A THEORY OF METRIC SPATIAL INFERENCE

Drew McDermott

Yale University
Department of Computer Science
New Haven, Connecticut

ABSTRACT*

Efficient and robust spatial reasoning requires that the properties of real space be taken seriously. One approach to doing this is to assimilate facts into a "fuzzy map" of the positions and orientations of the objects involved in those facts. Then many inferences about distances and directions may be made by "just looking" at the map, to determine bounds on quantities of interest. For flexibility, there must be many frames of reference with respect to which coordinates are measure. The resulting representation supports many tasks, including finding routes from one place to another.

****

In the past, AI researchers have often sought to reduce spatial reasoning to topological reasoning. [4, 6, 8] For example, the important problem of finding routes was analyzed as the problem of finding a path through a network or tree of known places. This sort of formulation throws away the basic fact that a route exists in real physical space regardless of our knowledge of any of the places along the way. So a network-based algorithm will fail to exhibit two important phenomena of route-finding:

> Often you know roughly what direction to go in without having any idea of the details of the path, or even if the path is physically possible.

> You can tell immediately that you don't know how to get to a place, just by verifying that you don't know the direction to that place.

There are many other problems that a topological approach fails to treat adequately. Here are some of the problems we (Ernie Davis, Mark Zbikowski and I) have worked on:

> How are metric facts, such as "A is about 5 miles from B" or "The direction from A to B is north" to be stored?

> How are queries such as "Is it farther from A to B than from A to C?" to be answered?

> Given a large set of objects and facts relating them, how do you find the objects that might be near some position? or with some orientation?

Some of these problems have received more of our attention than others. In what follows, I will sketch our approach, the details of various algorithms and data structures, and the results we have so far.

All of our solutions revolve around keeping track of the fuzzy coordinates of objects in various frames of reference. That is, to store metric facts about objects, the system tries to find, for each object, the ranges in which quantities like its X and Y coordinates, orientation and dimensions lie, with respect to convenient coordinate systems. The set of all the frames and coordinates is called a fuzzy map. We represent shapes as prototypes plus modifications. [3, 5] The domain we have used is the map of Yale University, from which most of my examples will be taken.

The advantage of the fuzzy-map approach is the efficiency to be gained from the fact that many inferences are performed by "just looking" at a fuzzy map, regardless of the complexity of the formulas that led to its creation. For example, Figure 1 represents a fuzzy map which locates Kline, Sterling Library, and Dunham with respect to Yale. The input which led to the creation of this map could have been a collection of entirely unruly constraints, such as "Dunham is nearly due south of Kline," "Dunham is 500 meters from Kline," "Dunham is in the block just northeast of the center of Yale," and "Sterling is due west of Dunham, about half as far away as Kline." It would be very difficult and time consuming to answer the question "What direction is it from Sterling to Kline?" given constraints in this form. However, once the data have been assimilated into a map, answering this question takes no more time than answering, "How far is Dunham from Sterling?", even though the answer to this second question was given explicitly.

To date we have written programs to do these tasks:

(1) Given a stream of metric relationships, create a fuzzy map of the objects involved.

Research supported by NSF under contract MCS7803599

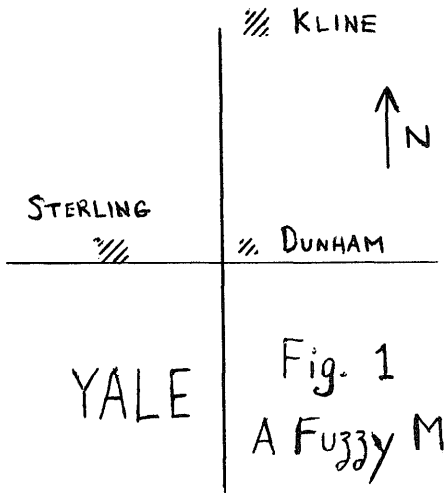


Fig. 1
A Fuzzy Map

(2) Given a fuzzy map, test the consistency of a relationship or find the value of a term.

(3) Given a fuzzy map, find objects with a position and orientation near some given value.

(4) Plot a course around objects or through conduits discovered using (3).

So far we have invested most of our effort in the study of task (2), what I described as "just looking" at the map to see what's true. This actually involves using hill climbing to see if a relationship can be satisfied, or to find the possible range of values of a term. So, in Figure 1, to answer the query "What's the distance from Kline to Sterling in meters?" the system plunks down two points in the fuzz boxes of Kline and Sterling, and moves them as close together, then as far apart, as it can. To answer the query "Is Kline closer to Dunham than to Sterling?" it looks for a configuration of points from the fuzz boxes of Kline, Dunham and Sterling in which Kline is further from Dunham than Sterling. (Since it fails to find it, the answer to the query is "Yes.")

The same hill-climbing algorithm is used for task (1), the assimilation of facts into a map. In this case, the object is to find the smallest and largest possible values of each "primitive term" involved in a relationship. (A primitive term is a quantity like (X A) or (LENGTH A) that characterizes an object's position, orientation or dimensions. More complicated terms, like (DIST A B), are functions of primitive terms.) The new, smaller range of a primitive term is then stored for future reference. This device, called fuzz constriction, suffices to capture many spatial facts.

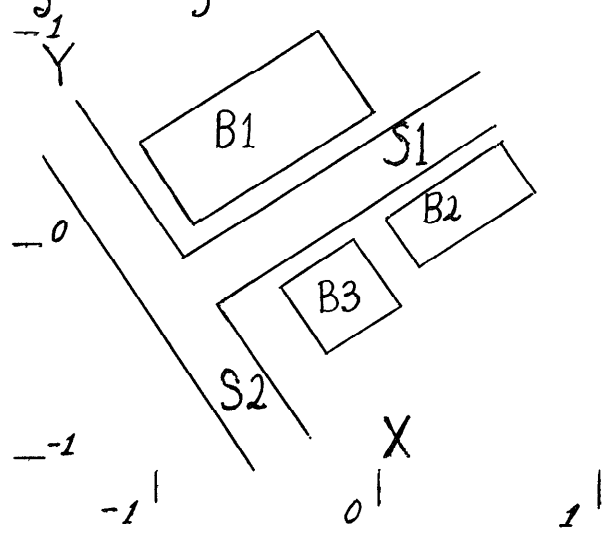
However, it can happen that the range of a primitive term does not change at all after a new fact is assimilated, especially when the fact relates objects about which little is known. For example, if we tell the system that the orientation of Sterling Library is the same as the orientation of Becton Library, when it knows nothing about their orientations with respect to Yale, this new fact doesn't constrain them any further. In this case, mere fuzz constriction has failed to capture the new fact. The solution is to introduce a new frame of reference F, and to indicate that (ORIENT STERLING) = (ORIENT BECTON) = 0.0 in this frame, while the orientation of F is completely fuzzy (from 0 to 2pi) with respect to Yale.

We are able to create flexible new configurations of frames to capture facts because we have a very clean mechanism for relating frames of reference. To begin with, every object can serve as a frame of reference, and every frame of reference can be considered an object, with a position, orientation, and scale. The system records the coordinates of each object with respect to one particular frame, its "parent" frame. For example, the parent frame of my office is Dunham Lab, because the system keeps track of (X OFFICE), (Y OFFICE), (ORIENT OFFICE) etc. with respect to Dunham Lab. That is, these quantities are measured with respect to an origin, an x-axis, and a scale which is peculiar to Dunham Lab. Yale is the parent frame of Dunham because (X DUNHAM), (Y DUNHAM) etc. are measured with respect to Yale. If the system ever needs to know the position of my office with respect to Yale (say, to plot a trip from my office to the library), it computes (X OFFICE) with respect to Yale on demand. At any given time, the "parentage" relation combines the objects into a tree; but the tree can be rearranged as more facts are added (see previous paragraph).

The machinery introduced so far enables us to retrieve characteristics of given objects. It is also important to be able to retrieve objects given their spatial characteristics (task (3)). For example, if you are trying to get from one place to another in a city, you will want to know what streets to use, i.e., how to find the nearest "long porous objects" with approximately the right position and orientation. This is a job for k-d trees of the kind devised by Bentley. [1, 2] In these trees, a large set of objects is broken down into manageable chunks by an obvious generalization of binary search: repeatedly discriminate on each coordinate. An example is shown in Figure 2.

The original version of k-d trees was designed to work on data bases in which all primitive terms have known values. In our application, most primitive terms can only be assigned ranges. To deal with this problem, we take the following tack: if a given attribute of an object is "very fuzzy" (e.g., its orientation is known only to lie between 0 and 2 pi), then we do not index it on that attribute. But if it is only moderately fuzzy,

Fig 2a Objects to be Indexed



then we index it as though its value were the midpoint of its range. This requires that on retrieval we be willing to look around a little for objects fitting our requirements. That is, if we need to find a street with a given orientation, we keep relaxing the requirement until we find one. Obviously, a street found after many relaxations is only a plausible candidate, and must be proven to actually work; and the process must be terminated when it is unlikely to give worthwhile results.

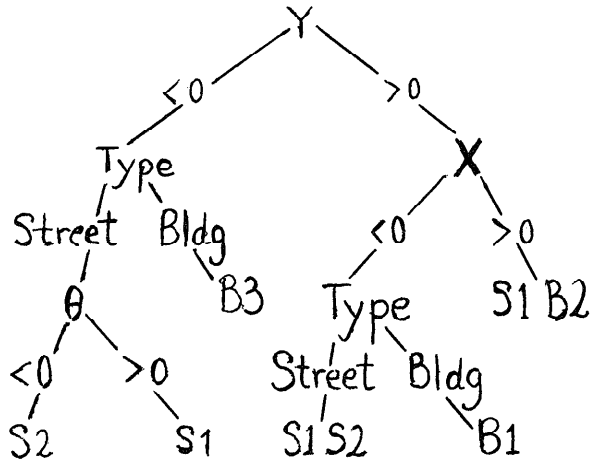


Fig 2b Index Tree

This algorithm for finding objects that might have given characteristics is used by our route-finding programs. Exactly what finding a route means depends on the density of the region to be traversed. If it is mostly open, then the problem is to plan to avoid obstacles; if it is mostly obstacle, then the problem is to plan to use conduits. Either way, the system must find objects with known properties (e.g., "open and pointing in the right direction" or "filled and lying in my way").

To summarize, our main results so far are these: representing space as a structure of multiple frames of reference, within which objects have fuzzy positions, is efficient and robust. In this context, assimilation is the process of constricting fuzz or creating new frames to capture a new fact. Route finding involves computing a fuzzy vector from where you are to where you want to be, then finding objects which can help or hinder your progress, and altering the plan to take them into account.

Many problems still remain. The assimilation algorithm needs improvement. The route finder has not yet been completed or connected with the assimilation and retrieval algorithms. As yet we have not implemented a (simulated) route executor, although this is a high priority.

Acknowledgements: Ernie Davis and Mark Zbikowski have helped develop many of the ideas in this paper, and made suggestions for improving the exposition.

REFERENCES

[1] Jon Bentley 1975 Multidimensional binary search trees used for associative searching, Comm. ACM 18, no. 9, pp. 509-517

[2] Jon Bentley and Jerome Friedman 1979 Data structures for range searching, Comput. Surveys 11, no. 4, pp. 397-409

[3] John Hollerbach 1975 Hierarchical shape description of objects by selection and modification of prototypes, Cambridge: MIT AI Laboratory Technical Report 346

[4] Benjamin Kuipers 1978 Modeling spatial knowledge, Cognitive Science 2, no. 2, p. 129

[5] David Marr and H. Keith Nishihara 1977 Representation and recognition of the spatial organization of three dimensional shapes, Cambridge: MIT AI Laboratory Memo 416

[6] Drew McDermott 1974 Assimilation of new information by a natural language-understanding program, Cambridge: MIT AI Laboratory Technical Report 291

[7] Drew McDermott 1980 Spatial inferences with ground, metric formulas on simple objects, New Haven: Yale Computer Science Research Report 173

[8] James Meehan 1976 The metanovel: writing stories by computer, New Haven: Yale Computer Science Research Report 74