A Medium Complexity Discrete Model for Uncertain Spatial Data

Erlend Tøssebro and Mads Nygård
Department of Computer Science, Norwegian University of Science and Technology
Email: tossebro, mads@idi.ntnu.no

Abstract

This paper presents a method for representing uncertainty in spatial data in a database. The model presented requires moderate amounts of storage space. To compute the probability that an object is at a particular place, the representation employs probability functions that can be computed quickly and efficiently. This is different from an advanced model presented by the same authors. This medium complexity model is less powerful, but requires much less storage space, and computing probabilities is much less complicated.

1. Introduction

In many cases, one does not have accurate measurements of the position and shape of a geographic or spatial object. However, one often knows roughly where the object may be, and how uncertain its position or shape is. When storing such objects in a spatial or spatiotemporal database, it is therefore important not just to be able to store the fact that the object is uncertain, but to somehow store how uncertain the object is.

Spatial and spatiotemporal objects may be uncertain because the measurements needed to place the object accurately are too expensive, or because exact measurements are impossible. Two examples of this are given below. In the first example, it is theoretically possible but practically infeasible to make exact measurements. In the second example, it is impossible to measure the objects exactly.

Example 1: Imagine you have scientists who are driving around making measurements in the Sahara desert to determine the extent of underground water reservoirs. The scientists themselves are uncertain points due to the imprecision of the positioning system that they use. The roads are uncertain lines because the roads in the Sahara desert are more like routes that shift as the sand dunes move than paved roads. The water reservoirs that the scientists are studying are uncertain regions because they are located deep underground and it is therefore not feasible to do more than a few measurements at each site. The scientists therefore lack the necessary information to define them precisely.

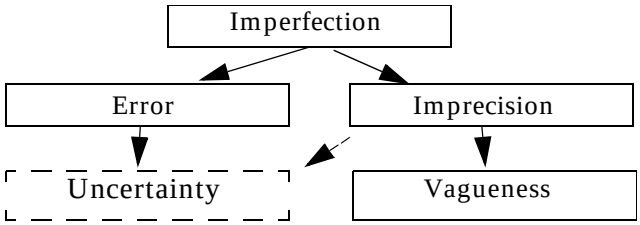


Figure 1 Hierarchy of the types of imperfection

Example 2: In a historical spatial database[1] one may need uncertainty about all three spatial data types. The extents of ancient empires are uncertain regions, because one may not know precisely where the borders were. Rivers were important for both trade and agriculture in the past, and these rivers have shifted in their beds several times. Therefore, many rivers should be stored as uncertain lines. Some ancient cities may only be known from written records, which may be imprecise about the location of the city. Such a city should be stored as an uncertain point.

Our model should be useful in representing and manipulating cases like those described in Example 1 and Example 2.

In [2], an ontology of different kinds of uncertainty is defined. Imperfection is considered to be the general form of uncertainty. Error is when measurements do not reflect reality. Imprecision is when measurements are lacking in specificity or are incomplete. [2] considers vagueness[2], to be a subcategory of imprecision.

The basic goal of this paper is representing uncertainty in the position or extent of an object, regardless of the source of that uncertainty. In the rest of this paper, uncertainty therefore means either measurement error or imprecision due to incomplete knowledge. This definition of uncertainty is shown by the dashed box in Figure 1.

1. A database containing map data about ancient civilizations.
2. An example of a vague statement would be that Bergen is in the south of Norway, because the south of Norway is not clearly defined.

In [11], we gave an overview of our work on uncertainty in spatial and spatiotemporal databases. This paper will present a medium complexity discrete model, being one of three different discrete models designated in that paper. A discrete model is a model that is directly implementable, unlike abstract models that often use infinite point sets. The new model will be compared to the advanced discrete model from [12] with respect to storage requirement, how easy it is to implement some operations on them, processing speed of the operations and modelling capabilities.

The data model described in this paper has been partially implemented by the first author. All three data types have been implemented as java classes, which include some important operations on these types.

2. Related work

There are two basic discrete methods for storing spatial data, the raster and vector models. The raster model divides space into a partition (typically a grid) and stores one value in each cell. For vague data, [6] and [7] present raster models in which a fuzzy membership value is stored in each cell. Fuzzy sets [17] is a type of set in which the membership of an individual may be fuzzy, that is, have a value between 0 and 1. The problem with raster models is that the data volume quickly becomes very large if the spatial objects are to be stored accurately. The advantage of raster models is that the values of arbitrarily complex functions may be stored explicitly in the cells. Also, overlay operations in which the values from two functions are combined are much easier to compute for raster models than for vector models.

Vector models, on the other hand, store the boundary of a region as a set of line segments. This form of storage is more complex, but usually takes much less space. An example of a vector model for uncertain regions is presented in [9]. However, you cannot store probability functions with this model. One approach for storing distinct probabilities in such a model is to store a number of different regions, one inside the other, where each successive region has a higher probability than the one before. This method is used to compute fuzzy intersection in [10].

An early model for uncertain points and lines is presented in [4]. In this model, an uncertain point is stored as a central point with a circular deviation. The probability of a point being at any one place follows a bi-gaussian distribution over the deviation. Lines are made up of a series of such points. The probability of each of these potential line segments is the product of the probabilities of the two points being at those precise locations.

[8] describes a way to model uncertainty in the location of the boundary of a region that uses probabilistic error bands. This means that on each side of the estimated border there is an area with a certain width in which the border can be. Additionally, the probability that a point p is inside the area is a function of the distance from the estimated border to p.

[15] describes a model for fuzzy boundaries between regions in which the fuzzy membership function indicates how sharp the boundary is. A membership of 1.0 indicates a crisp boundary.

[1] describes several ways to extract fuzzy objects from observations. These methods use a combination of fuzzy sets and probability theory. The model that they use is a raster model because fuzzy membership values are stored in each cell. However, they also group the cells into objects according to different criteria.

[16] uses rough sets to define the outer and inner boundaries of possibly imprecise spatial objects. [16] defines resolution objects that are partitions on the underlying space, and shows how to convert objects from one resolution to another. This process may introduce imprecision even if the original representation was precise because the object may only partially overlap the new partition parts.

A discussion of the last five models compared to our work will be given in Section 7.

3. Basis for the new model

As mentioned in Section 2, there are two basic modelling techniques for creating a discrete model: raster and vector. The vector approach is chosen in this paper because rasters require much more storage space for regions, and lines may become inaccurate when stored in a raster format. Additionally, this paper presents ways to compute probability functions in a vector model, so that a raster model is no longer needed to store probability values or fuzzy set values.

Spatial data types in this paper are given names in the following format: The type name begins with an A for abstract or a D for discrete. For the uncertain types, DA means advanced discrete and DM means medium-complexity discrete. After these initial letters, the type name is given in subscript, with an initial U if the type is uncertain. For instance DM_{UPoint} is a medium-complexity discrete uncertain point.

The types for crisp spatial data listed in Table 1 will be used to define the types in this paper. These are taken from [5] with the exception of D_{Seg} and D_{Curve}. These two are different because a more specialized definition of lines is needed in the uncertain case. In this paper, the following terminology will be used for lines. A line segment is a

Table 1: Types for crisp spatial data

Type name	Type definition
D_{Point}	A single point
D_{Points}	A set of points
D_{Seg}	A single line segment consisting of a start point and an end point
D_{Curve}	A set of D_{Seg} forming a continuous line
D_{Cycle}	A D_{Curve} where the start point and end point are the same
D_{Face}	Area that has an outer cycle and a number of disjoint hole cycles
D_{Region}	Consists of a number of disjoint faces

straight line going between two points. A curve is a single continuous line that does not intersect itself. A curve consists of a set of line segments. A line is a set of curves.

The basic idea from [11] is that all uncertain objects, regardless of type, are known to be within a certain crisp region. It may also be known where the object is most likely to be. This is modelled as a function over the plane for all three types[1].

The abstract uncertain point (A_{UPoint}) is modelled as a region with a probability density function indicating where the point is most likely to be. The abstract uncertain curve (A_{UCurve}) is modelled as a core line with gradient lines crossing it, and probability functions for both of these. The probability distribution function along the core line represents uncertainty about the existence of the line and the length of the line. The probability density function along the gradient lines represents the fact that the exact location of the line is not known. These gradient lines must form a crisp face which is the area in which the line might possibly be.

The abstract uncertain face (A_{UFace}) is modelled as a probability distribution function over the plane. The set of points with function value above 0 must form a crisp face.

4. Medium complexity model

The model presented in [12] manages to model many aspects of uncertainty, but at the cost of increased complexity and increased storage space, especially for points. For applications which requires the full power of the model from [12], that model would be good. However, for many applications a simpler model may be sufficient. This section presents a model that is simpler than the model from [12], while maintaining the goal of modelling all uncertain spatial data types.

In these definitions, *ProbMass*(*P*) indicates that *P*(x) is a probability mass function and *ProbFunc*(*P*) indicates that *P*(x) is a probability distribution function.

- Probability Mass Function:
  $ProbMass(P) \equiv (\forall x{:}P(x) \geq 0) \wedge \sum_x P(x) \leq 1$
- Probability Distribution Function:
  $ProbFunc(P) \equiv \forall x{:}(P(x) \geq 0 \wedge P(x) \leq 1)$

Probability mass functions are used whenever there is only one random variable. Because there is only one random variable, the sum of the probability masses of all possible singleton events should be at most one. For instance, an uncertain point can be at only one place. Therefore the probability distribution of an uncertain point should be a probability mass function. A simple way of storing a one dimensional probability mass function that is shaped as a series of steps is described in [3].

Probability distribution functions are used whenever there are multiple random variables. Each point that is a potential member of an uncertain region is a random variable of its own. Therefore the probability distribution of an uncertain region should be a probability distribution function.

Probabilities (DA_{Prob}) are floating point numbers between 0 and 1. In discussing the types, the number system used for coordinate values will be referred to as *CVS*. In these definitions, the *Core* is where the spatial object is known to exist. The *Support* is the region where the object has a probability greater than 0 of being. An *Alpha-Cut*(*object*, *alpha*) is the region in which the probability of the *object* being there is greater than *alpha*.

4.1. Uncertain Points

One major problem with the model for uncertain points from [12] is that the storage space needed may be far larger than that needed for crisp points. If the database models mainly points and if storage space is an issue, models requiring less storage space may be needed. One way to limit the amount of storage space needed is to limit the number of points that make up the boundary of support of the uncertain point. One natural way of doing this would be to store the distance from the central point to the boundary of the support at certain predefined angles, such as every 45 degrees. An example of such an uncertain point is shown in Figure 2.

The probability mass values of the uncertain point can now be computed based on the relative distance from the central point and the edge of the support.

1. Points, lines and regions.

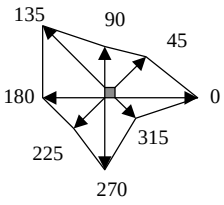


Figure 2: Medium complexity uncertain point with eight angles

Definition 1: The **uncertain point** is defined as follows:

$$DM_{UPoint} \equiv \{(a_0 a_1 a_2 \ldots a_{N-1}, cp, ps, pe) | \\ a_{i|i=0\ldots N-1} \in CVS \wedge cp \in D_{Point} \wedge \\ ProbMass(ps) \wedge pe \in DA_{Prob} \wedge \\ Increasing(ps)\}$$

The disadvantage of this model compared to the previous one is that it cannot model holes, and that it can only model uncertain points in which there is a straight line from the central point to any point in the support.

Another problem with this model is that the results of spatial set operations such as finding the intersection of an uncertain point and a face[1] are not necessarily members of the base type. They are not members because the result can have corners in different places than on the particular angles that are stored for the uncertain points. One solution is to store the normal point and to indicate that the probability mass function is the product of the normal probability mass function and the probability distribution function of the face. This problem does not occur in the advanced model from [12] because the support of the point can be an arbitrary face in that model.

One advantage of this model compared to the advanced one is that the storage space needed is known and bounded. Storing a single distance for every 45 degrees yields eight numbers. Because the central point needs two and the probability mass function requires one, this means that the uncertain point takes 11 numbers to store, or 5.5 times as much as a crisp point. For a number of angles *n*, the uncertain point takes $((n/2) + 1.5)$ times as much space.

An alternative to this model might be to store a variable number of distances and angles. This would require that the angle was also explicitly stored and would have the problem of an arbitrary number of angles. This means that such a solution is essentially the same as the one presented in [12]. To conserve storage space, one would have to limit the number of angles and distances stored.

The **uncertain points** set type is a set containing only DM_{UPoint} values.

1. This is the part of the support of the uncertain point that is inside the face.

4.2. Uncertain Lines

The storage cost for an uncertain curve in the model from [12] is 3.75 times that of a crisp curve. This section will describe a model which takes somewhat less space. Another advantage is that it is very easy to compute probabilities for this model. The method for doing this will be described in Section 5. One disadvantage is that the model presented here does not allow holes in the support. Another problem is that spatial set operations, such as finding the parts of an uncertain line that is inside a given region, may require some additional support. This problem is further discussed in [14].

The basic idea is to store a central line, as well as crossing lines for each stored point along the central line. These crossing lines are equally long on each side of the central line, and determine the extent of the support of the line. These crossing lines are the “gradient lines” from [11] for the end points of each line segment. In the interior of the line segment, the gradient lines have an angle which is linearly interpolated between the two gradient lines at the end. The support of the uncertain curve is determined by taking straight lines between the ends of all the crossing curves. Straight lines are used to make it easier to run plane-sweep algorithms on the support.

One example of a line stored in this fashion is shown in Figure 3. From this figure, one can clearly see how the crossing lines determine the shape of the support of the uncertain line.

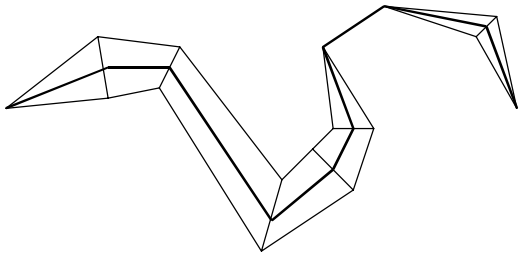


Figure 3: Medium complexity uncertain curve

The only aspects that need to be stored about the crossing lines (Hereafter called *CrossCurves*) are the length of the line and the angle between the line and the segment to which it belongs.

Definition 2: The **CrossCurve** is defined as follows:

$$D_{CCur} \equiv \{(a, b) | (a \in CVS \wedge b \in ANG)\}$$

In this formula, *ANG* is an angle. Angles are represented with floating-point numbers.

A line segment in this model may be defined as a single line segment of the central curve and the *CrossCurves* at

each end of it. Each line segment contains a probability distribution function indicating how likely it is that the actual line exists in the various parts of the segment as well as its probability of existing at the beginning and end. The line segment also contains a probability mass function which applies along the *CrossCurves*. Storing the probability mass function in the segment rather than for the entire curve makes one able to use different functions for different parts of the curve. However, there will be discontinuities in the probability values if the function changes. Therefore, most curves should use a single function throughout the curve. The option to use different probability functions is there to enable union and intersection operations on uncertain regions with different probability functions (See Section 4.3).

Definition 3: The medium complexity **uncertain segment** is defined as follows:

$$DM_{USeg} \equiv \{(cc, bc, ec, pb, pe, pf, pc) | \\ cc \in D_{Seg} \wedge bc \in D_{CCur} \wedge ec \in D_{CCur} \wedge pb \in DA_{Prob} \wedge \\ pe \in DA_{Prob} \wedge ProbFunc(pf) \wedge ProbMass(pc) \wedge \\ CenteredOn(bc, cc.sp) \wedge CenteredOn(ec, cc.ep)\}$$

CenteredOn means that the given point is on the middle of the *CrossCurve*.

Definition 4: The medium complexity **uncertain curve** is defined as follows:

$$DM_{Ucurve} \equiv \{(SS, pe) | \\ SS \subseteq DM_{USeg} \wedge pe \in DA_{Prob} \wedge \\ ContCurve(SS) \wedge \\ \forall a \in SS, \forall b \in SS, a \neq b \rightarrow \neg Ccross(a, b)\}$$

ContCurve is true if the set of uncertain segments forms a continuous curve. They form a continuous curve iff for all $a \in SS$ except possibly one[1], there exists a $b \in SS$ such that $(a.cc.endpoint = b.cc.begpoint) \wedge (a.ec = b.bc)$ is true. *Ccross* for uncertain segments is true iff their central curves cross.

In Figure 3 the *CrossCurve*s at the end have length 0. To model a crisp curve, all the *CrossCurves* should have length 0.

Compared to a crisp line, the *CrossCurves* and function references have an additional cost. A single *CrossCurve* contains two numbers. That means that the two crosscurves in the uncertain segment require four numbers. The uncertain segment requires four numbers. The probability values require two numbers. The probability functions require two numbers. Because a crisp line segment can be stored with four numbers, an uncertain line segments takes 12/4=3 times as much space as a crisp one. This is only slightly better than for the advanced model.

1. The last segment.

The **uncertain line** is defined as a set of uncertain curves.

4.3. Uncertain Regions

One problem with both the abstract model from [13] and the model presented in [12] is that the border of an uncertain region sometimes is not a valid uncertain line. This may be solved by defining the uncertain face the same way as the crisp face with the exception that uncertain cycles are used instead of crisp ones. An example of such a face is shown in Figure 4. An uncertain hole in the core can be stored in this model by using an uncertain cycle which is not certain to exist.

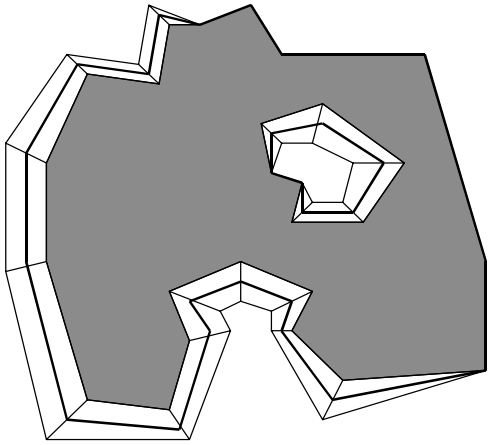


Figure 4: Medium complexity uncertain face

Definition 5: The **uncertain cycle** is defined as follows:

$$DM_{UCyc} \equiv \{sc \in DM_{UCurve} | \\ IsCycle(sc.SS) \wedge ConstProb(sc.SS)\}$$

In this definition, *IsCycle* means that the set of uncertain segments forms a cycle. A set of uncertain segments forms a cycle iff both the central line and the outer lines form cycles, all of the segments have $D_{USeg}.pf = 1.0$ for all the points along the segment, and the probabilities of existence of all the segments are the same. *ConstProb* means that all the line segments in the set have the same constant probability of existing.

A hole in the support of an uncertain region is stored like a regular hole. When computing probabilities or isolines, one computes these for both the outer line and the uncertain hole. If one uses fuzzy set mathematics, which are less accurate but easier to compute, alpha-cuts can be produced by taking the alpha-cut of the main face and subtracting the same alpha-cut of the hole. When using probability theory it becomes more complex because the

functions are multiplied together rather than taking the maximum or minimum.

This model cannot store a face with multiple core regions directly. This can only be done by storing several faces that have overlapping supports but non-overlapping cores in the same uncertain region.

However, it is very easy to determine the probability function at individual points as well as iso-lines of probability. In Section 5 it will be shown that this is even easier for these regions than it is for medium complexity uncertain lines.

The uncertain face also uses a different kind of probability function than the uncertain curve. In the gray area enclosed by the cycles, the probability of existence is always 1. Inside the supports of the uncertain cycles, the probability of existence is the sum of the probability mass function of the uncertain curve taken from outside and inward. To avoid having to compute this, it might be better to store this sum directly as a function rather than storing the probability mass function for the curves. If the function of the curves is also needed by the application, both may be stored together.

The definition of the uncertain face is taken from [5] and modified to use uncertain cycles.

Definition 6: The **uncertain face** is defined as follows:

$$
\begin{aligned}
&DM_{UFace} \equiv \{(bc, HS, ps, pe) | \\
&bc \in DM_{UCyc} \wedge HS \subseteq DM_{UCyc} \wedge \\
&ProbFunc(ps) \wedge pe \in DA_{Prob} \wedge \\
&\forall a \in HS \rightarrow EdgeInside(a, bc) \wedge \\
&\forall a \in HS, \forall b \in HS, a \neq b \rightarrow EdgeDisjoint(a, b) \wedge \\
&bc.sc.pe \equiv 1\}
\end{aligned}
$$

In this definition, *EdgeInside* means that all the line segments of *a* are in the interior of the cycle defined by *bc* with 100% certainty, and *EdgeDisjoint* means that the interiors of two cycles are certainly disjoint. The sum of a probability mass function is a probability distribution function. Therefore, *ps* is a probability distribution function. *Existence* is the probability that an uncertain object exists.

The definition of the **uncertain region** is a set of non-overlapping uncertain faces.

To make this model computationally closed under normal set operations, ways of dealing with uncertain curves that cross each other must be introduced. In Figure 5, the union and intersection of two example objects are shown. From this figure, one can see that the results of such set operations contain places in which one have to use just parts of some uncertain segments. The dotted lines from the figure shows where the segments are divided. This line also separates between segments that originated in the two curves.

However, this does not solve the problem when there is one or more *CrossCurves* inside the area in which the lines may possibly intersect. The solution to this problem is omitted due to space constraints. It is described in [14].

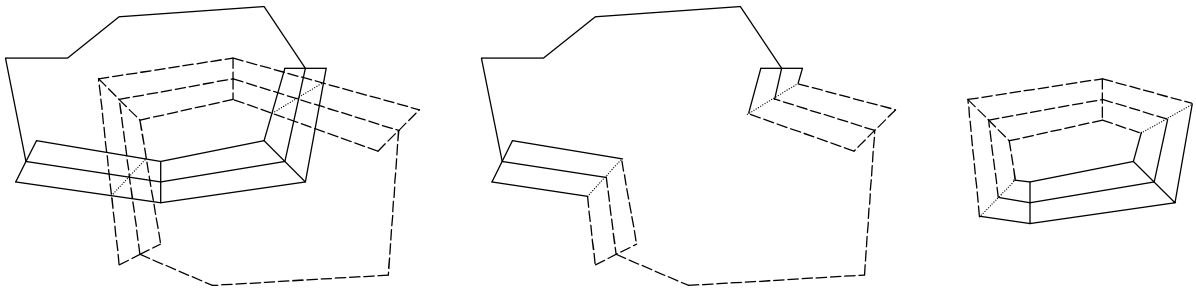
The increase in storage space for an uncertain region compared to a crisp region in this model is the same as for the uncertain curve (3X). This means that it actually costs slightly more storage space to store an uncertain region in this model than one stored in the model from [12].

However, for this slight increase in storage space, we gain the ability to compute alpha-cuts and the probability that a given crisp point is inside the region in an efficient and consistent manner. This is very difficult for the model from [12].

5. Storing and computing probability functions

The goal of this section is to find ways of using one-dimensional functions to compute the probabilities or probability densities rather than two-dimensional ones. This is done because one-dimensional functions are much easier to define for the user of the system and much easier to store and compute for the computer.

For the following algorithm to work, the probability functions should be functions that accept input values from



a) Original objects **b)** Union **c)** Intersection

Figure 5: Union and intersection of medium complexity regions

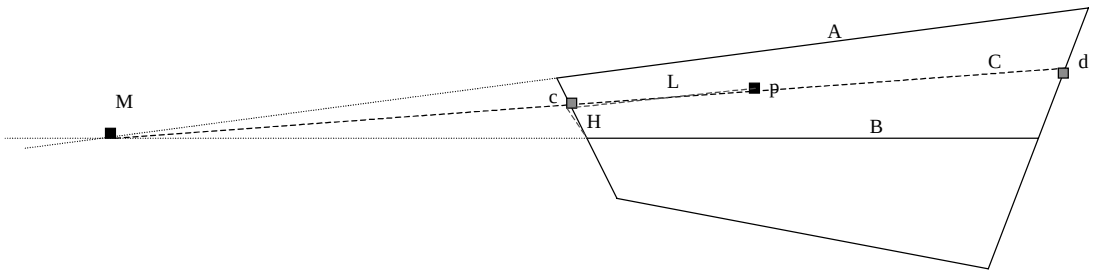


Figure 6: Computing probability values for medium complexity uncertain segments

0 to 1. The actual distances should be scaled to be between 0 and 1.

For the **lines** and **regions** in the medium complexity model, computing the probability functions is easy. In the end points of each uncertain line segment, the probability is computed along the *CrossCurves*. In the interior of **regions**, the function value for a point *p* should be computed as follows:

1. Take the lines *A* and *B* (the central line) from Figure 6 and compute the point *M* in which they intersect
2. Compute the line *C* from *M* to *p*.
3. Compute the point *c* or *d* where the line *C* crosses the *CrossCurves*.
4. Compute the distance *H* between the point *c* and the central curve *B* along the *CrossCurve*.
5. Find how long *H* is compared to the *CrossCurve*. This ratio determines the value of the probability distribution function for point *p*.

Two special cases need to be considered. The first is if the lines *A* and *B* are parallel, because in this case there is no meeting point *M*. In this case, the line *C* should be parallel to *A* and *B* and go through *p*.

The second is if the line *A* has length 0. In this case the line *A* becomes the point *a*. Compute the line *K* that passes through both *a* and *p*. *H* is now the distance from *p* to the line *B* along *K*, and the ratio from 5) uses the length of *K* instead of the length of the *CrossCurve*.

This method of computing probabilities has several advantages compared to other models. In Dutton's model for uncertain lines, one would need to take the sum of the probabilities of all the possible lines that might cross *p* to determine the probability that the line was in *p*. This is much more costly than the procedure outlined above.

Computing iso-lines of the probability is also simple in the model presented here. The iso-line passing through point *p* in Figure 6 is line *C*.

For **lines**, the process to compute individual probability values is similar to the process for regions described above. The process for lines requires one more step:

6. Find how long the distance *L* is compared to the distance between *c* and *d*. This ratio is used to find the probability that the curve exists at point *p*.

This probability is then multiplied with the probability value from step 5.

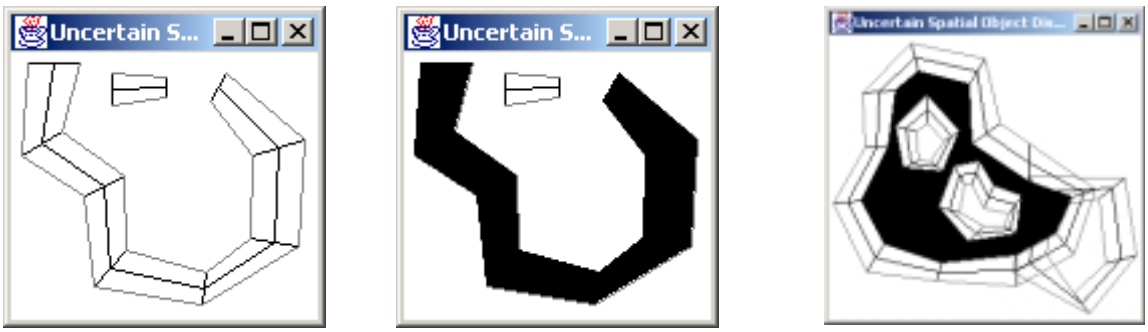
One also uses a probability mass function along the gradient lines rather than a distribution function. This mass function needs to be normalized to yield a sum of 1 regardless of the length of the gradient. Instead of using *H* and the length of the *CrossCurve* to determine the probability function, one has to find the length of the gradient that passes through *p*. The angle between this line and line *B* is linearly interpolated. Because the length of *L* is known, this ratio is easy to determine. One then needs to find where this gradient crosses *A* and *B*. Then the distance between *A* and *B* along the gradient and the location of *p* on the gradient is used to determine the value of the probability mass function.

For lines, computing iso-lines of a given probability is only slightly harder than for regions as long as the probability functions are linear. If both $DM_{USeg}.pf$ and $DM_{USeg}.pc$ are linear, the iso-lines will be straight lines. Then one just needs to compute the location of the iso-lines along the *CrossCurve*s.

6. Implementation

The model presented in this paper has been implemented as a set of java classes. A number of the operations from [14] have also been implemented for these types. The implementation was designed with one class for each type (point, points, curve, line, face, region) as well as superclasses for types with common properties. Some screenshots from the implementation are shown in Figure 7.

Although set operations for all types are fundamentally based on the same mathematical concepts, they must be implemented differently. The union of two sets of uncertain points can be performed normally. However, the union of two uncertain regions must check for overlaps between the member faces and join together those that overlap. This is because the semantics of union for regions is a union of the



a) Two uncertain curves **b)** Support of one of the uncertain curves **c)** Core of an uncertain face

Figure 7: Screen shots from the implementation

set of points that is contained in the region, not a union of the face sets.

As we have used fuzzy set mathematics instead of the more complex probability theory, some operations are fairly easy to implement. For instance, it is possible to handle holes in uncertain faces that overlap with the support. However, fuzzy set mathematics yields less accurate results for uncertainty. The full discussion on this is omitted due to space constraints, but may be found in [14].

7. Discussion

This paper has presented a new discrete model for storing uncertain spatial data. The advantages of this model over the advanced model from [12] is that it takes much less storage space and that the probability functions are much easier to compute. One problem with this model is that it cannot model holes in lines. It also cannot easily model faces with disjoint cores but single support. Thus it is less expressive than the advanced model. It also has the problem that a special construct has to be defined to make them computationally closed for regions. The advanced model does not need any special considerations for this.

Compared with earlier models, the major advantage of the new model for spatial data is that it can model all the normal spatial data types. [4] only describes modelling points and lines, although it is simple to extend his model to regions. An advantage of the model presented here over the model from [4] is that it is much easier to compute the probability values. In [4], only the points have a probability function, and to compute the probabilities of the line passing through a given point, one has to take the integral of all possible placements of the two end points which would yield a line passing through the given point. This integral is infeasible to perform in practise in a database system, although it could be done numerically in a simulator in which one has a lot of time available.

The approaches for storing imprecise regions in [8] and [15] are similar to our approach in that they store the uncertainty in the border line. However, they cannot store lines and points. These models are also more abstract in nature than this model as they base themselves on continuous lines rather than line segments.

The models from [1] and [16] are essentially rasters, which means that they can store complex functions but take a lot of storage space.

Compared to [9], our model is able to store more types of data and is also able to store probability functions. The method for computing fuzzy intersection from [10] is very costly if one desires high accuracy because one would have to store many regions, one inside the other. Our method produces fairly accurate results with a much lower storage requirement.

The different advantages and drawbacks of the model presented here and the advanced model presented in [12] show that none of the two models are obviously superior to the other. One should choose which to use based on what data one has and which functionality and operations one needs.

Storing spatial uncertainty takes more space than only storing crisp data. How much depends on the model chosen. For the model presented here, uncertain lines and region require three times as much space and uncertain points take 5.5 times more space than crisp ones. This is a smaller increase than in [12], but still significant.

Because the model presented here models points, lines and regions in a uniform way, it is better suited to databases like Example 1 than previous models which handles only one or two of the types. The types presented here are better integrated than a system consisting of a point model, a line model and a region model from different authors chosen because they are good models for the individual types. We are currently working on the issue of the completeness and computational closeness of this model. Converting from line to region (*enclosed_by* operation) and from region to

line (*border* operation) is trivial in this model because the border line of an uncertain face is explicitly stored in the face. Such conversion from point to line (*connecting_line* operation) and line to point (*end_points* operation) will be published elsewhere. Of course, conversion from point to region and vice versa is not applicable.

The time taken to process data depends on the operation. Plane-sweep algorithms typically take O(n*log(n)) time. However, these do not usually run on all the points at once. For regions with only spatial uncertainty, they run on the core region and the support region separately. Therefore the increase in processing cost is proportional to the increase in data stored. This is also true for uncertain lines, although the support here normally takes twice as much space as a crisp line. This means that an operation on the core will take somewhat longer compared to the size of the line than operations on crisp lines would. Only for points is there an increase in processing cost significantly greater than the data increase, because operations that before only had to operate on a single point now have to operate on the support of the point, which consists of eight points.

References

[1] T. Cheng, M. Molenaar, T. Bouloucos: Identification of Fuzzy Objects from Field Observation Data. In S. C. Hirtle and A. U. Frank (eds.) *Spatial Information Theory: A Theoretical Foundation for GIS*, LNCS vol. 1329, Springer-Verlag, 1997, pages 241-259.

[2] M. Duckham, K. Mason, J. Stell, M. Worboys: A formal approach to imperfection in geographic information. In *Computers, Environment and Urban Systems*, 25, 2001, pages 89-103.

[3] C. E. Dyreson and R. T. Snodgrass: Supporting Valid-time Indeterminacy. In *ACM Trans. on Database Systems*, vol. 23, no. 1, pages 1-57.

[4] G. Dutton: Handling Positional Uncertainty in Spatial Databases. In *Proc. SDH'92*, vol. 2, pages 460-469

[5] L. Forlizzi, R. H. Güting, E. Nardelli and M. Schneider: A Data Model and Data Structures for Moving Objects Databases. In *Proc. ACM SIGMOD Int. Conf. on Management og Data* (Dallas, Texas), pages 319-330, 2000

[6] P. Lagacherie, P. Andrieux and R. Bouzigues (1996): Fuzziness and Uncertainty of Soil Boundaries: From Reality to Coding in GIS: In *Geographic Objects with Indeterminate Boundaries*, GISDATA series vol. 2, Taylor & Francis, 1996, pages 155-169.

[7] K. Lowell: An Uncertainty-Based Spatial Representation for Natural Resources Phenomena. In *Proc. SDH'94* vol. 2, pages 933-944.

[8] D. M. Mark and F. Csillag: The nature of boundaries of 'area-class' maps. In *Cartographica*, 26, 1989, pages 65-77.

[9] M. Schneider: Modelling Spatial Objects with Undeterminate Boundaries using the Realm/ROSE Approach. In *Geographic Objects with Indeterminate Boundaries*, GISDATA series vol. 2, Taylor & Francis, 1996, pages 155-169.

[10] M. Schneider: Fuzzy Spatial Querying Based on Topological Predicates for Complex Crisp and Fuzzy Regions. In *Proc. ER2001 Conference*, pages 103-116.

[11] E. Tøssebro and M. Nygård: Abstract and Discrete Models for Uncertain Spatiotemporal Data. In *Proc. 14th Int. Conf. on Scientific and Statistical Databases (SSDBM)*, page 240, July 2002.

[12] E. Tøssebro and M. Nygård: An Advanced Discrete Model for Uncertain Spatial Data. In *Proc. 3rd Int. Conference on Web-Age Information Management (WAIM02)*, pages 37-51, 2002.

[13] E. Tøssebro and M. Nygård: Representing Uncertainty in Spatial Databases. *Submitted for publication.*

[14] E. Tøssebro: Representing uncertainty in spatiotemporal databases. Ph.D. Thesis.

[15] F. Wang and G. B: Hall: Fuzzy representation of geographical boundaries in GIS. In *Int. Journal of Geographical Information Systems*, 10(5), 1996, pages 573-590.

[16] M. F. Worboys: Imprecision in Finite Resolution Spatial Data. In *GeoInformatica*, 2(3), 1998, pages 257-279.

[17] L. A. Zadeh (1965): Fuzzy sets. In *Information and Control*, 8, pages 338-353, 1965