetic radiation range finders (laser, light-stripe, etc.), sonars, touch sensors, or even explicit data provided by a human user of the system.

1.3. RELATED WORK

While many spatial object-oriented data models have been suggested [15, 24, 26, 29], they lack facilities for modeling uncertain information which is essential in our application, such as uncertain location of objects. Dutta [9, 10] has suggested a fuzzy semantics for the description of some spatial attributes and relationships in geographical information systems. For example, he suggests a fuzzy description of the notion of one object being "far away" from another. While fuzzy semantics is useful for the representation of such concepts, the representation of uncertain locations of objects is best handled by probabilistic means, as one has to supply a quantitative distribution of the possible location of the object [11, 23].

Using probabilities, we can take advantage of many well-understood tools, such as optimal estimators. Additionally, using probabilities makes it possible for the robot to make intelligent *decisions* in the presence of uncertainty, such as selecting one of a set of possible actions, through classical decision-theoretic methods. It is not clear how that can be done with any of the other existing uncertainty formalisms. For a more detailed discussion of why we use Bayesian probability rather than Dempster-Shafer probabilities or fuzzy sets, see [21].

Previous research on the probabilistic modeling of uncertain information has concentrated on the relational model (where tuples could have probabilistic attributes), e.g., [4, 7, 14], whose structuring capabilities are too poor for spatial database applications. Most importantly, these models incorporated only finite discrete probabilities over atomic types, e.g., integer, string, and did not consider continuous distributions of attribute values. However, these are necessary for representing uncertain spatial information.

Jagadish and O'Gorman [17] have suggested an image data model to aid in image interpretation. The description of uncertainty in object interpretation is through a confidence (probabilistic) attribute attached to each recognized object. However, they have not addressed the important problem of modeling and processing uncertain spatial information such as object pose in three dimensions.

The rest of the paper is organized as follows. Section 2 briefly describes the object-oriented aspects of our data model. Modeling spatial information of the world model is described in Section 3. The mechanisms for describing uncertainty of information within the data model are presented in Section 4. Representation and manipulation of uncertain spatial information is discussed in Section 5. Our conclusions are presented in Section 6.

2. AN OBJECT-ORIENTED DATA MODEL

In this section, we describe the object-oriented aspects of our data model that will be needed throughout the paper. The data model is similar to the ODE [1] and O_2 [3] object-oriented data models, and an extension of our previous work [21]. As its details are fairly standard, we describe it briefly through examples.

The data model includes *values* and *objects*. A value has a *type*. This type is recursively definable from atomic types and type constructors. An object has an *identity*, a value (its state), and a behavior defined by its *methods*. It belongs to a class.

An object or a value may refer to other objects via their identities. Similarly, an object or a value may be composed of subvalues. In the latter case, however, a subvalue is part of its container and cannot be shared by other objects. In other words, assigning a value yields a copy operation, while assigning an object gives a new reference to this object. Note that by including object identities in values, we may build multiple hierarchies over the same set of objects. Such multiple hierarchies are often required in image interpretation tasks.

Types and Values. We assume a standard set of atomic types such as boolean, integer, and real. Complex types may be defined recursively using the *tuple*, *list*, *array*, *set*, and *function* constructors. The latter defines a function value, which describes operations associated with other objects. Such function values are created by giving the name of a function of appropriate signature implemented in a standard programming language. Figure 1 is an example of a complex type. For each type, its *domain* is the set of possible values of the type.

Classes and Objects. A class is defined by its *type* and its *methods*, which define the structure of the objects in the class and the operations that can manipulate them, respectively. Figure 2 gives an example of the declaration of a class 'OfficeObject,' that has four attributes and two methods. Some of the attributes of 'OfficeObject' are 'csg_tree,' an object of class CSG, providing constructive solid geometry (CSG) representations of solid objects, and 'parts,' a list of objects which are part of the OfficeObject. The method 'projection' takes one argument of class 'Window' and returns a result of class 'ImageRegion.'2

The declaration of methods in a class does not include their implementation due to lack of space. Such an implementation is to be done in a standard programming language. A method is invoked in the scope of an object on a tuple of arguments, returns an answer, and (possibly) changes the internal state of that object. Both the declaration and the implementation of methods in our model are as in other common object-oriented data models [1, 3]. For example, the application of the (geometric) *projection*

type *desk* : **tuple** (*location* : *3DCoord*, *surface* : *Surface*,
color : *Color*, *texture* : *Texture*, *legs* : **list**(*leg*), *csg_tree* : *CSG*)

Fig. 1. A desk type.

```
class OfficeObject
    type tuple(name : string, csg_tree : CSG, location : 3DCoord, parts : set(WorldObject))
    method predict_features(angle : real, region : ImageRegion) : set(feature),
           projection(w : Window) : ImageRegion
end;
```

Fig. 2. A class.

method to a desk object with respect to (w.r.t.) a window *window*1 is denoted

$$region23 := desk45.projection(window1)$$

and returns an image region as a result.

Each object that belongs to a class is called an instance of the class, and the set of instances of a class is called its *extent*. The extent of a class changes dynamically as new objects are being created or destroyed.

Inheritance. A class may inherit its type and methods from other classes. Such inheritance declarations generate an inheritance hierarchy over the classes. An inherited type or method can be redefined locally, as long as this redefinition respects the *subtyping* semantics:

- New attributes may be added to a tuple, like 'surface' in Figure 3.
- The type of an attribute, or of a method parameter (more generally, its *signature*), can be specialized to a subtype. This is the case for the attribute 'location,' whose type is specialized from *3DCoord* to *Office-Coord* in Figure 3.
- Attributes or method names may be explicitly *renamed*, exemplified by the declaration of *legs* shown in Figure 3 (which also serves for avoiding name collisions).

3. SPATIAL ASPECTS OF THE DATA MODEL

Systems that need to perform some kind of spatial reasoning, such as our visual tracking system prototype, require a description of the geometric properties of objects in the world model, and their predicted image features. This motivates the incorporation of built-in classes with associ-

```
class  WorldObject
       type      tuple(name : string, csg_tree : CSG, location : 3DCoord,
                       texture : Texture, parts : list(PolyhedralObject))
       method    predict_features(angle : real, region : ImageRegion) : set(feature)
end;
class  desk      inherit WorldObject
       rename    parts as legs
       type      tuple(location : OfficeCoord, surface : Surface, color : Color)
end;
```

Fig. 3. Inheritance.

ated methods for describing geometric information, such as location, size, transformations between local coordinate systems, and CSG representations. Other classes define image features such as corners, edges, and texture, and relationships between features, such as collinearity. The aggregation hierarchy defining the logical composition of a complex world object from its parts is built using the type constructors. Correspondingly, the geometric description of a complex object, such as its CSG tree, is defined as a refinement of the aggregation hierarchy by means of the type constructors. Similarly, we describe the position of parts of objects relative to their containing structure.

3.1. GEOMETRIC MODELS

We opt for several kinds of geometric models, which are known in the computer graphics and vision literature. These include enhanced constructive solid geometry (CSG) representations, representation by extents, and generic models [2, 12, 22]. Their representation using our powerful data model is straightforward so we omit their description here, and demonstrate them through examples. A CSG representation is constructed from primitive objects using union, intersection and similar operations, and described as a tree whose leaves are the primitive objects and whose internal nodes are the operations.

A generic model [22] consists of a collection of objects in space, whose spatial layout is constrained by various constraints on their relative loca-

tion, orientation, and other spatial attributes. A generic model may describe the various parts of a floor in an office building, such as the walls and the doors, and may include a constraint that the sides of the hallway are parallel. A generic model is represented as a class whose parts define the objects in the model, and where constraints are represented by boolean function attributes on the parts of the model. Some of these constraints may require computing parameters from the parameters of the objects related through the constraint, such as the angle between planes computed from their sizes.

EXAMPLE 1. Let us consider a representation of a part of the generic model of a building from [22] in our data model. A hallway consists of two walls, where each wall contains a set of doors and a set of windows. The hallway and wall classes are shown in Figure 4.

Articulated models built from several rigid parts with constraints on their relative motion are represented using the type constructors for building a complex value, and associating appropriate transformations and constraints (represented as function-valued attributes) with these parts. The transformation is assumed to be of their coordinate system w.r.t. the coordinate system of their aggregation-hierarchy parent, or any other coordinate system. Such transformation are 6-vectors, with three parameters for translation and three for rotation. Constraints exist between the transformations of the subparts, as well as their parameters. Note that there may be more than one parameterization for an object type. For example, 2D lines can be represented either by the parameters (a, b) of the line equation $y = ax + b$, or by distance from the coordinate system origin and a slope angle, etc.

```
class Hallway inherit Building
    type tuple (name : string, side1, side2 : Wall, floor : plane)
end;
class Wall inherit Planar_surface
    type tuple (location : 3DCoord, length : real, doors : set(Door), windows : set(Window))
end;
```

Fig. 4. Part of a generic building model.

EXAMPLE 2. Consider a desk that has a part representing its top surface. It is represented in the data model as:

```
class  3DCoord
       type  tuple(X, Y, Z : real)
end;

class  Desk
       type  tuple( position : 3DCoord, surface : ObjectSurface,
                    legs : tuple( elements : list(tuple( leg : PolyhedralObject,
                      position : 3DCoord)),
                    constraint_legs : list(3DCoord) → Bool))
end;

class  ObjectSurface inherit PolyhedralObject
       type tuple( position : 3DCoord)
end;
```

where the positions of the surface and legs of the desk are relative to that of the desk. The parameters and transformations of the legs are to be constrained as follows: the legs must be at the corners, and protrude from the bottom of the desk surface (for transformations inconsistent with these intuitions, the constraint should return false). This constraint is implemented through the *constraint_legs* component of the legs attribute. The actual constraint function depends on the actual surface parameters of *desk*45.

To compute the absolute position of the surface, we employ the method *part_abs_position* defined on the class *Desk* (where *desk*45 is a particular desk):

$$desk45.part\_abs\_position(desk45.surface).$$

3.2. IMAGE FEATURES

Images are either 2D (standard camera images) or 3D (results of stereo-pair reconstruction, depth maps from scan-line devices, etc.). Features are partitioned according to their dimensionality: 0D (points, corners), 1D (lines, curves), 2D (surfaces, textures), and 3D (volumes). Image objects are described with appropriate objects together with relationships between these objects, such as collinearity, rectilinearity, coplanarity, etc. The image features can be computed from database objects and a projection specification. We also allow for finding 3D objects from image

features, this being an ill-posed problem in the general case. Vision systems need both kinds of manipulations.

EXAMPLE 3. Consider the desk from the previous example. Suppose that our camera focus is at (X_c, Y_c, Z_c) looking along an axis oriented according to $(\theta_X, \theta_Y, \theta_Z)$ (ignoring other camera parameters, such as focal length, for simplicity, and just using a central projection). To begin tracking the desk, we would like to find its position in the projected image. This is done using the *transformation* method defined on projections as shown in Figure 5. Naturally, this example is oversimplified. In fact, the desk surface may generate many edges and corners, and perhaps texture features as well as a surface.

4. MODELING UNCERTAIN INFORMATION

Most information incorporated in the data model is known only incompletely and is subject to imprecision. Thus, values of attributes such as object location in the world, or parameters of transformations, are only estimated subject to errors. Similarly, the existence of a particular object in the scene is uncertain due to imprecision of the object recognition modules. This probabilistic knowledge is employed by the vision system to make important inferences guiding the robot in its task.

Examining the first kind of uncertain information, that is, spatial uncertain information, one can observe that it amounts to uncertainty in the values of simple attributes, or of vectors of attributes. Thus, the location attribute of some world object which is a 3D coordinate (i.e., a value in $\Re^3$) is estimated with a certain error. This location attribute should be accordingly represented as a probabilistic distribution over $\Re^3$. Similarly, uncertainty of parameters of scaling or distortion should be represented as a random vector, and hence a vector (array) of uncertain

```
class Projection
    type tuple(axis : tuple(pitch, yaw, roll : angle), origin : tuple(X, Y, Z : distance))
    method transformation(p : 3DCoord) : 2DCoord
end;
p000 : Projection
p000.transformation(desk45.position)
```

Fig. 5. Transforming an object with a central projection.

attributes. More generally, we could have any attribute of a class be uncertain, in which case its value is a probabilistic distribution over some set of values. We have presented a general model of uncertain attributes in object-oriented data models in [21], and we will employ its general mechanisms here.

The second kind of uncertain information is that of uncertainty in object existence and uncertainty in the composition of complex values (e.g., sets). Uncertain existence of objects is represented as an *existence* attribute giving the probability that the object exists, while uncertain composition of complex values is represented by modeling their elements as pairs whose components are the element and a *membership* attribute giving the probability that the element is indeed a component of the complex value [4].

4.1. UNCERTAIN ATTRIBUTES

Generally, an uncertain attribute A denotes a probabilistic distribution over the instances of some type T. Such a distribution is a mapping f_A from the domain of T to $[0 \ldots 1]$, where for each $v \in domain(T)$, $f_A(v) = Pr(A = v)$. The description of this mapping depends on the type T. For the purpose of the spatial data model, we are mainly concerned with those attributes whose values are from a basic type such as the real numbers, or vectors of basic types. These attributes are the most useful for representing uncertain spatial information. However, as they range over a potentially infinite set of instances, this distribution is infinite and continuous. We would represent such a distribution by finite means such as its expected value and covariance (see Example 4). It turns out that the useful manipulations of these attributes can be expressed by means of this finite representation as shown in Section 5 below.

Practically, such representations are implemented as classes of *probability distributions* whose instances describe instances of such distributions.

EXAMPLE 4. The position of a desk could be represented as a vector of random variables (X, Y, Z) that are jointly Gaussian in distribution. It is sufficient to provide a vector of expected values $\mu = (\mu_x, \mu_y, \mu_z)$ and the covariance matrix Λ as parameters to completely specify the distribution. Accordingly, we have the *3DCoord_Gaussian* class:

class *3DCoord_Gaussian*
  type **tuple**(*expected_value* : **tuple**(*mu_x*, *mu_y*, *mu_z* : *real*)
    covariance : **array**[3, 3] (*real*))
end;

The robot now measures the position of the table (an uncertain measurement). We wish to represent the position of the desk given the measurement (a posteriori distribution). It turns out that the result is a Gaussian distribution with different parameters, which can be computed using Kalman filter techniques [2]. This is implemented with the method (of the 3DCoord_Gaussian class):

method *Kalman_filter*(*measurement*: *3DCoord_Gaussian*):

3DCoord_Gaussian.

4.2. UNCERTAIN RESULTS OF METHODS

Methods may yield a probabilistic distribution as a result, and may obtain probabilistic distributions as arguments. Intuitively, one can regard methods as parameterized attributes, and hence a distribution computed by such a method is parameterized by its input arguments. For example, consider the distribution of the position of an edge shared by two adjacent surfaces such as the walls of a room. Imprecision in the position of the edge depends on the imprecision in the positions of the two surfaces. This is computed by the method:

edge_position(*position_surface*1, *position_surface*2:

Distribution_plane_parameters):

Distribution_edge_parameters.

4.3. UNCERTAINTY IN OBJECT EXISTENCE AND COMPOSITION

Objects are recognized in a scene with a certain degree of imprecision and consequently their existence is uncertain. We describe this situation through an *existence* attribute of such an object. Semantically, this attribute is associated with the probability that the object in which it appears exists, i.e., *Pr*(*o exists*). For example, the occlusion relationship between predicted image features is often uncertain due to imprecise estimates of feature locations. Accordingly, we associate an existence attribute with the occlusion relationship objects, giving the probability that one feature occludes another:

type *occlusion*: **tuple**(*f*1, *f*2: *ImageFeature*, *existence*: *Probability*)

where *Probability* is implemented as

class *Probability*
type **tuple**(*value*: *real*)
method *threshold*(*cutoff*: *real*): *boolean*
end;

Composition of complex values, built using the type constructors, may be uncertain, and requires associating a probability with each component, the probability that is part of the complex value. Accordingly, we assume that such complex values are built from pairs, whose first component is the element, while the second is called a *membership* attribute and contains the probability that the element is part of the complex value, e.g., **set**(**tuple**(*element*: T, *membership*: *Probability*)), where for a set S and $t \in S$, $t.membership = Pr(t \in S)$.

4.4. *INHERITANCE OF UNCERTAIN INFORMATION*

Uncertain attributes and methods manipulating distributions are inherited along the inheritance hierarchy using the data model standard inheritance mechanism. Similarly, distribution classes are organized in an inheritance hierarchy which includes some basic useful discrete and continuous distributions, such as the Gaussian distribution, important for uncertain spatial information.

5. HANDLING UNCERTAIN SPATIAL DATA

In order to handle spatial probabilistic information, we need to define the kinds of uncertain spatial data, how their probability distributions are represented, and what manipulations are performed on them.

Uncertain information arises in the following cases of geometric information:

1. Uncertainty of position, both global and local, 2D or 3D.
2. Uncertainty of parameters of scaling or distortion.
3. Uncertainty of object (curve, plane, etc.) parameters that are not positional in nature.

As objects may be parts of other objects, their position can be represented by a transformation of their coordinate system w.r.t. the coordinate system of their aggregation-hierarchy parent, or any other coordinate system. Such transformation are 6-vectors, with three parameters for translation and three for rotation (as discussed in Section 3). Uncertain

transformations are thus six-dimensional random vectors. Scaling and distortion are defined w.r.t. the shape that is the parent in the aggregation hierarchy. As above, uncertainty in scaling and distortion is represented as a random vector. Uncertainty in parametric representations used for points, curves, and surfaces is represented by a probabilistic distribution of the parameter.

5.1. PROBABILISTIC REPRESENTATIONS

Having defined where distributions occur, we need to represent them. Probability distributions are either discrete or continuous. In the discrete case, they are represented either by enumeration (for a small, finite set of possible values) or by specifying a known distribution type and parameters. In the continuous case, they are represented by distribution type and parameters. In any event, the distribution types are classes, of which the actual distributions are specific object instances.

We considered several well-known distributions for representing uncertain values:

1. Representation by extents: this usually implies uniform distribution.
2. Discrete representation: enumeration of the probability of each case.
3. Joint Gaussian distribution: representation by expected values and covariance matrix.

We judge the Gaussian distribution to be the best representation for our purposes (and thus use it throughout, unless otherwise specified), for the following reasons:

1. Measurement errors frequently have Gaussian distributions.
2. Gaussian distributions are closed over linear operations. This is not the case for most other distributions.
3. The central limit theorem shows that the sum of many random distributions converges to Gaussian. Thus, it seems to be the simplest, most reasonable approximation for many cases where the distribution is not Gaussian, or unknown.[1]
4. Much current research on performing manipulations on spatial distributions uses Gaussian distributions [2, 23, 16, 20]. We intend to employ results of such research essentially as "plug-in modules" in our system.

---

[1] Counterexamples exist, such as badly bimodal distributions. We consider the treatment of these cases to be beyond the scope of this paper.

5. Efficiency. For example, computing posterior probabilities in Bayesian belief networks (a well-known probability space structuring tool that we intend to use [25]) is NP-hard in the general case, but $O(n^3)$ for Gaussian distributions, for n random variables.

5.2. *DATA MANIPULATIONS INVOLVING PROBABILITIES*

There are several ways in which the vision system manipulates uncertain spatial information represented in the data model. These operations on probability distributions are implemented as methods of the probability distributions' classes. These methods may be overridden in particular distribution classes. We consider the following such operations:

1. **Composition of distributions:** for example, composition of two or more uncertain spatial transformations.
2. **Expectation:** finding the expected value of a distribution (trivial in the case of Gaussian distributions, as it is given in our representation).
3. **Evidential reasoning (probability updating):** finding posterior distributions given evidence (or measurement).
4. **Plausibility evaluation:** finding the probability that an attribute vector is within certain extents. This is done by integrating (or summing, in the discrete case) probability densities over the extents. In certain special cases, one can use well-known cumulative distribution functions, such as using χ^2 distributions in the case of Gaussian random vectors.
5. **Spatial retrieval:** finding all objects that have at least a probability p of being within certain spatial extents. This requires plausibility evaluation, but also a spatial indexing scheme for efficiency.
6. **Reparameterization:** finding the distribution over parameters using a different parameterization than the given one, e.g., specifying a line in 2D using (r, θ) instead of the parameters (a, b) of the line equation $y = ax + b$.

We begin with a simple composition example, where everything is nice and linear.

EXAMPLE 5. Consider our desk *desk*5, at uncertain position $P = (X_P, Y_P, Z_P)$, P being a three-dimensional random vector with jointly Gaussian distribution. (We shall ignore rotations for simplicity, for the moment.) Let $\mu_P = (\mu_{X_P}, \mu_{Y_P}, \mu_{Z_P})$ be the expected values, and Λ_P be the covariance matrix of P.

Let the surface of *desk*5 be uncertainly positioned relative to the desk at $Q = (X_Q, Y_Q, Z_Q)$, with Q likewise Gaussian, with μ_Q and Λ_Q the parameters of the distribution.

We may now wish to find the absolute position R of the desk surface, in order for the robot to place something on it. The result is a composition of the above transformations, in this case a simple addition of the random vectors: $R = P + Q$. Assuming that P and Q are statistically independent (as we usually do, unless we know otherwise), R is also a Gaussian random vector, with μ_R and Λ_R defined as follows (the equations are true even for non-Gaussian independent random vectors):

$$\mu_R = \mu_P + \mu_Q, \tag{1}$$

$$\Lambda_R = \Lambda_P + \Lambda_Q. \tag{2}$$

This is implemented using the method of the Desk class:

part_abs_position(*position*: *3Dcoord_Gaussian*).

Figure 6 depicts the above situation for the two-dimensional case (along the horizontal plane), for $\mu_P = (2,2)$, $\mu_Q = (1,0)$, with the following covariance matrices:

$$\Lambda_P = \begin{pmatrix} 0.4 & 0 \\ 0 & 0.1 \end{pmatrix}, \qquad \Lambda_Q = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.2 \end{pmatrix}.$$

The ellipses are some of the isoprobability contours of the distributions.

Practically, not every transformation is linear. Rotations introduce nonlinearities, as do perspective transformations. Hence, even if the trans-

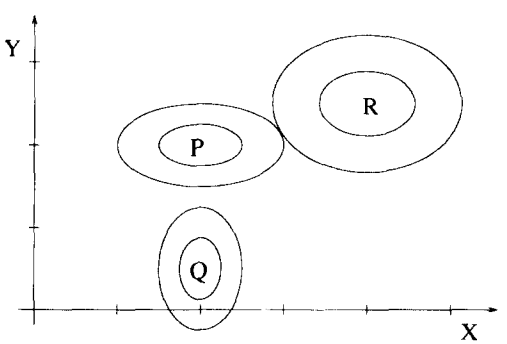

Fig. 6. Isoprobability contours for P, Q, R.

formation has no uncertainty, the uncertainty of a given object may become non-Gaussian. Assuming that the uncertainty is small, we can linearize the transformation around its expected value, to get an approximately Gaussian distribution of the transformed object.

EXAMPLE 6. An estimate of the desk's location in an image is required, in order to begin tracking it. Assume for simplicity a central projection from (0,0,0) looking along the Z axis (usually the projection is more general, and the parameters may be uncertain). The transformation is

$$p(P) = \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -\frac{X}{Z} \\ -\frac{Y}{Z} \end{pmatrix}. \tag{4}$$

The location of the desk $p = (x_p, y_p)$ in the image is (almost) Gaussian, with μ_p and Λ_p defined as follows (for small uncertainties):

$$\mu_p = \begin{pmatrix} -\frac{\mu_{X_p}}{\mu_{Z_p}} \\ -\frac{\mu_{Y_p}}{\mu_{Z_p}} \end{pmatrix}, \tag{5}$$

$$\Lambda_p = J_{p,P} \Lambda_P J_{p,P}^T, \tag{6}$$

where superscript T stands for "transposed," and $J_{p,P}$ is the Jacobian matrix of the perspective transformation $p(P)$ evaluated at μ_P, i.e., the 2×3 matrix

$$J_{p,P} = \begin{pmatrix} \frac{\partial x}{\partial X} & \frac{\partial x}{\partial Y} & \frac{\partial x}{\partial Z} \\ \frac{\partial y}{\partial X} & \frac{\partial y}{\partial Y} & \frac{\partial y}{\partial Z} \end{pmatrix} = \begin{pmatrix} -\frac{1}{Z} & 0 & \frac{X}{Z^2} \\ 0 & -\frac{1}{Z} & \frac{Y}{Z^2} \end{pmatrix}$$

$$= \begin{pmatrix} -\frac{1}{\mu_{Z_p}} & 0 & \frac{\mu_{X_p}}{\mu_{Z_p}^2} \\ 0 & -\frac{1}{\mu_{Z_p}} & \frac{\mu_{Y_p}}{\mu_{Z_p}^2} \end{pmatrix}. \tag{7}$$

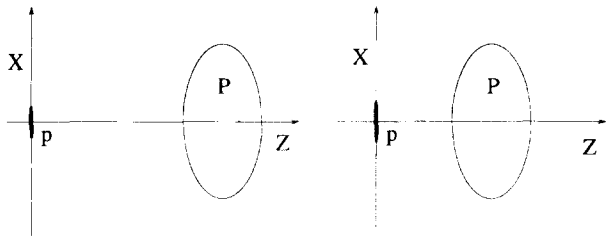

Fig. 7. Isoprobability contours for projection.

Figure 7 shows the isoprobability contours projected on the XZ plane, for two cases, where the covariances are 0.

To update distributions after observation, we use well-known results from estimation theory on maximum-likelihood estimators (see, e.g., [13], a textbook on estimation). When using the least-squares error criterion over a number of observations, essentially one needs to remember the results of every observation. Under certain assumptions (Gaussian random variables), it is sufficient to keep the results of all previous observations as a single distribution, and incorporate each new observation in turn. This is particularly useful for the data model, in which we currently have no concept of time and thus cannot keep track of a history of observations (but even if we did, not having to keep track of all observations saves considerable memory space.) The technique that allows for this kind of updating is known as Kalman filtering.

In a Kalman filter, the previous system estimate, S_n, and its error estimate (a covariance matrix Λ_n), are updated by the system model of motion (which may have an error as well), to get a *prediction* S^p_{n+1} and its error model. An observation O_{n+1} is made at time $n+1$, and its related error model is stored temporarily (another covariance matrix). Based on the error models for the prediction and observation, a Kalman gain matrix is computed. The new system model S_{n+1} is computed as a weighted average between the prediction and observation, where the weight is a function of the Kalman gain matrix. The new error model Λ_{n+1} is also computed. A complication is that the variables we can observe (in O_{n+1}) usually do not coincide with all (or any) of the system variables in S_n. Kalman filtering addresses this problem as well; for more details, see [2, 16]. Instead of delving into estimation theory, we explain our use of the method by example.

Our system uses a special case of the Kalman filter techniques of [2, 16], which is straightforward, assuming linear transformations. We choose to

represent uncertainty using Gaussian random variables, again for the reasons stated above. Note that we do not need to keep track of previous observations, or the time at which they were made. Thus S_n and Λ_n are simply the parameters stored in the database for the object under consideration, and these values are simply overwritten by (the new) S_{n+1} and Λ_{n+1}.

EXAMPLE 7. Our *desk*5 has been measured to be at an uncertain location $M=(X_M, Y_M, Z_M)$, where M is a random vector with jointly Gaussian distribution (this is our observation O_{n+1}). Assume that the measurement is independent of the distribution P from the previous example (the state of the system, S_n in the Kalman filter description.) We would like to update the database as to the location of the desk (under the assumption that the desk is stationary, for simplicity, and thus our prediction of the desk location is also P.) Call the new uncertain location P', again a Gaussian random vector.

Using the Kalman filter equations, we have

$$\mu_{P'} = \mu P + K(\mu_M - \mu_P), \tag{8}$$

$$\Lambda_{P'} = (I_3 - K)\Lambda_P, \tag{9}$$

with I_n being an $n \times n$ identity matrix, and K being the "Kalman gain matrix," defined as $K=\Lambda_P(\Lambda_P+\Lambda_M)^{-1}$ in this special case. Figure 8

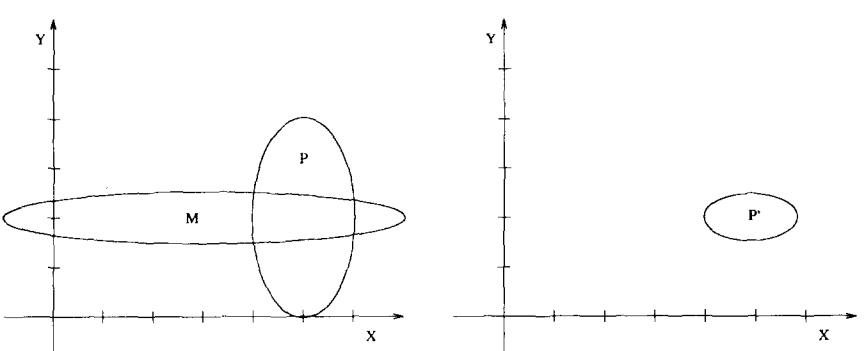

Fig. 8. Isoprobability contours for M, P, P'.

shows the isoprobability contours projected on the XY plane, where again the covariances are 0.

5.3. COMPLEXITY OF UPDATES

Updating one object's spatial distribution can percolate in the model, and may force us to update distributions for other object parameters. Define the generalized constraint graph to be $G=(V,E)$, where there is a node in V for every spatial parameter of every object, and $(v_1,v_2)\in E$ whenever v_1 and v_2 stand for parameters that are mutually constrained (e.g., desk and leg locations) or are directly conditionally (probabilistically) dependent (e.g., desk surface location and desk location). The complexity of the update operation is as follows. If the constraint graph is unconnected, with k the number of objects in the component at which the update is originated, only k objects need to be considered. If the graph is connected, but not highly connected, updates can still be done reasonably efficiently: in singly connected constraint graphs, updates take linear time. In a multiply connected graph, updates can be done in $O(k^3)$ in the worst case [16, 25]. With m multiply connected components of maximum size k (assuming the multiply connected components are connected), the complexity of updates is $O(k^3m)$. For multiple updates, for l updates occurring in the same multiply connected component G', it is more efficient to propagate updates only in G', and only once all these updates are complete, to propagate the changes to the rest of G. Correctness is assured, and the complexity for l updates in this case is only $O(k^3(m+l))$, rather than $O(k^3ml)$.

For better performance, it should be possible to postpone updates until query time. Additionally, if a query accesses objects in the same multiply connected component G' as all the updates, it is possible to update only G', even at query time. Nevertheless, a query may need information from several components, or the component in question may be unknown at query time. For example, the query may attempt to retrieve all objects likely to be within certain spatial extents. This can usually be done efficiently with reasonable indexing. However, an update may mandate a change in the index, which cannot be predicted in general without actually carrying out the update. A possible solution is to carry out updates only if the (local) change in spatial parameters is over a certain threshold, which would require a more permissive indexing scheme. This issue is in our scope of future research. Needless to say, if the assumptions of independence and Gaussian random variables and constraints are dropped, the update problem becomes NP-hard in the general case. Fortunately, spatial location equality constraints (such as: desk leg origin constrained to be

exactly at corner of desk) is a special case of a Gaussian constraint, with a null covariance matrix.

Similar update methods apply to uncertain transformations, including rotations and projections. In certain nonlinear cases, if the resulting uncertainty is small (which is possible if there is a large number of measurements, even if they each have a large uncertainty), it is possible to overcome the problem using iterations that converge to the correct results [16]. The general problem of handling large uncertainties, in the presence of nonlinearities, remains open. We choose to ignore the problem, and instead focus future research on being able to handle a richer language of spatial object types and parameterizations.

6. CONCLUSIONS

We have presented a probabilistic spatial data model whose novel contribution is in the description of uncertain spatial information through probability distributions. A prototype has been implemented in CLOS (Common LISP Object-System), an object-oriented package of Common LISP. The implementation provides the distribution classes useful for our current visual tracking prototype, such as representation by extents and joint Gaussian distribution, together with methods for their manipulations, such as composition of distributions. Based on these classes, representations of office objects, such as those mentioned above, have been constructed.

This paper significantly extends the data model of [21], but its most important addition is in the handling of uncertain *spatial* data. The latter is an extremely complicated problem, but by introducing certain assumptions often made in the research community (Gaussian distributions, and independent measurements), we can use standard estimation theory tools, in particular a special case of Kalman filtering, to update the database as observations are made. Extending the model to handle distributions that are significantly different from Gaussian, or that disobey the independence assumption, is an issue for future research.

We intend the database to provide the basic functionality of filtering objects using size, rough location, and aggregation-hierarchy information, in order to facilitate the feature prediction effort in the robot navigation and tracking application. Another application of the database is for a robot touring the environment and learning it, i.e., through updating its distributions of object locations. Recent research performs multi-image fusion of data using constraints and Kalman filtering [2, 16]. Managing the results of

such fusion and providing the constraints over a large environment is the responsibility of the probabilistic spatial database system.

This work is partially supported by the Paul Ivanier Center for Robotics and Production Management, Ben-Gurion University.

REFERENCES

1. R. Agrawal and N. Gehani, ODE (object database and environment): The language and the data model, in *ACM SIGMOD Int. Conf. on Management of Data*, 1989, pp. 36–45.
2. N. Ayache, *Artificial Vision for Mobile Robots*, MIT Press, Cambridge, MA, 1991.
3. F. Bancilhon et al., The design and implementation of O_2, an object-oriented database system, in K. Dittrich, Ed., *Proc. Int. Workshop on Object-Oriented Database Systems*, volume 334 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 1988, pp. 1–22.
4. D. Barbara, H. Garcia-Molina, and D. Porter, A probabilistic relational data model, in *Proc. Int. Conf. on Extending Database Technology*, volume 416 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 1990, pp. 60–74.
5. J. Brolio et al., ISR: A database for symbolic processing in computer vision, *IEEE Computer*, pp. 22–30 (1989).
6. R. A. Brooks, Model-based three-dimensional interpretations of two-dimensional images, *IEEE Trans. PAMI* 5(2):140–149 (1984).
7. R. Cavallo and M. Pittarelli, The theory of probabilistic databases, in *Proc. Int. Conf. on Very Large Data Bases*, 1987.
8. R. T. Chin and C. R. Dyer, Model-based recognition in robot vision, *Comput. Surv.* 18-1 (1986).
9. S. Dutta, Qualitative spatial reasoning: A semi-quantitative approach using fuzzy logic, in A. Buchmann et al., Eds., *Design and Implementation of Large Spatial Databases*, volume 409 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 1989, pp. 345–364.
10. S. Dutta, Topological constraints: A representational framework for approximate spatial and temporal reasoning, in O. Gunther and H. Schek, Ed., *Advances in Spatial Databases*, volume 525 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 1991.
11. A. Elfes, Using occupancy grids for mobile robot perception and navigation, *IEEE Computer*, pp. 46–57 (1989).
12. J. D. Foley and A. V. Dam, *Fundamentals of Interactive Computer Graphics*, Addison-Wesley, Reading, MA, 1982.
13. A. Gelb, *Applied Optimal Estimation*, MIT Press, Cambridge, MA, 1974.
14. E. Gelenbe and G. Hebrail, A probability model of uncertainty in databases, in *Proc. IEEE Data Engineering Conf.*, 1986.
15. A. Goodman, R. Haralick, and L. Shapiro, Knowledge-based computer vision: Integrated programming language and data management system design, *IEEE Computer*, pp. 43–54 (1989).
16. Y. Hel-Or and M. Werman, Absolute orientation from uncertain point data: A unified approach, in *Proc. CVPR Conf.*, 1992.

17. H. Jagadish and L. O'Gorman, An object model for image understanding, *IEEE Computer*, pp. 33–41 (1989).
18. P. Kahn, L. Kitchen, and E. M. Riseman, A fast line finder for vision-guided robot navigation, *IEEE Trans. PAMI* 12(11):1098–1102 (1990).
19. J. J. Koenderick and A. J. V. Doorn, The singularities of the visual mapping, *Biol. Cyber.* 32:51–59 (1976).
20. Y. Kornatzky and S. E. Shimony, Predicting image features for tracking and navigation, in *Proc. Int. Conf. on Automation, Robotics and Computer Vision*, 1992.
21. Y. Kornatzky and S. E. Shimony, A probabilistic object-oriented data model, *Data & Knowledge Eng.* 12:143–166 (1994).
22. D. Kriegman, T. Binford, and T. Sumanaweera, Generic models for robot navigation, in *Proc. DARPA Image Understanding Workshop*, 1988, pp. 453–460.
23. L. Matthies, R. Szeliski, and T. Kanade, Kalman filter-based algorithms for estimating depth from image sequences, *Int. J. Comput. Vision* 3(3):209–238 (1989).
24. J. Orenstein and F. Manola, PROBE: Spatial data modeling and query processing in an image database application, *IEEE Trans. Software Eng.* 14(5):611–629 (1988).
25. J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Mateo, CA, 1988.
26. N. Roussopoulos, C. Faloustsos, and T. Sellis, An efficient pictorial database system for SQL, *IEEE Trans. Software Eng.* 14(5):639–650 (1988).
27. P. Suetens, P. Fua, and A. Hanson, Computational strategies for object recognition, *ACM Comput. Sur.* 24(1):5–62 (1992).
28. A. J. Vayda and A. C. Kak, A robot vision system for recognition of generic shaped objects, *CVGIP: Image Understanding* 54(1):1–46 (1991).
29. A. Wolf, How to fit geo-objects into databases—An extensibility approach, in *Proc. Int. Conf. on Extending Database Technology*, volume 580 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 1992.

Received 1 June 1994; revised 24 June 1995