SPATIOTEMPORAL INTERPOLATION METHODS IN GIS

by

Lixin Li

A DISSERTATION

Presented to the Faculty of
The Graduate College at the University of Nebraska
In Partial Fulfillment of Requirements
For the Degree of Doctor of Philosophy

Major: Computer Science

Under the Supervision of Professor Peter Revesz

Lincoln, Nebraska

May, 2003

SPATIOTEMPORAL INTERPOLATION METHODS IN GIS

Lixin Li, Ph.D.

University of Nebraska, 2003

Advisor: Peter Revesz

In this dissertation, spatiotemporal interpolation of geographic data is considered. Two methods are discussed which are the reduction and the extension methods. The reduction method treats time as an independent dimension, whereas the extension method treats time equivalent to a spatial dimension. Both 2-D and 3-D shape functions are adopted, usually used in finite element methods, for the spatiotemporal interpolation of 2-D spatial & 1-D temporal and 3-D spatial & 1-D temporal data sets. Domains are divided into a finite number of sub-domains (such as triangles and tetrahedra) in which local shape functions are assumed. New 4-D shape functions that can be applied for each 4-D Delaunay Tessellation element are developed using the extension method for 3-D spatial & 1-D temporal problems. The visualization of shape function interpolation results is also explained and illustrated.

Using an actual real estate data set with house prices, we compare these methods with other spatiotemporal interpolation methods based on inverse distance weighting and kriging. We compare these methods with respect to interpolation accuracy, error-proneness to time aggregation, invariance to scaling on the coordinate axes, and the type of constraints used in the representation of the interpolated data. Our experimental results show that the extension method based on shape functions is the most accurate and the overall best spatiotemporal interpolation method.

Constraint databases provide a general approach to express and solve constraint problems. We show that spatiotemporal interpolation data can be represented in constraint databases efficiently and accurately. The advantage of constraint databases

is that many queries that could not be done in traditional GIS systems can now be easily expressed and evaluated in constraint database systems.

Finally, the constraint database system MLPQ (Management of Linear Programming Queries) is used to animate and query some spatiotemporal data examples. A translation algorithm between ArcGIS shape files and MLPQ input data files is also discussed.

ACKNOWLEDGEMENTS

I would like to thank my academic advisor, Professor Peter Revesz, for his expert advice. He has been a very motivating advisor and given me a lot of help and encouragement through my Ph.D. studies. He devoted a lot of time to fruitful discussions.

Thanks also to the other members of my Ph.D. dissertation committee, Professor Fayad, Professor Samal and Professor Narayanan. They have advised me during various stages of this work.

Special thanks to my old friends in China and new friends I met at UNL. They have been a source of moral support for me. In particular, I am grateful to my Master advisor Professor Huaxin Zeng, who helped me to come to UNL.

Finally, I would like to thank my husband Reiner for his understanding and love during the past three years. His loving support was in the end what made this dissertation possible. My parents receive my deepest gratitude and love for their dedication through my whole life. I regret very much that my father can not see me complete my Ph.D. degree since he passed away during my first year in the Ph.D. program. I dedicate my dissertation to my dear parents.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Geographic Information System (GIS) (Demers 2000, Longley, Goodchild, Maguire & Rhind 2001, Worboys 1995) applications increasingly require the use of spatiotemporal data, that is, data that combine both space and time (Langran 1992). For example, land-use change through time is a typical spatiotemporal data. Future GIS systems need to efficiently manage spatiotemporal databases (STDBs) containing such data. The study of the representation and the algorithmic methods to query and visualize spatiotemporal data is still a growing research area.

GIS applications often require *spatiotemporal interpolation* of an input data set. Spatiotemporal interpolation requires the estimation of the unknown values at unsampled location-time pairs with a satisfying level of accuracy. For example, suppose that we know the recording of temperatures at different weather stations at different instances of time. Then spatiotemporal interpolation would estimate the temperature at unsampled locations and times.

Spatial interpolation is already frequently used in GIS. There are many *spatial interpolation* algorithms for spatial (2-D or 3-D) data sets. Shepard (1968) discusses

in detail *inverse distance weighting*, Deutsch & Journel (1998) *kriging*, Goodman & O'Rourke (1997) *splines*, Zurflueh (1967) *trend surfaces*, and Harbaugh & Preston (1968) *Fourier series*. Lam (1983) gives a review and comparison of spatial interpolation methods.

There are surprisingly few papers that consider the topic of spatiotemporal interpolation in GIS. In fact, we could only find papers in spatiotemporal interpolation that estimate the motion of moving objects, which is a major concern in human vision but unrelated to GIS. One exception is Miller (1997), which utilizes kriging for spatiotemporal interpolation.

Most GIS researchers assume that spatiotemporal interpolation is reducible to a sequence of spatial interpolations. This reduction is convenient only if we sample the same locations at the same times. For example, this may be true for the above temperature data set if each weather station records temperature at same times. Then we can do a separate spatial interpolation for each time instance for which we have the temperatures at the weather stations.

However, irregular data sets are also quite common. For example, consider a data set that records the price of houses sold in a city. For each day of sale, this data set can give us only the exact price of a set of houses (those that are sold that day). This subset varies day by day. This is unlike the set of weather stations which are fixed. For such irregular data sets the above reduction method is unnatural to apply.

A spatiotemporal interpolation method for both regular and irregular GIS data is designed in this dissertation. After the comparison with other common methods, such as inverse distance weighting and kriging, the designed method, which utilizes shape functions, is shown to be the best in the selected examples.

Existing GIS software packages usually do not have the ability to visualize spatiotemporal data (Li & Revesz 2001). In this dissertation, the designed spatiotemporal interpolation result is visualized by using an experimental test data-set. Also, the

constraint database system MLPQ (Management of Linear Programming Queries) is used to animate and query some interesting spatiotemporal data.

1.2 Related Work

It is already becoming clear that we can distinguish three major types of GIS-oriented spatiotemporal databases. Spatiotemporal data can be represented by three basic primitives: points, regions, and constraints (Li & Revesz 2003 to appear). It should be emphasized that these three types of spatiotemporal databases are only alternative representations of the same data. Any one type of spatiotemporal database can be translated into another type as shown in Figure 1.1. In the figure, each edge represents a translation algorithm. For example, edge A represents the algorithm that converts a point-based spatiotemporal database into a constraint-based spatiotemporal database.

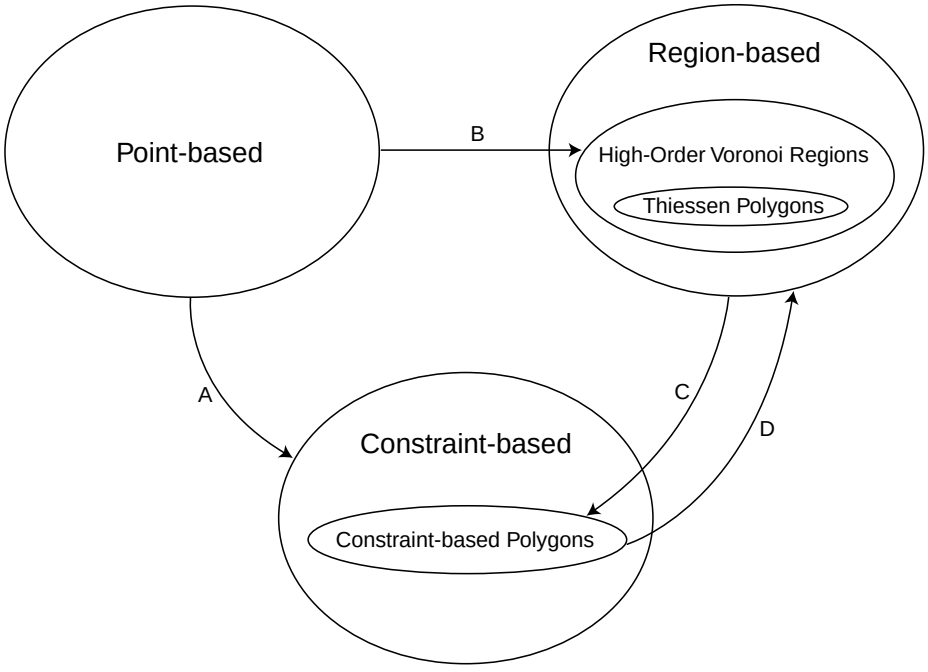

Figure 1.1: The relationship among the spatiotemporal databases.

1.2.1 Point-Based Spatiotemporal Databases

A point-based STDB consists of a set of point-based spatiotemporal relations. For 2-D space and 1-D time problems, a point-based spatiotemporal relation should have the schema of (x, y, t, w_1, w_2, ..., w_m). The attributes (x, y) specify point locations and t specifies a time instance. In GIS applications, (x, y) may be given as UTM (Universal Transverse Mercator) coordinates for easting and northing. The last m attributes w_i $(1 \leq i \leq m)$ record the other features at location (x, y) and time t.

Example 1.2.1 The point-based spatiotemporal relation *Drought_Point(x, y, year, SPI)* stores the average yearly SPI (Standardized Precipitation Index) values sampled by 48 major weather stations in Nebraska from year 1992 to 2002. SPI is one of the common and simple measures of drought which is based solely on the probability of precipitation for a given time period. Values of SPI range from 2.00 and above (extremely wet) to -2.00 and less (extremely dry) with near normal conditions ranging from 0.99 to -0.99. A drought event is defined when the SPI is continuously negative and reaches a value of -1.0 or less, and continues until the SPI becomes positive. We obtained the *Drought_Point* relation as shown in Table 1.1 from the Unified Climate Access Network (UCAN).

Drought_Point

x (easting)	y (northing)	year	SPI
-315515.56	2178768.67	1992	0.27
-315515.56	2178768.67	1993	-0.17
⋮	⋮	⋮	⋮
-315515.56	2178768.67	2002	0.19
-163932.36	2075263.16	1992	-0.19
⋮	⋮	⋮	⋮
-133759.02	1985122.32	2002	-0.22

Table 1.1: A point-based spatiotemporal relation.

However, from this example, we can see that the disadvantage of point-based STDBs is that the measured information, such as SPI, only exists at certain sampled locations and times. Therefore, we are not able to query the values at unsampled locations and times.

1.2.2 Region-Based Spatiotemporal Databases

A region-based STDB database has both spatial and temporal parts. The spatial part has schema *(region-id, boundary)*. The *region-id* is a unique identifier of each polygonal shaped region, and the *boundary* is the sequence of its corner vertices. The spatial part can be stored in an ArcGIS database (Johnston, Hoef, Krivoruchko & Lucas 2001). The temporal part has schema (*region-id*, t, w_1, w_2, ..., w_m), where t is the time attribute and each w_i represents some other characteristics of the region.

Nebraska_Corn_Space_Region

county	boundary
1	{ (-656160.3, 600676.8), (-652484.0, 643920.3), (-607691.1, 639747.6), (-608934.8, 615649.0), (-607875.6, 615485.8), (-610542.0, 576509.1), (-607662.7, 576138.5), (-611226.9, 537468.5), (-607807.7, 536762.1), (-608521.1, 527084.0), (-660885.4, 531441.2), (-661759.8,532153.1) }
⋮	⋮

Nebraska_Corn_Time_Region

county	year	practice	acres	yield	production
1	1947	irrigated	2700	49	132300
1	1947	non-irrigated	81670	18	1470060
1	1947	total	84370	19	1602360
⋮	⋮	⋮	⋮	⋮	⋮
1	2000	irrigated	141300	161	22749300
1	2000	non-irrigated	27900	73	2036700
1	2000	total	169200	146.5	24786000
⋮	⋮	⋮	⋮	⋮	⋮

Table 1.2: A region-based spatiotemporal database with separate spatial and temporal relations.

Example 1.2.2 A NASS (National Agricultural Statistics Service) region-based spatiotemporal data-base shows the yearly corn yield and production in each county of the state of Nebraska. The spatial part of the database is shown in the upper half of Table 1.2 which uses the vector representation of counties in Nebraska, while the temporal part is shown in the lower half of Table 1.2. Note that in the relation *Nebraska_Corn_Time_Region*, {*county*, *year*, *practice*} is the primary key, because it is the minimal set of attributes that functionally determine the other three attributes. The attributes used in these two tables are the following:

- *county*, the common attribute between the spatial and temporal relations, is the Federal Information Processing Standards (FIPS) id that is unique for each county in a state.
- *boundary* is a sequence of corner vertices on the boundary of a county. For example, the county with id 1 is a polygon that is represented by its 12 corner vertices.
- *year* is the time.
- *practice* is one of the following types: irrigated, non-irrigated, and total.
- *acres* is the number of acres that are harvested with total = irrigated + non-irrigated.
- $yield = bushels/acres$.
- $production = acres \times yield = total\ bushels$ (total = irrigated + non-irrigated).

Although region-based STDBs cover the continuous area and do not have the problem of missing information as in point-based STDBs, the information accuracy in region-based STDBs is reduced by assigning one uniform value to each region.

1.2.3 Voronoi Region-Based Spatiotemporal Databases

Sometimes, region-based STDBs can be derived from point-based STDBs. Voronoi diagrams are a special type of region-based STDBs generated from point-based STDBs. There are two types of Voronoi diagrams: (i) ordinary Voronoi diagrams, (ii) higher-order Voronoi diagrams. The ordinary Voronoi diagram of a finite set S of points in the plane is a partition of the plane so that each region of the partition is the locus of points which are closer to *one member* of S than to any other member. Higher-order Voronoi diagrams generalize ordinary Voronoi diagrams by dealing with k closest points. The higher-order Voronoi diagram of a finite set S of points in the plane is a partition of the plane into regions such that points in each region have the *same closest members* of S. As in an ordinary Voronoi diagram, each Voronoi region is still a convex polygon in a higher-order Voronoi diagram (Preparata & Shamos 1985). From the definition of higher-order Voronoi diagrams, it can be seen that the problem of finding the k closest neighbors for a given point in the whole domain, which is closely related to the IDW interpolation method with $N = k$, is equivalent to constructing k-th order Voronoi diagrams. Note that ordinary Voronoi diagrams can be viewed as special higher-order Voronoi diagrams when $k = 1$. Revesz & Li (2002*a*) discusses the details about how to transfer point-based STDBs to higher-order Voronoi diagrams and use them for IDW interpolation in constraint databases.

If the Voronoi diagram consists of ordinary Voronoi polygons (each Voronoi polygon is associated with one nearest neighbor), it is also called Thiessen polygons. In this case, each Voronoi region is assigned the value of the nearest neighbor inside the region. If the Voronoi diagram consists of kth-order ($k > 1$, each Voronoi polygon is associated with k nearest neighbors) Voronoi regions, then the average of the k nearest neighbors will be taken to assign each region. Thus, either ordinary or higher-order Voronoi regions are assigned a uniform value for all the points inside.

Example 1.2.3 Assume that in the point-based STDB *Drought_Point* in Example 1.2.1, the 48 weather stations have not changed their locations through the last 10 years and had SPI values measured every year. The spatial and temporal parts of the ordinary Voronoi region-based relation of *Drought_Point* are shown in Table 1.3; while the spatial and temporal parts of 2nd-order Voronoi region-based relation of *Drought_Point* are shown in Table 1.4.

Drought_Vo1_Space

$\{(x, y)\}$	boundary
{ (-133759.02, 1985122.32) }	{ (33051.50, 2031044.16), (-42164.24, 2073164.97), (-32238.03, 2124781.27), (33051.50, 2184847.63) }
⋮	⋮
{ (-133759.02, 1985122.32) }	{ (-355511.65, 2122185.61), (-416623.67, 2173220.63), (-405804.69, 2227674.50), (-286944.03, 2227674.50), (-265126.60, 2179386.65), (-273957.47, 2110648.67), (-275412.74, 2109314.41), (-286304.46, 2104106.55) }

Drought_Vo1_Time

$\{(x, y)\}$	year	SPI
{ (-315515.56, 2178768.67) }	1992	0.27
{ (-315515.56, 2178768.67) }	1993	-0.17
⋮	⋮	⋮
{ (-133759.02, 1985122.32) }	2001	-0.19
{ (-133759.02, 1985122.32) }	2002	-0.22

Table 1.3: An ordinary Voronoi region-based database.

1.2.4 Constraint-Based Spatiotemporal Databases

Constraint databases (Kanellakis, Kuper & Revesz 1995, Revesz 2002) provide a natural representation for spatiotemporal objects when their trajectory can be described as simple mathematical functions. Constraint-based STDBs can represent both point-based and region-based spatiotemporal objects, where each attribute is associated

Drought_Vo2_Space

$\{(x_1, y_1), (x_2, y_2)\}$	boundary
{ (-9820.18, 1929867.40) , (-42164.88, 1915035.54) }	{ (-17122.48, 2203344.58), (3014.51, 2227674.50) (33051.50, 2227674.50), (33051.5, 2140801.51) }
⋮	⋮
{ (-507929.66, 2216998.17), (-247864.81, 1946777.44) }	{ (-274044.43, 2109969.45), (-273957.42, 2110648.46), (-245744.29, 2136492.60), (-205869.74, 2142110.52), (-198942.71, 2115609.44), (-227141.62, 2099273.50) }

Drought_Vo2_Time

$\{(x_1, y_1), (x_2, y_2)\}$	year	avgSPI
{ (-9820.18, 1929867.4), (-42164.88, 1915035.54) }	1992	-0.47
{ (-9820.18, 1929867.4), (-42164.88, 1915035.54) }	1993	0.71
⋮	⋮	⋮
{ (-507929.66, 2216998.17), (-247864.81, 1946777.44) }	2001	0.65
{ (-507929.66, 2216998.17), (-247864.81, 1946777.44) }	2002	-0.03

Table 1.4: A 2nd-order Voronoi region-based database.

with an attribute variable. For example, the spatial attributes can be associated with x and y variables for representing point-based spatiotemporal objects or with a *region_id* variable for region-based spatiotemporal objects, while the temporal attributes can be associated with *year* or other time variables. A constraint-based relation is a finite set of constraint tuples. The value of the attributes in a relation is specified implicitly using (arithmetic) constraints such that each constraint tuple is a conjunction of constraints using the same set of attribute variables.

Example 1.2.4 The region-based NASS corn data in Example 1.2.2 can be represented by spatial and temporal parts in the constraint-based STDB shown in Table 1.5. In the spatial part of Table 1.5, each county polygon can be either convex or concave, we break each county polygon into a set of adjacent triangles for the convenience of implementation, where each triangle can be represented by a conjunction of three linear arithmetic constraints. For example, the county with FIPS code 1 consists of three triangles.

Nebraska_Corn_Space_Constraint

county	easting	northing	
1	x	y	$81.15x - y \leq -15827909.86$, $-0.02x - y \geq -356911.63$, $-1.11x - y \leq -108714.89$
1	x	y	$7.69x - y \geq -2116031.10$, $33.21x - y \geq -7938392.36$, $29.71x - y <= -7135823.80$
1	x	y	$-0.03x - y \leq -315048.67$, $-1.11x - y \geq -108714.89$, $29.71x - y >= -7135823.80$
⋮	⋮	⋮	⋮

Nebraska_Corn_Time_Constraint

county	year	practice	acres	yield	production	
1	t	irrigated	a	y'	p	$1947 \leq t$, $t \leq 2000$, $a = 3453t - 6729678$, $y' = 2t - 3890$, $p = 540016t - 1054525400$
1	t	non-irrigated	a	y'	p	$1947 \leq t$, $t \leq 2000$, $a = -1000t + 1995747$, $y' = 2t - 3200$, $p = 6839t - 12608086$
1	t	total	a	y'	p	$1947 \leq t$, $t \leq 2000$, $a = 2453t - 4733931$, $y' = 3t - 5069$, $p = 546854t - 1067133500$
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 1.5: A constraint-based spatiotemporal database with separate spatial and temporal relations.

The temporal part of Table 1.5 compresses the region-based *Nebraska_Corn_Time_Region* relation in Example 1.2.2 by storing the linear regression functions of *acres*, *yield* and *production* according to *year*. The linear regression functions show the trend of these attributes during the past years. So they are useful to predict the future events. For example, what will the the yield of irrigated corn in county 1 approximately be in 2010? The entire *Nebraska_Corn_Time_Constraint* relation consists of only 279 constraint tuples, while *Nebraska_Corn_Time_Region* contains 14,788

original tuples. Although the accuracy in this example is not very high because of the approximation by only one linear regression functions, we can improve it by inserting more segments of lines. Revesz, Chen & Ouyang (2001) discusses the detail of how to use piece-wise linear functions to compress data in constraint databases. We also could use non-linear approximation functions to increase the accuracy. Note that because of the tradeoff between the accuracy and storage, the above improvements will increase the number of tuples in the temporal relation (i.e. *Nebraska_Corn_Time_Constraint*) of Table 1.5.

In summary, users can query the measured information at any location and time in constraint-based STDBs. This is similar to region-based STDBs: there is no missing information. Moreover, constraint-based STDBs are more sophisticated than region-based STDBs. In region-based STDBs, all the points in a region can only be given one uniform value. However, in constraint-based STDBs, the points in a region can be assigned with different values. If we could find appropriate functions (constraints) of x, y and t to interpolate the values inside each region and store them in the constraint-based STDBs, we can get a good result.

1.2.5 The Relationship among Spatiotemporal Databases

Now we are ready to explain the edges in the Figure 1.1 as follows:

(A) Point-based STDBs can be converted to constraint-based STDBs by certain interpolation methods, such as shape functions (Zienkiewics & Taylor 2000), spline functions (Goodman & O'Rourke 1997), IDW (Shepard 1968), and Kriging (Deutsch & Journel 1998). The details about using shape functions to convert point-based STDBs to constraint-based STDBs can be found in (Revesz & Li 2002*b*, Li & Revesz 2003 in press); and the translation details using IDW can be found in (Revesz & Li 2002*a*, Revesz & Li 2003 to appear). The comparison

of using shape functions and IDW to represent point-based data in constraint-based STDBs is analyzed in (Li & Revesz 2002) based on a specific set of data.

(B) Point-based STDBs can be used to generate region-based STDBs, including Voronoi region-based STDBs, by treating a region as a unit and assigning a uniform value for each region. A special case of region-based STDBs is the Worboys relations (Worboys 1995) where each region is divided into a set of triangles.

(C) Region-based STDBs can be converted to constraint-based STDBs by using linear arithmetic constraints to represent region boundaries and compressing the temporal part by some types of functions in time, such as linear regression functions in Example 1.2.4.

(D) Constraint-based STDBs (Revesz 2002) can be converted to region-based STDBs by simply assigning a uniform value to each region, such as the average of all the nearest neighbors.

1.3 Overview of Contributions of This Dissertation

The main contributions of this dissertation are the following.

- We design shape function-based 2-D space & 1-D time and 3-D space & 1-D time spatiotemporal interpolation methods for GIS data.
- We compare the shape function-based spatiotemporal interpolation methods with other spatiotemporal interpolation methods based on inverse distance weighting and kriging. Based on an actual real estate data set with house prices,

the experimental results show that one method based on shape functions is the most accurate and the overall best spatiotemporal interpolation method.

- We represent the spatiotemporal interpolation results in constraint databases, which have the advantage of accurate representation, efficient storage, and powerful queries.

- We use the constraint database system MLPQ (Management of Linear Programming Queries) to animate and query some spatiotemporal data examples.

The detailed contributions of this dissertation according to each chapter can be described as follows.

1. Chapter 2, based on (Li & Revesz 2002, Li & Revesz 2003 in press), starts by describing two general methods for spatiotemporal interpolations. The *reduction method* treats time independently from the spatial dimensions. The *extension method* treats time as equivalent to a spatial dimension.

   (a) In Section 2.1, we design several shape function-based spatiotemporal interpolation methods that are suitable for both regular and irregular GIS data.

      i. For 2-D space and 1-D time spatiotemporal interpolation problems, the reduction approach using the combination of 2-D shape functions for space and 1-D shape functions for time is illustrated in Section 2.1.1.1, and the extension approach using 3-D shape functions where the first two dimensions are for space and the third dimension is for time is described in Section 2.1.1.2.

      ii. For 3-D space and 1-D time spatiotemporal interpolation problems, the reduction method using the combination of 3-D shape functions for space and 1-D shape functions for time is illustrated in Section 2.1.2.1.

For the extension method, based on dividing the 4-D domain by a 4-D Delaunay Tesselation, new 4-D shape functions, which can be used for each 4-D Delaunay Tesselation element, are developed (see Section 2.1.2.2).

(b) In Section 2.2, we discuss spatiotemporal interpolation methods based on inverse distance weighting.

(c) In Section 2.3, we discuss spatiotemporal interpolation methods based on kriging.

2. Chapter 3 is based on (Revesz & Li 2002*b*, Revesz & Li 2002*a*, Revesz & Li 2003 to appear). We discuss constraint databases according to the efficient and accurate representation of both input data and interpolation result, as well as the powerful querying.

(a) In Section 3.1.1, we design an example to show how to represent input data in constraint databases.

(b) We illustrate the representation of 2-D shape function interpolation of the same example in constraint databases in Section 3.1.2.

(c) We illustrate the representation of inverse distance weighting interpolation of the same example in constraint databases in Section 3.1.3.

(d) We give some Datalog query examples in Section 3.2.

3. Chapter 4 is based on (Li & Revesz 2002, Li & Revesz 2003 in press). Based on a set of actual real estate data, we present a comparison between shape function-based spatiotemporal interpolation methods with inverse distance weighting and kriging. We show that the extension method with shape functions is the most accurate spatiotemporal interpolation method as measured by mean absolute

error (MAE) and root mean square error (RMSE). It is also the only method which can be represented using linear constraints.

The extension method, which treats time as another dimension, has a potential problem, namely that there is no easy way to compare one temporal unit with one spatial unit. Depending on the unit measure, we may get a different value for the estimated results. Are there spatiotemporal interpolations that are invariant with respect to the choice of units in the spatial and temporal axes? We show that only shape functions-based spatiotemporal interpolation is invariant.

Also, in the real estate data instead of recording the precise date of sale of houses we may have only records of monthly, bimonthly or even yearly sales, that is, all the houses sold in that time interval are listed together. We show experimentally that this time aggregation has a serious negative effect on the accuracy of the reduction method.

Finally, in Section 4.6, we illustrate visualization of the shape function interpolation result. In Section 4.7, with the height of each house also recorded, we give an example of using 4-D shape functions by considering an extension of the real estate data.

4. In Chapter 5, which is based on (Li & Revesz 2003 to appear, Li 2001, Li & Revesz 2001), we use the constraint database system MLPQ (Management of Linear Programming Queries) to animate (see Section 5.3) and query (see Section 5.4) some spatiotemporal data examples. A translation algorithm between ArcGIS shape files and MLPQ input data files is briefly discussed in Section 5.2.

Chapter 2

Spatiotemporal Interpolation Methods

There are two general methods for spatiotemporal interpolations: reduction and extension (Li & Revesz 2002). The *reduction method* treats time independently from the spatial dimensions. The *extension method* treats time as equivalent to a spatial dimension. These methods can be described briefly as follows:

Reduction This method reduces the spatiotemporal interpolation problem to a regular spatial interpolation case. First, at each sample point, we interpolate (using any 1-D interpolation in time) separately all the measured value over time. Then using the time functions for the sample point values in the spatial interpolation functions, we can get spatiotemporal interpolation results.

Extension This method deals time as another regular dimension in space and therefore extends the spatiotemporal interpolation problem into a higher-dimensional spatial interpolation problem.

In this chapter, we consider 2-D space & 1-D time and 3-D space & 1-D time spatiotemporal interpolation problems. The main contribution of this chapter is to

apply shape functions to spatiotemporal interpolation, which will be described in Section 2.1 as follows:

- We address the 2-D space & 1-D time problems using shape functions in Section 2.1.1. We illustrate the reduction approach using a combination of 2-D shape functions for space and 1-D shape functions for time, and the extension approach using 3-D shape functions where the first two dimensions are for space and the third dimension is for time.

- We consider the 3-D space & 1-D time problems using shape functions in Section 2.1.2. For the reduction method we use the combination of 3-D shape functions for space and 1-D shape functions for time. For the extension method we first divide the 4-D domain by a 4-D Delaunay Tesselation. Then we develop new 4-D shape functions that can be applied for each 4-D Delaunay Tesselation element.

Besides shape functions, other spatial interpolation methods may also have reduction and extension approaches for spatiotemporal problems. Inverse Distance Weighting (IDW) and Kriging are common spatial interpolation methods. Spatiotemporal interpolation methods based on IDW and Kriging will be discussed in Section 2.2 and 2.3, respectively. The comparison of shape function with IDW and Kriging based spatiotemporal interpolation methods will be analyzed later in this dissertation based on several sets of data.

2.1 Shape Functions

Shape functions, which can be viewed as a spatial interpolation method, are popular in engineering applications, for example, in finite element algorithms (Zienkiewics & Taylor 2000, Buchanan 1995). There are various types of 2-D and 3-D shape

functions. We are interested in 2-D shape functions for triangles and 3-D shape functions for tetrahedra, both of which are linear approximation methods. In this section, we discuss new approaches to spatiotemporal interpolation by shape function based reduction and extension methods for 2-D space & 1-D time and 3-D space & 1-D time problems.

2.1.1 2-D Space and 1-D Time Problems

2.1.1.1 Reduction Approach: ST Product Method

Since this approach is obtained by multiplying two interpolation functions in space and time, we call this method ST (space time) product method. This approach for 2-D space and 1-D time problems can be described by two steps: 2-D spatial interpolation by shape functions for triangles and approximation in space and time. Although there exists similar shape function based ST product methods such as the temperature distribution function in time-dependent heat conduction problems (Huebner 1975), we discuss in this section an ST product method which combines 2-D shape function in space and 1-D shape function in time.

2-D Shape Functions for Triangles First of all, when dealing with complex 2-D geometric domains, it is convenient to divide the total domain into a finite number of simple sub-domains which can have triangular or quadrilateral shapes. Mesh generation using triangular or quadrilateral domains is important in finite element discretization of engineering problems. For the generation of triangular meshes, quite successful algorithms have been developed. A popular method for the generation of triangular meshes is the "Delaunay Triangulation" (Goodman & O'Rourke 1997, Preparata & Shamos 1985, Shewchuk 1996). Delaunay triangulation is related to the construction of the so called "Voronoi diagram", which is related to "Convex Hull" problems. We embedded in our system the Delaunay triangulation algorithm available from the

public website `www.geom.umn.edu/software/~qhull`.

A linear approximation function for a triangular area can be written in terms of three shape functions N_1, N_2, N_3, and the corner values w_1, w_2, w_3. In Figure 2.1, two triangular finite elements, I and II, are combined to cover the whole domain considered.

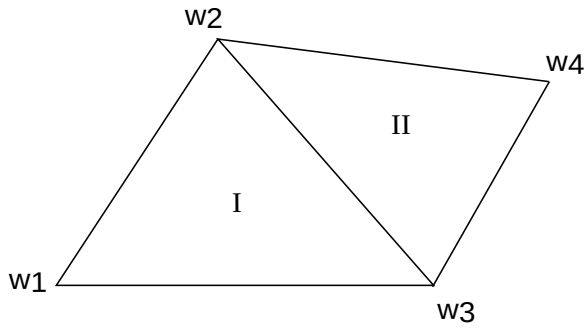

Figure 2.1: Linear interpolation in 2-D space for triangular elements.

In this example, the function in the whole domain is interpolated using four discrete values w_1, w_2, w_3, and w_4 at four locations. A particular feature of the chosen approximation method is that the function values inside the sub-domain I can be obtained by using only the three corner values w_1, w_2 and w_3, whereas all function values for the sub-domain II can be constructed using the corner values w_2, w_3, and w_4. The linear interpolation function for the sub-domain of element I can be written as

$$w(x,y) = N_1(x,y)w_1 + N_2(x,y)w_2 + N_3(x,y)w_3 = [N_1 \ N_2 \ N_3] \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} \tag{2.1}$$

where N_1, N_2 and N_3 are the following shape functions:

$$\begin{aligned} N_1(x,y) &= \frac{[(x_2y_3 - x_3y_2) + x(y_2 - y_3) + y(x_3 - x_2)]}{2\mathcal{A}} \\ N_2(x,y) &= \frac{[(x_3y_1 - x_1y_3) + x(y_3 - y_1) + y(x_1 - x_3)]}{2\mathcal{A}} \\ N_3(x,y) &= \frac{[(x_1y_2 - x_2y_1) + x(y_1 - y_2) + y(x_2 - x_1)]}{2\mathcal{A}} \ . \end{aligned} \tag{2.2}$$

The area $\mathcal{A}$ used for the shape functions in equation (2.2) can be computed using the corner coordinates (x_i, y_i) $(i = 1, 2, 3)$ in the determinant of a 3×3 matrix according to

$$\mathcal{A} = \frac{1}{2}\ det \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix} \ . \tag{2.3}$$

For the sub-domain II, the local approximation takes a similar form as the expression (2.1): we just have to replace the corner values w_1, w_2 and w_3 with the new values w_2, w_3 and w_4.

It should be noted that for every sub-domain, a local approximation function similar to expression (2.1) is used. Each local approximation function is constrained to the local triangular sub-domain. For example, the function w of equation (2.1) is valid only for sub-domain I.

Alternatively, considering only sub-domain I, the 2-D shape function (2.2) can also be expressed as follows (Revesz & Li 2002*b*):

$$N_1(x,y) = \frac{\mathcal{A}_1}{\mathcal{A}}, \quad N_2(x,y) = \frac{\mathcal{A}_2}{\mathcal{A}}, \quad N_3(x,y) = \frac{\mathcal{A}_3}{\mathcal{A}} \ . \tag{2.4}$$

$\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$ are the three sub-triangle areas of sub-domain I as shown in Figure 2.2, and $\mathcal{A}$ is the area of the outside triangle $w_1w_2w_3$ which can be computed by equation (2.3). All the $\mathcal{A}_i$'s $(1 \leq i \leq 3)$ can also be computed in the similar way as equation (2.3) by using the appropriate coordinate values.

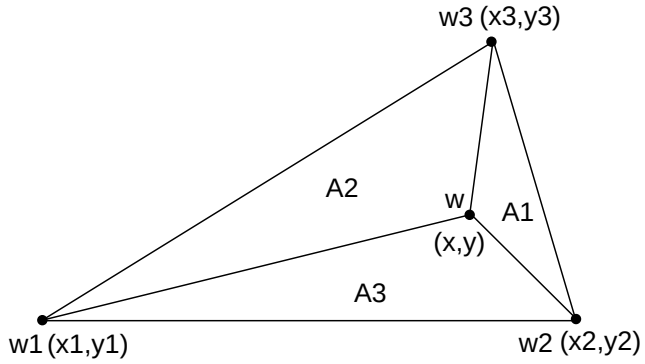

Figure 2.2: Computing 2-D shape functions by area divisions.

Example 2.1.1 Let $w_1 = 1$, $w_2 = 2$, and $w_3 = 3$ be the values measured at locations $(x_1, y_1) = (0, 0)$, $(x_2, y_2) = (10, 0)$, and $(x_3, y_3) = (10, 5)$, respectively. Using Equations (2.1) and (2.4) we can interpolate the unknown value w at location $(x, y) = (8, 2)$ as follows:

$$\begin{aligned} w &= \frac{\mathcal{A}_1}{\mathcal{A}} w_1 + \frac{\mathcal{A}_2}{\mathcal{A}} w_2 + \frac{\mathcal{A}_3}{\mathcal{A}} w_3 \\ &= \frac{1}{5} \times 1 + \frac{2}{5} \times 2 + \frac{2}{5} \times 3 \\ &= 2.2 \end{aligned}$$

Approximation in Space and Time Since in the reduction approach, we model time independently, approximation in space and time can be implemented by combining a time shape function with the space approximation function (2.1).

Assume the value at the node i at time t_1 is w_{i1}, and at time t_2 the value is w_{i2}. The value at the node i at any time between t_1 and t_2 can be approximated using a 1-D time shape function in the following way:

$$w_i(t) = \frac{t_2 - t}{t_2 - t_1} w_{i1} + \frac{t - t_1}{t_2 - t_1} w_{i2} \ . \tag{2.5}$$

Using the example shown in Figure 2.1 and utilizing formulas (2.1) and (2.5), the approximation function for any point constraint to the sub-domain I at any time between t_1 and t_2 can be expressed as follows (Li & Revesz 2002):

$$\begin{aligned} w(x,y,t) \ = \ & N_1(x,y)\left[\frac{t_2 - t}{t_2 - t_1} w_{11} + \frac{t - t_1}{t_2 - t_1} w_{12}\right] \\ & + N_2(x,y)\left[\frac{t_2 - t}{t_2 - t_1} w_{21} + \frac{t - t_1}{t_2 - t_1} w_{22}\right] \\ & + N_3(x,y)\left[\frac{t_2 - t}{t_2 - t_1} w_{31} + \frac{t - t_1}{t_2 - t_1} w_{32}\right] \\ = \ & \frac{t_2 - t}{t_2 - t_1}\left[N_1(x,y)w_{11} + N_2(x,y)w_{21} + N_3(x,y)w_{31}\right] \\ & + \frac{t - t_1}{t_2 - t_1}\left[N_1(x,y)w_{12} + N_2(x,y)w_{22} + N_3(x,y)w_{32}\right] \ . \end{aligned} \tag{2.6}$$

Since the space shape functions (N_1, N_2 and N_3) and the time shape functions (2.5) are linear, the spatiotemporal approximation function (2.6) is not linear, but quadratic.

2.1.1.2 Extension Approach: 3-D Method

This method treats *time* as a regular third dimension. Since it extends 2-D problems to 3-D problems, we call this method 3-D method. This method is very similar to the linear approximation in space of the reduction approach for 3-D problem (Section 2.1.2.1), which is based on tetrahedral meshes. The only modification is to substitute the z variable by the time variable t (see equations (2.7)–(2.10)).

2.1.2 3-D Space and 1-D Time Problems

2.1.2.1 Reduction Approach: ST Product Method

This shape function based reduction spatiotemporal interpolation in 3-D space and 1-D time will be described in the following two steps: 3-D spatial interpolation by shape functions for tetrahedra and approximation in space and time.

3-D Shape Functions for Tetrahedra Three-dimensional domains can be divided into finite number of simple sub-domains. For example, we can use tetrahedral or hexahedral sub-domains. Tetrahedral meshing is of particular interest. With a large number of tetrahedral elements, we can also approximate complicated 3-D objects. Figure 2.3 shows a tetrahedral mesh of a 3-D object. This object has a cutout (one quarter of a cylinder) behind the boundary defined by the points $ABCD$.

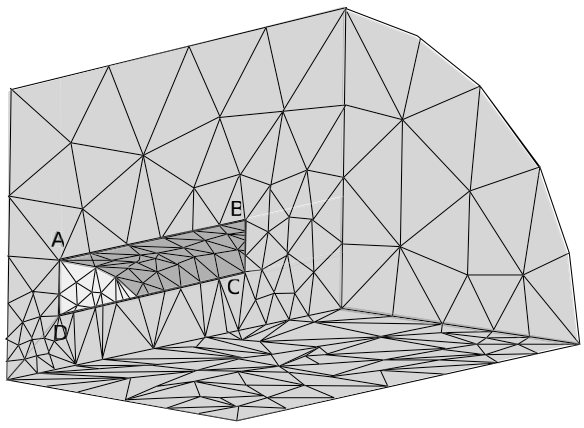

Figure 2.3: A tetrahedral mesh.

It is a difficult topic to automatically generate tetrahedral meshes. Fortunately, there exist several methods to generate automatic tetrahedral meshes, such as the ordinary 3-D Delaunay tetrahedrilization and some tetrahedral mesh improvement methods to avoid poorly-shaped tetrahedra. For example, the tetrahedral mesh gen-

eration by Delaunay refinement (Shewchuk 1998) and tetrahedral mesh improvement using swapping and smoothing (Freitag & Gooch 1997).

Similarly to the linear approximation function for the 2-D problem solved in Section 2.1.1, a linear approximation function for a 3-D tetrahedral element can be written in terms of four shape functions N_1, N_2, N_3, N_4 and the corner values w_1, w_2, w_3, w_4. In Figure 2.4, two tetrahedral elements, I and II, cover the whole domain considered.

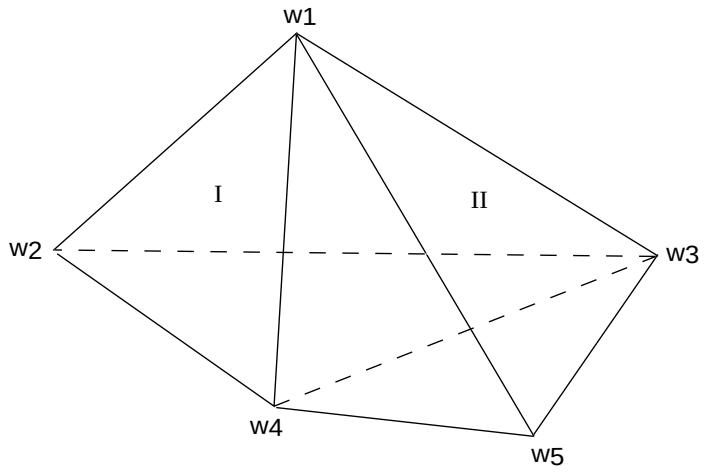

Figure 2.4: Linear interpolation in 3-D space for tetrahedral elements.

In this example, the function in the whole domain is interpolated using five discrete values w_1, w_2, w_3, w_4, and w_5 at five locations in space. To obtain the function values inside the tetrahedral sub-domain I, we can use the four corner values w_1, w_2, w_3 and w_4. Similarly, all function values for the sub-domain II can be constructed using the corner values w_1, w_3, w_4 and w_5. The linear interpolation function for the sub-domain

of element I can be written as

$$\begin{aligned} w(x,y,z) &= N_1(x,y,z)w_1 + N_2(x,y,z)w_2 + N_3(x,y,z)w_3 + N_4(x,y,z)w_4 \\ &= [N_1\ N_2\ N_3\ N_4] \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} \end{aligned} \tag{2.7}$$

where N_1, N_2 N_3 and N_4 are the following shape functions:

$$\begin{aligned} N_1(x,y,z) = \frac{a_1 + b_1x + c_1y + d_1z}{6\mathcal{V}}\ , N_2(x,y,z) = \frac{a_2 + b_2x + c_2y + d_2z}{6\mathcal{V}}\ , \\ N_3(x,y,z) = \frac{a_3 + b_3x + c_3y + d_3z}{6\mathcal{V}}\ , N_4(x,y,z) = \frac{a_4 + b_4x + c_4y + d_4z}{6\mathcal{V}}\ . \end{aligned} \tag{2.8}$$

The volume $\mathcal{V}$ of the tetrahedron used for the shape functions in (2.8) can be computed using the corner coordinates (x_i, y_i, z_i) $(i = 1, 2, 3, 4)$ in the determinant of a 4×4 matrix according to

$$\mathcal{V} = \frac{1}{6}\ det \begin{bmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{bmatrix}\ . \tag{2.9}$$

By expanding the other relevant determinants into their cofactors, we have

$$a_1 = det \begin{bmatrix} x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \\ x_4 & y_4 & z_4 \end{bmatrix} \qquad b_1 = -det \begin{bmatrix} 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \\ 1 & y_4 & z_4 \end{bmatrix}$$

$$c_1 = -det \begin{bmatrix} x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \\ x_4 & 1 & z_4 \end{bmatrix} \qquad d_1 = -det \begin{bmatrix} x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \\ x_4 & y_4 & 1 \end{bmatrix}$$

with the other constants defined by cyclic interchange of the subscripts in the order 4, 1, 2, 3 (Zienkiewics & Taylor 1989).

Alternatively, considering only the tetrahedral sub-domain I, the 3-D shape function (2.8) can also be expressed as follows:

$$N_1(x,y,z) = \frac{\mathcal{V}_1}{\mathcal{V}}, N_2(x,y,z) = \frac{\mathcal{V}_2}{\mathcal{V}}, N_3(x,y,z) = \frac{\mathcal{V}_3}{\mathcal{V}}, N_4(x,y,z) = \frac{\mathcal{V}_4}{\mathcal{V}} \qquad (2.10)$$

$\mathcal{V}_1$, $\mathcal{V}_2$, $\mathcal{V}_3$ and $\mathcal{V}_4$ are the volumes of the four sub-tetrahedra $ww_2w_3w_4$, $w_1ww_3w_4$, $w_1w_2ww_4$, and $w_1w_2w_3w$, respectively, as shown in Figure 2.5; and $\mathcal{V}$ is the volume of the outside tetrahedron $w_1w_2w_3w_4$ which can be computed by equation (2.9). All the $\mathcal{V}_i$'s $(1 \leq i \leq 4)$ can also be computed in the similar way as equation (2.9) by using the appropriate coordinate values.

Example 2.1.2 Suppose $(x_1, y_1, z_1) = (8, 5, 10)$, $(x_2, y_2, z_2) = (0, 6, 0)$, $(x_3, y_3, z_3) = (20, 10, 0)$, and $(x_4, y_4, z_4) = (10, 0, 0)$ are the four sampled corner vertices of a tetrahedron which encloses the point $(x, y, z) = (10, 4, 4)$, as shown in Figure 2.5. Let $w_1 = 1$, $w_2 = 2$, $w_3 = 3$, and $w_4 = 4$ be the measured values at the corner vertices. Using Equations (2.7) and (2.10), we can interpolate the unknown value w at location

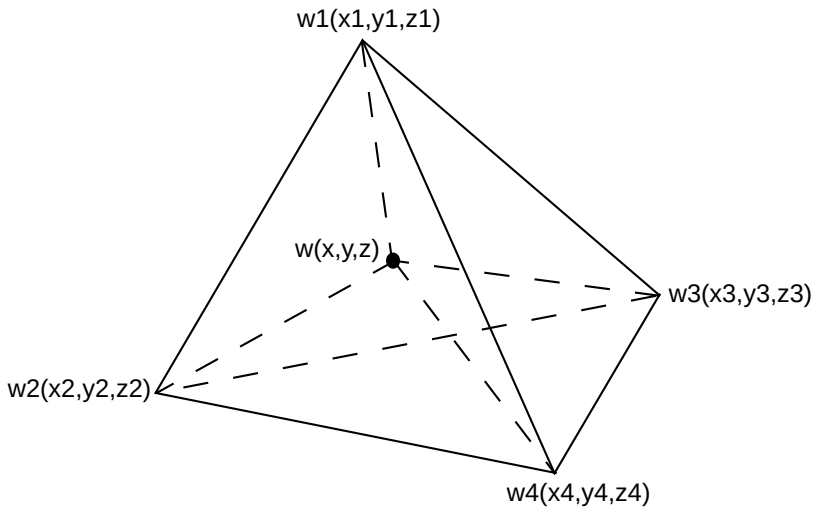

Figure 2.5: Computing 3-D shape functions by volume divisions.

(x, y, t) by the tetrahedral method as:

$$\begin{aligned} w &= \frac{\mathcal{V}_1}{\mathcal{V}} w_1 + \frac{\mathcal{V}_2}{\mathcal{V}} w_2 + \frac{\mathcal{V}_3}{\mathcal{V}} w_3 + \frac{\mathcal{V}_4}{\mathcal{V}} w_4 \\ &= \frac{640}{1600} \times 1 + \frac{120}{1600} \times 2 + \frac{248}{1600} \times 3 + \frac{592}{1600} \times 4 \\ &= 2.495 \end{aligned}$$

Approximation in Space and Time Similarly to the reduction approach to 2-D space & 1-D time problems, 3-D space & 1-D time interpolation can be implemented by combining the time shape function (2.5) with the space approximation function (2.7). Using the example shown in Figure 2.4, the linear approximation function for any point constraint to the sub-domain I at any time between t_1 and t_2 can be

expressed as follows:

$$
\begin{aligned}
w(x,y,z,t) \;=\;& N_1(x,y,z)\left[\frac{t_2-t}{t_2-t_1}w_{11}+\frac{t-t_1}{t_2-t_1}w_{12}\right] \\
&+ N_2(x,y,z)\left[\frac{t_2-t}{t_2-t_1}w_{21}+\frac{t-t_1}{t_2-t_1}w_{22}\right] \\
&+ N_3(x,y,z)\left[\frac{t_2-t}{t_2-t_1}w_{31}+\frac{t-t_1}{t_2-t_1}w_{32}\right] \\
&+ N_4(x,y,z)\left[\frac{t_2-t}{t_2-t_1}w_{41}+\frac{t-t_1}{t_2-t_1}w_{42}\right] \qquad (2.11)
\end{aligned}
$$

$$
\begin{aligned}
=\;& \frac{t_2-t}{t_2-t_1}[N_1(x,y,z)w_{11}+N_2(x,y,z)w_{21}+ \\
&\qquad N_3(x,y,z)w_{31}+N_4(x,y,z)w_{41}] \\
&+ \frac{t-t_1}{t_2-t_1}[N_1(x,y,z)w_{12}+N_2(x,y,z)w_{22}+ \\
&\qquad N_3(x,y,z)w_{32}+N_4(x,y,z)w_{42}]\ .
\end{aligned}
$$

Since the space shape functions (N_1, N_2, N_3 and N_4) and the time shape functions (2.5) are linear, the spatiotemporal approximation function (2.11) is quadratic.

2.1.2.2 Extension Approach: 4-D Method

This method treats *time* as a regular fourth dimension. We develop new linear 4-D shape functions to solve this problem. In the engineering area, the highest number of dimensions of shape functions is three because there are no higher dimensional real objects. By developing 4-D shape functions, we will be able to interpolate an unsampled value at location *(x,y,z)* and time *t*. For example, the location can be house locations, including the elevation *z*. In a flat city the elevation is not important. In a hilly city the elevation may be important (for example, nice ocean view may be preferred).

Our linear 4-D shape functions are based on 4-D Delaunay tesselation. The De-

launay tesselation in 4-D space is a special case for n-D space Delaunay tesselation when $n = 4$. The n-D Delaunay tesselation is defined as a space-filling aggregate of n-simplices (Watson 1981). Each Delaunay n-simplex can be represented by an (n+1)-tuple of indices to the data points. We can use Matlab to computate the n-D Delaunay tesselation by function *Delaunayn*. $\mathbf{T} = delaunayn(\mathbf{X})$ computes a set of n-simplices such that no data points of $\mathbf{X}$ are contained in any n-D hyperspheres of the n-simplices. The set of n-simplices forms the n-D Delaunay tessellation. $\mathbf{X}$ is an $m \times n$ array representing m points in n-D space. $\mathbf{T}$ is an $s \times (n+1)$ array where s is the number of n-simplices after the n-D Delaunay tesselation. Each row of $\mathbf{T}$ contains the indices into $\mathbf{X}$ of the vertices of the corresponding n-simplex. In order to solve 4-D Delaunay tesselation in Matlab, we just need to give the *Delaunayn* function proper $\mathbf{X}$ array with size $m \times 4$. An example of 4-D Delaunay tesselation by Matlab is give below.

Example 2.1.3 Assume $\mathbf{X}$ is an array that contains seven 4-D points ($m = 7$, $n = 4$) as follows:

$$\mathbf{X} = \begin{bmatrix} 115 & 1525 & 500 & 16 \\ 890 & 1880 & 750 & 36 \\ 1120 & 1650 & 300 & 22 \\ 730 & 1660 & 600 & 13 \\ 725 & 1320 & 780 & 42 \\ 880 & 1140 & 678 & 69 \\ 1610 & 2570 & 890 & 95 \end{bmatrix}$$

Then $\mathbf{T} = delaunayn(\mathbf{X})$ will return the following set of nine 5-simplices ($s = 9$):

$$\mathbf{T} = \begin{bmatrix} 2 & 3 & 6 & 7 & 1 \\ 2 & 4 & 3 & 7 & 1 \\ 2 & 4 & 3 & 6 & 1 \\ 5 & 4 & 3 & 6 & 1 \\ 5 & 2 & 6 & 7 & 1 \\ 5 & 2 & 4 & 6 & 1 \\ 5 & 2 & 3 & 6 & 7 \\ 5 & 2 & 4 & 3 & 7 \\ 5 & 2 & 4 & 3 & 6 \end{bmatrix}$$

We develop new 4-D shape functions using two different approaches. Although they yield mathematically equivalent results, the first approach yields very long symbolic expressions whereas the second approach gives simple expressions.

Approach I Since we want to develop linear 4-D shape functions to do the 4-D approximation, we can assume that within each element we have some constants a, b, c, d and e such that:

$$w(x, y, z, t) = a + bx + cy + dz + et \ .$$

Let $\mathbf{\Phi}(x, y, z, t) = [1, x, y, z, t]$ and $\mathbf{f}^{\mathbf{T}} = [a, b, c, d, e]$, we have

$$w(x, y, z, t) = \mathbf{\Phi}(x, y, z, t)\, \mathbf{f} \ . \tag{2.12}$$

We use the five known nodal values (w_i's, $1 \leq i \leq 5$) to calculate $\mathbf{f}$ as follows:

$$\begin{aligned}
\mathbf{\Phi}(x_1, y_1, z_1, t_1)\ \mathbf{f} &= w_1 \\
\mathbf{\Phi}(x_2, y_2, z_2, t_2)\ \mathbf{f} &= w_2 \\
\mathbf{\Phi}(x_3, y_3, z_3, t_3)\ \mathbf{f} &= w_3 \\
\mathbf{\Phi}(x_4, y_4, z_4, t_4)\ \mathbf{f} &= w_4 \\
\mathbf{\Phi}(x_5, y_5, z_5, t_5)\ \mathbf{f} &= w_5
\end{aligned}$$

This can be written as $\mathbf{Af} = \mathbf{w}$, where $\mathbf{A} = \begin{bmatrix} \mathbf{\Phi}(x_1, y_1, z_1, t_1) \\ \mathbf{\Phi}(x_2, y_2, z_2, t_2) \\ \mathbf{\Phi}(x_3, y_3, z_3, t_3) \\ \mathbf{\Phi}(x_4, y_4, z_4, t_4) \\ \mathbf{\Phi}(x_5, y_5, z_5, t_5) \end{bmatrix}$,

and $\mathbf{w^T} = [w_1, w_2, w_3, w_4, w_5]$. We obtain the solution for $\mathbf{f}$ as:

$$\mathbf{f} = \mathbf{A}^{-1}\mathbf{w}\ . \tag{2.13}$$

Let

$$\mathbf{N}(x, y, z, t) = \mathbf{\Phi}(x, y, z, t)\mathbf{A}^{-1}\ . \tag{2.14}$$

After substituting (2.13) into (2.12), we have

$$\begin{aligned} w(x,y,z,t) &= \mathbf{\Phi}(x,y,z,t)\mathbf{A}^{-1}\mathbf{w} \\ &= \mathbf{N}(x,y,z,t)\mathbf{w} \\ &= [N_1\ N_2\ N_3\ N_4\ N_5] \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{bmatrix} \end{aligned} \tag{2.15}$$

Now it is clear that (2.14) is the shape function matrix that we need to find. We calculated the result of (2.14) by Matlab. Since $\mathbf{A}$ is a 5×5 matrix in symbolic form, its inverse is very complicated and messy. The expression result of $\mathbf{N}$ based on the x_i's, y_i's, z_i's, t_i's and w_i's $(1 \leq i \leq 5)$ is very redundant and unreadable. Each shape function expression N_i $(1 \leq i \leq 5)$ covers about four pages. Next, we introduce a second approach which is based on the linear 3-D shape functions (2.8) or (2.10) and yields a neat symbolic expression.

Approach II The idea in the second approach is to reduce the 4-D case to a 3-D case. This can be done if the deletion of a dimension does not collapse two nodes into one. For example, if we have (x, y, z, t) data points and we delete z coordinates, then we should not get two points with the same (x, y, t) values. Let us denote the 3-D shape functions by $\hat{N}_i(x, y, z)$ $(1 \leq i \leq 4)$. Then the 4-D linear approximation in terms of these can be expressed as follows:

$$w(x,y,z,t) = \hat{a}\hat{N}_1(x,y,z) + \hat{b}\hat{N}_2(x,y,z) + \hat{c}\hat{N}_3(x,y,z) + \hat{d}\hat{N}_4(x,y,z) + \hat{e}t\ .$$

Let $\hat{\mathbf{\Phi}}(x,y,z,t) = \left[\hat{N}_1(x,y,z), \hat{N}_2(x,y,z), \hat{N}_3(x,y,z), \hat{N}_4(x,y,z), t\right]$ and

$\hat{\mathbf{f}}^{\mathbf{T}} = \left[\hat{a}, \hat{b}, \hat{c}, \hat{d}, \hat{e}\right]$, we have:

$$w(x, y, z, t) = \hat{\mathbf{\Phi}}(x, y, z, t)\, \hat{\mathbf{f}} \ . \tag{2.16}$$

We use the five known nodal values (w_i's, $1 \leq i \leq 5$) to calculate $\hat{\mathbf{f}}$ as follows:

$$\begin{aligned}
\hat{\mathbf{\Phi}}(x_1, y_1, z_1, t_1)\ \hat{\mathbf{f}} &= w_1 \\
\hat{\mathbf{\Phi}}(x_2, y_2, z_2, t_2)\ \hat{\mathbf{f}} &= w_2 \\
\hat{\mathbf{\Phi}}(x_3, y_3, z_3, t_3)\ \hat{\mathbf{f}} &= w_3 \\
\hat{\mathbf{\Phi}}(x_4, y_4, z_4, t_4)\ \hat{\mathbf{f}} &= w_4 \\
\hat{\mathbf{\Phi}}(x_5, y_5, z_5, t_5)\ \hat{\mathbf{f}} &= w_5
\end{aligned}$$

Assuming $m_i = \hat{N}_i(x_5, y_5, z_5)$ $(1 \leq i \leq 4)$, this can be written as $\mathbf{B}\hat{\mathbf{f}} = \mathbf{w}$, where

$$\mathbf{B} = \begin{bmatrix} \hat{\mathbf{\Phi}}(x_1, y_1, z_1, t_1) \\ \hat{\mathbf{\Phi}}(x_2, y_2, z_2, t_2) \\ \hat{\mathbf{\Phi}}(x_3, y_3, z_3, t_3) \\ \hat{\mathbf{\Phi}}(x_4, y_4, z_4, t_4) \\ \hat{\mathbf{\Phi}}(x_5, y_5, z_5, t_5) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & t_1 \\ 0 & 1 & 0 & 0 & t_2 \\ 0 & 0 & 1 & 0 & t_3 \\ 0 & 0 & 0 & 1 & t_4 \\ m_1 & m_2 & m_3 & m_4 & t_5 \end{bmatrix} \text{ and } \mathbf{w}^{\mathbf{T}} = [w_1, w_2, w_3, w_4, w_5].$$

We obtain the solution for $\hat{\mathbf{f}}$ as:

$$\hat{\mathbf{f}} = \mathbf{B}^{-1}\mathbf{w} \ . \tag{2.17}$$

Let

$$\mathbf{N}(x, y, z, t) = \hat{\mathbf{\Phi}}(x, y, z, t)\mathbf{B}^{-1} \ . \tag{2.18}$$

After substituting (2.17) into (2.16), we have

$$\begin{aligned} w(x,y,z,t) &= \hat{\mathbf{\Phi}}(x,y,z,t)\mathbf{B}^{-1}\mathbf{w} \\ &= \mathbf{N}(x,y,z,t)\mathbf{w} \\ &= [N_1\ N_2\ N_3\ N_4\ N_5] \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{bmatrix} \end{aligned} \tag{2.19}$$

The shape function result of (2.18) can be calculated as follows:

$$N_i = \hat{N}_i + \frac{m_i h}{detB}\ (1 \le i \le 4)\ \ and\ \ N_5 = \frac{h}{detB}\ , \tag{2.20}$$

where $detB = -m_1t_1 - m_2t_2 - m_3t_3 - m_4t_4 + t_5$ is the determinant of $\mathbf{B}$ and $h = \hat{N}_1t_1 + \hat{N}_2t_2 + \hat{N}_3t_3 + \hat{N}_4t_4 - t$. This method can be generalized to derive shape functions of n dimension from shape functions of $n-1$ dimensions.

2.2 Inverse Distance Weighting

Inverse Distance Weighting (IDW) interpolation (Shepard 1968) is based on the assumption that things that are close to one another are more alike than those that are farther apart. Revesz & Li (2002*a*) uses IDW to visualize spatial interpolation data. In IDW, the measured values (known values) closer to a prediction location will have more influence on the predicted value (unknown value) than those farther away. More specifically, IDW assumes that each measured point has a local influence that diminishes with distance. Thus, points in the near neighborhood are given high weights, whereas points at a far distance are given small weights.

According to Johnston et al. (2001), the general formula of IDW interpolation is the following:

$$w(x,y) = \sum_{i=1}^{N} \lambda_i w_i \quad , \qquad \lambda_i = \frac{(\frac{1}{d_i})^p}{\sum_{k=1}^{N}(\frac{1}{d_k})^p} \quad , \tag{2.21}$$

where $w(x,y)$ is the predicted value at location (x,y), N is the number of nearest known points surrounding (x,y), λ_i are the weights assigned to each known point value w_i at location (x_i,y_i), d_i are the Euclidean distances between each (x_i,y_i) and (x,y), and p is the exponent, which influences the weighting of w_i on w.

Like shape functions, IDW is originally a spatial interpolation method and we can extend it by reduction and extension approaches to solve spatiotemporal interpolation problems.

2.2.1 2-D Space and 1-D Time Problems

2.2.1.1 Reduction Approach: ST Product Method

Assume we are interested in the value of the unsampled point at location (x,y) and time t. This approach first finds the nearest neighbors of for each unsampled point and calculates the corresponding weights λ_i. Then, it calculates for each neighbor the value at time t by some time interpolation method. If we use 1-D shape function interpolation in time, the time interpolation will be similar to (2.5). The formula of the this approach can be expressed as:

$$w(x,y,t) = \sum_{i=1}^{N} \lambda_i w_i(t) \quad , \qquad \lambda_i = \frac{(\frac{1}{d_i})^p}{\sum_{k=1}^{N}(\frac{1}{d_k})^p} \tag{2.22}$$

where $d_i = \sqrt{(x_i - x)^2 + (y_i - y)^2}$ and

$$w_i(t) = \frac{t_{i2} - t}{t_{i2} - t_{i1}} w_{i1} + \frac{t - t_{i1}}{t_{i2} - t_{i1}} w_{i2} \ . \tag{2.23}$$

Each neighbor may have different beginning and ending times t_{i1} and t_{i2} in (2.23) if each point is sampled at different times.

2.2.1.2 Extension Approach: 3-D Method

Since this method treats time as a third dimension, the IDW based spatiotemporal formula is of the form of (2.21) with $d_i = \sqrt{(x_i - x)^2 + (y_i - y)^2 + (t_i - t)^2}$.

2.2.2 3-D Space and 1-D Time Problems

2.2.2.1 Reduction Approach: ST Product Method

For 3-D space and 1-D time problems, we are interested in the value of the unsampled point at location (x, y, z) and time t. This approach is very similar to the IDW reduction approach for 2-D space & 1-D time problems in Section 2.2.1.1. Its formula is:

$$w(x, y, z, t) = \sum_{i=1}^{N} \lambda_i w_i(t) \quad , \qquad \lambda_i = \frac{(\frac{1}{d_i})^p}{\sum_{k=1}^{N} (\frac{1}{d_k})^p} \tag{2.24}$$

where $d_i = \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2}$ and $w_i(t)$ takes the same form as in (2.23).

2.2.2.2 Extension Approach: 4-D Method

Since this method treats time as a fourth dimension, the IDW based spatiotemporal formula is of the form of (2.21) with $d_i = \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2 + (t_i - t)^2}$.

2.3 Kriging

Kriging is an important interpolation method by using geostatistical analysis which provides a minimum error-variance estimate of any unsampled value. It was initially introduced by D. G. Krige as an optimal interpolation method in the mining industry (Krige 1951). It was later developed by G. Matheron as the *theory of regionalized variables* (Matheron 1971). Using Kriging as an interpolation method in GIS was discussed by Oliver & Webster (1990).

Kriging is similar to IDW in the sense that it uses a weighting mechanism that assigns more influence to the nearer data points to interpolate values at unknown locations. However, instead of using inverse distance weighting approach, Kriging uses variograms. As a measure of spatial variability, a variogram replaces the Euclidean distance by a structural distance that is specific to the attribute and the field under study (Deutsch & Journel 1998). Assume u is a location vector where the data value is unsampled. The variogram distance measures the average degree of dissimilarity between $w(u)$ and a nearby known data value. For example, given two sampled data values w_1 and w_2 at two different locations $u + h_1$ and $u + h_2$, the more "dissimilar" sample value should receive less weight in the estimation of $w(u)$.

2.3.1 2-D Space and 1-D Time Problems

2.3.1.1 Reduction Approach: ST Product Method

This is not a feasible approach for Kriging. According to (Lam 1983), a variogram $(2r)$ can be defined as

$$2r = \frac{1}{N}\sum_{i=1}^{N}[w(u_i + h) - w(u_i)]^2 \tag{2.25}$$

where h is the distance between two samples can N is the number of pairs of samples having the same distance.

From equation (2.25), we can see that not only do variograms depend on the location distribution (h) of samples, but also depend on the sample values (w). Since weights are determined by variograms, weights are also both location and information dependent. That is, weights can not be calculated without knowing the values of sample points. So, if we want to use reduction approach, we have to know in advance which sampled points will be used in Kriging for each unsampled points and then use some temporal interpolation method to estimate the sample values at the time the unsampled point is interested in. However, different unknown points may share some same sample points. This leads to the ambiguity about the values at what time shall be used for those sample points. Therefore, the reduction approach of spatiotemporal interpolation is not feasible for Kriging.

2.3.1.2 Extension Approach: 3-D Method

Since Kriging can be generalized into high dimension, the extension approach of Kriging is a natural approach for spatiotemporal interpolation. There are multiple types of Kriging, such as simple Kriging, ordinary Kriging, universal Kriging, and factorial Kriging. Ordinary Kriging is the most commonly used variant of simple Kriging and it has been the anchor algorithm of geostatistics (Deutsch & Journel 1998). If we choose 3-D ordinary Kriging, the estimation for unknown location u is calculated as:

$$w(u) = \sum_{i=1}^{N} \lambda_i w_i \quad , \qquad \sum_{i=1}^{N} \lambda_i = 1 \quad , \tag{2.26}$$

where weights λ_i are determined by variograms to minimize the error variance.

2.3.2 3-D Space and 1-D Time Problems

2.3.2.1 Reduction Approach: ST Product Method

Like the Kriging reduction approach for 2-D space & 1-D time problems in Section 2.3.1.1, this is also not a feasible approach for 3-D space and 1-D time problems.

2.3.2.2 Extension Approach: 4-D Method

Since Kriging can be generalized into 4-D, like in Section 2.3.1.2, the extension approach of Kriging is also a natural approach for 3-D space and 1-D time spatiotemporal interpolation. Since formulas of different types of Kriging are different and they are not the emphasis in this dissertation, further discussion is omitted.

Chapter 3

Constraint Databases

3.1 Constraint Databases

Early work on constraint databases in logic programming has been done by Jaffar & Lassez (1987). The concepts of constraint data model and query language have been explored by Kanellakis, Kuper & Revesz (1990) and Kanellakis et al. (1995). Constraint databases is a growing area. The recently books on constraint databases are Kuper, Libkin & Paredaens (2000) and Revesz (2002).

Constraint databases generalize relational databases by finitely representable infinite relations. In the constraint data model, each attribute is associated with an attribute variable and the value of the attributes in a relation is specified implicitly using constraints.

A *constraint database* is a finite set of constraint relations. A *constraint relation* is a finite set of constraint tuples, where each *constraint tuple* is a conjunction of *atomic constraints* using the same set of attribute variables. Atomic constraints includes *arithmetic atomic constraints* and *boolean atomic constraints*.

The most basic types of arithmetic atomic constraints are the following: Equality, Inequality, Lower Bound, Upper Bound, Order, Gap–Order, Difference, Half–

Addition, Linear, and Polynomial. If a constraint database does not include polynomial arithmetic constraints, it can be called a linear constraint database.

The basic types of boolean atomic constraints are the following: Equality, Inequality, Monotone Equality, Monotone Inequality, Precedence, and Exclusive–Or.

3.1.1 Ultraviolet Radiation Input Database Example

Suppose that we have the following two sets of sensory data in our database:

1. $Incoming(y, t, u)$ records the amount of incoming ultraviolet radiation u for each pair of latitude degree y and time t, where time is measured in days.

2. $Filter(x, y, r)$ records the ratio r of ultraviolet radiation that is usually filtered out by the atmosphere above location (x, y) before reaching the earth.

Suppose that Figure 3.1 shows the locations of the (y, t) and (x, y) pairs where the measurements for u and r, respectively, are recorded. Then Tables 3.1 and 3.2 could be instances of these two relations in a relational database.

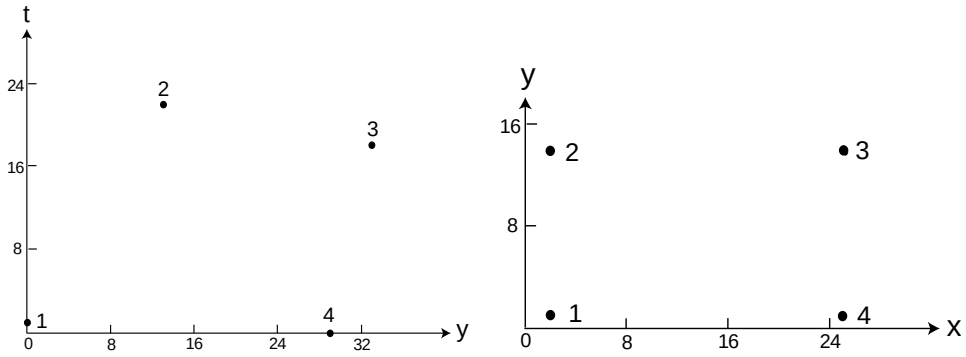

Figure 3.1: The spatial sample points for *Incoming* (left) and *Filter* (right).

The above relational database can be translated into a constraint database with the two constraint relations shown in Tables 3.3 and 3.4.

Although any relational relation can be translated into a constraint relation as above, not all the constraint relations can be converted back to relational databases. A

Table 3.1: Relational Incoming (y, t, u).

ID	Y	T	U
1	0	1	60
2	13	22	20
3	33	18	70
4	29	0	40

Table 3.2: Relational Filter (x, y, r).

ID	X	Y	R
1	2	1	0.9
2	2	14	0.5
3	25	14	0.3
4	25	1	0.8

constraint relation can store infinite number of solutions. For example, we can represent the interpolation of u and r for all the points in the domains for $Incoming(y, t, u)$ and $Filter(x, y, r)$ in a constraint database.

As described in Chapter 2, there are many types of interpolations. 2-D shape function and IDW interpolation methods will be used next for the ultraviolet radiation example to illustrate how constraint databases can represent infinitive interpolation results. Let $INCOMING(y, t, u)$ be the constraint relation that represents the interpolation of the $Incoming$ relation. Similarly, let $FILTER(x, y, r)$ be the constraint relation that represents the interpolation of the $Filter$ relation.

3.1.2 Representation of 2-D Shape Function Interpolation in Constraint Databases

According to Section 2.1, triangulation of the set of sampled points is the first step to use 2-D shape functions. Figures 3.2 and 3.3 shows the Delaunay triangulations for the sample points in $Incoming(y, t, u)$ and $Filter(x, y, r)$ (Revesz & Li 2002*b*).

Table 3.3: Constrain Incoming (y, t, u).

ID	Y	T	U	
id	y	t	u	$id = 1$, $y = 0$, $t = 1$, $u = 60$
id	y	t	u	$id = 2$, $y = 13$, $t = 22$, $u = 20$
id	y	t	u	$id = 3$, $y = 33$, $t = 18$, $u = 70$
id	y	t	u	$id = 4$, $y = 29$, $t = 0$, $u = 40$

Table 3.4: Constraint Filter (x, y, r).

ID	X	Y	R	
id	x	y	r	$id = 1$, $x = 2$, $y = 1$, $r = 0.9$
id	x	y	r	$id = 2$, $x = 2$, $y = 14$, $r = 0.5$
id	x	y	r	$id = 3$, $x = 25$, $y = 14$, $r = 0.3$
id	x	y	r	$id = 4$, $x = 25$, $y = 1$, $r = 0.8$

The area of a Delaunay triangle can be represented by a conjunction C of three linear inequalities corresponding to the three sides of the triangle. Then, by the shape function (2.2) in Section 2.1.1, the value w of any point x, y inside the triangle can be represented by the linear constraint tuple:

$$\begin{aligned} R(x,y,w) \quad :- C, \quad w = \quad & [((y_2 - y_3)w_1 + (y_3 - y_1)w_2 + (y_1 - y_2)w_3)/(2\mathcal{A})]x + \\ & [((x_3 - x_2)w_1 + (x_1 - x_3)w_2 + (x_2 - x_1)w_3)/(2\mathcal{A})]y + \\ & [((x_2y_3 - x_3y_2)w_1 + (x_3y_1 - x_1y_3)w_2 + (x_1y_2 - x_2y_1)w_3)/(2\mathcal{A})]. \end{aligned}$$

where $\mathcal{A}$ is the constant value of the triangle area. By representing the interpolation in each triangle by a separate constraint tuple, we can find in linear time a constraint relation to represent the whole interpolation.

Tables 3.5 and 3.6 illustrate the constraint representation for the interpolation result *INCOMING* and *FILTER* using 2-D shape functions.

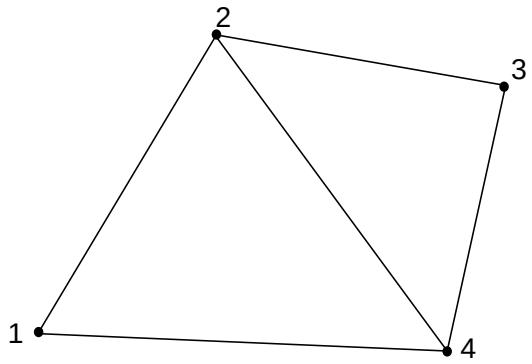

Figure 3.2: Delaunay triangulations for *Incoming*.

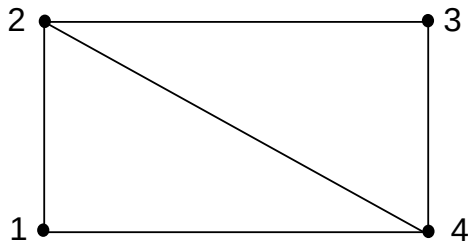

Figure 3.3: Delaunay triangulations for *Filter*.

3.1.3 Representation of IDW Interpolation in Constraint Databases

3.1.3.1 Higher-order Voronoi Diagrams

To represent the IDW interpolation, we need first to find the nearest neighbors for a given point. Therefore, we borrow the idea of higher-order Voronoi diagrams (or k-th order Voronoi diagrams) from computational geometry. Higher-order Voronoi diagrams generalize ordinary Voronoi diagrams by dealing with k closest points. The ordinary Voronoi diagram of a finite set S of points in the plane is a partition of the plane so that each region of the partition is the locus of points which are closer to *one member* of S than to any other member (Preparata & Shamos 1985). The higher-order Voronoi diagram of a finite set S of points in the plane is a partition of the plane into regions such that points in each region have the *same closest members* of S. As in an ordinary Voronoi diagram, each Voronoi region is still convex in a

Table 3.5: INCOMING (y, t, u) using 2-D shape functions.

Y	T	U	
y	t	u	$21x - 13y + 13 \geq 0$, $x + 29y - 29 \leq 0$, $11x + 8y - 319 \geq 0$, $u = 0.69x - 1.45y + 62.33$
y	t	u	$11x + 8y - 319 \leq 0$, $x + 5y - 97 \leq 0$, $9x - 2y - 261 \leq 0$, $u = 2.71x - 1.06y + 79.95$

Table 3.6: FILTER (x, y, r) using 2-D shape functions.

X	Y	R	
x	y	r	$13x - 23y + 296 \geq 0$, $x \geq 2$, $y \geq 1$, $r = 0.0004x - 0.0031y + 0.1168$
x	y	r	$13x - 23y + 296 \leq 0$, $x \leq 25$, $y \leq 14$, $r = 0.0013x - 0.0038y + 0.1056$

higher-order Voronoi diagram.

From the definition of higher-order Voronoi diagrams, it is obvious to see that the problem of finding the k closest neighbors for a given point in the whole domain, which is closely related to the IDW interpolation method with $N = k$, is equivalent to constructing k-th order Voronoi diagrams.

Although higher-order Voronoi diagrams are very difficult to create by imperative languages, such as C, C++, and Java, they can be easily constructed by declarative languages, such as Datalog. For example, we can express a second-order Voronoi region for points (x_1, y_1), (x_2, y_2) in Datalog as follows.

At first, let $P(x, y)$ be a relation that stores all the points in the whole domain. Also let $Dist(x, y, x_1, y_1, d_1)$ be a Euclidean distance relation where d_1 is the distance between (x, y) and (x_1, y_1). It can be expressed in Datalog as:

$$Dist(x, y, x_1, y_1, d_1) \quad :- \quad d_1 = \sqrt{(x - x_1)^2 + (y - y_1)^2} \ .$$

Note that any point (x, y) in the plane does *not* belong to the 2nd order Voronoi region of the sample points (x_1, y_1) and (x_2, y_2) if there exists another sample point (x_3, y_3) such that (x, y) is closer to (x_3, y_3) than to either (x_1, y_1) or (x_2, y_2). Using this idea, the complement can be expressed as follows:

$$\begin{aligned} Not\_2Vor(x, y, x_1, y_1, x_2, y_2) \ :- \ & P(x_3, y_3), \\ & Dist(x, y, x_1, y_1, d_1), \\ & Dist(x, y, x_3, y_3, d_3), \\ & d_1 > d_3. \end{aligned}$$

$$\begin{aligned} Not\_2Vor(x, y, x_1, y_1, x_2, y_2) \ :- \ & P(x_3, y_3), \\ & Dist(x, y, x_2, y_2, d_2), \\ & Dist(x, y, x_3, y_3, d_3), \\ & d_2 > d_3. \end{aligned}$$

Finally, we take the negation of the above to get the 2nd order Voronoi region as follows:

$$2Vor(x, y, x_1, y_1, x_2, y_2) : -not\ Not\_2Vor(x, y, x_1, y_1, x_2, y_2).$$

The second-order Voronoi diagram will be the union of all the nonempty second-order Voronoi regions. Similarly, to the 2nd order, we can also construct any kth-order Voronoi diagram.

3.1.3.2 IDW in Constraint Databases

After finding the closest neighbors for each point by constructing higher-order Voronoi diagrams, we can represent IDW interpolation in constraint databases. In this section, we describe how to represent the IDW interpolation with $N = 2$ and $p = 2$. The

representation of other IDW interpolations in constraint databases is straightforward to get. The representation is obtained by constructing the appropriate Nth-order Voronoi diagram (where $N \geq 2$) and using Equation 2.21 with the proper p.

Based on the previous section, assume that the second-order Voronoi region for points (x_1, y_1), (x_2, y_2) is stored by the relation *Vor_2nd(x, y, x_1, y_1, x_2, y_2)*, which is a conjunction C of some linear inequalities corresponding to the edges of the Voronoi region. Then, the value w of any point (x, y) inside the Voronoi region can be expressed by the cubic constraint tuple as follows:

$$\begin{aligned} R(x, y, w) \ :- \ & ((x - x_2)^2 + (y - y_2)^2 + \\ & (x - x_1)^2 + (y - y_1)^2)\ w \\ & = \\ & ((x - x_2)^2 + (y - y_2)^2)w_1 + \\ & ((x - x_1)^2 + (y - y_1)^2)w_2\ , \\ & Vor\_2nd(x, y, x_1, y_1, x_2, y_2). \end{aligned} \tag{3.1}$$

or equivalently as,

$$\begin{aligned} R(x, y, w) \ :- \ & ((x - x_2)^2 + (y - y_2)^2 + \\ & (x - x_1)^2 + (y - y_1)^2)\ w \\ & = \\ & ((x - x_2)^2 + (y - y_2)^2)w_1 + \\ & ((x - x_1)^2 + (y - y_1)^2)w_2\ , \\ & C. \end{aligned} \tag{3.2}$$

In the above polynomial constraint relation, there are three variables x, y, and

w. The highest order terms in the relation are $2x^2w$ and $2y^2w$, which are both cubic. Therefore, this is a cubic constraint tuple.

3.1.3.3 Application

Let us return now to the ultraviolet radiation example in Section 3.1.1. Figures 3.4 and 3.5 show the second-order Voronoi diagrams for the sample points in *Incoming(y,t,u)* and *Filter(x,y,r)*, respectively. Please note that some second-order Voronoi regions are empty. For example, there is no $(1, 3)$ region in Figure 3.4, and there are no $(1, 3)$ and $(2, 4)$ regions in Figure 3.5 (Revesz & Li 2002*a*).

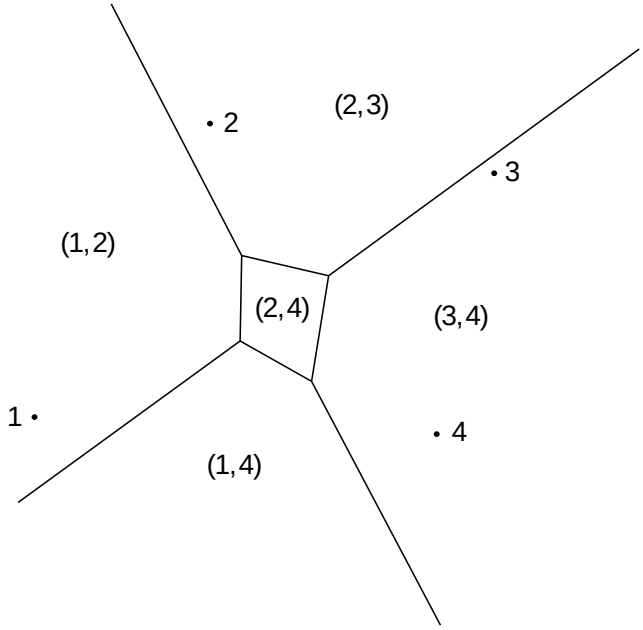

Figure 3.4: The 2nd order Voronoi diagram for *Incoming*.

Based on Equation 3.2, *INCOMING(y,t,u)* and *FILTER(x,y,r)*, which are the IDW interpolation for *Incoming(y,t,u)* and *Filter(x,y,r)*, can be represented in constraint databases as shown in Tables 3.7 and 3.8. Note that the five tuples in Table 3.7 represent the five second-order Voronoi regions in Figure 3.4. These five regions are $(1, 2)$, $(1, 4)$, $(3, 4)$, $(2, 3)$ and $(2, 4)$. Similarly, the four tuples in Table 3.8 represent

Table 3.7: INCOMING (y, t, u) using IDW.

Y	T	U	
y	t	u	$13y + 7t - 286 \leq 0,$ $2y - 3t - 12 \leq 0,$ $y \leq 15,$ $((y-13)^2 + (t-22)^2)60 + (y^2 + (t-1)^2)20$ $= ((y-13)^2 + (t-22)^2 + y^2 + (t-1)^2)u$
y	t	u	$2y - 3t - 12 \geq 0,$ $2y + 5t - 60 \leq 0,$ $2y + t - 44 \leq 0,$ $((y-29)^2 + t^2)60 + (y^2 + (t-1)^2)40$ $= ((y-29)^2 + t^2 + y^2 + (t-1)^2)u$
y	t	u	$2y + t - 44 \geq 0,$ $7y - t - 136 \geq 0,$ $8y - 11t - 47 \geq 0,$ $((y-29)^2 + t^2)70 + ((y-33)^2 + (t-18)^2)40$ $= ((y-29)^2 + t^2 + (y-33)^2 + (t-18)^2)u$
y	t	u	$8y - 11t - 47 \leq 0,$ $y + 3t - 54 \geq 0,$ $13y + 7t - 286 \geq 0,$ $((y-33)^2 + (t-18)^2)20 + ((y-13)^2 + (t-22)^2)70$ $= ((y-33)^2 + (t-18)^2 + (y-13)^2 + (t-22)^2)u$
y	t	u	$y \geq 15,$ $y + 3t - 54 \leq 0,$ $7y - t - 136 \leq 0,$ $2y + 5t - 60 \geq 0,$ $((y-29)^2 + t^2)20 + ((y-13)^2 + (t-22)^2)40$ $= ((y-29)^2 + t^2 + (y-13)^2 + (t-22)^2)u$

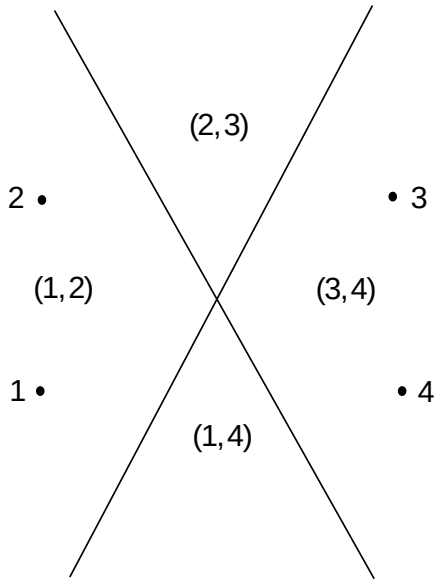

Figure 3.5: The 2nd order Voronoi diagram for *Filter*.

the four second-order Voronoi regions in Figure 3.5. These four regions are $(1, 2)$, $(1, 4)$, $(3, 4)$ and $(2, 3)$.

3.2 Datalog Query Languages

There are several ways to query constraint databases, such as the relational algebra, SQL, and Datalog. Among them, Datalog is the most powerful query language. Therefore, in this section, Datalog is chosen to show how to query constraint databases.

Datalog is a nonprocedural query language that is based on the logic-programming language Prolog. Recursion is allowed in Datalog. Datalog with recursion has more expressive power than Datalog without recursion. Nonrecursive Datalog expressions without arithmetic operations are equivalent in expressive power to expressions using the basic operations in relational algebra. Therefore Datalog has more expressive power than the relational algebra.

Table 3.8: FILTER (x, y, r) using IDW.

X	Y	R	
x	y	r	$2x - y - 20 \leq 0,\ 12x + 7y - 216 \leq 0,$ $((x-2)^2 + (y-14)^2)0.9+$ $((x-2)^2 + (y-1)^2)0.5 =$ $(2(x-2)^2 + (y-14)^2 + (y-1)^2)r$
x	y	r	$2x - y - 20 \geq 0,\ 12x + 7y - 216 \leq 0,$ $((x-25)^2 + (y-1)^2)0.9+$ $((x-2)^2 + (y-1)^2)0.8 =$ $(2(y-1)^2 + (x-25)^2 + (x-2)^2)r$
x	y	r	$2x - y - 20 \geq 0,\ 12x + 7y - 216 \geq 0,$ $((x-25)^2 + (y-14)^2)0.8+$ $((x-25)^2 + (y-1)^2)0.3 =$ $(2(x-25)^2 + (y-14)^2 + (y-1)^2)r$
x	y	r	$2x - y - 20 \leq 0,\ 12x + 7y - 216 \geq 0,$ $((x-25)^2 + (y-14)^2)0.5+$ $((x-2)^2 + (y-14)^2)0.3 =$ $(2(y-14)^2 + (x-25)^2 + (x-2)^2)r$

3.2.1 Ultraviolet Radiation Query Example

Query 3.2.1 *For the ultraviolet radiation example in Section 3.1.1, find the amount of ultraviolet radiation for each ground location (x, y) at time t.*

Since the input relations in Tables 3.1 and 3.2 only record the incoming ultraviolet radiation u and filter ratio r on a few sample points, these cannot be used directly to answer the query. Therefore, to answer this query, we need the interpolation result $INCOMING(y, t, u)$ and $FILTER(x, y, r)$. To write queries, we do not need to know precisely what kind of interpolation method is used and what are the constraints used in the representation interpolation. The above query can be expressed in Datalog as follows:

$$GROUND(x, y, t, i) \ :- \ INCOMING(y, t, u),$$
$$FILTER(x, y, r),$$
$$i = u(1 - r).$$

The above query could be also expressed in SQL style or relational algebra. Whatever language is used, it is clear that the evaluation of the above query requires a join of the *INCOMING* and *FILTER* relations. Unfortunately, join operations are difficult to express in most GIS systems, including the ArcGIS system. However, join processing is very natural in constraint database systems.

If we use IDW interpolation, the final result of the Datalog query, *GROUND(x, y, t, i)*, can be represent by Table 3.9. Since there are five tuples in *INCOMING(y,t,u)* in Table 3.7 and four tuples in *FILTER(x,y,r)* in Table 3.8, there should be twenty tuples in *GROUND(x, y, t, i)* in Table 3.9. Note that the constraint relations can be easily joined by taking the conjunction of the constraints from each pair tuples of the two input relations. Finally, in a constraint database system the constraint in each tuple are automatically simplified by eliminating the unnecessary variables u and r. We do not show the result of the simplification step.

3.2.2 Fire Management Plan Query Example

Suppose that we have the following relations in the input constraint database:

1. $Vegcover(x, y, v)$: recording the vegetation cover type v at each location (x, y).
2. $Elevation(x, y, e)$: recording the elevation e at each location (x, y).
3. $Slope(x, y, s)$: recording the slope s at each location (x, y).
4. $Aspect(x, y, a)$: recording the aspect a at each location (x, y).

Based on the historical data that summarize the fire reports by four spatial variables: (i) vegetation type, (ii) elevation, (iii) slope, and (iv) aspect, we have the following assumption:

Table 3.9: GROUND (x, y, t, i) using IDW.

X	Y	T	I	
x	y	t	i	$2x - y - 20 \leq 0,$ $12x + 7y - 216 \leq 0,$ $13y + 7t - 286 \leq 0,$ $2y - 3t - 12 \leq 0,$ $y \leq 15,$ $((x-2)^2 + (y-14)^2)0.9+$ $((x-2)^2 + (y-1)^2)0.5 =$ $(2(x-2)^2 + (y-14)^2 + (y-1)^2)r,$ $((y-13)^2 + (t-22)^2)60 + (y^2 + (t-1)^2)20$ $= ((y-13)^2 + (t-22)^2 + y^2 + (t-1)^2)u,$ $i = u(1-r)$
x	y	t	i	$2x - y - 20 \geq 0,$ $12x + 7y - 216 \leq 0,$ $13y + 7t - 286 \leq 0,$ $2y - 3t - 12 \leq 0,$ $y \leq 15,$ $((x-25)^2 + (y-1)^2)0.9+$ $((x-2)^2 + (y-1)^2)0.8 =$ $(2(y-1)^2 + (x-25)^2 + (x-2)^2)r,$ $((y-13)^2 + (t-22)^2)60 + (y^2 + (t-1)^2)20$ $= ((y-13)^2 + (t-22)^2 + y^2 + (t-1)^2)u,$ $i = u(1-r)$
x	y	t	i	$\vdots$ $\vdots$
x	y	t	i	$2x - y - 20 \leq 0,$ $12x + 7y - 216 \geq 0,$ $y \geq 15,$ $y + 3t - 54 \leq 0,$ $7y - t - 136 \leq 0,$ $2y + 5t - 60 \geq 0,$ $((x-25)^2 + (y-14)^2)0.5+$ $((x-2)^2 + (y-14)^2)0.3 =$ $(2(y-14)^2 + (x-25)^2 + (x-2)^2)r,$ $((y-29)^2 + t^2)20 + ((y-13)^2 + (t-22)^2)40$ $= ((y-29)^2 + t^2 + (y-13)^2 + (t-22)^2)u,$ $i = u(1-r)$

1. For the *Vegcover* relation, "coniferous forests", "grass lands", "mixed forests", "deciduous forests", and "disturbed (a mixed class including urban lands, surface mines, exposed rock and soil)" are considered to have the fire risk rating of 4, 3, 2, 1, and 0, respectively.

2. For the *Elevation* relation, the areas with elevation values in $4000 - 5999ft$ are considered to have the fire risk rating of 4, the areas with elevation values in $2000 - 2999ft$ or $6000 - 6999ft$ are considered to have the fire risk rating of 3, the areas with elevation values in $3000 - 3999ft$ are considered to have the fire risk rating of 2, and the areas with elevation values in $0 - 1999ft$ or $\geq 7000ft$ are considered to have the fire risk rating of 1.

3. For the *Slope* relation, the areas with slope in $0° - 20°$, $21° - 40°$, $41° - 60°$, and $\geq 60°$ are considered to have the fire risk rating of 4, 3, 2, and 1, respectively.

4. For the *Aspect* relation, the areas with aspect in S (south) are considered to have the fire risk rating of 4, the areas with aspect in SW (southwest) are considered to have the fire risk rating of 3, the areas with aspect in E (east), or SE (southeast), or W (west) are considered to have the fire risk rating of 2, and the areas with aspect in N (north), or NE (northeast), or NW (northwest) are considered to have the fire risk rating of 1.

Please note that the fire risk rating is on a scale of $0 \sim 4$, where 4 indicates the highest possibility to catch fire and 0 indicates the lowest possibility to catch fire.

Query 3.2.2 *This query is to get the composite* $Fire\_Risk(x, y, r)$ *relation to analyze the composite fire risk* r *which combines the individual fire risks based on vegetation type, elevation, slope, and aspect. First of all, we need to reclassify the relations* $Vegcover(x, y, v)$, $Elevation(x, y, e)$, $Slope(x, y, s)$, *and* $Aspect(x, y, a)$ *according to their own fire risk ratings as follows:*

$$Fire\_Risk\_Veg(x, y, r\_v) \ :- \ Vegcover(x, y, v),\ v = \text{“}conifer\text{”},\ r\_v = 4\ .$$

$$Fire\_Risk\_Veg(x, y, r\_v) \ :- \ Vegcover(x, y, v),\ v = \text{“}grass\text{”},\ r\_v = 3\ .$$

$$Fire\_Risk\_Veg(x, y, r\_v) \ :- \ Vegcover(x, y, v),\ v = \text{“}mixed\text{”},\ r\_v = 2\ .$$

$$Fire\_Risk\_Veg(x, y, r\_v) \ :- \ Vegcover(x, y, v),\ v = \text{“}deciduous\text{”},\ r\_v = 1\ .$$

$$Fire\_Risk\_Veg(x, y, r\_v) \ :- \ Vegcover(x, y, v),\ v = \text{“}disturbed\text{”},\ r\_v = 0\ .$$

$$Fire\_Risk\_Ele(x, y, r\_e) \ :- \ Elevation(x, y, e),\ e \geq 4000,\ e \leq 5999,\ r\_e = 4\ .$$

$$Fire\_Risk\_Ele(x, y, r\_e) \ :- \ Elevation(x, y, e),\ e \geq 2000,\ e \leq 2999,\ r\_e = 3\ .$$

$$Fire\_Risk\_Ele(x, y, r\_e) \ :- \ Elevation(x, y, e),\ e \geq 6000,\ e \leq 6999,\ r\_e = 3\ .$$

$$Fire\_Risk\_Ele(x, y, r\_e) \ :- \ Elevation(x, y, e),\ e \geq 3000,\ e \leq 3999,\ r\_e = 2\ .$$

$$Fire\_Risk\_Ele(x, y, r\_e) \ :- \ Elevation(x, y, e),\ e \geq 0,\ e \leq 1999,\ r\_e = 1\ .$$

$$Fire\_Risk\_Ele(x, y, r\_e) \ :- \ Elevation(x, y, e),\ e \geq 7000,\ r\_e = 1\ .$$

$$Fire\_Risk\_Slo(x, y, r\_s) \ :- \ Slope(x, y, s),\ e \geq 0^{\circ},\ e \leq 20^{\circ},\ r\_s = 4\ .$$

$$Fire\_Risk\_Slo(x, y, r\_s) \ :- \ Slope(x, y, s),\ e \geq 21^{\circ},\ e \leq 40^{\circ},\ r\_s = 3\ .$$

$$Fire\_Risk\_Slo(x, y, r\_s) \ :- \ Slope(x, y, s),\ e \geq 41^{\circ},\ e \leq 60^{\circ},\ r\_s = 2\ .$$

$$Fire\_Risk\_Slo(x, y, r\_s) \ :- \ Slope(x, y, s),\ e \geq 61^{\circ},\ r\_s = 1\ .$$

$$Fire\_Risk\_Asp(x, y, r\_a) \ :- \ Aspect(x, y, a),\ a = \text{“}S\text{”},\ r\_a = 4\ .$$

$$Fire\_Risk\_Asp(x, y, r\_a) \ :- \ Aspect(x, y, a),\ a = \text{“}SW\text{”},\ r\_a = 3\ .$$

$$Fire\_Risk\_Asp(x, y, r\_a) \ :- \ Aspect(x, y, a),\ a = \text{“}E\text{”},\ r\_a = 2\ .$$

$$Fire\_Risk\_Asp(x, y, r\_a) \ :- \ Aspect(x, y, a),\ a = \text{“}SE\text{”},\ r\_a = 2\ .$$

$$Fire\_Risk\_Asp(x, y, r\_a) \ :- \ Aspect(x, y, a),\ a = \text{“}W\text{”},\ r\_a = 2\ .$$

$$Fire\_Risk\_Asp(x, y, r\_a) \ :- \ Aspect(x, y, a),\ a = \text{“}N\text{”},\ r\_a = 1\ .$$

$$Fire\_Risk\_Asp(x, y, r\_a) \ :- \ Aspect(x, y, a),\ a = \text{“}NE\text{”},\ r\_a = 1\ .$$

$$Fire\_Risk\_Asp(x, y, r\_a) \ :- \ Aspect(x, y, a),\ a = \text{“}NW\text{”},\ r\_a = 1\ .$$

Next, we can produce a temporary composite $Temp\_Fire\_Risk(x, y, \hat{r})$ relation as follows based on the assumption that the four factors (vegetation type, elevation, slope, and aspect) contribute with the same weight to the composite fire risk:

$$\begin{aligned} Temp\_Fire\_Risk(x, y, \hat{r}) \ :- \ & Fire\_Risk\_Veg(x, y, r\_v),\ Fire\_Risk\_Ele(x, y, r\_e), \\ & Fire\_Risk\_Slo(x, y, r\_s),\ Fire\_Risk\_Asp(x, y, r\_a), \\ & \hat{r} = r\_v + r\_e + r\_s + r\_a\ . \end{aligned}$$

The fire risk attribute $\hat{r}$ in $Temp\_Fire\_Risk(x, y, \hat{r})$ should have values ranging from 3 to 16, with the higher number indicating higher fire risk. This is because the highest fire risk ratings for $Fire\_Risk\_Veg$, $Fire\_Risk\_Ele$, $Fire\_Risk\_Slo$, and $Fire\_Risk\_Asp$ are all 4, while the lowest ratings for $Fire\_Risk\_Veg$ are 0 and 1 for $Fire\_Risk\_Ele$, $Fire\_Risk\_Slo$, and $Fire\_Risk\_Asp$.

Finally, we can produce the $Fire\_Risk(x, y, r)$ relation by reclassifying $\hat{r}$ in *Temp_Fire_Risk (x,y,$\hat{r}$)* into four new classes according to the following rule: new class 1 = old classes 3-5 (low risk), new class 2 = old classes 6-9 (moderate low risk), new class 3 = old classes 10-12 (moderate high risk), and new class 4 = old classes 13-16 (high risk). This can be expressed as:

$$\begin{aligned} Fire\_Risk(x, y, r) \ &:- \ Temp\_Fire\_Risk(x, y, \hat{r}),\ \hat{r} \geq 3\ , \hat{r} \leq 5,\ r = 1\ . \\ Fire\_Risk(x, y, r) \ &:- \ Temp\_Fire\_Risk(x, y, \hat{r}),\ \hat{r} \geq 6\ , \hat{r} \leq 9,\ r = 2\ . \\ Fire\_Risk(x, y, r) \ &:- \ Temp\_Fire\_Risk(x, y, \hat{r}),\ \hat{r} \geq 10\ , \hat{r} \leq 12,\ r = 3\ . \\ Fire\_Risk(x, y, r) \ &:- \ Temp\_Fire\_Risk(x, y, \hat{r}),\ \hat{r} \geq 13\ , \hat{r} \leq 16,\ r = 4\ . \end{aligned}$$

Suppose that there is another relation in the input constraint database, $Road(x, y, r)$, which records the road information r. It is assumed that once fire occurs, fire crews rush to the scene from the roads. Therefore, the accessibility to the road network

should be considered as a factor for the fire hazard analysis. Based on the distance to the road network, we assume that: the areas 0-500 meters from a road are considered to have the fire risk rating of 1, the areas 501-1000 meters from a road are considered to have the fire risk rating of 2, the areas 1001-1500 meters from a road are considered to have the fire risk rating of 3, and the areas greater than 1500 meters from a road are considered to have the fire risk rating of 4. Again, the fire risk rating is on a scale of $0 \sim 4$, with higher number indicating higher fire hazard.

Query 3.2.3 *This query is to analyze the fire hazard by considering both fire risk based on the historical data and road accessibility. First of all, we need to produce the relation* $Access(x, y, h\_a)$ *to record the fire hazard according to road accessibility as follows:*

$$
\begin{aligned}
Access(x,y,h\_a) \;&:-\; road(x',y',r),\ \sqrt{(x-x')^2+(y-y')^2} \le 500,\ h\_a = 1\ .\\
Access(x,y,h\_a) \;&:-\; road(x',y',r),\ \sqrt{(x-x')^2+(y-y')^2} \ge 501,\\
&\qquad \sqrt{(x-x')^2+(y-y')^2} \le 1000,\ h\_a = 2\ .\\
Access(x,y,h\_a) \;&:-\; road(x',y',r),\ \sqrt{(x-x')^2+(y-y')^2} \ge 1001,\\
&\qquad \sqrt{(x-x')^2+(y-y')^2} \le 1500,\ h\_a = 3\ .\\
Access(x,y,h\_a) \;&:-\; road(x',y',r),\ \sqrt{(x-x')^2+(y-y')^2} \ge 1501,\ h\_a = 4\ .
\end{aligned}
$$

Next, we can produce a temporary composite $Temp\_Fire\_Hazard(x, y, \hat{h})$ relation as follows based on the assumption that fire risk and road accessibility contribute with the same weight to the fire hazard:

$$
Temp\_Fire\_Hazard(x,y,\hat{h}) \;:-\; Fire\_Risk(x,y,r),\ Access(x,y,h\_a),\ \hat{h} = r + h\_a\ .
$$

The fire hazard attribute $\hat{h}$ in $Temp\_Fire\_Hazard(x, y, \hat{h})$ should have values ranging from 2 to 8, with the higher number indicating higher fire hazard. This is

because the highest fire hazard ratings for $Fire\_Risk$ and $Access$ are both 4, while the lowest ratings for them are both 1.

Finally, we can produce the $Fire\_Hazard(x, y, h)$ relation by reclassifying $\hat{h}$ in *Temp_Fire_Hazard (x,y,*$\hat{h}$*)* into four new classes according to the following rule: new class 1 = old classes 2-3 (low hazard), new class 2 = old classes 4-5 (moderate low hazard), new class 3 = old classes 6-7 (moderate high hazard), and new class 4 = old classes 8 (high hazard). This can be expressed as:

$$
\begin{aligned}
Fire\_Hazard(x, y, h) \ &:- \ Temp\_Fire\_Hazard(x, y, \hat{h}),\ \hat{h} \geq 2\ , \hat{r} \leq 3,\ h = 1\ . \\
Fire\_Hazard(x, y, h) \ &:- \ Temp\_Fire\_Hazard(x, y, \hat{h}),\ \hat{h} \geq 4\ , \hat{r} \leq 5,\ h = 2\ . \\
Fire\_Hazard(x, y, h) \ &:- \ Temp\_Fire\_Hazard(x, y, \hat{h}),\ \hat{h} \geq 6\ , \hat{r} \leq 7,\ h = 3\ . \\
Fire\_Hazard(x, y, h) \ &:- \ Temp\_Fire\_Hazard(x, y, \hat{h}),\ \hat{h} = 8,\ h = 4\ .
\end{aligned}
$$

3.2.3 Greenness Data Temporal Query Example

Let $Fire(x, y, t)$ record the location (x, y) that have wild fire activity at time t and $Greenness(x, y, g, t)$ record the greenness index g at location (x, y) at time t. Greenness data record the seasonal characteristics of vegetation. Usually in greenness maps, the reddish-brown to green colors represent the gradation from sparse to dense live vegetation and from weak to vigorous growth. The Normalized Difference Vegetation Index (NDVI) is the most widely used vegetation greenness index in satellite data analysis, which can be calculated by the following formula:

$$NDVI = \frac{NIR - RED}{NIR - RED}$$

where NIR is the brightness in the near infra-red wavebands wavebands and RED is the brightness in the red wavebands. It can be seen that the NDVI greenness index is normalized and nonlinear. Its range varies between -1 and +1.

The study of greenness data is an important and interesting topic, for example, it can be used in fire assessment systems. Yang, Yang & Merchant (1997) is another example on the research of greenness data. Next two querying examples on greenness data are given.

Query 3.2.4 *To find the locations that had fire in 1999 with greenness index value* $\in [-1.0, 0)$*, and calculate the total area, we can do the following datalog query:*

$$\begin{aligned} Find\_neg\_green\_1999(x, y) \ &:- \ Fire(x, y, t), Greenness(x, y, g, t), t \text{ in } 1999, \\ &\qquad g < 0, g \geq -1.0 \ . \\ Fire\_Area2(area < x, y >) \ &:- \ Find\_neg\_green\_1999(x, y) \ . \end{aligned}$$

Query 3.2.5 *To find the locations that had fire in 1999 with greenness index value* $\in [0, 1.0]$*, and calculate the total area, we can do the following datalog query:*

$$\begin{aligned} Find\_pos\_green\_1999(x, y) \ &:- \ Fire(x, y, t), Greenness(x, y, g, t), t \text{ in } 1999, \\ &\qquad g \geq 0, g \leq 1.0 \ . \\ Fire\_Area3(area < x, y >) \ &:- \ Find\_pos\_green\_1999(x, y) \ . \end{aligned}$$

Chapter 4

Spatiotemporal Interpolation Methods for House Price Data in Constraint Databases

In this chapter, based on a set of 2-D space and 1-D time actual real estate data, the shape function based methods with IDW (Inverse Distance Weighting) and Kriging interpolation methods in both reduction and extension approaches will be compared.

4.1 Experimental Data

The experimental test data consisting of a set of real estate data obtained from the Lancaster county assessor's office in Lincoln, Nebraska. House sale histories since 1990 are recorded in the real estate data set and include sale prices and times.

We randomly select 126 residential houses from a quarter of a section of a township, which covers an area of 160 acres. Furthermore, from these 126 houses, we randomly select 76 houses as sample data, and the remaining 50 houses are used as test data. Figure 4.1 shows the 76 houses with circles and the 50 remaining houses

with pentagons.

Tables 4.1 and 4.2 show instances of these two data-sets. Based on the fact that the earliest sale of the houses in this neighborhood is in 1990, we encode the time in such a way that 1 represents January 1990, 2 represents February 1990, ..., 148 represents April 2002. Note that some houses are sold more than once in the past, so they have more than one tuples. For example, the house at the location $(2215, 110)$ was sold three times in the past at time 27, 77, and 114 (which represent 3/1992, 5/1996, and 6/1999).

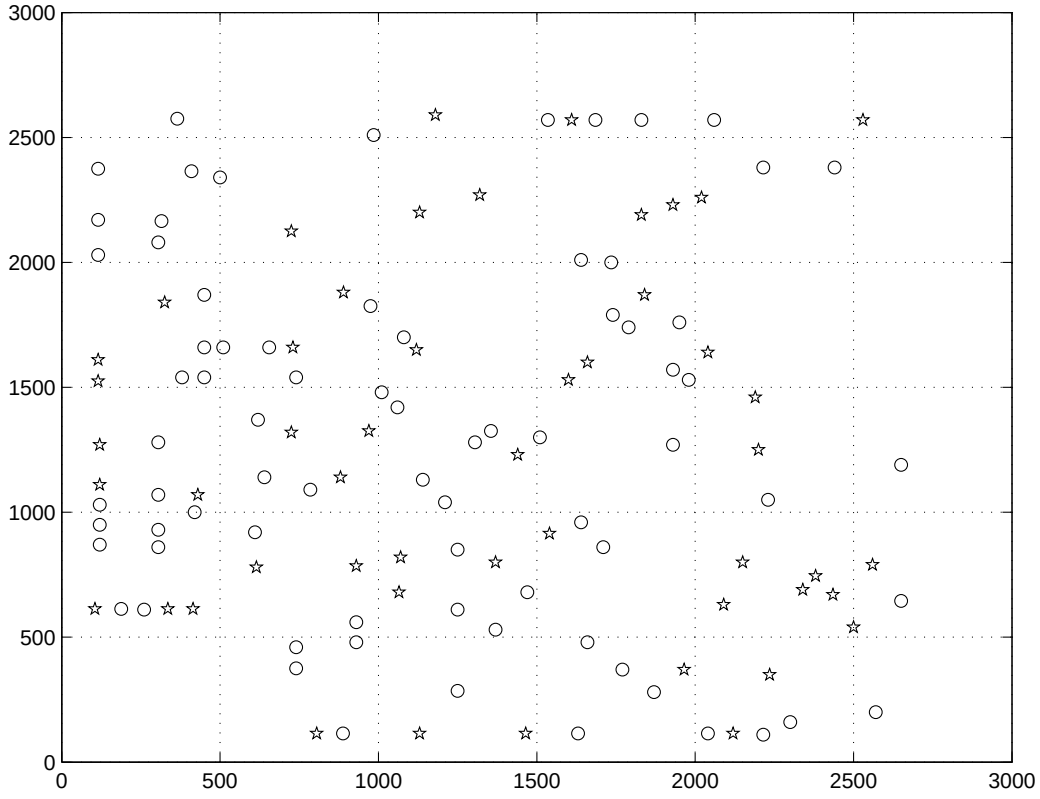

Figure 4.1: 76 sample houses ($\circ$) and 50 test houses ($\star$).

Table 4.1: Sample (x, y, t, p).

X	Y	T	P (price/square foot)
888	115	4	56.14
888	115	76	76.02
1630	115	118	86.02
1630	115	123	83.87
⋮	⋮	⋮	⋮
2240	2380	51	91.87
2650	1190	43	63.27

Table 4.2: Test (x, y, t).

X	Y	T
115	1525	16
115	1525	58
115	1525	81
115	1610	63
⋮	⋮	⋮
120	1110	30
615	780	59

4.2 Experimental Result of Shape Function Based Methods

4.2.1 Accuracy

We compare the estimated values of price per square foot with the true values for each sale instance of the 50 test houses according to Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). The definition of MAE and RMSE is as follows:

$$MAE = \frac{\sum_{i=1}^{N} \mid I_i - O_i \mid}{N} \qquad RMSE = \sqrt{\frac{\sum_{i=1}^{N} (I_i - O_i)^2}{N}}$$

Table 4.3: Comparison results.

Method		MAE	RMSE	Slope		Constraint	Invariance
				MAE	RMSE		
Reduction (ST Product)	Shape Func	8.98	11.34	9.69	13.08	polynomial	yes
	IDW(n=3, p=1)	10.05	11.96	9.49	13.62	polynomial	no
Extension (3-D)	Shape Func	7.92	10.11	0.34	0.69	linear	yes
	IDW(n=3, p=1)	11.14	13.63	0.06	0.07	polynomial	no
	Kriging	10.25	12.59	0.07	0.08	polynomial	no

where N is the number of test houses, I_i is the interpolated house price, and O_i is the original house price.

In Table 4.3, the *MAE* and *RMSE* columns summarize the accuracy analysis of the methods. We can see that the ST product method yields a slightly better accuracy (less MAE and RMSE values) than the 3-D method for shape function based interpolation.

4.2.2 Error-Proneness to Time Aggregation

The unit of time is a special issue for spatiotemporal data. For example, the following questions are of interest:

1. For a specific spatiotemporal data-set, how fine should the granularity of time be to obtain the best result of interpolation?

2. For some data-sets that only have a coarse granularity of time, what kind of spatiotemporal interpolation methods should be used?

To answer these questions, the error criteria of MAE and RMSE have been measured according to twelve different ways of time aggregation of the house price data. The twelve approaches of time aggregation include monthly, bimonthly, quarterly, ...,

yearly. That is, each month is treated as a different time instance in monthly aggregation, every two months are treated a different time instance in bimonthly aggregation, ..., each year is treated as a different time instance in yearly aggregation.

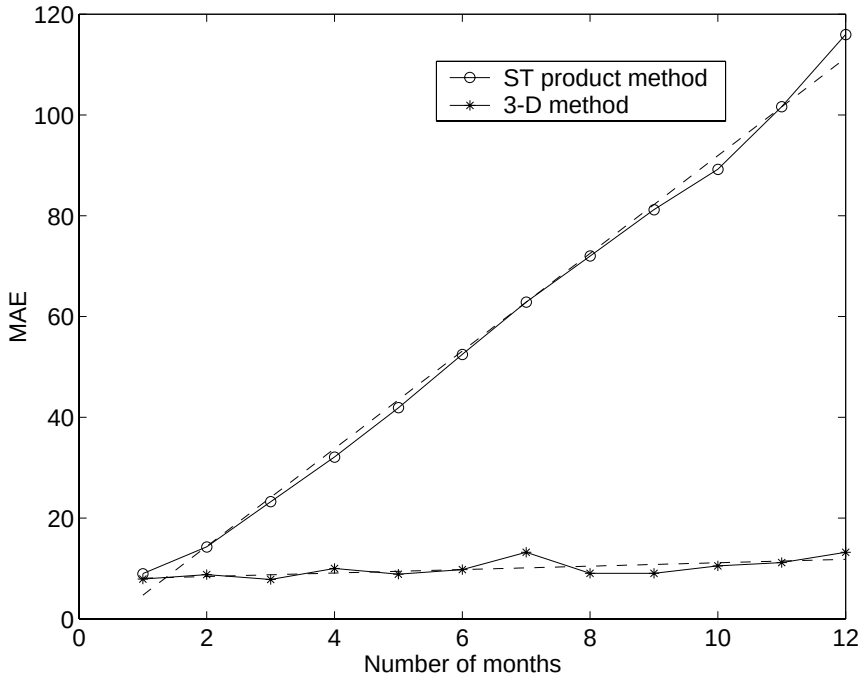

Figure 4.2: Shape function susceptibility to time aggregation according to MAE (Mean Absolute Error). The solid lines are the actual result, while the dashed lines are the linear regression functions that best approximate the tendency of MAE.

Figures 4.2 and 4.3 show the experimental results of MAE and RMSE for error proneness to time aggregation of the shape function based methods. The Matlab function *polyfit* has been used to calculate the linear regression functions. In Table 4.3, the column *Slope* summarizes their slopes. Steeper slope indicates less error-proneness to time aggregation. It is shown that the 3-D method is much less error-prone than the ST product method.

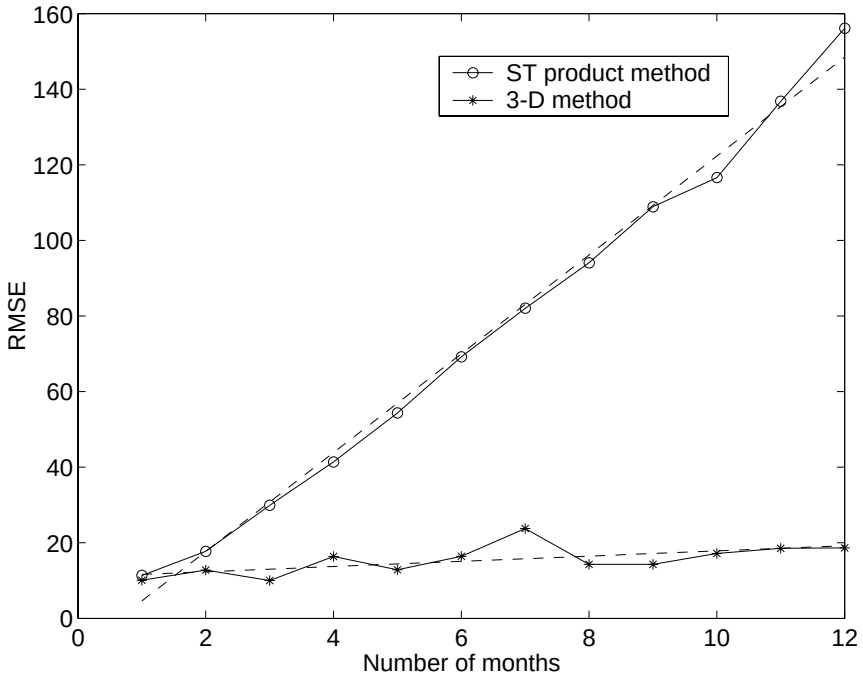

Figure 4.3: Shape function susceptibility to time aggregation according to RMSE (Root Mean Square Error). The solid lines are the actual result, while the dashed lines are the linear regression functions that best approximate the tendency of RMSE.

4.2.3 Constraint Types

For the ST product method, assume we have the following input relations after Delaunay triangulation and time approximation (which can be piecewise linear approximation) for each nodal points:

1. $Triangle(x_1, y_1, x_2, y_2, x_3, y_3, x, y)$ indicates that the point (x, y) is inside the triangle with three corner vertices (x_1, y_1), (x_2, y_2) and (x_3, y_3).

2. $Point\_Approx(x_1, y_1, t, w_1)$ stores the interpolated value w_1 for corner vertex (x_1, y_1) at time t.

The value w of any point (x, y) inside each triangle can be represented by the following polynomial constraint tuple (Li & Revesz 2002):

$$\begin{aligned} Triangle\_Approx(x, y, t, w) \ :- \ & Triangle(x_1, y_1, x_2, y_2, x_3, y_3, x, y) \ , \\ & Point\_Approx(x_1, y_1, t, w_1) \ , \\ & Point\_Approx(x_2, y_2, t, w_2) \ , \\ & Point\_Approx(x_3, y_3, t, w_3) \ , \\ & Area(x_1, y_1, x_2, y_2, x_3, y_3, a) \ , \\ & Area(x, y, x_2, y_2, x_3, y_3, a_1) \ , \\ & Area(x_1, y_1, x, y, x_3, y_3, a_2) \ , \\ & Area(x_1, y_1, x_2, y_2, x, y, a_3) \ , \\ & wa = w_1 a_1 + w_2 a_2 + w_3 a_3 \ . \end{aligned} \tag{4.1}$$

where the $Area(x_1, y_1, x_2, y_2, x_3, y_3, a)$ relation calculates the area of the triangle with vertices (x_1, y_1), (x_2, y_2) and (x_3, y_3) and can be defined as follows:

$$Area(x_1, y_1, x_2, y_2, x_3, y_3, a) : -a = \frac{\mid (x_1 - x_3)(y_2 - y_3) - (y_1 - y_3)(x_2 - x_3) \mid}{2} \ .$$

In equation (4.1), since w_i's $(1 \leq i \leq 3)$ are linear functions of t and a_i's $(1 \leq i \leq 3)$ are linear functions of x and y, $Triangle\_Approx$ is not linear, but quadratic. If we use a polynomial function of t to approximate the w_i's, we will get even higher polynomial functions for w_i's. Therefore, the constraint type of this ST product method is polynomial.

Example 4.2.1 *For the house price data, represent the interpolation by the ST product method in constraint databases.*

Table 4.4 shows the representation result. Since there are 137 triangles generated after the Delaunay triangulation, there should be 137 tuples in Table 4.4. Note that the first constraint tuple corresponds to the first 7 sample points in the *Sample(x,y,t,p)* relation in Table 4.1. In this first constraint tuple, the first three linear inequality constraints define a triangle, the next three linear constraints are derived from piecewise linear interpolation, the area values are calculated from the above *Area* Equation, and the last constraint is polynomial which calculates the interpolation value of the house price.

Table 4.4: House(x,y,t,p) interpolation by ST product method. The first constraint tuple corresponds to the first 7 sample points in the Sample(x,y,t,p) relation.

X	Y	T	P	
x	y	t	p	$5x + 1327y - 157045 \geq 0$, $x + 117y - 15085 \leq 0$, $y \leq 115$, $p_1 = 0.28(t-4) - 56$, $p_2 = -0.43(t-118) + 86$, $p_3 = 0.17(t-27) + 60$, $a = 1855$, $a_1 = 0.5 \mid 5x + 585y - 75425 \mid$, $a_2 = 0.5 \mid -5x - 1327y + 157045 \mid$, $a_3 = 0.5 \mid 742y - 85330 \mid$, $pa = p_1a_1 + p_2a_2 + p_3a_3$.
x	y	t	p	⋮ ⋮

For the 3-D method, assume we have the following input relations after tetrahedral tesselation:

1. $Tetra(x_1, y_1, t_1, x_2, y_2, t_2, x_3, y_3, t_3, x_4, y_4, t_4, x, y, t)$ indicates that the point (x, y, t) is inside the tetrahedron with four corner vertices (x_1, y_1, t_1), (x_2, y_2, t_2), (x_3, y_3, t_3) and (x_4, y_4, t_4).

2. $Point\_Value(x_1, y_1, t_1, w_1)$ stores the nodal value w_1 for corner vertex (x_1, y_1, t_1).

The value w of any point (x, y, t) inside each tetrahedron can be represented by the following linear constraint tuple (Li & Revesz 2002):

$$\begin{aligned} Tetra\_Approx(x,y,t,w) \ :- \ & Tetra(x_1,y_1,t_1,x_2,y_2,t_2,x_3,y_3,t_3,x_4,y_4,t_4,x,y,t) \ , \\ & Point\_Value(x_1,y_1,t_1,w_1) \ , \\ & Point\_Value(x_2,y_2,t_2,w_2) \ , \\ & Point\_Value(x_3,y_3,t_3,w_3) \ , \\ & Point\_Value(x_4,y_4,t_4,w_4) \ , \\ & Volume(x_1,y_1,t_1,x_2,y_2,t_2,x_3,y_3,t_3,x_4,y_4,t_4,v) \ , \\ & Volume(x,y,t,x_2,y_2,t_2,x_3,y_3,t_3,x_4,y_4,t_4,v_1) \ , \\ & Volume(x_1,y_1,t_1,x,y,t,x_3,y_3,t_3,x_4,y_4,t_4,v_2) \ , \\ & Volume(x_1,y_1,t_1,x_2,y_2,t_2,x,y,t,x_4,y_4,t_4,v_3) \ , \\ & Volume(x_1,y_1,t_1,x_2,y_2,t_2,x_3,y_3,t_3,x,y,t,v_4) \ , \\ & wv = w_1v_1 + w_2v_2 + w_3v_3 + w_4v_4 \ . \end{aligned} \tag{4.2}$$

where the $Volume(x_1, y_1, t_1, x_2, y_2, t_2, x_3, y_3, t_3, x_4, y_4, t_4, v)$ relation calculates the volume v of the tetrahedron with vertices (x_1, y_1, t_1), (x_2, y_2, t_2) , (x_3, y_3, t_3) and (x_4, y_4, t_4). It can be defined as follows:

$$\begin{aligned} & Volume(x_1,y_1,t_1,x_2,y_2,t_2,x_3,y_3,t_3,x_4,y_4,t_4,v) \ :- \ v = \\ \frac{1}{6} \ & | \ (x_1-x_2)(y_1-y_3)(t_1-t_4) + (x_1-x_4)(y_1-y_2)(t_1-t_3) \quad + \\ & (x_1-x_3)(y_1-y_4)(t_1-t_2) - (x_1-x_4)(y_1-y_3)(t_1-t_2) \quad - \\ & (x_1-x_3)(y_1-y_2)(t_1-t_4) - (x_1-x_2)(y_1-y_4)(t_1-t_3) \ | \quad . \end{aligned}$$

Since in equation (4.2), all the w_i's $(1 \leq i \leq 4)$ are constants and all the v_i's $(1 \leq i \leq 4)$ are linear in terms of x, y, t, the relation $Tetra\_Approx$ is linear. By representing the tetrahedral method in each tetrahedron with a separate constraint tuple, we can find in linear time a constraint relation to represent the whole interpolation.

Example 4.2.2 *For the house price data, represent the interpolation by the tetrahedral method in constraint databases.*

Table 4.5 shows the representation result. Since there are 733 tetrahedra generated after the tetrahedral tesselation, there should be 733 tuples in Table 4.5. Note that the first constraint tuple corresponds to the last 4 sample points in the *Sample(x,y,t,p)* relation in Table 4.1. In this first constraint tuple, the first four linear inequality constraints define a tetrahedron, and the volume values are calculated from the above *Volume* Equation.

Table 4.5: House(x,y,t,p) interpolation by the tetrahedral method. The first constraint tuple corresponds to the last 4 sample points in the Sample(x,y,t,p) relation.

X	Y	T	P	
x	y	t	p	$38160(x-2650) - 450(y-1190) - 599300(t-78) \geq 0,$ $16460(x-2650) + 9700(y-1190) - 599300(t-43) \geq 0,$ $-19950(x-2650) - 24500(y-1190) \leq 0,$ $-41650(x-2650) - 14350(y-1190) \leq 0,$ $v = 4160916.67$, $v_1 = 1/6 \mid -16460x - 7700y + 599300t + 29392100 \mid,$ $v_2 = 1/6 \mid -41650x - 7350y + 127449000 \mid,$ $v_3 = 1/6 \mid 19950x + 24500y - 82022500 \mid,$ $v_4 = 1/6 \mid 38160x - 9450y - 599300t - 53843100 \mid,$ $pv = 65.44v_1 + 70.34v_2 + 91.87v_3 + 63.27v_4$.
x	y	t	p	$\vdots$ $\vdots$

In Table 4.3, the *Constraint Type* column summarizes the type of constraints of these methods. For shape function based approaches, since the 3-D method yields

only linear constraints and the ST product yields polynomial constraints, the 3-D method has an advantage over the ST product method: query evaluation is more efficient.

4.2.4 Invariance to Coordinate Scale

Shape functions for triangles and tetrahedra are invariant to coordinate scale, which means their results will remain the same even if the scale of a dimension (or dimensions) changes. Being invariance to coordinate scale is a very charming characteristic of a spatiotemporal interpolation method especially when we want to use the extension approach. This is because we don't have to worry about what time unit should be used when mixing the space and time dimension. In Table 4.3, the column *Invariance* summarizes whether the method is invariant to coordinate scale. We prove that 2-D triangular shape functions are invariant to scale in below. The proof for the invariance of 3-D tetrahedral shape functions can be similarly obtained.

Proof 4.2.1 *2-D triangular shape functions are invariance to coordinate scale.*

Consider $N_1(x, y)$ in the triangular shape function (2.2). After substituting the determinant result of $\mathcal{A}$, we have

$$N_1(x,y) = \frac{[(x_2y_3 - x_3y_2) + x(y_2 - y_3) + y(x_3 - x_2)]}{x_2y_3 - y_2x_3 - x_1y_3 + y_1x_3 + x_1y_2 - y_1x_2} .$$

Assume that the scale in x dimension enlarges to n times of the original scale. Then N_1 will be as follows after scaling

$$N_1'(x,y) = \frac{[(nx_2y_3 - nx_3y_2) + nx(y_2 - y_3) + y(nx_3 - nx_2)]}{nx_2y_3 - ny_2x_3 - nx_1y_3 + ny_1x_3 + nx_1y_2 - ny_1x_2} ,$$

which is obviously the same result as before scaling. Invariance to y scale is straightforward too. Similarly, we can prove that N_2 and N_3 are also invariant to coordinate

scale.

4.3 Experimental Result of IDW Based Methods

4.3.1 Accuracy

From the *MAE* and *RMSE* columns in Table 4.3, we can see that as different from the shape function based methods, the 3-D method yields a slightly better accuracy (less MAE and RMSE values) than the ST product method for IDW based interpolation.

4.3.2 Error-Proneness to Time Aggregation

Similarly to the analysis of shape function based methods, we test the same twelve ways of time aggregation for the IDW based methods.

Figures 4.4 and 4.5 show the experimental results of MAE and RMSE for error proneness to time aggregation of the IDW based methods when the number of near neighbors is 3. Again, the Matlab function *polyfit* has been used to calculate the linear regression functions. From the the column *Slope* in Table 4.3, we can see that similarly to the shape function based methods, the 3-D method is much less error-prone than the ST product method for IDW based approaches.

4.3.3 Constraint Types

The constraint type for both the IDW based ST product and 3-D methods are polynomial.

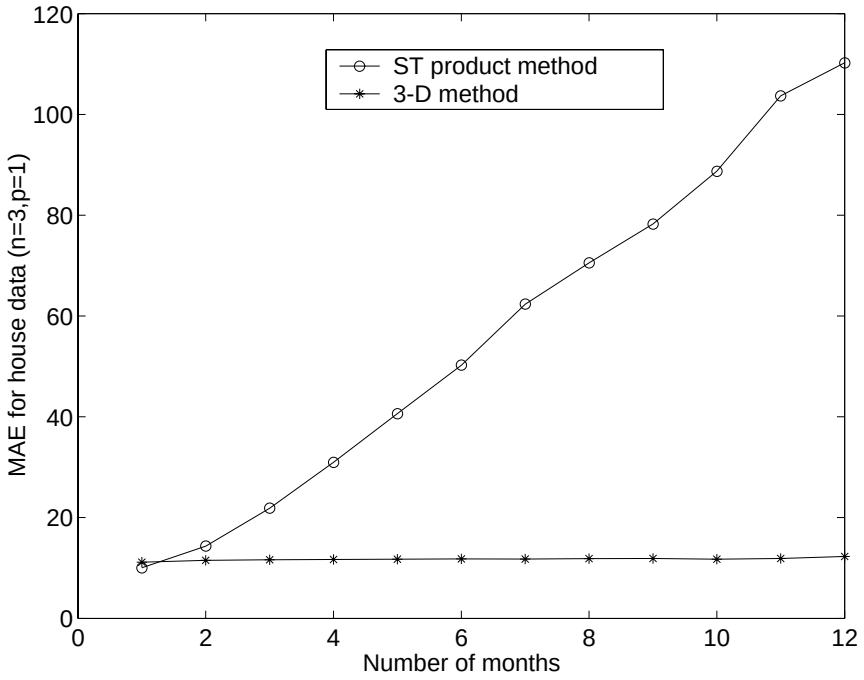

Figure 4.4: IDW susceptibility to time aggregation according to MAE (Mean Absolute Error). The solid lines are the actual result, while the dashed lines are the linear regression functions that best approximate the tendency of MAE.

4.3.4 Not Invariance to Coordinate Scale

IDW is not invariant to coordinate scale. Consider the IDW interpolation with 2 neighbors and power 2, based on equation (2.21), we have

$$\lambda_1 = \frac{(x - x_2)^2 + (y - y_2)^2}{(x - x_1)^2 + (y - y_1)^2 + (x - x_2)^2 + (y - y_2)^2} \ .$$

Assume that the x dimensional scale enlarges to n times. Then after scaling, λ_1 will be

$$\lambda_1' = \frac{n^2(x - x_2)^2 + (y - y_2)^2}{n^2(x - x_1)^2 + (y - y_1)^2 + n^2(x - x_2)^2 + (y - y_2)^2} \ ,$$

which is not the same result as before scaling. Therefore, IDW is not invariant to coordinate scale.

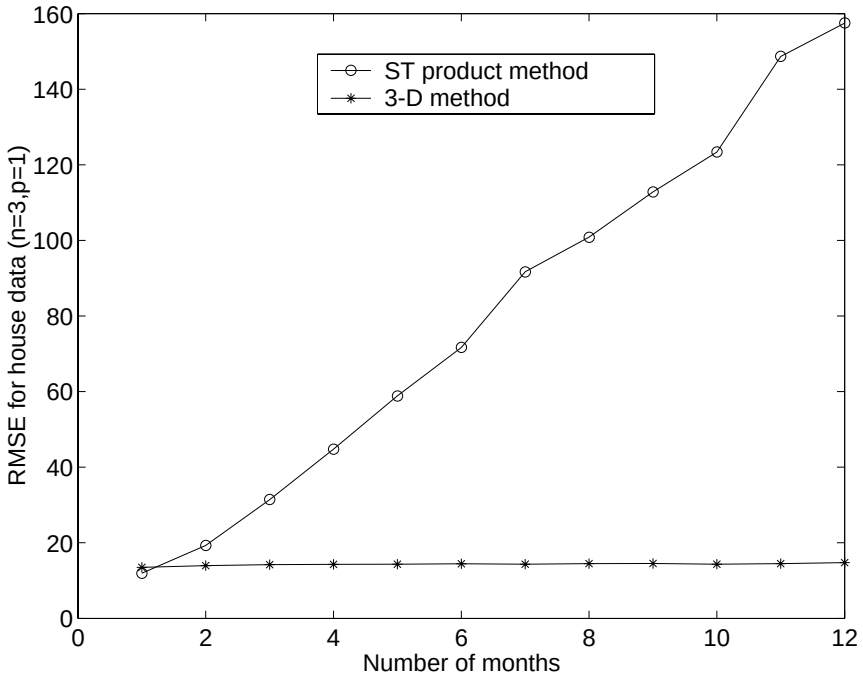

Figure 4.5: IDW susceptibility to time aggregation according to RMSE (Root Mean Square Error). The solid lines are the actual result, while the dashed lines are the linear regression functions that best approximate the tendency of RMSE.

4.4 Experimental Result of Kriging Based Methods

We use the *Matlab Kriging Toolbox (version 4.0)* provided by Gratton to do the experiments. It is available from http://www.inrs-eau.uquebec.ca/ activites/repertoire/ yves_gratton/krig.htm. This toolbox is almost entirely made up of functions from Deutsch & Journel (1998) and Marcotte (1991). It actually implemented high dimensional Cokriging with Matlab. Cokriging is the multi-variable extension of Kriging. It means Kriging with more than one variables. When the Cokriging program is called with only one variable, it will return the Kriging result. Since we have only one variable, the house price, we only need Kriging.

Since the reduction approach of Kriging for spatiotemporal interpolation is not

a feasible approach (see Section 2.3), the following comparison is referring to the extension approach (i.e. 3-D Method) of Kriging.

4.4.1 Accuracy

With the search radius be 500, the number of nearest neighbors be 10, and some other default input parameters for *point cokriging*, we have tested several choices of variogram models. The result of linear model with nugget effect has been the best. We put the result of this model into Table 4.3. The MAE and RMSE values of Kriging based 3-D method are slightly better than IDW based 3-D method. But they are worse than shape function based both ST product and 3-D methods.

4.4.2 Error-Proneness to Time Aggregation

Similarly to the analysis of shape function and IDW based methods, we test the same twelve ways of time aggregation for Kriging based 3-D method. Figure 4.6 shows the experimental results of both MAE and RMSE. From the column *Slope* in Table 4.3, we can see that Kriging based 3-D method is not error-prone.

4.4.3 Constraint Types

The constraint type for Kriging based 3-D method is polynomial since the calculation of variograms by equation (2.25) is already quadratic.

4.4.4 Not Invariance to Coordinate Scale

Since Kriging is similar to IDW in the weighting mechanism that is influenced by distances, Kriging is also not invariant to coordinate scale.

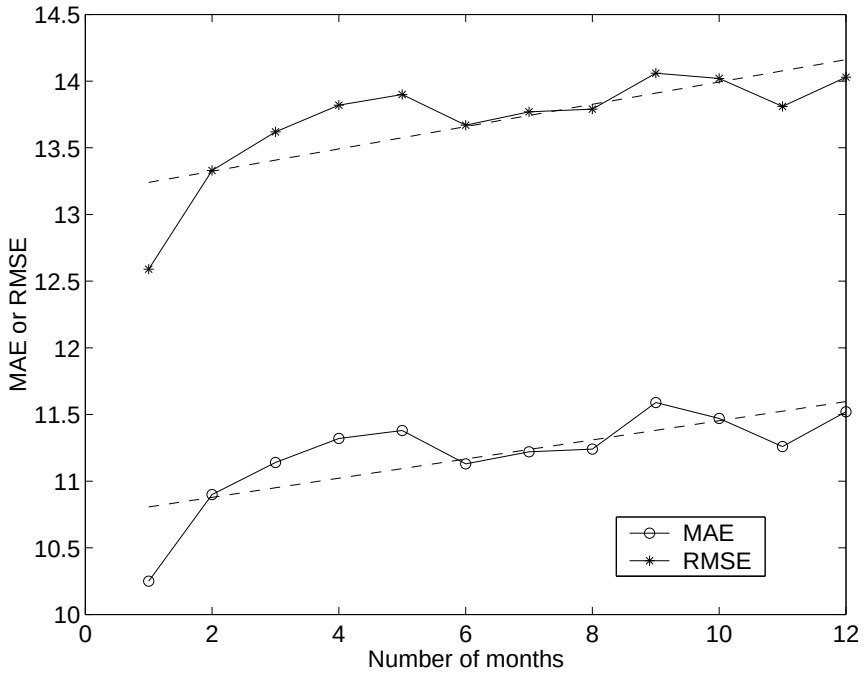

Figure 4.6: Kriging susceptibility to time aggregation according to MAE (Mean Absolute Error) and RMSE (Root Mean Square Error). The solid lines are the actual result of MAE and RMSE, while the dashed lines are the linear regression functions that best approximate their tendency.

4.5 Comparison Summary

Table 4.3 shows that the extension method with shape functions is the most accurate spatiotemporal interpolation method as measured by mean absolute error (MAE) and root mean square error (RMSE). It is also the only one which can be represented using linear constraints.

The extension method, which treats time as another dimension, has a potential problem, namely that there is no easy way to compare one temporal unit with one spatial unit. Depending on the unit measure, we may get a different value for the estimated results. Are there spatiotemporal interpolations that are invariant with respect to the choice of units in the spatial and temporal axes? The column *Invariance* in Table 4.3 shows that only shape functions-based spatiotemporal interpolation are

invariant.

Finally, in the real estate data instead of recording the precise date of sale of houses we may have only records of monthly, bimonthly or even yearly sales, that is, all the houses sold in that time interval are listed together. The column *Slope* in Table 4.3 shows experimentally that this time aggregation has a serious negative effect on the accuracy of the reduction method.

4.6 Visualization of Shape Function Interpolation Result

4.6.1 Shape Function Reduction Approach: ST Product Method

The spatiotemporal interpolation result from this approach can be visualized in a 2-D display at different time instances. We illustrate the visualization result using the same set of real estate data as described in Section 4.1.

For the color plot, six basic colors are chosen: red, yellow, green, turquoise, blue, and purple. The 24-bit RGB values for these colors are the following: red = (255, 0, 0), yellow = (255, 255, 0), green = (0, 255, 0), turquoise = (0, 255, 255), blue = (0, 0, 255), purple = (255, 0, 255). The colors are used to represent interpolated values. The following two versions of color rendering have been used in the program implementation:

Version 1: Use of 400 Smoothly Changing Colors In this version, a 1-D linear shape function interpolation scheme is used between each pair of the basic colors. Five simple linear interpolations are chosen for the color changes between red and yellow, yellow and green, green and turquoise, turquoise and blue, blue and purple. This version yields a smooth change of colors in the visualization, hence

it avoids sharp color transitions. We give an example of one color interpolation below.

Example 4.6.1 *Suppose that between* red$(255, 0, 0)$ *and* yellow$(255, 255, 0)$*, we use 80 intermediate colors of the form* $(255, G, 0)$*. Here the possible values of* G *can be found using the following linear function:*

$$G = (1 - \frac{x}{80}) * StartGValue + (\frac{x}{80}) * EndGValue \ ,$$

where $x \in [0, 80]$*.*

In this example between **red** and **yellow** we have $StartGValue = 0$ and $EndGValue = 255$. For other intervals, the values of $StartGValue$ and $EndGValue$ can be changed accordingly.

In Figure 4.7, the graphical output for the presentation of measured house price data using smoothly changing colors is illustrated.

Version 2: Use of 6 Colors In this version, only the six basic colors are used in the plots. The color red is assigned for the smallest function value and the color purple is assigned for the largest value. Each color covers 1/6 of the total range of values for the house price/square foot. This version results in visualizations that show distinct boundaries between colors. Although this version seems to have less information than the first color rendering version, for users this may be more convenient in categorizing house price differences. Actually, this version can be considered as an extreme case of the previous version with *no* intermediate colors.

Figure 4.8 shows the graphical output for the measured house price data using 6 colors.

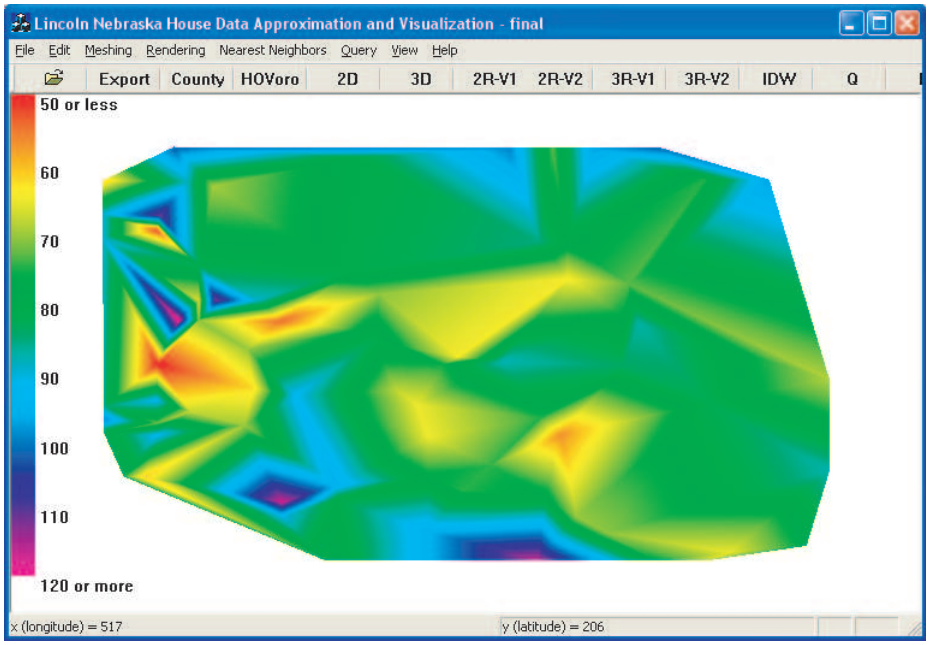

Figure 4.7: Shape function reduction approach visualization version 1: continuous color rendering for the house price data of Lincoln, Nebraska in October 1995.

4.6.2 Shape Function Extension Approach: 3-D Method

The spatiotemporal interpolation result from this approach can be visualized in a vertical profile display. Using the same real estate data example as in Section 4.1, the graphical output from this extension approach is illustrated in Figures 4.9. The three slices in the figure corresponds to house price visualizations at three time instances: August 1991, October 1995 and December 1999. They are obtained by intersecting three horizontal time planes with the tetrahedral mesh of the 76 sample houses. Note that after tetrahedral meshing, each slice has different coverage of area. Figure 4.9 was produced by Matlab 6.0.

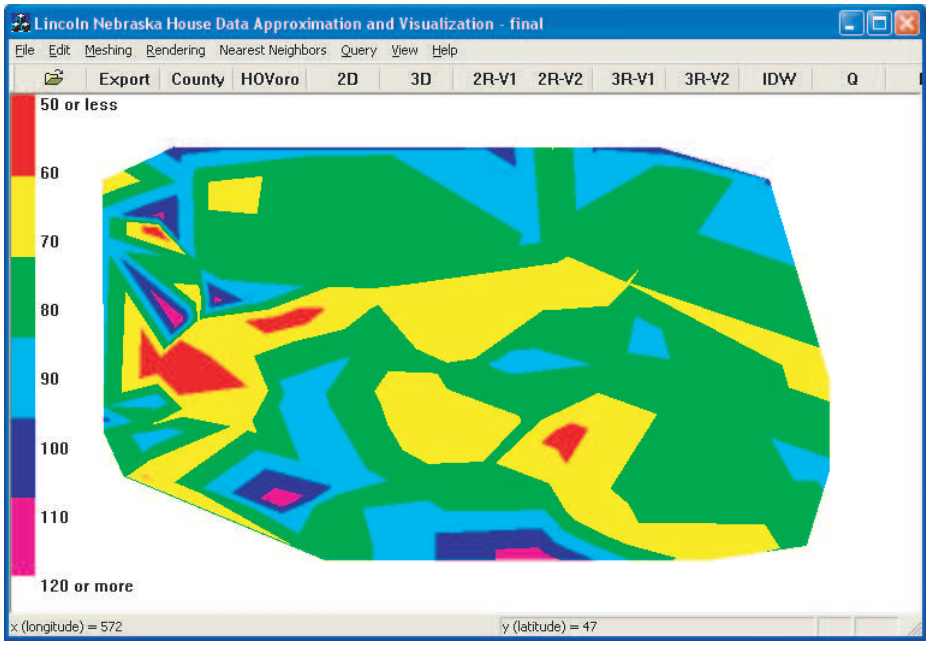

Figure 4.8: Shape function reduction approach visualization 2: rendering with 6 discrete colors for the house price data of Lincoln, Nebraska in October 1995.

4.7 4-D Shape Function Example for 3-D Space and 1-D Time Problems

We implemented the 4-D shape functions in Section 2.1.2.2 by Matlab. We also extended the 2-D space and 1-D time real estate example to a 3-D space and 1-D time problem by adding the elevation information to each house as shown in Tables 4.6 and 4.7.

We used our Matlab program to interpolate the 4-D test data and compared it with the original values according to MAE and RMSE. The result of MAE is 8.54 and the result of RMSE is 10.25. These results are slightly worse than the 3-D shape functions methods in Table 4.3. This can be explained by the fact that the elevations of those houses in the selected test area are similar and the house elevation is not a factor to contribute to house prices. Therefore, it adds noisy to the interpolation by considering the house elevation.

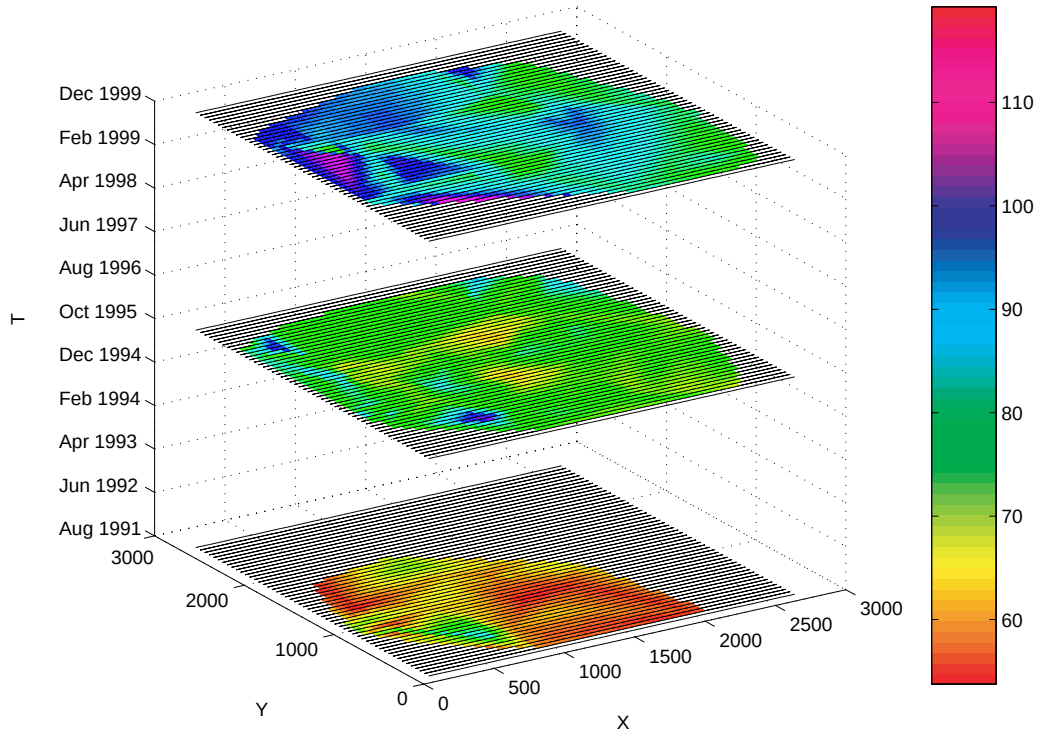

Figure 4.9: Shape function extension approach visualization: vertical profile of house price data of Lincoln, Nebraska in August 1991, October 1995 and December 1999.

4.8 Query Examples

In this section, we give some sample queries. We assume that the input constraint relations are *House(x,y,t,p)* and *Built (x,y,t)*. *House(x,y,t,p)* represents the interpolation result of house price data, and *Built (x,y,t)* records the time t (in month) when the house at location (x, y) was built. The *Built* relation can be usually easily obtained from real estate or city planning agencies.

We write queries in the Datalog query language, which is a rule based-language used in several constraint database systems (Kanellakis et al. 1995, Kuper et al. 2000, Revesz 2002). Next we give a few sample queries:

Query 4.8.1 *For each house, find the starting sale price when the house was built.*

Table 4.6: Sample_4D (x, y, z, t, p).

X	Y	Z	T	P (price/square foot)
888	115	1305	4	56.14
888	115	1305	76	76.02
1630	115	1294	118	86.02
1630	115	1294	123	83.87
⋮	⋮	⋮	⋮	⋮
2240	2380	1295	51	91.87
2650	1190	1288	43	63.27

Table 4.7: Test_4D (x, y, z, t).

X	Y	Z	T
115	1525	1294	16
115	1525	1294	58
115	1525	1294	81
115	1610	1293	63
⋮	⋮	⋮	⋮
120	1110	1300	30
615	780	1306	59

This can be expressed in Datalog as follows:

$$\begin{aligned} Start\_price(x, y, p) \ :- \ & Built(x, y, t), \\ & House(x, y, t, p). \end{aligned}$$

Query 4.8.2 *Suppose that we know house prices in general decline for some time after the first sale. For each house, find the first month when it become profitable, that is, the first month when its price exceeded its initial sale price.*

This can be expressed as follows:

$$\begin{aligned} not\_Profitable(x,y,t) &:- Built(x,y,t). \\ not\_Profitable(x,y,t_2) &:- not\_Profitable(x,y,t_1), \\ &\quad House(x,y,t_2,p_2), \\ &\quad Start\_price(x,y,p), \\ &\quad t_2 = t_1 + 1,\ p_2 \leq p. \end{aligned}$$

$$\begin{aligned} Profitable(x,y,t_2) &:- not\_Profitable(x,y,t_1), \\ &\quad House(x,y,t_2,p_2), \\ &\quad Start\_price(x,y,p), \\ &\quad t_2 = t_1 + 1,\ p_2 > p. \end{aligned}$$

Query 4.8.3 *How many months did it take for each house to become profitable?*

This translates as:

$$\begin{aligned} Time\_to\_Profit(x,y,t_3) &:- Built(x,y,t_1), \\ &\quad Profitable(x,y,t_2), \\ &\quad t_3 = t_2 - t_1. \end{aligned}$$

All of the above queries could be a part of a more complex data mining or decision support task. For example, a buyer may want to find out which builders tend to build houses that become profitable in a short time or best keep their values. We do not illustrate such an analysis in more details, because that would lead too far from the main GIS issues. It is enough to see that constraint database systems provide a flexible, yet powerful rule-based query language to use in building a full decision

support system.

Chapter 5

Implementation

5.1 MLPQ System

MLPQ (Management of Linear Programming Queries) system is a constraint database system for rational linear constraint databases (Revesz & Li 1997, Kanjamala, Revesz & Wang 1998, Revesz 2002). This system has a graphic user interface (GUI) which support Datalog-based and icon-based queries, as well as visualization and animations. One of the main application areas of the MLPQ system is managing GIS spatial and spatiotemporal data. It can outdo the popular ArcGIS system (Johnston et al. 2001) by powerful queries (such as join and recursive queries) and the ability to animate spatiotemporal data.

The MLPQ system has been kept improving in the past several years. For example, Syed (2002) did a good job to enhance the MLPQ system by adding the SQL query ability into the system, a color-animation algorithm that allows visualization of the changing value of each cell in map displays, and a translation algorithm between ArcGIS shape files and MLPQ input data files.

5.2 Data Translation between ArcGIS System and MLPQ System

This is a joint work with Mohiuddin Syed (Syed 2002) and Andrew Rutlege at GIS Works Inc. A translation algorithm between ArcGIS shape files and MLPQ input data files is implemented. ArcGIS is an integrated system for geographic data creation, management, integration, and analysis. It is developed by ESRI, the world leading vendor for GIS softwares such as ArcView and ArcInfo. ArcGIS is a new system which can include ArcView and ArcInfo packages. Since ArcGIS system is a very popular GIS software, it is important to convert the data from ArcGIS to MLPQ and vice versa. Our work focused on the conversion between ArcGIS shape data and MLPQ input data files.

The Arc shape data describes the region information. Shape data is set of three files: (i) a shape file which contains vector information of the regions, (ii) an index file, and (iii) a dBase file containing information about the region. During the translation from Arc shape files to MLPQ input data files, these three files are first converted into an intermediate form (Li 2001), and then from the intermediate form to the MLPQ input data form. In the intermediate form, each region is represented as a polygon with coordinates in acyclic form, appended with respective information from the dBase file. Each line in the intermediate file represents a polygon or cell, terminated by a semi-colon (";") except that the last line which is terminated by period (".."). This intermediate file is then converted in to MLPQ input data format.

The translation from Arc shape format to the intermediate format is done under the help of Andy Rutledge at the GISWORKSHOP. The rest conversion job is done by Mohiuddin Syed. The details of this algorithm for this translation can be found in Syed (2002).

5.3 Visualization of Spatiotemporal Data in MLPQ

5.3.1 Point-based and Region-based Spatiotemporal Data Visualization

Point-based and region-based spatial-temporal data can be visualized by animated histograms. For point-based data, each bar in a histogram represents a point and the bar height is animated with time according to the measured value at the point. For region-based data, each bar represents a region and the height of the bar changing with time according to the measured value of the region. For example, for Example 1.2.2 in Chapter 1, we can visualize the total corn yield animation in 24 counties in Nebraska from 1947 to 2000. Figures 5.1 and 5.2 show two snapshots during the animation.

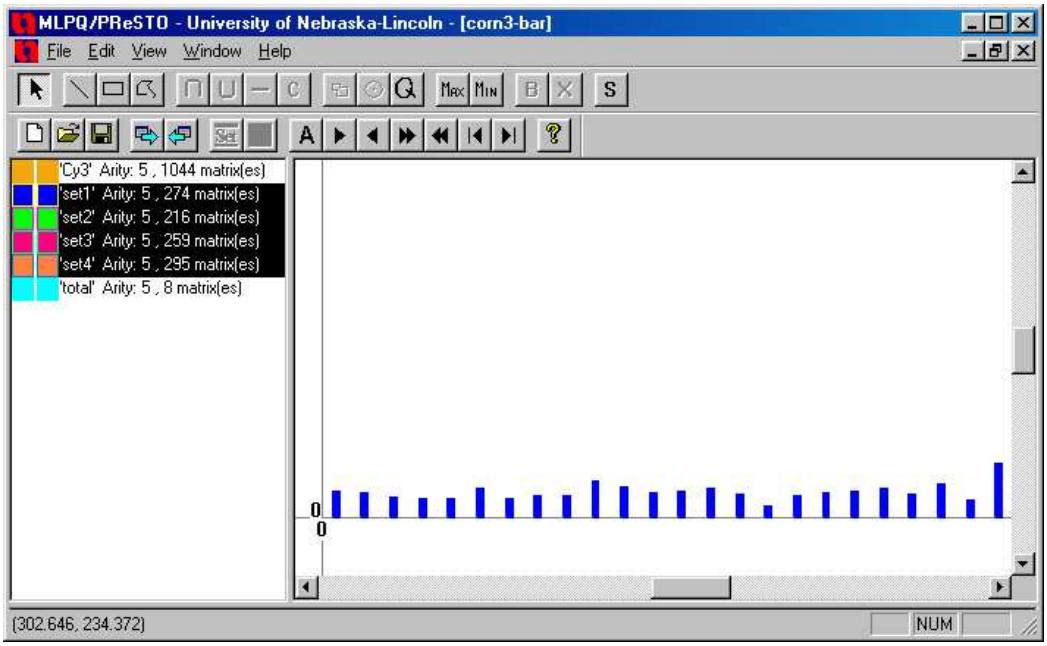

Figure 5.1: The total yield of corn in 1947.

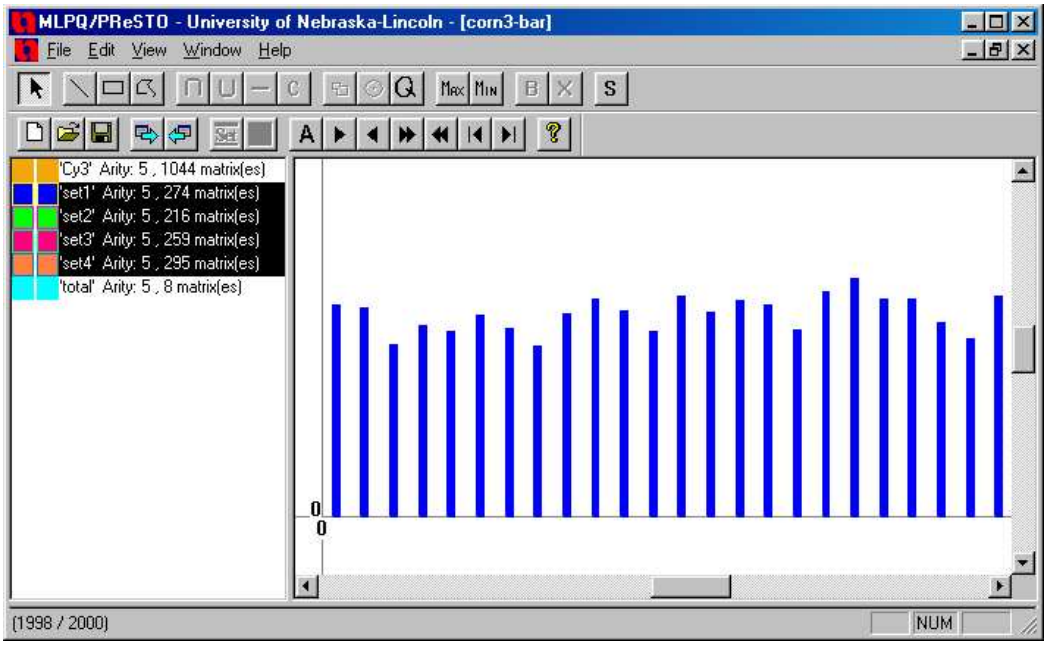

Figure 5.2: The total yield of corn in 1998.

5.3.2 Constraint-based Spatiotemporal Data Visualization

Since constraint-based spatiotemporal data can be either created from point-based or region-based spatiotemporal databases as shown by edges a and c in Figure 1.1 in Chapter 1, we discuss constraint-based spatiotemporal data visualization in two categories: (i) translated from point-based data and (ii) translated from region-based data.

5.3.2.1 Translated from Point-Based Data

In order to translate spatiotemporal data from point-based STDBs into constraint-based STDBs, we need to use some spatiotemporal interpolation methods. Choosing proper spatiotemporal interpolation methods will lead to more accurate representation of the data.

For example, we can use shape functions to interpolate each of the 12 snapshots of the monthly SPI data in 1992 in Nebraska. Since this is a 2-D spatial problem, the

corresponding shape function is based on triangulation of the sample weather station locations. For each triangles, we will have a shape function to approximate SPI values of all the inside points. A sample shape function for a triangle is given as follows:

```
test(id, x, y, w)  :-  id = 0,
                       -13.39x + y >= 9929075.15,
                       -1.67x + y <= 3187523.83,
                       -1.02x + y >= 2759839.18,
                       151.50x - 137.53y + 10000000w = -385625717.32.
```

5.3.2.2 Translated from Region-Based Data

We use animated color maps to visualize constraint data translated from region-based spatiotemporal databases. In such a map, each region is represented by a polygon with a certain color. The color of a polygon represents the measured value of the region and it changes during the animation.

Voronoi Regions For Example 1.2.3 in Chapter 1, we can use animated color maps to visualize those Voronoi diagrams. The color map animation has been implemented in MLPQ. Users only need to push the color animation button and input the following three parameters: the beginning time instance, ending time instance and step size. Then the color of each region of the map will be animated according to its value at a specific time instance. Figure 5.3 shows the 116 2nd-order Voronoi diagram for the 48 weather stations in Nebraska at the snapshot when $t = 1992$.

Other Regions Besides using Voronoi regions, there are many other types of regions. For example, to visualize the total corn yield animation in each county in Nebraska during a given period, we can use county-based regions. First of all, we need to represent each county polygon in MLPQ. Since such county vector data in

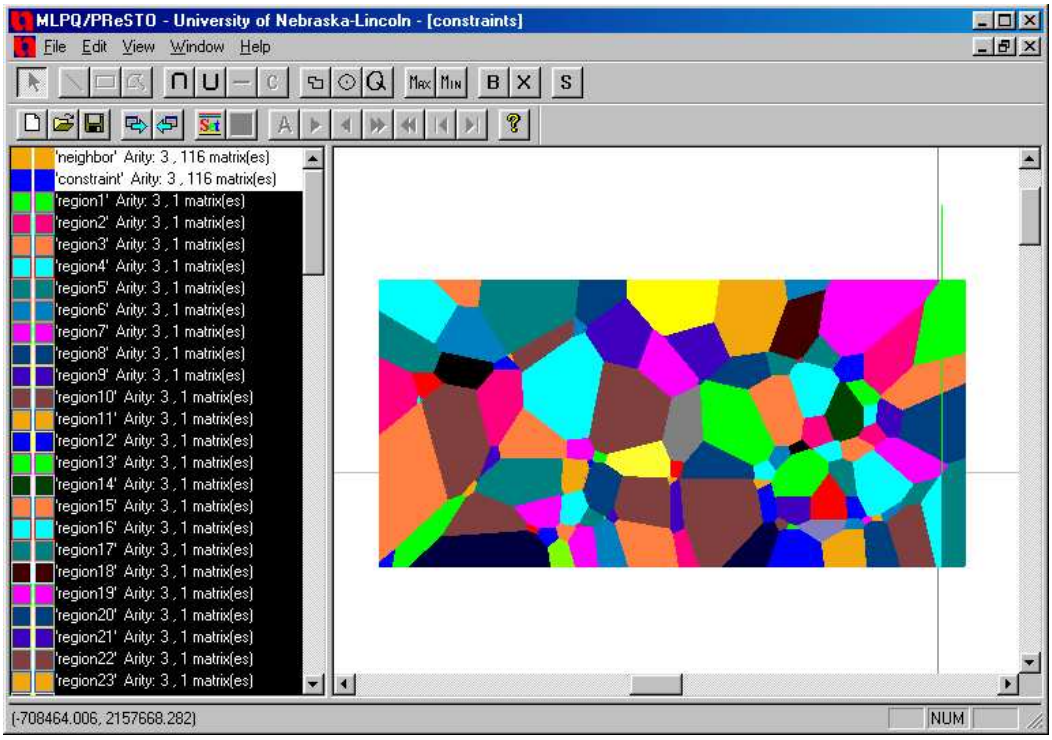

Figure 5.3: The 2rd order Voronoi diagram for 48 weather stations in Nebraska, which consists of 116 regions.

US are usually available in ArcView shape file format, we did a program to convert ArcView shape files to MLPQ input text files. The conversion from MLPQ files to shape files was also implemented. With such two-way conversion programs to transfer data between ArcView and MLPQ, on one hand, we can explore the special features of MLPQ such as querying on real data sets from ArcView databases; on the other hand, newly generated data in MLPQ can be translated back to ArcView data format. Figures 5.4 and 5.5 show two snapshots during the color map animation when $t = 1947$ and $t = 1998$.

5.4 Query in MLPQ/PReSTO

Query 5.4.1 *Find those counties which will be more productive (more yield) than county 109 (Lancaster county) in irrigated corn in 2010.*

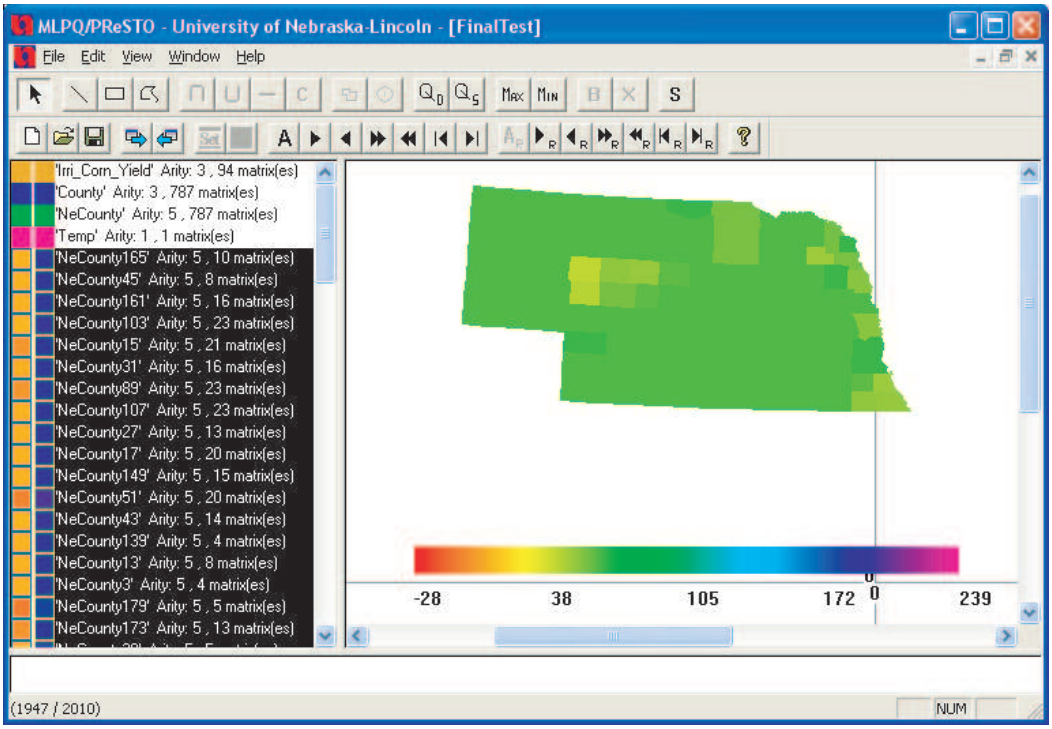

Figure 5.4: A Snapshot of Color Map Animation for County-based Corn Yield in Nebraska when t=1947.

This is an unusual query since 2010 is a future instance and its information does not exist in any traditional databases. However, constraint databases are a natural approach to this problem. We can use constraint databases and store the NASS corn data as in Table 1.5. Since the attribute *yield*, as *acres* and *production*, is stored as a linear regression function for each county, we can estimate easily the corn yield of a county at any time (even in the future) by substituting the right t value into the linear regression function. This query can be expressed in SQL as follows:

```
Select C2.county From Corn_Constraint as C1, Corn_Constraint as C2
Where C1.practice = 'irrigated' and C2.practice = 'irrigated'
      and C1.year = 2010 and C2.year = 2010
      and C1.county = 109 and C2.yield > C.yield.
```

Alternatively, this can be expressed in Datalog as:

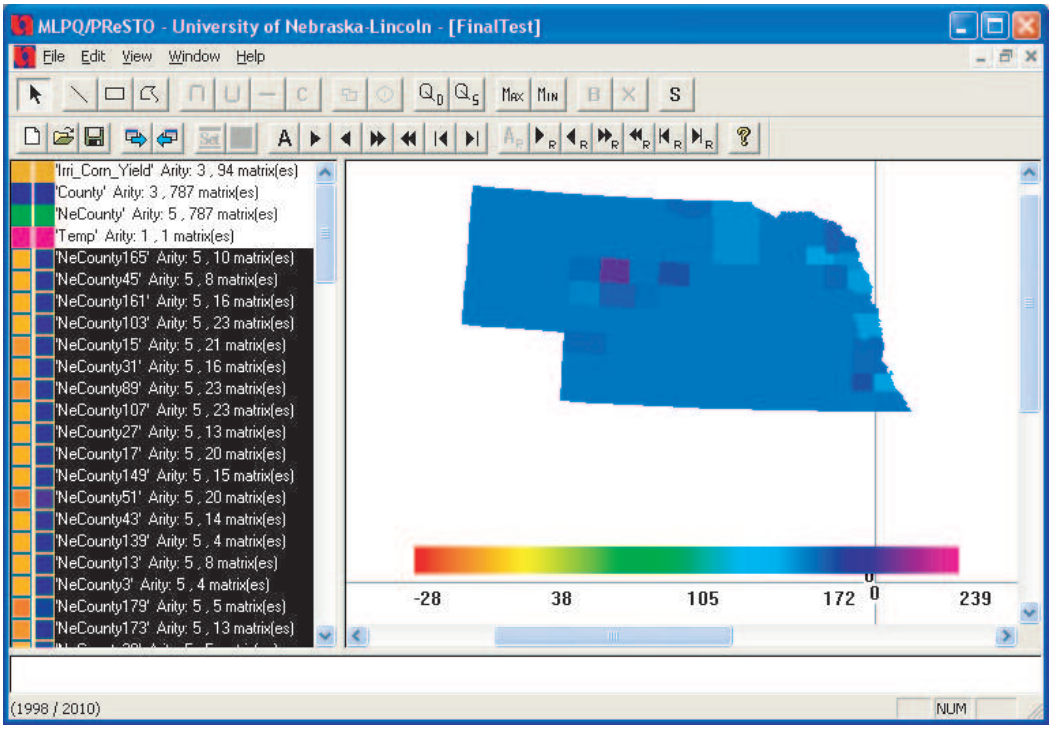

Figure 5.5: A Snapshot of Color Map Animation for County-based Corn Yield in Nebraska when t=1998.

```
Productive(county2) :- Corn_Constraint(109, 2010, 'irrigated',
                                       arce1, yield1, production1),
                       Corn_Constraint(county2, 2010, 'irrigated',
                                       arce2, yield2, production2),
                       yield2 > yield1.
```

Query 5.4.2 *Assume there is fire in an area of a county. How will the irrigated corn production be affected by the fire in this county?*

Region-based Exposure Analysis This approach assumes uniform production for all the locations in each county. Suppose a county has 1 unit of irrigated corn production. This county is of a rectangular shape and the fire covers also a rectangular area as shown in Figure 5.6. Since the area of this county is $6 \times 3 = 18$ units, the irrigated corn production of the whole county without considering the corn loss by

fire can be calculated by the volume of a cube with base area 18 and height 1, which is $18 \times 1 = 18$. The production lost by fire can be calculated by the volume of a cube with base area $2 \times 1 = 2$ and height 1, which is $2 \times 1 = 2$. Therefore, the adjusted irrigated corn production considering fire is $18 - 2 = 16$ by this approach.

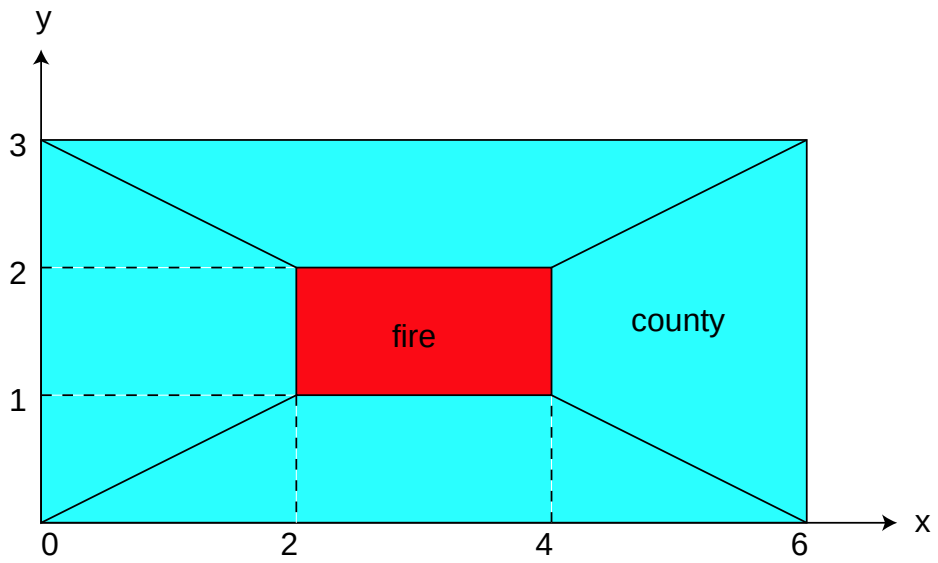

Figure 5.6: Region-based exposure analysis.

Constraint-based Exposure Analysis This approach does not assume that all the locations in a county have the same production. Instead, it uses a variable to represent the corn production. It is a more realistic approach since different locations in a county should not always have the same production because of many factors, such as different soil conditions. For example, rocky areas or poorly drained soils have low capability to produce corn; but areas with soil pH value in the 6.0 to 7.0 range have high capability for corn production since they have optimum benefits from applied fertilizer and herbicides (MWPS-45 2000). Suppose that the irrigated corn production of the county without considering the corn loss by fire is also 18 units, which is the same as in the *Region-based Exposure Analysis*, and the fire happens in the area with the highest production as shown in Figure 5.7. The production lost by

fire can be calculated by the sum of two volumes of a pyramid and a cube, which is shown in red in Figure 5.7, where the fire intersects with the county. As a result, the fire volume is $2/3 + 4 = 14/3$. Therefore, the adjusted irrigated corn production considering fire is $18 - 14/3 = 40/3$ by this approach.

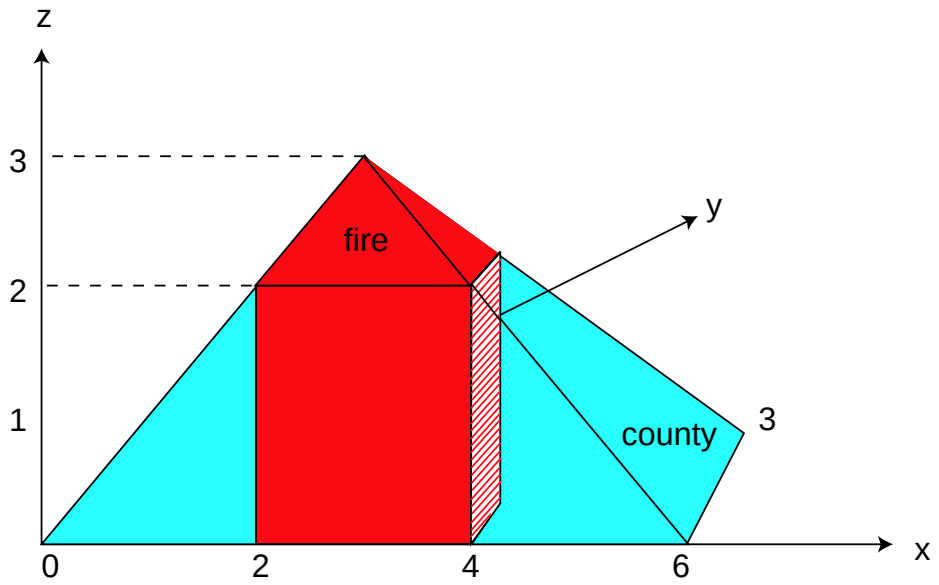

Figure 5.7: Constraint-based exposure analysis.

We can see that these two approach yield different results. Since constraint-based exposure analysis is a more realistic approach, its result is more accurate than the region-based exposure analysis. For the constraint-based exposure analysis, if the *volume* aggregate operator (Revesz 2002) is implemented, which takes three variables x, y and z as arguments and calculates the volume of the domain defined by x, y and z, this query can be expressed in Datalog as follows:

```
Adjusted(volume<x, y, z>) :- County(x, y, z), not Fire(x, y, z).
```

Chapter 6

Conclusion and Future Work

This dissertation discusses the shape function-based spatiotemporal interpolation methods: the reduction and the extension methods. Based on a set of actual real estate data, the shape function based methods are compared with the IDW and the kriging methods. Our experimental results show that the extension method based on shape functions is the most accurate and the overall best spatiotemporal interpolation method.

For the extension method based on shape functions, the resulting spatiotemporal interpolation data can be represented using linear equality and inequality constraints. While there are many ways of storing this representation, *constraint databases* (Kanellakis et al. 1995, Kuper et al. 2000, Revesz 2002) are a convenient alternative. Linear constraint databases such as the DEDALE system (Grumbach, Rigaux & Segoufin 2000) and the MLPQ system – see Chapter 18 in (Revesz 2002) – are particularly natural for this type of interpolated data. The advantages of using MLPQ include compact data storage, convenient database querying, and the availability of a number of built-in visualization tools, including some for spatiotemporal animation.

In constraint databases, the details of the interpolation are at a lower level, that is, hidden from the users. This results in a high level data abstraction, which makes

querying and visualization easier for the users.

The work discussed in this dissertation can be extended into different directions. First of all, polynomial constraints should be implemented in the future. This will allow applications of IDW and kriging interpolation in constraint databases. The representation of IDW and kriging using polynomial constraints is feasible in theory. For example, Revesz & Li (2002*a*) discussed the representation of IDW interpolation in polynomial constraint databases. Kriging is similar to IDW with the difference that the weights are derived using error statistics of the data. It is also representable in constraint databases if the variogram, or the statistically derived function of weight and distance, is representable using constraints. If we take some (distance, weight) samples from the variogram, then we get data in form of a time series, which can be interpolated and translated into a linear constraint relation using the algorithm in Revesz et al. (2001).

Second, more data sets should be used to test shape function-based spatiotemporal interpolation methods. It seems that the main characteristic of the house price data is that it is dense in space but sparse in time. That is, the houses are close together but are sold only infrequently. More tests are needed to evaluate how the density in space and time affect the accuracy of each interpolation method. It may be that for input data with opposite characteristics, that is, sparse in space but dense in time, the reduction method could be better than the extension (i.e. tetrahedral) method.

Last, the visualization and animation of spatiotemporal interpolation should be improved. For example, a "Volume" function should be added in the MLPQ system. This will contribute to queries that need to calculate volumes, see the constraint-based exposure analysis approach in Query 5.4.2. Also, for 3-D space and 1-D time problems, 3-D animation should be implemented to display a series of 3-D snapshots during the visualization.

Bibliography

Buchanan, G. R. (1995), *Finite Element Analysis*, McGraw-Hill, New York.

Demers, M. N. (2000), *Fundamentals of Geographic Information Systems*, 2nd edn, John Wiley & Sons, New York.

Deutsch, C. V. & Journel, A. G. (1998), *GSLIB: Geostatistical Software Library and User's Guide*, 2nd edn, Oxford University Press, New York.

Freitag, L. A. & Gooch, C. O. (1997), 'Tetrahedral mesh improvement using swapping and smoothing', *International Journal for Numerical Methods in Engineering* **40**, 3979–4002.

Goodman, J. E. & O'Rourke, J., eds (1997), *Handbook of Discrete and Computational Geometry*, CRC Press, Boca Raton, New York.

Grumbach, S., Rigaux, P. & Segoufin, L. (2000), Manipulating interpolated data is easier than you thought, *in* 'Proc. IEEE International Conference on Very Large Databases', pp. 156–165.

Harbaugh, J. W. & Preston, F. W. (1968), *Fourier Analysis in Geology*, Prentice-Hall, Englewood Cliffs, pp. 218–238.

Huebner, K. H. (1975), *The Finite Element Method for Engineers*, John Wiley and Sons, pp. 117–118.

Jaffar, J. & Lassez, J. L. (1987), Constraint logic programming, *in* 'Proc. 14th ACM Symposium on Principles of Programming Languages', pp. 111–119.

Johnston, K., Hoef, J. M. V., Krivoruchko, K. & Lucas, N. (2001), *Using ArcGIS Geostatistical Analyst*, ESRI Press.

Kanellakis, P. C., Kuper, G. M. & Revesz, P. (1990), Constraint query languages, *in* 'Proc. ACM Symposium on Principles of Database Systems', pp. 299–313.

Kanellakis, P. C., Kuper, G. M. & Revesz, P. (1995), 'Constraint query languages', *Journal of Computer and System Sciences* **51**(1), 26–52.

Kanjamala, P., Revesz, P. & Wang, Y. (1998), MLPQ/GIS: A GIS using linear constraint databases, *in* C. S. R. Prabhu, ed., 'Proc. 9th COMAD International Conference on Management of Data', pp. 389–393.

Krige, D. G. (1951), A statistical approach to some mine valuations and allied problems at the witwatersrand, Master's thesis, University of Witwatersrand, South Africa.

Kuper, G. M., Libkin, L. & Paredaens, J., eds (2000), *Constraint Databases*, Springer-Verlag.

Lam, N. S. (1983), 'Spatial interpolation methods: A review', *The American Cartographer* **10**(2), 129–149.

Langran, G. (1992), *Time in Geographic Information Systems*, Taylor and Francis, London.

Li, L. (2001), Exporting file format for arc coverages or export data, Technical report, University of Nebraska-Lincoln.

Li, L. & Revesz, P. (2001), Visualization for temporally changing 3d data, *in* 'Proc. of the 3rd Annual Midwest Meeting of the American Association for the Advancement of Science, Southwestern and Rocky Mountain Division', Vol. 41, American Association for the Advancement of Science, p. 21.

Li, L. & Revesz, P. (2002), A comparison of spatio-temporal interpolation methods, *in* M. Egenhofer & D. Mark, eds, 'Proc. of the Second International Conference on GIScience 2002', Vol. 2478 of *Lecture Notes in Computer Science*, Springer, pp. 145–160.

Li, L. & Revesz, P. (2003 in press), 'Interpolation methods for spatio-temporal geographic data', *Journal of Computers, Environment and Urban Systems* .

Li, L. & Revesz, P. (2003 to appear), The relationship among gis-oriented spatiotemporal databases, *in* 'Proc. of the Third National Conference on Digital Government Research', Boston.

Longley, P. A., Goodchild, M. F., Maguire, D. J. & Rhind, D. W. (2001), *Geographic Information Systems and Science*, John Wiley, Chichester.

Marcotte, D. (1991), 'Cokriging with matlab', *Computer & Geosciences* **17**(9), 1265–1280.

Matheron, G. (1971), 'The theory of regionalized variables and its applications', *Les Cahiers du Centre de Morphologie Mathematique de Fontainebleau* **5**, 221 pp.

Miller, E. J. (1997), Towards a 4D GIS: Four-dimensional interpolation utilizing kriging, *in* Z. Kemp, ed., 'Innovations in GIS 4: Selected Papers from the Fourth National Conference on GIS Research U.K, Ch. 13', Taylor & Francis, London, pp. 181–197.

MWPS-45 (2000), *Conservation Tillage Systems and Management*, 2nd edn, Midwest Plan Service.

Oliver, M. A. & Webster, R. (1990), 'Kriging: A method of interpolation for geographical information systems', *International Journal of Geographical Information Systems* **4**(3), 313–332.

Preparata, F. P. & Shamos, M. I. (1985), *Computational Geometry: An Introduction*, Springer-Verlag.

Revesz, P. (2002), *Introduction to Constraint Databases*, Springer, New York.

Revesz, P., Chen, R. & Ouyang, M. (2001), Approximate query evaluation using linear constraint databases, *in* 'Proc. Symposium on Temporal Representation and Reasoning', Cividale del Friuli, Italy, pp. 170–175.

Revesz, P. & Li, L. (2002*a*), Constraint-based visualization of spatial interpolation data, *in* 'Proc. of the Sixth International Conference on Information Visualization', IEEE Press, London, England, pp. 563–569.

Revesz, P. & Li, L. (2002*b*), Representation and querying of interpolation data in constraint databases, *in* 'Proc. of the Second National Conference on Digital Government Research', Los Angeles, California, pp. 225–228.

Revesz, P. & Li, L. (2003 to appear), *Advances in Geometric Modeling*, John Wiley, chapter Constraint-Based Visualization of Spatiotemporal Databases.

Revesz, P. & Li, Y. (1997), MLPQ: A linear constraint database system with aggregate operators, *in* 'Proc. 1st International Database Engineering and Applications Symposium', IEEE Press, pp. 132–137.

Shepard, D. (1968), A two-dimensional interpolation function for irregularly spaced data, *in* 'Proc. 23nd National Conference ACM', ACM, pp. 517–524.

Shewchuk, J. R. (1996), Triangle: Engineering a 2D quality mesh generator and delaunay triangulator, *in* 'Proc. First Workshop on Applied Computational Geometry', Philadelphia, Pennsylvania, pp. 124–133.

Shewchuk, J. R. (1998), Tetrahedral mesh generation by delaunay refinement, *in* 'Proc. 14th Annual ACM Symposium on Computational Geometry', Minneapolis, Minnesota, pp. 86–95.

Syed, M. (2002), Enhancement of MLPQ/PReSTO constraint database system, Master's thesis, University of Nebraska-Lincoln.

Watson, D. F. (1981), 'Computing the n-dimensional delaunay tesselation with application to voronoi polytopes', *The Computer Journal* **24**(2), 167–172.

Worboys, M. F. (1995), *GIS: A Computing Perspective*, Taylor & Francis.

Yang, W., Yang, L. & Merchant, J. (1997), 'An assessment of AVHRR/NDVI-ecoclimatological relationships in Nebraska, USA', *International Journal of Remote Sensing* **18**, 2161–2180.

Zienkiewics, O. C. & Taylor, R. L. (1989), *Finite Element Method, Vol. 1, The Basic Formulation and Linear Problems*, McGraw-Hill.

Zienkiewics, O. C. & Taylor, R. L. (2000), *Finite Element Method, Vol. 1, The Basis*, Butterworth Heinemann, London.

Zurflueh, E. G. (1967), 'Applications of two-dimensional linear wavelength filtering', *Geophysics* **32**, 1015–1035.