PERGAMON

Computers, Environment and Urban Systems
28 (2004) 201–227

Computers, Environment and Urban Systems

www.elsevier.com/locate/compenvurbsys

Interpolation methods for spatio-temporal geographic data

Lixin Li, Peter Revesz*

Computer Science and Engineering Department, University of Nebraska-Lincoln, Lincoln, NE 68588, USA

Abstract

We consider spatio-temporal interpolation of geographic data using both the reduction method, which treats time as an independent dimension, and the extension method, which treats time as equivalent to a spatial dimension. We adopt both 2-D and 3-D shape functions from finite element methods for the spatio-temporal interpolation of 2-D spatial and 1-D temporal data sets. We also develop new 4-D shape functions and use them for the spatio-temporal interpolation of 3-D spatial and 1-D temporal data sets. Using an actual real estate data set with house prices, we compare these methods with other spatio-temporal interpolation methods based on inverse distance weighting and kriging. The comparison criteria include 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 spatio-temporal interpolation method. New color rendering algorithms are also developed for the visualization of time slices of the interpolated spatio-temporal data. We show some visualization results of the real estate data set including the vertical profile of house prices.

Keywords: Spatio-temporal interpolation; Shape functions; Constraint databases

1. Introduction

Geographic information system (GIS) applications often require *spatio-temporal interpolation* of an input data set. Spatio-temporal 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

* Corresponding author. Tel.: +1-402-472-3488; fax: +1-402-472-7767.
E-mail addresses: revesz@cse.unl.edu (P. Revesz), lli@cse.unl.edu (L. Li).

0198-9715/03/$ - see front matter
doi:10.1016/S0198-9715(03)00018-8

at different weather stations at different instances of time. Then spatio-temporal 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 and Journel (1998) *kriging*, Goodman and O'Rourke (1997) *splines*, Zurflueh (1967) *trend surfaces*, and Harbaugh and 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 spatio-temporal interpolation in GIS. In fact, we could only find papers in spatio-temporal 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, chapter 13), which utilizes kriging for spatio-temporal interpolation.

Most GIS researchers assume that spatio-temporal 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 the Monday noon temperature on each Monday. 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.

The outline of our paper and our main contributions are the following.

(1) In Section 2 we give a literature review about using 2-D and 3-D shape functions to approach spatial interpolation problems.
(2) In Section 3 we start by describing two general methods for spatio-temporal interpolations. The *reduction method* treats time independently from the spatial dimensions. The *extension method* treats time as equivalent to a spatial dimension.
(3) In Section 3.1 we consider 2-D space and 1-D time spatio-temporal interpolation. We illustrate the reduction approach using a combination of 2-D shape functions for space and 1-D shape functions for time (Section 3.1.1). We also illustrate the extension approach using 3-D shape functions where the first two dimensions are for space and the third dimension is for time (Section 3.1.2). In both cases we consider visualization of the spatio-temporal interpolation. For the reduction method we give a new color rendering scheme which utilizes 1-D shape functions.
(4) In Section 3.2 we consider 3-D space and 1-D time spatio-temporal interpolation. For the reduction method we use the combination of 3-D shape functions for space and 1-D shape functions for time (Section 3.2.1). For the extension method we first divide the 4-D domain by a 4-D Delaunay Tesselation (see Section 2.3). Then we develop new 4-D (Section 3.2.2) shape functions that can be applied for each 4-D Delaunay Tesselation element.

(5) Section 4 compares our interpolation methods with the inverse distance weighting and kriging methods based on the same actual real estate data. We show that the extension method with shape functions is the most accurate spatio-temporal 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 spatio-temporal interpolations that are invariant with respect to the choice of units in the spatial and temporal axes? We show that only shape functions-based spatio-temporal 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. We show experimentally that this time aggregation has a serious negative effect on the accuracy of the reduction method, while the extension method is barely affected.
(6) In Section 5 we give an example of using 4-D shape functions by considering an extension of the real estate data where the height of each house is also recorded.
(7) Finally, in Section 6 we discuss some future work.

2. Literature review

In this section, we give a literature review about 2-D and 3-D shape functions as well as 4-D Delaunay tesselation. Shape functions, which can be viewed as a spatial interpolation method, are popular in engineering applications, for example, in finite element algorithms (Buchanan, 1995; Zienkiewics & Taylor, 2000). There are various types of 2-D and 3-D shape functions. In this section, we are only interested in 2-D shape functions for triangles and 3-D shape functions for tetrahedra, both of which are linear approximation methods.

2.1. 2-D shape functions for triangles

2.1.1. Triangular meshes

When dealing with complex 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 in the case of 2-D problems. 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). We embedded in our system the Delauney triangulation algorithm available from the public Website http://www.geom.umn.edu/software/~qhull and used this for one of our spatio-temporal data approximation methods which will be described in Section 3.1.1.

2.1.2. Linear approximation in 2-D space

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 Fig. 1, two triangular finite elements, I and II, are combined to cover the whole domain considered.

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

$$w(x, y) = N_1(x, y)w_1 + N_2(x, y)w_2 + N_3(x, y)w_3 = [N_1 N_2 N_3]\begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} \tag{1}$$

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

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

The area $\mathcal{A}$ of element II in Eq. (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

$$\mathcal{A} = \frac{1}{2}\det\begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix}. \tag{3}$$

It should be noted that for every sub-domain, a local approximation function similar to expression (1) is used. Each local approximation function is constrained to the local triangular sub-domain. For example, the function w of Eq. (1) is valid only

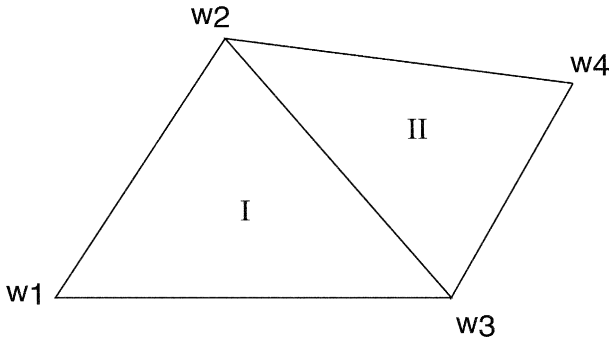

Fig. 1. Linear interpolation in space for triangular elements.

for sub-domain I. For sub-domain II, the local approximation takes a similar form as the expression (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.

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

$$N_1(x, y) = \frac{\mathcal{A}_1}{\mathcal{A}}, N_2(x, y) = \frac{\mathcal{A}_2}{\mathcal{A}}, N_3(x, y) = \frac{\mathcal{A}_3}{\mathcal{A}}, \tag{4}$$

where $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$ are the three sub-triangle areas of sub-domain I as shown in Fig. 2, and $\mathcal{A}$ is the area of the outside triangle $w_1w_2w_3$ which can be computed by Eq. (3). All the $\mathcal{A}_i$s ($1 \leqslant i \leqslant 3$) can also be computed similarly to Eq. (3) by using the appropriate coordinate values.

2.2. 3-D shape functions for tetrahedra

2.2.1. Tetrahedral meshes

Three-dimensional domains can be divided into a 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. Fig. 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*.

There exist several methods to generate automatic tetrahedral meshes, such as the 3-D Delaunay tetrahedrilization and some tetrahedral mesh improvement methods to avoid poorly shaped tetrahedra. For example, the tetrahedral mesh generation by Delaunay refinement (Shewchuk, 1998) and tetrahedral mesh improvement using swapping and smoothing (Freitag & Gooch, 1997).

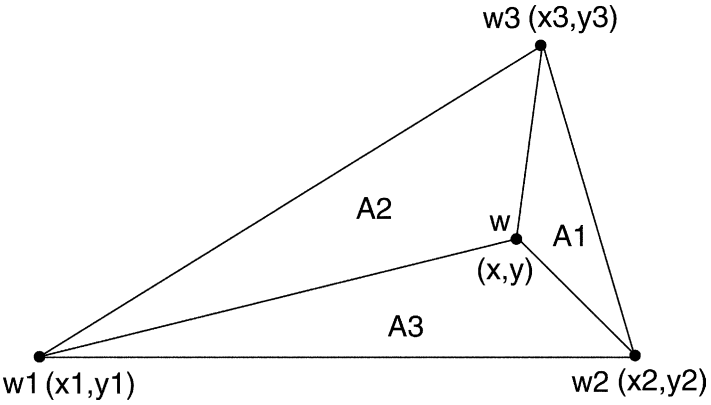

Fig. 2. Computing shape functions by area divisions.

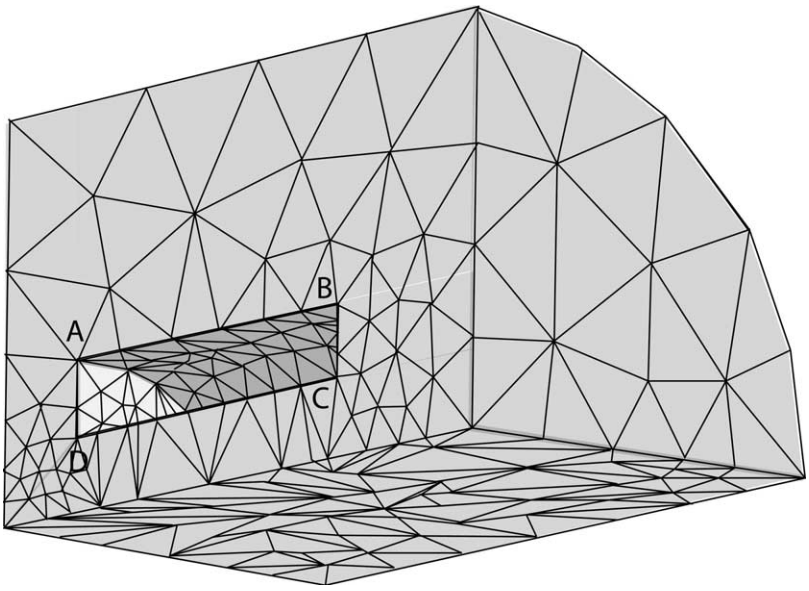

Fig. 3. A tetrahedral mesh.

2.2.2. Linear approximation in 3-D space

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 Fig. 4, two tetrahedral elements, I and II, cover the whole domain considered.

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 element I, we can use the four corner values w_1, w_2, w_3 and w_4. Similarly, all function values for element II can be constructed using the corner values w_1, w_3, w_4 and w_5. The linear interpolation function for element I can be written as:

$$w(x,y,z) = N_1(x,y,z)w_1 + N_2(x,y,z)w_2 + N_3(x,y,z)w_3 + N_4(x,y,z)w_4$$

$$= [N_1 N_2 N_3 N_4] \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} \tag{5}$$

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

$$N_1(x,y,z) = \frac{a_1 + b_1 x + c_1 y + d_1 z}{6\mathcal{V}}, N_2(x,y,z) = \frac{a_2 + b_2 x + c_2 y + d_2 z}{6\mathcal{V}},$$
$$N_3(x,y,z) = \frac{a_3 + b_3 x + c_3 y + d_3 z}{6\mathcal{V}}, N_4(x,y,z) = \frac{a_4 + b_4 x + c_4 y + d_4 z}{6\mathcal{V}}. \tag{6}$$

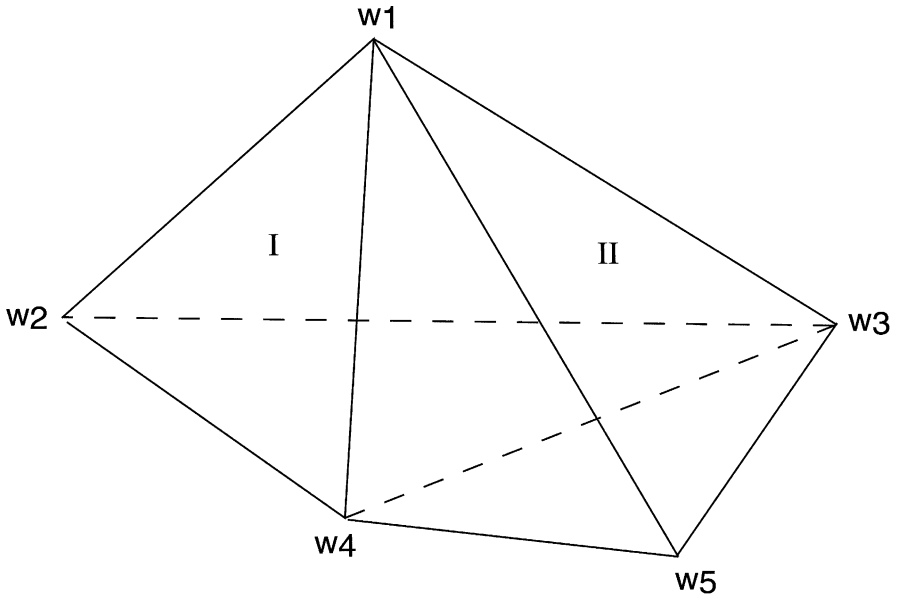

Fig. 4. Linear interpolation in space for tetrahedral elements.

The volume $\mathcal{V}$ of the tetrahedron used for the shape functions in (6) 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

$$\mathcal{V} = \frac{1}{6}\det\begin{bmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{bmatrix}. \tag{7}$$

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

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

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

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 element I, the 3-D shape function (6) can also be expressed as follows:

$$N_1(x, y, z) = \frac{\mathcal{V}_1}{\mathcal{V}}, N_2(x, y, z) = \frac{\mathcal{V}_2}{\mathcal{V}}, N_3(x, y, z) = \frac{\mathcal{V}_3}{\mathcal{V}}, N_4(x, y, z) = \frac{\mathcal{V}_4}{\mathcal{V}} \tag{8}$$

$\mathcal{V}_1$, $\mathcal{V}_2$, $\mathcal{V}_3$ 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 Fig. 5; and $\mathcal{V}$ is the volume of the outside tetrahedron $w_1w_2w_3w_4$ which can be computed by Eq. (7). All the $\mathcal{V}_i$s ($1 \leqslant i \leqslant 4$) can also be computed similarly to Eq. (7) by using the appropriate coordinate values.

2.3. 4-D Delaunay tesselation

The Delaunay tesselation in 4-D space is a special case of 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 compute 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 need to give the *delaunayn* function proper $\mathbf{X}$ array with size $m \times 4$. An example of a 4-D Delaunay tesselation by Matlab is given later.

Example 2.1. Assume $\mathbf{X}$ is an array that contains seven 4-D points ($m=7$, $n=4$) as follows:

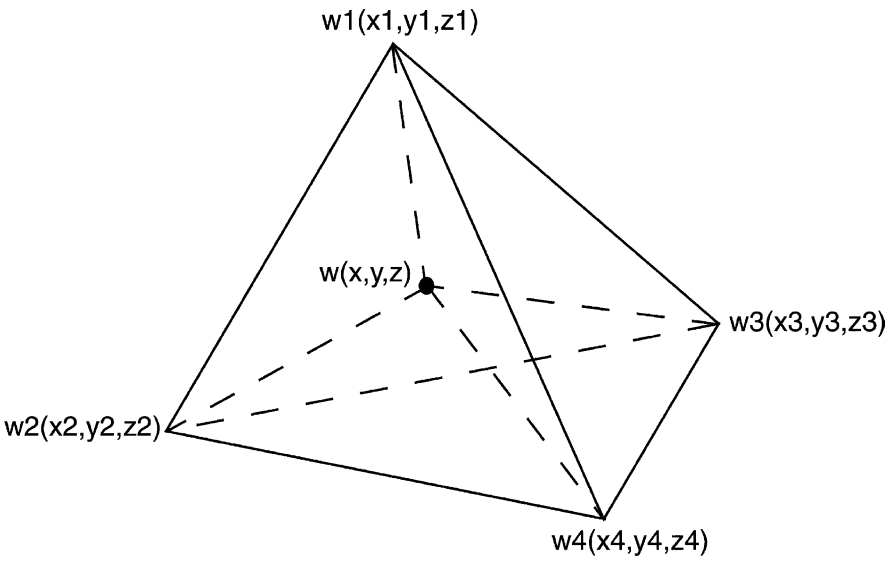

Fig. 5. Computing shape functions by volume divisions.

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

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

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

3. New approaches to spatio-temporal interpolation

There are two fundamentally different ways for spatio-temporal interpolation: reduction and extension (Li & Revesz, 2002) These models can be descrbed briefly as follows:

Reduction. This approach reduces the spatio-temporal interpolation problem to a regular spatial interpolation case. First, we interpolate (using any 1-D interpolation in time) the measured value over time at each sample point. Then by substituting the desired time instant into some regular spatial interpolation functions, we can get spatio-temporal interpolation results.
Extension. This approach deals with time as another dimension in space and extends the spatio-temporal interpolation problem into a one-higher dimensional spatial interpolation problem.

In this section, we discuss new approaches to spatio-temporal interpolation by shape function based reduction and extension methods for 2-D space and 1-D time and 3-D space and 1-D time problems.

3.1. 2-D space and 1-D time

3.1.1. 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 (Section 2.1) and approximation in space and time (Section 3.1.1.1). 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 paper an ST product method which combines 2-D shape function in space and 1-D shape function in time.

3.1.1.1. 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 (1).

Assume the value at 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:

$$w_i(t) = \frac{t_2 - t}{t_2 - t_1} w_{i1} + \frac{t - t_1}{t_2 - t_1} w_{i2}. \tag{9}$$

Using the example shown in Fig. 1 and utilizing formulas (1) and (9), the approximation function for any point constraint to element I at any time between t_1 and t_2 can be expressed as follows (Li & Revesz, 2002)

$$\begin{aligned} w(x, y, t) &= N_1(x, y)\left[\frac{t_2 - t}{t_2 - t_1} w_{11} + \frac{t - t_1}{t_2 - t_1} w_{12}\right] \\ &\quad + N_2(x, y)\left[\frac{t_2 - t}{t_2 - t_1} w_{21} + \frac{t - t_1}{t_2 - t_1} w_{22}\right] \\ &\quad + N_3(x, y)\left[\frac{t_2 - t}{t_2 - t_1} w_{31} + \frac{t - t_1}{t_2 - t_1} w_{32}\right] \\ &= \frac{t_2 - t}{t_2 - t_1}[N_1(x, y)w_{11} + N_2(x, y)w_{21} + N_3(x, y)w_{31}] \\ &\quad + \frac{t - t_1}{t_2 - t_1}[N_1(x, y)w_{12} + N_2(x, y)w_{22} + N_3(x, y)w_{32}]. \end{aligned} \tag{10}$$

Since the space shape functions (N_1, N_2 and N_3) and the time shape functions (9) are linear, the spatio-temporal approximation function (10) is not linear but quadratic.

3.1.1.2. Visualization. The spatio-temporal interpolation result from this approach can be visualized in a 2-D display at different time instances. We illustrate the visualization result using 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 (60%) houses as sample data, and the remaining 50 (40%) houses are used as test

data. Tables 1 and 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 were sold more than once in the past, so the sales corresponds to different tuples. For example, the house at the location (2215, 110) was sold at times 27, 77, and 114 (which represent 3/1992, 5/1996, and 6/1999).

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 are used in the program implementation:

Version 1: Use of 400 Smoothly Changing Colors. 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 later.

Example 3.1. 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:

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

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.

Version 2: Use of six Colors. 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

Table 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 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

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.

In Figs. 6 and 7, the graphical output for the presentation of measured house price data is illustrated.

3.1.2. Extension approach: 3-D method

This method treats *time* as a regular third dimension. Since it extends 2-D problems to 3-D problems, this method is very similar to the linear approximation by 3-D shape functions for tetrahedra (Section 2.2). The only modification is to substitute variable z in Eqs. (5)–(8) by the time variable t.

3.1.2.1. Visualization. The spatio-temporal interpolation result from this approach can be visualized in a vertical profile display. Using the same real estate data example as in Section 3.1.1, the graphical output from this extension approach is illustrated in Fig. 8. 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 a different coverage of area. Fig. 8 was produced by Matlab 6.0.

3.2. 3-D space and 1-D time

In this section, we discuss the shape function based reduction and extension approaches for 3-D space and 1-D time spatio-temporal problems.

3.2.1. Reduction approach: ST product method

This shape function based reduction data approximation in 3-D space and 1-D time can be described in the following two steps: 3-D spatial interpolation by shape functions for tetrahedra (Section 2.2) and approximation in space and time.

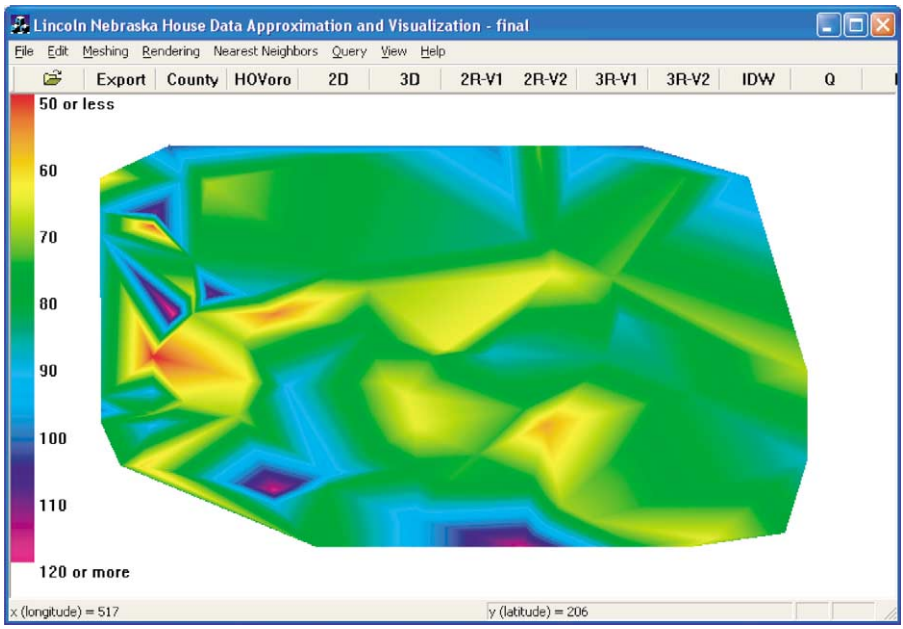

Fig. 6. Version 1: continuous color rendering for the house price data of Lincoln, Nebraska in October 1995.

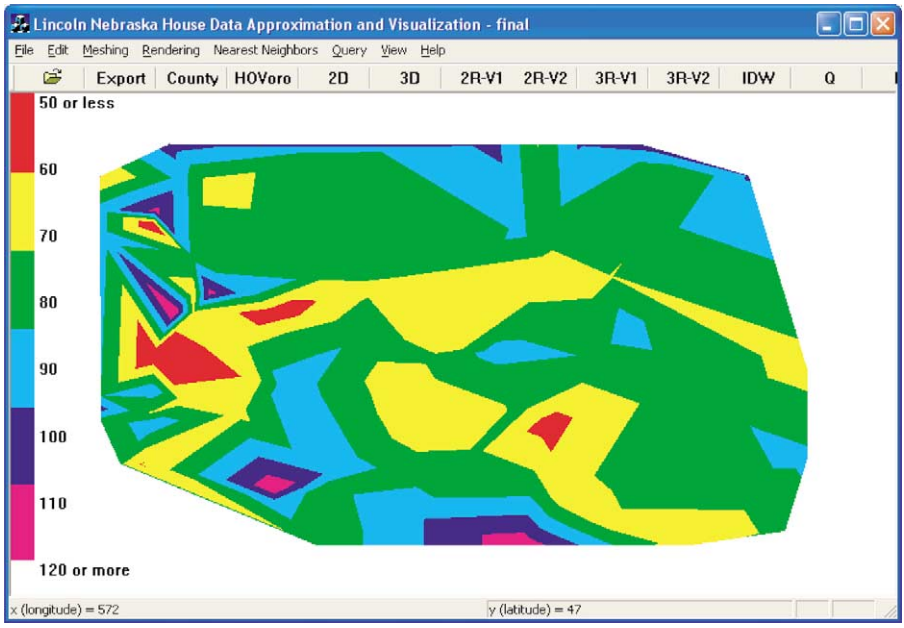

Fig. 7. Version 2: rendering with six discrete colors for the house price data of Lincoln, Nebraska in October 1995.

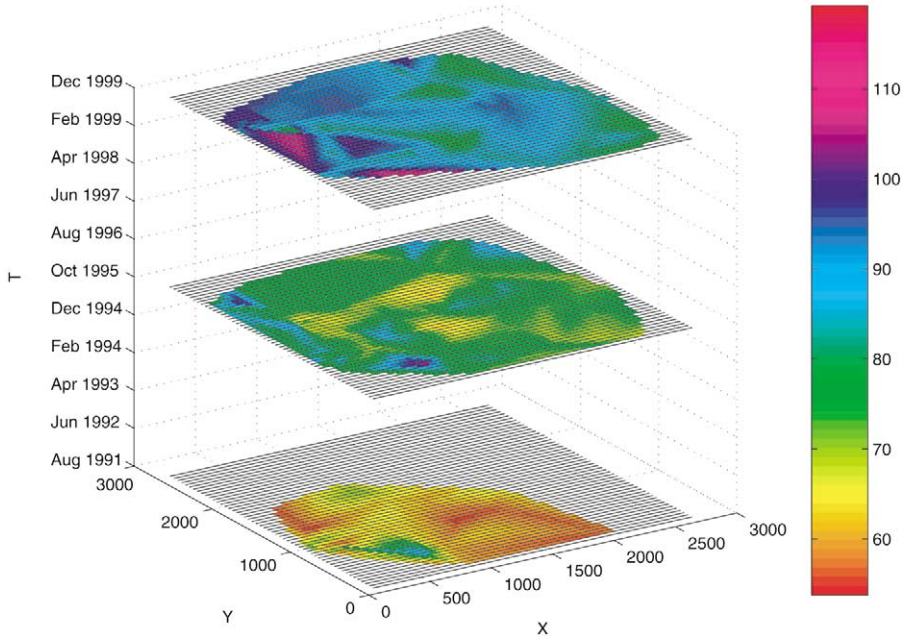

Fig. 8. Vertical profile of house price data of Lincoln, Nebraska in August 1991, October 1995 and December 1999.

3.2.1.1. Approximation in space and time. Similarly to the reduction approach to 2-D problems, 3-D approximation in space and time can be implemented by combining the time shape function (9) with the space approximation function (5). Using the example shown in Fig. 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:

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

Since the space shape functions (N_1, N_2, N_3 and N_4) and the time shape functions (9) are linear, the spatio-temporal approximation function (11) is quadratic.

3.2.2. 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 (e.g. nice ocean view may be preferred).

Our linear 4-D shape functions are based on 4-D Delaunay tesselation, which is briefly described in Section 2.3. In this section, 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.

3.2.2.1. 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

$$w(x, y, z, t) = \mathbf{\Phi}(x, y, z, t)\mathbf{f}. \tag{12}$$

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

$$\mathbf{\Phi}(x_1, y_1, z_1, t_1)\mathbf{f} = w_1$$

$$\mathbf{\Phi}(x_2, y_2, z_2, t_2)\mathbf{f} = w_2$$

$$\mathbf{\Phi}(x_3, y_3, z_3, t_3)\mathbf{f} = w_3$$

$$\mathbf{\Phi}(x_4, y_4, z_4, t_4)\mathbf{f} = w_4$$

$$\mathbf{\Phi}(x_5, y_5, z_5, t_5)\mathbf{f} = w_5$$

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

$$\mathbf{f} = \mathbf{A}^{-1}\mathbf{w}. \tag{13}$$

Let

$$\mathbf{N}(x, y, z, t) = \mathbf{\Phi}(x, y, z, t)\mathbf{A}^{-1}. \tag{14}$$

After substituting (13) into (12), we have

$$\begin{aligned} w(x, y, z, t) &= \mathbf{\Phi}(x, y, z, t)\mathbf{A}^{-1}\mathbf{w} \\ &= \mathbf{N}(x, y, z, t)\mathbf{w} \\ &= [N_1 N_2 N_3 N_4 N_5]\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{bmatrix} \end{aligned} \tag{15}$$

Now it is clear that (14) is the shape and function matrix that we need to find. We calculated the result of (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_is, y_is, z_is, t_is and w_is ($1 \leqslant i \leqslant 5$) is very redundant and unreadable. Each shape function expression N_i ($1 \leqslant i \leqslant 5$) covers about four pages. Next, we introduce a second approach which is based on the linear 3-D shape functions (6) or (8) and yields a neat symbolic expression.

3.2.2.2. 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) $(1 \leqslant i \leqslant 4)$. 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:

$$w(x, y, z, t) = \hat{\mathbf{\Phi}}(x, y, z, t)\hat{\mathbf{f}}. \tag{16}$$

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

$$\hat{\mathbf{\Phi}}(x_1, y_1, z_1, t_1)\hat{\mathbf{f}} = w_1$$

$$\hat{\mathbf{\Phi}}(x_2, y_2, z_2, t_2)\hat{\mathbf{f}} = w_3$$

$$\hat{\mathbf{\Phi}}(x_3, y_3, z_3, t_3)\hat{\mathbf{f}} = w_3$$

$$\hat{\mathbf{\Phi}}(x_4, y_4, z_4, t_4)\hat{\mathbf{f}} = w_4$$

$$\hat{\mathbf{\Phi}}(x_5, y_5, z_5, t_5)\hat{\mathbf{f}} = w_5$$

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

$$\mathbf{B} = \begin{bmatrix} \hat{\mathbf{\Phi}}(x_1, y_1, z_1, t_1) \\ \hat{\mathbf{\Phi}}(x_2, y_2, z_2, t_2) \\ \hat{\mathbf{\Phi}}(x_3, y_3, z_3, t_3) \\ \hat{\mathbf{\Phi}}(x_4, y_4, z_4, t_4) \\ \hat{\mathbf{\Phi}}(x_5, y_5, z_5, t_5) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & t_1 \\ 0 & 1 & 0 & 0 & t_2 \\ 0 & 0 & 1 & 0 & t_3 \\ 0 & 0 & 0 & 1 & t_4 \\ m_1 & m_2 & m_3 & m_4 & t_5 \end{bmatrix}$$

and $\mathbf{w}^{\mathbf{T}} = [w_1, w_2, w_3, w_4, w_5]$.

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

$$\hat{\mathbf{f}} = \mathbf{B}^{-1}\mathbf{w}. \tag{17}$$

Let

$$\mathbf{N}(x, y, z, t) = \hat{\mathbf{\Phi}}(x, y, z, t)\mathbf{B}^{-1}. \tag{18}$$

After substituting (17) into (16), we have

$$\begin{aligned} w(x, y, z, t) &= \hat{\mathbf{\Phi}}(x, y, z, t)\mathbf{B}^{-1}\mathbf{w} \\ &= \mathbf{N}(x, y, z, t)\mathbf{w} \\ &= [N_1 N_2 N_3 N_4 N_5] \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{bmatrix} \end{aligned} \tag{19}$$

The shape function result of (18) can be calculated as follows:

$$N_i = \hat{N}_i + \frac{m_i h}{\det B}\ (1 \leqslant i \leqslant 4) \quad \text{and} \quad N_5 = \frac{h}{\det B}, \tag{20}$$

where $\det B = -m_1 t_1 - m_2 t_2 - m_3 t_3 - m_4 t_4 + t_5$ is the determinant of $\mathbf{B}$ and $h = \hat{N}_1 t_1 + \hat{N}_2 t_2 + \hat{N}_3 t_3 + \hat{N}_4 t_4 - t$. This method can be generalized to derive shape functions of n dimension from shape functions of $n-1$ dimensions.

4. Comparison with IDW and kriging for 2-D space and 1-D time problems

So far we have discussed the reduction and extension approaches for the shape function based interpolation methods. Other spatial interpolation methods may also

have reduction and extension approaches for spatio-temporal problems. In this section, based on the same set of actual real estate data as used in Sections 3.1.1 and 3.1.2, we will compare the above shape function based methods with inverse distance weighting (IDW) and kriging interpolation methods in both reduction and extension approaches.

4.1. Experimental result of shape function based methods

4.1.1. 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 MAE and RMSE. The definition of MAE and RMSE is as follows:

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

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 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.

4.1.2. Error-proneness to time aggregation

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

(1) For a specific spatio-temporal 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 spatio-temporal 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.

Fig. 9 shows the experimental results of RMSE for error proneness to time aggregation of the shape function based methods. The results of MAE are very similar to RMSE. The Matlab function *polyfit* has been used to calculate the linear regression functions. In Table 3, the column *Slope* summarizes the slopes of MAE and RMSE linear regression functions. Steeper slope indicates less error-proneness

Table 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

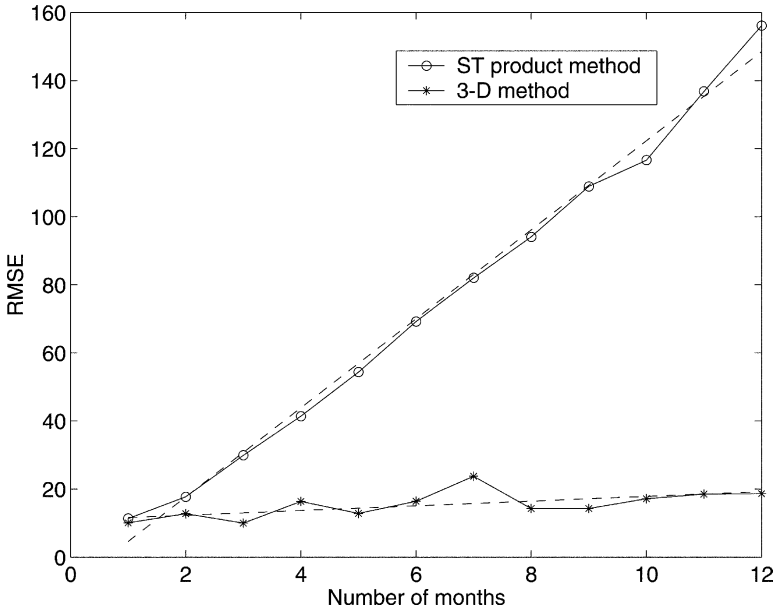

Fig. 9. Shape function susceptibility to time aggregation according to root mean square error (RMSE). The solid lines are the actual result, while the dashed lines are the linear regression functions that best approximate the tendency of RMSE.

to time aggregation. It is shown that the 3-D method is much less error-prone than the ST product method.

4.1.3. Constraint types

For the ST product method, if w_is ($1 \leqslant i \leqslant 3$) are linear functions of t, the constraint types are quadratic; if we use a polynomial function of t to approximate the w_is, we will get even higher polynomial functions for w_is. For the 3-D method,

we can find in linear time a constraint relation to represent the whole interpolation by representing the tetrahedral method in each tetrahedron with a separate constraint tuple (Li & Revesz, 2002).

In Table 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.

4.1.4. Invariance to coordinate scaling

Shape functions for triangles and tetrahedra are invariant to coordinate scaling, 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 spatio-temporal interpolation method especially when we want to use the extension approach. This is because we do not have to worry about what time unit should be used when mixing the space and time dimension. In Table 3, the column *Invariance* summarizes whether the method is invariant to coordinate scaling. We prove that 2-D triangular shape functions are invariant to scaling in below. The proof for the invariance of 3-D tetrahedral shape functions can be similarly obtained.

Proof 4.1. 2-D triangular shape functions are invariance to coordinate scaling.

Consider $N_1(x,y)$ in the triangular shape function (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 scaling. □

4.2. Experimental result of IDW-based methods

IDW interpolation is based on the assumption that things that are close to one another are more alike than those that are farther apart. Revesz and Li (2002a) uses IDW to visualize spatial interpolation data. In IDW, the measured values (known values) closer to 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 distance. Thus, points in the

near neighbourhood are given high weights, whereas points at a far distance are given small weights.

According to Johnston, Hoef, Krivoruchko, and Lucas (2001), the general formula of IDW interpolation is the following:

$$w(x,y)=\sum_{i=1}^{N}\lambda_i w_i,\quad \lambda_i=\frac{\left(\frac{1}{d_i}\right)^p}{\sum_{k=1}^{N}\left(\frac{1}{d_k}\right)^p}, \tag{21}$$

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.

Since the experimental data is 2-D, next we briefly discuss the IDW based reduction and extension approaches to 2-D problem. For 3-D problem, the formulae can be similarly derived.

Reduction Approach. 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 shape function interpolation in time, the time interpolation will be similar to (9). The formula of this approach can be expressed as:

$$w(x,y,t)=\sum_{i=1}^{N}\lambda_i w_i(t),\quad \lambda_i=\frac{\left(\frac{1}{d_i}\right)^p}{\sum_{k=1}^{N}\left(\frac{1}{d_k}\right)^p} \tag{22}$$

where

$$w_i(t)=\frac{t_{i2}-t}{t_{i2}-t_{i1}}w_{i1}+\frac{t-t_{i1}}{t_{i2}-t_{i1}}w_{i2}. \tag{23}$$

Each neighbor may have different beginning and ending times t_{i1} and t_{i2} in (23) if each point is sampled at different times.

Extension Approach Since this method treats time as a third dimension, the IDW based spatio-temporal formula is of the form of (21) with $d_i=\sqrt{(x_i-x)^2+(y_i-y)^2+(t_i-t)^2}$.

4.2.1. Accuracy

From the MAE and RMSE columns in Table 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.

4.2.2. Error-proneness to time aggregation

Similarly to the analysis of shape function-based methods, we test the same 12 ways of time aggregation for the IDW-based methods. Fig. 10 show the experimental results of RMSE for error proneness to time aggregation of the IDW based methods when the number of near neighbors is 3. The results of MAE are very similar to RMSE. From the column *Slope* in Table 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.

4.2.3. Constraint types

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

4.2.4. Non-invariance to coordinate scaling

IDW is not invariance to coordinate scaling. Consider the IDW interpolation with 2 neighbors and power 2, based on Eq. (21), we have

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

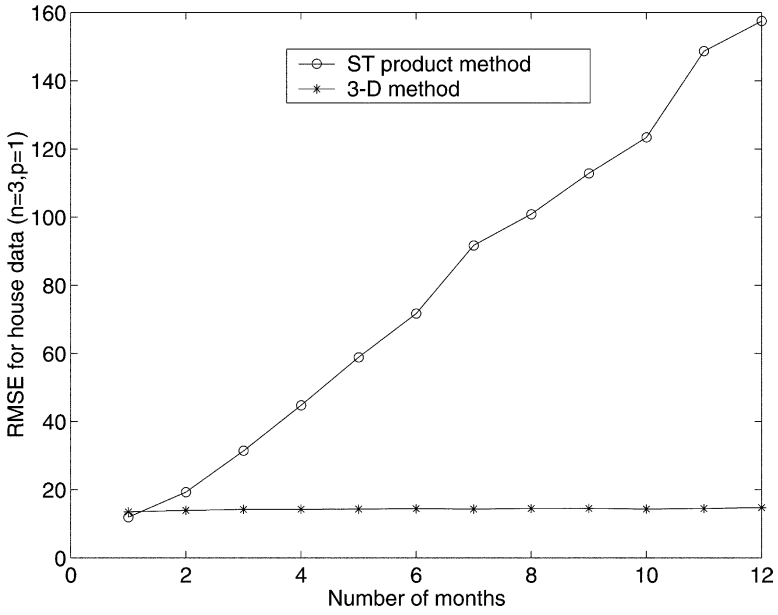

Fig. 10. IDW susceptibility to time aggregation according to root mean square error (RMSE). The solid lines are the actual result, while the dashed lines are the linear regression functions that best approximate the tendency of RMSE.

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 scaling.

4.3. Experimental result of kriging-based methods

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 and 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 $w(u)$.

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

$$2r = \frac{1}{N}\sum_{i=1}^{N}[w(u_i+h)-w(u_i)]^2 \tag{24}$$

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

From Eq. (24), 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 spatio-temporal interpolation is not feasible for kriging.

Extension Approach. Since kriging can be generalized into high dimension, the extension approach of kriging is a natural approach for spatio-temporal interpolation. There are multiple types of kriging, such as simple kriging, ordinary kriging, universal kriging, and factorial kriging. Because ordinary kriging is the most commonly used variant of simple kriging and it has been the anchor algorithm of geostatistics (Deutsch & journel, 1998), we choose 3-D ordinary kriging to interpolate the house experimental data. By ordinary kriging, the estimation for unknown location u is calculated as:

$$w(u) = \sum_{i=1}^{N} \lambda_i w_i, \quad \sum_{i=1}^{N} \lambda_i = 1\,, \tag{25}$$

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

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 and 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.

4.3.1. Accuracy

With the search radius being 500, the number of nearest neighbors will 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 3. The MAE and RMSE values of kriging based 3-D method are slightly better than the IDW-based 3-D method. But they are worse than shape function-based both ST product and 3-D methods.

4.3.2. Error-proneness to time aggregation

Similarly to the analysis of shape function and IDW-based methods, we test the same 12 ways of time aggregation for the kriging based 3-D method. Fig. 11 shows the experimental results of both MAE and RMSE. From the column *Slope* in Table 3, we can see that the kriging based 3-D method is not error-prone.

4.3.3. Constraint types

The constraint type for the kriging-based 3-D method is polynomial since the calculation of variograms by Eq. (24) is already quadratic.

4.3.4. Non-invariance to coordinate scaling

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

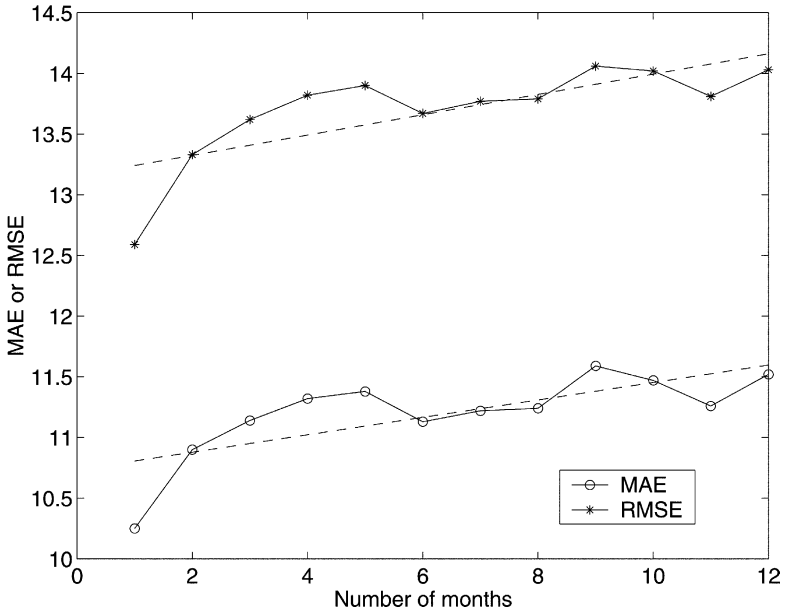

Fig. 11. Kriging susceptibility to time aggregation according to mean absolute error (MAE) and root mean square error (RMSE). The solid lines are the actual result of MAE and RMSE, while the dashed lines are the linear regression functions that best approximate their tendency.

5. 4-D Shape function example for 3-D space and 1-D time problems

We implemented the 4-D shape functions in Section 3.2.2 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 and 5.

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 3. This can be explained by the fact that the elevations

Table 4
Sample 4-D (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 5
Test 4-D (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

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 noise to the interpolation by considering the house elevation.

6. Future work

For the extension method based on shape functions the resulting spatio-temporal interpolation data can be represented using linear equality and inequality constraints. While there are many ways of storing this representation, constraint databases (Kanellakis, Kuper, & Revesz 1995; Kuper, Libkin, & Paredaens, 2000; Revesz, 2002) are a convenient alternative. Linear constraint databases used in 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 spatio-temporal animation. For future work, we plan to use this representation for the real estate data set and also experiment with other data sets.

Acknowledgements

The authors would like to thank Professor Reinhard Piltner for the discussion of shape functions and finite elements.

References

Buchanan, G. R. (1995). *Finite element analysis*. New York: McGraw-Hill.

Deutsch, C. V., & Journel, A. G. (1998). *GSLIB: geostatistical software library and user's guide* (2nd ed.). New York: Oxford University Press.

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*. Boca Raton, NY: CRC Press.

Grumbach, S., Rigaux, P., & Segoufin, L. (2000). Manipulating interpolated data is easier than you thought. In *Proc. of IEEE International Conference on very large databases* (pp. 156–165).

Harbaugh, J. W., & Preston, F. W. (1968). *Fourier analysis in geology*. Englewood Cliffs: Prentice-Hall.

Huebner, K. H. (1975). *The finite element method for engineers*. New York: John Wiley and Sons.

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. (1995). Constraint query languages. *Journal of Computer and System Sciences*, *51*(1), 26–52.

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.

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, pp. 145–160). Springer-Verlag.

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.

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 UK* (pp. 181–197). London: Taylor & Francis.

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-Verlag.

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

Revesz, P., & Li, L. (2002b). Representation and querying of interpolation data in constraint databases. In *Proc. of the Second National Conference on digital government research* (pp. 225–228). Los Angeles, CA.

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

Shewchuk, J. R. (1996). Triangle: engineering a 2D quality mesh generator and delaunay triangulator. In *Proc. First Workshop on applied computational geometry* (pp. 124–133). Philadelphia, PA.

Shewchuk, J. R. (1998). Tetrahedral mesh generation by delaunay refinement. In *Proc. 14th Annual ACM Symposium on computational geometry* (pp. 86–95). Minneapolis, MN.

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

Zienkiewics, O. C., & Taylor, R. L. (1989). Finite element method. In *The basic formulation and linear problems* (Vol. 1). McGraw-Hill.

Zienkiewics, O. C., & Taylor, R. L. (2000). Finite element method. In *The basis* (Vol. 1). London: Butterworth Heinemann.

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