QUALITATIVE SPATIAL REASONING UNDER UNCERTAINTY IN GEOGRAPHICAL INFORMATION SYSTEMS

UTE LOERCH *HANS W. GUESGEN*
Computer Science Department
University of Auckland
Private Bag 92019, Auckland, New Zealand
Phone: +64-9-373-7599
Fax: +64-9-373-7453
E-mail: ute, hans@cs.auckland.ac.nz

ABSTRACT

The capability of geographical information systems (GIS) to display images and analyse them is one of the most powerful, because people have a very consistent ability to reason about visual features according to their own knowledge model. As images are accepted as a symbolic view of the world, with familiar and intuitive entities, the set of tasks we desire for the performance of the GIS includes image understanding and image manipulation.
Usually the abilities of GIS are restricted dealing with quantitative data only, so that they fail whenever exact matches can't be found. A solution for this problem is not only to allow quantitative information, but also qualitative one. Proximity factors such as 'close to' and 'far from'. The qualitative approach offers a more natural way for the user of the system to specify what he actually wants, by using proximity operators.
This paper is an attempt of integrating qualitative spatial reasoning into geographic information systems.
The idea is to associate qualitative information with fuzzy sets and to use the values of these fuzzy sets for spatial reasoning. The authors are concerned with the problem of image interpretation, which stands for the geometric and semantic recognition of the objects contained in one image.

KEYWORDS

Geographical information systems, qualitative reasoning, spatial querying, fuzzy sets, hill climbing, simulated annealing, colour spaces.

1. INTRODUCTION

Qualitative spatial reasoning is a way to reason about spatial information that is very close to human reasoning.
However, the major criticism of qualitative reasoning is that the approach imposes some degree of uncertainty in the reasoning process, as it is primarily reasoning with deduced information. There are a number of methods that have been developed to deal with uncertainty.
Numerically orientated approaches (including Bayesian probability [1], certainty factors, Dempster-Schafer theory of evidence [2], fuzzy logic [3]) and symbolic approaches (including Cohen's Theory of Endorsements [4] and Fox's semantic system [5], [6]) are the two main streams in the area of reasoning under uncertainty. None of these methods is ideal for all applications. The purpose of this paper is not to argue which uncertainty handling technique is the best. Instead we will concentrate on fuzzy logic, one of the numerically orientated approaches, which has been applied successfully in many related areas. The paper will concentrate on proximity operators like 'far' and 'close' and the question of getting the right set of factors that make up the membership functions and determine the significance of each of them. There are several physical factors that will affect the human recognition of closeness like absolute distance, effect of scale, frame of reference and so forth. We will achieve the goal to impact the problem complexity on proximity through empirical research and a survey. The paper also focuses on the computational costs and the efficiency of the implementation. The most dominant factor that impacts on human recognition of closeness will be revised in the computational point of view, and a description of the algorithm will be given.
The aim of spatial querying is to provide the user with a solution to his query. By using a fuzzy-based qualitative query, searching for a suitable location will be one of the primary aims of the user. The most suitable location is the best location, and therefore we are talking about an optimal solution.
Since spatial querying is always concerned with huge spaces, such as in a geographical information system, the problem and solution space are the ones to be large. Searching for an optimal solution in a two or higher-dimensional space can make exhaustive searching techniques too expensive in terms of computational costs. In this paper, hill climbing and simulated annealing will be described briefly, two probabilistic searches that are general strategies for tackling problems that cannot be solved in polynomial time.

2. PROXIMITY FUZZY PREDICATES

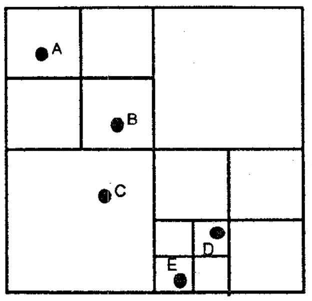
Overview

Generally, analysis of spatial data involves the usage of either the connectivity or the similarity of spatial objects. From this point of view, analysis of point data, where only the location of points is given without any attributes, is a rather peculiar task, which often requires special tools.

A data set consisting of irregularly distributed points within a region is referred as (spatial) point pattern. The objective of analysis of a point pattern may involve test of complete spatial randomness (CSR), estimation of intensity, stochastic model fitting etc. For detailed discussion of point patterns see, e.g., [9] or [10]. Various distance as well as quadrat methods differ in definition of this 'generalized connectivity'.
The absolute distance might be the most important factor that affects human recognition of closeness. It can be obtained simply by applying the distance formula $d=\sqrt{(y_p-y_r)^2+(x_p-x_r)^2}$ to a primary object with the coordinates (x_p, y_p) and a reference object with the coordinates (x_r, y_r). However, the problem gets much more complex if the spatial query does not deterministically specify the reference object. For instance, consider the following spatial querying: "Find a place far from a city". The query does not specify which particular city it should be. With such non-deterministic reference objects, the problem of obtaining the absolute distance becomes a computationally expensive problem.

Quadtree

A quadtree is a hierarchical data structure that is tailored for spatial data.
The basic principle of a quadtree is to cover a planar region of interest by a square, then recursively partition squares into smaller squares until each square contains a suitably uniform subset of the input; for instance this can be used to compress bitmap images by subdividing until each square has the same color value. This and related partitions have many applications in computational geometry and geometric applications, including data clustering, shape representation and many more.
It is useful for many operations manipulating two dimensional spatial data (see Figure 1).
Figure 1 also shows how the set of points are stored in an actual tree structure.
Notice that each of the leaf nodes is either empty or contains a single data point.



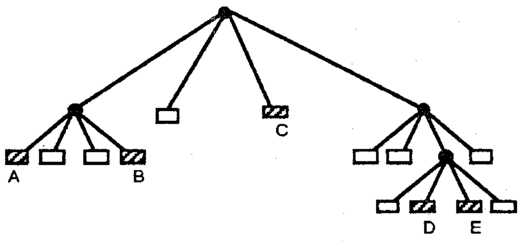


Fig 1: Quadtree and the records it represents.

Figure 2 shows an image containing a triangle, and how it is represented as a quadtree (down to 5 levels of division). Each square is either filled with grey or white.
One drawback of the quadtree is that the depth of the tree does not directly depend on the number of data items in the tree. Two data points which are extremely close together may require an enormous amount of subdivisions in order to assign different regions to them.

However, it is known that the upper bound for the depth of an *m x m* quadtree is $\log_2 m$.

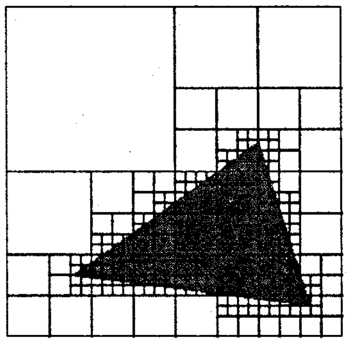


Fig 2: Quadtree of an object.

Description of the algorithm for finding the nearest reference object

The idea of the algorithm is to cut down the search space based on the distribution of the reference objects. This is achieved by performing the following three steps:

1. Find a region as small as possible that guarantees to contain the nearest reference object.

2. Pull out all the reference objects in that region.

3. Calculate the distance from each of the objects pulled out in step 2 and get the nearest neighbour.

Step 1 can be implemented as shown in Figure 3 (see [11] for details):

```
function regionContainNearestObj( location P) :region ;
{getting a region as small as possible that quarentees to
contain the closest spatial object from P}

begin
  QuadTreeNode temp;
  real distance;
  temp ← leaf node containing location P;
  if temp is not empty then
    distance ← distance between P and the location of
    the spatial object stored
  else
    distance ←opposite corner of the parent of the temp;
    return new circular region centred at P with radius of
    distance;
end;
```

Fig 3: Algorithm for getting a region as small as possible that contains the nearest reference object.

The main principle underlying step2 of the algorithm is to 'divide and conquer'. Starting with the root node it repeatedly verifies if the current node intersects with the starting region. If this is the case, it recursively calls the four children in the next level until it reaches the lowest level.

```
recursive procedure GetObjInRegion( region r;
QuadtreeNode currentNode);
{getting all the spatial object within the region r}
begin
   if r intersects with the area represented by
      currentNode then
   begin
      if currentNode is leaf node then
      begin
         if currentNode is not empty and the data point
            inside r then
            include data point;
      end;
      else
         recursively call the four children of
         currentNode;
   end;
end;
```

Fig 4: Algorithm for finding all the objects within a region in a quadtree.

It is obvious that the efficiency of the algorithm depends on the area obtained in step 1. The smaller the area is the less operation steps have to be carried out.
But the computational costs of the algorithm are dominated by the second step. An analysis of the computational complexity gives us the following result:

- The best case and average case complexity is $O(\log_2 m + F)$, where F is the number of data points pulled out from the searching region.

- The worst case complexity is $O(m^2 + N)$, where N is the number of all spatial objects.

3. FUZZIFYING THE DISTANCES BETWEEN COLOURS

Reasoning about geographic information, i.e., the extraction of new information from stored spatial data, is usually quantitative in nature. On the other hand, humans often prefer a qualitative analysis over a quantitative one, as this is more adequate in many cases from the cognitive point of view. Our model of qualitative spatial reasoning is based on fuzzy maps.
We assume that the information to produce these maps is already available in electronic form. This assumption is certainly valid for a large amount of geographic information. In the following, we will develop a framework for converting 'fuzzy' maps into a representation that can be incorporated with a system for fuzzy spatial reasoning. The main idea of our approach is to determine a set of reference colours, each colour corresponding to an object of a particular type.
Instead of searching for a particular colour, we look for colours in a particular range, like all areas from light to deep blue. In other words, we are looking for a colour that is close to a given reference colour, which means that we need to define distances between colours. The smaller the distance of some colour to the reference colour, the more we consider this colour to be the same as the reference colour.
By mapping distances to fuzzy values such that a distance of 0 is mapped to a fuzzy value of 1, we are able to define a fuzzy set for the reference colour. Some colour spaces are perceptually linear, i.e. a 10 unit change in stimulus will produce the same change in perception wherever it is applied.

Colour Spaces

Many colour spaces, particularly in computer graphics are not linear in this way. The CIE Luv is a perceptually uniform colour space, in which two colours that are equally distant are perceived as equally distant by viewers. The L component defines the luminancy and u and v the chrominancy. CIE Luv is often used in calculation of small colours or colour differences, especially with additive colours.

Most of the pictures available in electronical format use a representation in the RGB colour space, where the colour of a pixel is split into its red, green, and blue component.
Most visible colours can be generated that way. Since the RGB colour space is nonlinear, coordinates in the RGB colour space have to be transformed into coordinates in the CIE Luv colour space by first mapping them to coordinates (x,y,z) in the CIE space and then mapping the result to coordinates (L,u,v) in the CIE Luv space.
In particular, the transformations are as follows. Given a color (r,g,b) in the RGB colour space, a transformation matrix M can be used to calculate the corresponding colour (x,y,z) in the CIE space:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = M \begin{pmatrix} r \\ g \\ b \end{pmatrix} \text{ with } M = \begin{pmatrix} x_r & x_g & x_b \\ y_r & y_g & y_b \\ z_r & z_g & z_b \end{pmatrix}$$

where , x_r, x_g, x_b are the weights applied to the monitor's RGB colours to find x, and so on. The following matrix has been used for the conversion of the RGB colour space into the XYZ-space:

$$M = \begin{pmatrix} 0.431 & 0.342 & 0.178 \\ 0.222 & 0.707 & 0.071 \\ 0.020 & 0.130 & 0.939 \end{pmatrix}$$

The values in the matrix are based on a reommendation defined by the CCIR (French acronym for Comite Consultatif International des Radio-communications) for the conversion from the RGB colour space into the CIE colour space. The transforms are only valid if the equipment matches the required specifications, or the images used have been encoded to these standards. If this is not the case, the calculated CIE values will not be true CIE. One of these conditions is that the white point coordinates in the CIE colour space are fixed as $(x_n;\ y_n) =$ =(0.312713; 0.329016).

Figure 5 shows representation of the RGB cube in CIE space. The x-, y- and z-axes are indicated with the left, right and vertical arrow.

After obtaining the coordinates of the colour in the CIE space, we are able to calculate the coordinates (L,u,v) by applying the following formulas:

$$L = 116\sqrt[3]{y/y_n} - 16$$

$$u = 13L(u' - u_n') \qquad v = 13L(v' - v_n')$$

$$u' = \frac{4x}{x + 15y + 3z} \qquad v' = \frac{9y}{x + 15y + 3z}$$

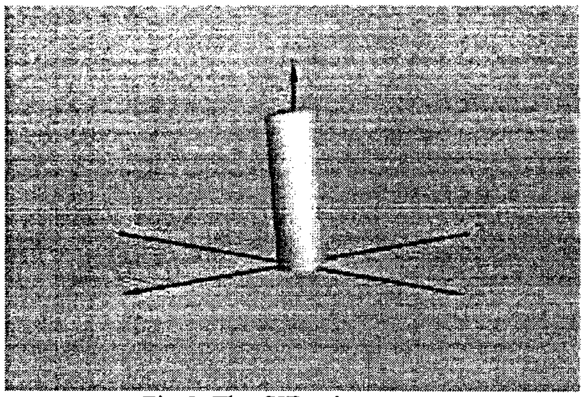


Fig 5: The CIE colour space.

$$u_n' = \frac{4x_n}{x_n + 15y_n + 3z_n} \qquad v_n' = \frac{9y_n}{x_n + 15y_n + 3z_n}$$

where (x_n, y_n, z_n) are the coordinates of the colour that is defined as white. The new coordinates are in a colour space that is perceptually uniform, which means that the distance between, for example, two reds is comparable with the distance between two greens, etc. In other words, given a reference colour for a certain area like water, we can now associate with each point on the map a fuzzy value that indicates the degree to which the area is considered to be water. Figure 6 shows a representation of the RGB cube in Luv space. The u-axis is indicated by the left arrow, the v-axis by the right arrow and the L-axis by the vertikal arrow.

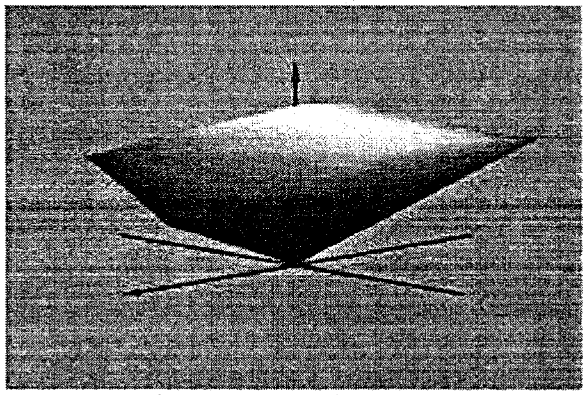


Fig 6: The Luv colour space.

4. OPTIMISATION IN FUZZY CONSTRAINT NETWORKS

The result of spatial queries without qualitative extensions is mostly a set of spatial objects that satisfy the constraints. A location can only either satisfy or not satisfy the given set of boolean constraints. To obtain

the most suitable location is one of the most common types of qualitative spatial querying.
Searching for the best location is an optimisation problem. There are generally two categories of approaches for the optimisation problem: exhaustive searching techniques and probabilistic searching techniques.
Since exhaustive searching techniques, such as back-tracking and branch-and-bound, can only guarantee to obtain the best location by computing the degree of satisfaction for every single location within the frame of conference, we adopted a probabilistic searching technique for our problem.
Hill climbing is one of the general purpose searching methods. The whole idea is to continously climb up to a better neighbouring configuration until no better neighbouring configuration exists or a satisfactory level has been reached.

Hill climbing

The algorithm for simple hill climbing is outlined in Figure 7.
It is a local optimisation method, which means that a solution found by this method might not be the best solution possible. Hill climbing might terminate without reaching the optimal configuration state when getting to a state from which a better neighbouring state cannot be generated. When hitting a local maxima, the search will terminate since we can no longer make any positive transitions.
There are a number of ways to tackle local maxima in hill climbing. And yet none of them can guarantee to get an optimal solution, as they to only avoid getting trapped in a local maximum. These methods are:

- Backtracking
- Relaxing the criteria of neighbourhood
- Multiple starting locations
- Simulated Annealing

```
function SimpleHillClimb (function f) : configuration;
{search for the best configuration of f using simple hill
climbing}

begin
   configuration currentConfig;
   configuration neighbourConfig;
   currentConfig ← randomly choose a starting
                   configuration;
   repeat
       neighbourConfig←randomly select a neighbour
                   configuration that has not yet
          been selected;
       if  f(neighbourConfig) is better than
          f(currentConfig)          then
              currentConfig ← neighbourconfig;
   until no better neighbour configuration exists;
   return currentConfig;
end;
```

Fig 7: Simple Hill Climbing Algorithm.

It is beyond the scope of this paper to discuss all these methods in details, instead we want to briefly describe the optimisation methods used for our purposes.

Simulated Annealing

Simulated Annealing is a randomised variation of hill climbing. By doing some exploration of the landscape early at the beginning of the search, the result becomes relatively insensitive to the starting location and the likelihood of being caught by a local maxima can be reduced. Starting from an initial point, the algorithm computes a new neighbouring state. The new state is accepted if it is better than the current state. If this is not the case, a probability is calculated for accepting the new state even though it is a worse state, thus avoiding to get trapped in a local maximum. As the optimisation process proceeds, the length of the steps decline and the algorithm closes in on the global optimum. Since the algorithm makes very few assumptions regarding the function to be optimised, it is quite robust with respect to non-quadratic surfaces. The degree of robustness can be adjusted by the user. In fact, simulated annealing can be used as a local optimiser for difficult functions.
Both, hill climbing and simulated annealing support fuzzy-based spatial queries that search for a suitable location with a set of constraints.

Figure 8 shows an outline of the algorithm.

```
function SimulatedAnneal (function f) : configuration;
{search for the best configuration of f using simulated
annealing}

begin
   configuration currentConfig ← randomly choos a
                                 starting configuration;
   configuration bestConfig ← currentConfig;
   configuration neighbourConfig;
   temperature temp ← initialize temperature;
   repeat
      neighbourConfig ← randomly select a neighbour
                        configuration that has not
                        yet been selected;
      if f(neighbourConfig) ← is better then
                        f(currentConfig) then
      begin
        currentConfig ← neighbourConfig;
        if f(currentConfig) is better than f(bestConfig)
          then
          bestConfig ← currentConfig;
      end
      else
```

```
        perform currentConfig ← neighbourConfig with
                                some probability based
                                on temperature;
        cooling the temperature;
    until some stopping criteria is made;
    return bestConfig;
end;
```

Fig 8: Algorithm of Simulated Annealing.

CONCLUSION

This paper has presented a qualitative approach for spatial reasoning based on fuzzy logic. The advantages and disadvantages of qualitative and quantitative reasoning have been discussed and a few implementation strategies have been presented.
In this paper, we also outlined a system for fuzzy spatial reasoning. The system takes a picture (geographic map, satellite image, etc.) and analyses the colours in that picture by first transforming the colours into a perceptually uniform colour space and then associating fuzzy values with the colours. These fuzzy values are combined with the fuzzy values that are obtained from analysing spatial relation like *'extremely close to'*. As a result, we get a fuzzy classifications of all points in the pictures.

Clearly this paper is only the beginning. Many aspects of fuzzy-based qualitative spatial querying will need further development.

REFERENCES

[1] C. Howson, *Scientific reasoning: the Bayesian approach',* Chicago : Open Court, 1993

[2] E. Rich and K. Knight, *Artificial Intelligence,* MacGraw-Hill

[3] S. Gottwald, *Fuzzy sets and fuzzy logic: the foundations of application from a mathematical point of view,* Berlin, New York: Springer Verlag, 1993

[4] P. Cohen, *Heuristic Reasoning about Uncertainty: An Artificial Intelligence Approach,* London: Pitman

[5] J. Fox, Knowledge , Decision Making and Uncertainty in Artificial Intelligence and Statistics, ed. A. Gale. Reading, Ma: Addison - Wesley, pp 57-76

[6] J. Fox , Three Frameworks for Extending the Framework of Probability, *In Uncertainty and Artificial Intelligene*, ed. N. Kanal and J. F. Lemmer. New York, NY: Elsevier, pp 447-458

[7] H. Samet, *The Design and Analysis of Spatial Data Structures,* Addison-Wesley, 1990

[8] H.W. Guesgen and J.W. Histed, Towards qualitative spatial reasoning in geographic information systems. *In Proc. AAAI-96 Workshop on Spatial and Temporal Reasoning,* pp 39-46.

[9] B. D. Ripley, *Spatial Statistics.* Wiley, 1981

[10] N. Cressie, *Statistics for Spatial Data* , Wiley, 1991

[11] D. Poon, *Incorporate Fuzzy Approach in Supporting Qualitative Spatial Querying,* Thesis, University of Auckland, 1996