Constraint-Based Visualization of Spatial Interpolation Data*

Peter Revesz Lixin Li
Computer Science and Engineering Department
University of Nebraska-Lincoln
Lincoln, NE 68588, U.S.A.
{revesz, lli}@cse.unl.edu

Abstract

We propose using a constraint relational representation for spatial data derived using an inverse distance weighting interpolation method. The advantage of our approach is that many queries that could not be done in traditional GIS systems can now be easily expressed and evaluated in constraint database systems. The data visualization can also be based on constraint techniques.

1. Introduction

To visualize and query a set of spatial data in GIS (Geographic Information Systems) applications, we often need *spatial interpolation*, that is, to estimate the unknown values at unsampled locations with a satisfying level of accuracy. For example, suppose 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.

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

Table 1. Incoming.

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

Table 2. Filter.

*This research was supported in part by NSF grant EIA-0091530 and by a Gallup Research Professorship.

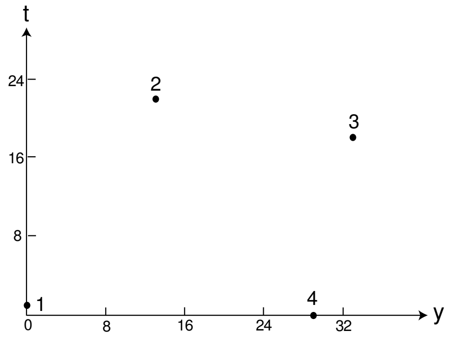

Figure 1. The sample points in Incoming.

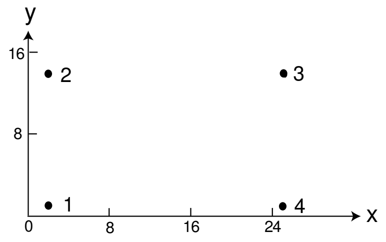

Figure 2. The sample points in Filter.

Figures 1 and 2 illustrate the locations of the (y, t) and (x, y) pairs where the measurements for u and r are recorded. Tables 1 and 2 show the corresponding instances of these two relations.

Since *Incoming(y,t,u)* and *Filter(x,y,r)* only record incoming ultraviolet u and filter ratio r at a few sampled locations, they cannot be displayed directly. Some spatial interpolation is needed to estimate u and r for all the locations in the domain. The spatial interpolation is usually used to calculate the interpolation values at each pixel to be displayed. This will result pixel-based data.

Pixel-based data are of great use for GIS applications, where the basic idea is to map each data value to a pixel in display. Many algorithms developed for pixel-based data stem from the graphics and image processing areas, such as the algorithms for planar transformation, shape filling, and clipping [5]. An overview of the pixel-based visualization techniques is given in [11]. However, the resulting pixel-based data file of interpolation has some potential problems. For example, the number of pixels in display is limited. In some applications, the number of data values may exceed the number of available pixels. In this case, the pixel file will not have complete information. Therefore, it is difficult to use to answer many queries. For example, consider the following query:

Query 1.1 *Find the amount of ultraviolet radiation for each ground location (x, y) at time t.*

Let *INCOMING(y,t,u)* and *FILTER(x,y,r)* be the relations that represent the interpolations of *Incoming(y,t,u)* and *Filter(x,y,r)*, respectively. Then the above query can be expressed in Datalog as follows:

$$\begin{aligned} GROUND(x,y,t,i) \quad :- \quad & INCOMING(y,t,u), \\ & FILTER(x,y,r), \\ & i = u(1-r) \ . \end{aligned} \tag{1}$$

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 perform on pixel-based files and are not supported by most GIS systems, including the ArcInfo/ArcView systems.

Several authors noted that interpolation constraints can be stored in *constraint relations*, which can be easily joined together, making the evaluation of queries like Query 1.1 feasible. (The textbook [15] discusses the relationship of constraint databases and GIS data models.) Also, in contrast to the pixel data representation, the constraint representation is capable of an arbitrary precision.

Chen et al. [2] and Revesz et al. [16] considered piecewise linear interpolation of time series data. Grumbach et al. [8] considered linear interpolation between snapshots of moving points and the interpolation of a landscape surface based on TIN (triangular irregular network) elevation data. Chen & Revesz [3] used a similar linear interpolation for landscape elevation, aspect, slope, and related data. All of these interpolations are represented in linear constraint databases.

Cai et al. [1], and Tossebro & Güting [17] considered the interpolation of snapshots of moving regions. Cai et al. [1] represent the interpolation by sets of *parametric rectangles*, and Tossebro & Güting [17] represent the interpolation by a *sliced representation* that was introduced by Forlizzi et al. [6]. Both parametric rectangles and sliced representations can be translated into linear constraint relations.

However, many practical spatial interpolations, such as inverse distance weighting [4], Kriging [13], splines [7], trend surfaces [18], and Fourier series [9], require non-linear constraints. In this paper, we focus on the inverse distance weighting (IDW) interpolation, which is non-linear, relatively easy, and gives good results in practice [12]. We also look at visualization, which is generally ignored in the earlier papers.

The rest of this paper is organized as follows. Section 2 discusses how to represent IDW interpolation in polynomial constraint databases. Section 3 describes the application of IDW in constraint databases to the example in this section. Section 4 gives some visualization results. Finally, in Section 5, we present some ideas for future work.

2. Representation of IDW in constraint databases

The rationale for IDW is consistent with most natural properties of spatial data, in particular, that their values vary continuously and tend to be similar at closer than at further locations. 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 reference [10], the general formula of IDW interpolation is the following:

$$w(x,y) = \sum_{i=1}^{N} \lambda_i w_i \quad , \qquad \lambda_i = \frac{(\frac{1}{d_i})^p}{\sum_{k=1}^{N} (\frac{1}{d_k})^p} \quad . \tag{2}$$

As shown in Figure 3, w is the predicted value for 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 distances between each (x_i, y_i) and (x, y), and p is the exponent, which influences the weighting of w_i on w. The optimal value of the exponent is dependent on the statistical characteristics of the data set. Please note that in Equation 2, $\sum_{i=1}^{N} \lambda_i = 1$.

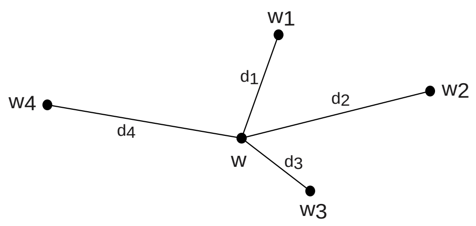

Figure 3. IDW Interpolation.

Example 2.1 *Suppose* $(x_1, y_1) = (0, 0)$*,* $(x_2, y_2) = (10, 0)$*, and* $(x_3, y_3) = (10, 5)$ *are the three closest sampled locations to the location* $(x, y) = (8, 2)$*, as shown in Figure 4. Let* $w_1 = 1$*,* $w_2 = 2$*, and* $w_3 = 3$ *be the values of the three sampled locations. We can interpolate the unknown value* w *at location* (x, y) *by IDW with* $N = 3$ *and* $p = 2$ *as*

$$\begin{aligned} w &= \sum_{i=1}^{3} \lambda_i w_i \\ &= \frac{(\frac{1}{d_1})^2}{(\frac{1}{d_1})^2 + (\frac{1}{d_2})^2 + (\frac{1}{d_3})^2} w_1 + \\ &\quad \frac{(\frac{1}{d_2})^2}{(\frac{1}{d_1})^2 + (\frac{1}{d_2})^2 + (\frac{1}{d_3})^2} w_2 + \\ &\quad \frac{(\frac{1}{d_3})^2}{(\frac{1}{d_1})^2 + (\frac{1}{d_2})^2 + (\frac{1}{d_3})^2} w_3 \\ &= 0.07 \times 1 + 0.35 \times 2 + 0.58 \times 3 \\ &= 2.51 \end{aligned}$$

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

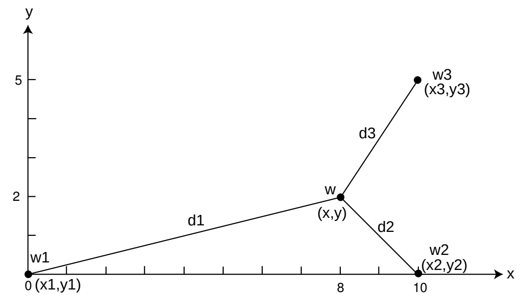

Figure 4. IDW example with three neighbors.

2.1. 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 [14]. 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 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:

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

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

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.

2.2. 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 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:

$$\begin{aligned} R(x,y,w) \quad :- \quad & ((x-x_2)^2 + (y-y_2)^2 + \\ & (x-x_1)^2 + (y-y_1)^2)\ w \\ & = \\ & ((x-x_2)^2 + (y-y_2)^2)w_1 + \\ & ((x-x_1)^2 + (y-y_1)^2)w_2\ , \\ & Vor\_2nd(x,y,x_1,y_1,x_2,y_2). \end{aligned} \tag{3}$$

or equivalently as,

$$\begin{aligned} R(x,y,w) \quad :- \quad & ((x-x_2)^2 + (y-y_2)^2 + \\ & (x-x_1)^2 + (y-y_1)^2)\ w \\ & = \\ & ((x-x_2)^2 + (y-y_2)^2)w_1 + \\ & ((x-x_1)^2 + (y-y_1)^2)w_2\ , \\ & C. \end{aligned} \tag{4}$$

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. Application

Let us return now to the example in Section 1. Figures 5 and 6 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 5, and there are no $(1,3)$ and $(2,4)$ regions in Figure 6.

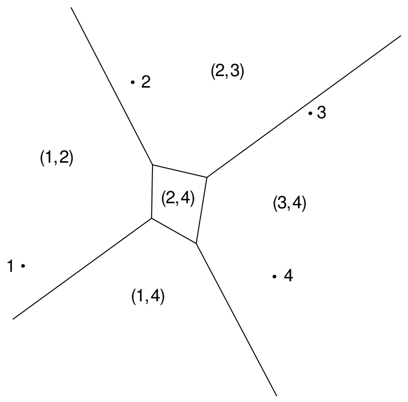

Figure 5. The 2nd **order Voronoi diagram for** *Incoming***.**

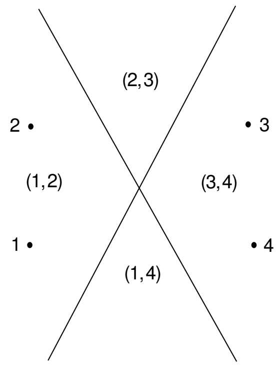

Figure 6. The 2nd **order Voronoi diagram for** *Filter***.**

Based on Equation 4, *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 Table 3 and 4. Note that the five tuples in Table 3 represent the five second-order Voronoi regions in Figure 5. These five regions are $(1,2)$, $(1,4)$, $(3,4)$, $(2,3)$ and $(2,4)$. Similarly, the four tuples in Table 4 represent the four second-order Voronoi regions in Figure 6. These four regions are $(1,2)$, $(1,4)$, $(3,4)$ and $(2,3)$.

The final result of the Datalog query, *GROUND(x, y, t, i)*, can be represent by Table 5. Since there are five tuples in *INCOMING(y,t,u)* and four tuples in *FILTER(x,y,r)*, there should be twenty tuples in

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	$y \geq 15,$ $y + 3t - 54 \leq 0,$ $7y - t - 136 \leq 0,$ $2y + 5t - 60 \geq 0,$ $((y-29)^2 + t^2)20 + ((y-13)^2 + (t-22)^2)40$ $= ((y-29)^2 + t^2 + (y-13)^2 + (t-22)^2)u$

Table 3. INCOMING (y,t,u).

GROUND(x, y, t, i). 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.

4. Visualization

In Section 2, we have described how to represent IDW interpolation in constraint databases. In Section 1, we have seen it is very easy to express queries (such as join operation) in constraint database on interpolation data. In this section, we will discuss how to visualize interpolation data and give some analysis on the quality of IDW interpolation.

For visualization, six basic colors are chosen: red, yellow, green, turquoise, blue, and purple. The 24 bits 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). 400 smoothly changing colors have been used for the color plot. These 400 colors are created by a linear interpolation scheme which is used between each of the following pair of the basic colors:

- red and yellow,

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

Table 4. FILTER (x,y,r).

- yellow and green,
- green and turquoise,
- turquoise and blue,
- blue and purple.

This color rendering yields a smooth change of colors in the visualization, hence it avoids sharp color transitions.

In Figures 7 and 8, the graphical interface for the presentation of IDW interpolation data is illustrated. Specifically, these two figures illustrate IDW interpolation with $n = 3$ and $p = 2$ on randomly selected DEM (Digital Elevation Model) data over the same area. Figure 7 visualizes the interpolation data based on 255 input points, while Figure 8 visualizes the interpolation data based on 1271 input points.

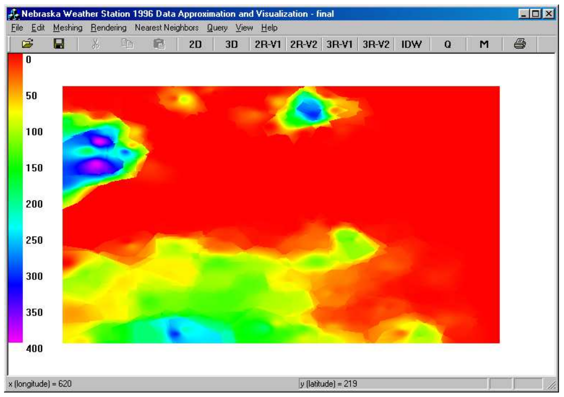
Nebraska Weather Station 1996 Data Approximation and Visualization - final
File Edit Meshing Rendering Nearest Neighbors Query View Help
2D 3D 2R-V1 2R-V2 3R-V1 3R-V2 IDW Q M
0
50
100
150
200
250
300
350
400
x (longitude) = 620
y (latitude) = 219

Figure 7. IDW ($n = 3$, $p = 2$) on 255 points.

Experiments have been conducted to analyze the quality of IDW interpolation according to Mean Absolute Error (MAE) and Root Mean Square Error

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

Table 5. GROUND (x, y, t, i).

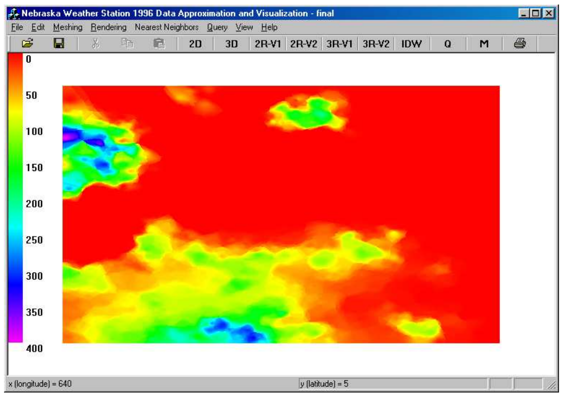
Nebraska Weather Station 1996 Data Approximation and Visualization - final
File Edit Meshing Rendering Nearest Neighbors Query View Help
2D 3D 2R-V1 2R-V2 3R-V1 3R-V2 IDW Q M
0
50
100
150
200
250
300
350
400
x (longitude) = 640
y (latitude) = 5

Figure 8. IDW ($n = 3$, $p = 2$) on 1271 points.

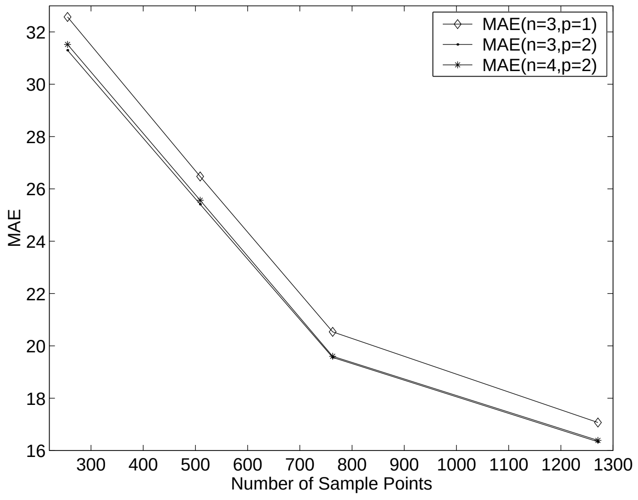

Figure 9. MAE result.

(RMSE). A set of sample points have been selected from a DEM surface in the northern part of San Francisco, which has 1525991 original points. The number of randomly selected points are 255, 509, 763 and 1271. For each dataset, three kinds of IDW interpolation methods with different n (the number of neighbors) and p (exponent) have been experimented: (i) $n = 3$, $p = 1$; (ii) $n = 3$, $p = 2$; (iii) $n = 4$, $p = 2$. The number of pixels in display is between 214775 to 215380.

Figures 9 and 10 illustrate the quality analysis of IDW interpolation based on different sets of randomly selected points from the original DEM data. We can see that under the condition of randomly selecting points, both MAE and RMSE almost decrease to half when the number of sample points increases from 255 to 1271. In particular, when $n = 4$ and $p = 2$ and the dataset contains 1271 points, the MAE is 16.34, which is approximately 17.3% of 94.55, the original average elevation value. This is a very good result, considering that in this case the size of input points is condensed over 1200 times, that is from 1525991 to 1271.

Although we only discuss the visualization of interpolation data in this section, the same visualization technique can apply to animating, that is, visualizing for each time instance, a query result, such as $GROUND(x, y, t, i)$ in the Datalog query in Formula 1.

5. Conclusion and future work

This paper discusses the representation, querying and visualization of IDW interpolation in polynomial constraint databases. In constraint databases, the details of the interpolation are at a lower level, which is transparent for the users. This property makes querying and visualization easy in constraint databases.

Kriging [13] is similar to IDW but the weights are derived using error statistics of the data. Beside IDW,

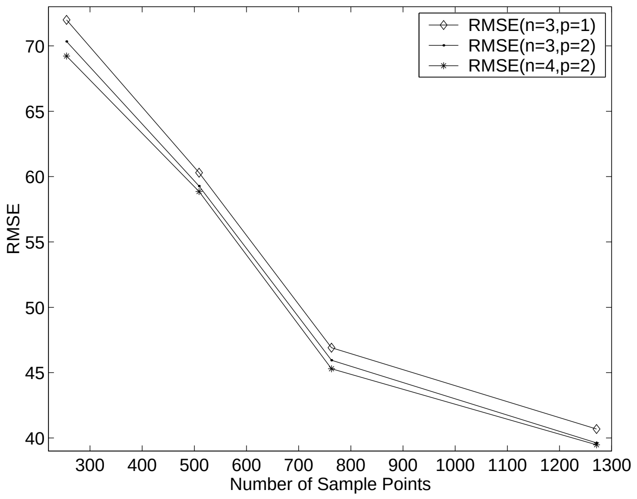

Figure 10. RMSE result.

it is easy to see that Kriging 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 a time series like data, which can be interpolated and translated into a linear constraint relation using the algorithm in Revesz et al. [16].

We are currently working on extending our work to animation. In animation we would use a 3D spatial interpolation that is the combination of a 2D spatial interpolation and a function of time. That is, at each sample point we would no longer have a constant value measured, but we would have a time series of the measurements. If the time series is itself interpolated, then we get a function of time that can be combined with the spatial interpolation to get a 3D spatio-temporal interpolation that is also representable in constraint relations.

References

[1] M. Cai, D. Keshwani, and P. Revesz. Parametric rectangles: A model for querying and animating spatiotemporal databases. In *Proc. 7th International Conference on Extending Database Technology*, volume 1777 of *Lecture Notes in Computer Science*, pages 430–44. Springer-Verlag, 2000.

[2] R. Chen, M. Ouyang, and P. Revesz. Approximating data in constraint databases. In *Proc. Symposium on Abstraction, Reformulation and Approximation*, volume 1864 of *Lecture Notes in Computer Science*, pages 124–143. Springer-Verlag, 2000.

[3] R. Chen and P. Revesz. Geo-temporal data transformation and visualization. In *The First International Conference on Geographic Information Science*, pages 240–242, Savannah, Georgia, USA, 2000.

[4] M. N. Demers. *Fundamentals of Geographic Information Systems.* John Wiley & Sons, New York, 2nd edition, 2000.

[5] J. D. Foley, A. V. Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics : Principles and Practice, Second Edition in C.* Addison-Wesley, 1996.

[6] L. Forlizzi, R. H. Güting, E. Nardelli, and M. Schneider. A data model and data structure for moving object databases. In *Proc. ACM SIGMOD International Conference on Management of Data*, pages 319–30, 2000.

[7] J. E. Goodman and J. O'Rourke, editors. *Handbook of Discrete and Computational Geometry.* CRC Press, Boca Raton, New York, 1997.

[8] S. Grumbach, P. Rigaux, and L. Segoufin. Manipulating interpolated data is easier than you thought. In *Proc. IEEE International Conference on Very Large Databases*, pages 156–65, 2000.

[9] J. W. Harbaugh and F. W. Preston. *Fourier Analysis in Geology*, pages 218–238. Prentice-Hall, Englewood Cliffs, 1968.

[10] K. Johnston, J. M. V. Hoef, K. Krivoruchko, and N. Lucas. *Using ArcGIS Geostatistical Analyst.* ESRI Press, 2001.

[11] D. A. Keim. Pixel-oriented database visualizations. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 25(4):35–39, 1996.

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

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

[14] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, 1985.

[15] P. Revesz. *Introduction to Constraint Databases.* Springer-Verlag, 2002.

[16] P. Z. Revesz, R. Chen, and M. Ouyang. Approximate query evaluation using linear constraint databases. In *Proc. Symposium on Temporal Representation and Reasoning*, pages 170–175, Cividale del Friuli, Italy, 2001.

[17] E. Tossebro and R. H. Güting. Creating representation for continuously moving regions from observations. In *Proc. 7th International Symposium on Spatial and Temporal Databases*, pages 321–344, Redondo Beach, CA, 2001.

[18] E. G. Zurflueh. Applications of two-dimensional linear wavelength filtering. *Geophysics*, 32:1015–1035, 1967.