The Use of Fuzzy Measures as a Data Fusion Tool in Geographic Information Systems: Case Study

Cynthia Campos, G. Randy Keller, Vladik Kreinovich, Luc Longpré,
Francois Modave, Scott A. Starks, and Roberto Torres
NASA Pan-American Center for Earth and Environmental Studies (PACES)
University of Texas at El Paso, El Paso, TX 79968, USA
contact emails {vladik,fmodave}@cs.utep.edu

Abstract

Geospatial databases generally consist of measurements related to points (or pixels in the case of raster data), lines, and polygons. In recent years, the size and complexity of these databases have increased significantly and they often contain duplicate records, i.e., two or more close records representing the same measurement result. In this paper, we use fuzzy measures to address the problem of detecting duplicates in a database consisting of point measurements. As a test case, we use a database of measurements of anomalies in the Earth's gravity field that we have compiled. We show that a natural duplicate deletion algorithm requires (in the worst case) quadratic time, and we propose a new asymptotically optimal $O(n \cdot \log(n))$ algorithm. These algorithms have been successfully applied to gravity databases. We believe that they will prove to be useful when dealing with many other types of point data.

1. General Introduction

The current state of remote sensing technology enables us to acquire huge amount of geospatial data. Moreover, the rate with which we acquire geospatial data is constantly increasing: for example, geospatial satellites of a new generation produce images in several hundred wavelengths as opposed to seven wavelengths for Landsat satellites. As a result, the rate of data acquisition is much higher than the rate with which we are able to analyze this data.

One of the natural approaches to solving this problem is to take into account the experience of experts–geologists and geophysicists–in processing the geospatial data. When the experts analyze a region, they consider data about this region coming from different sources–gravity measurements, magnetic measurements, terrestrial geophysical measurements–and combine ("fuse") this data into a single geological map. Therefore, to be able to process all available geospatial data, it is desirable to design a computer-based system with the ability to fuse all this data.

There exist many useful data fusion techniques; some of these techniques–e.g., statistical methods of data fusion–come from the detailed mathematical analysis of the corresponding problems. However, since the main objective of our particular data fusion application is to emulate the experts, we strongly believe that a more prospective approach is to take into consideration experts' uncertainty and use techniques that have been specifically designed to handle this uncertainty–the techniques of fuzzy logic.

Fuzzy techniques have been actively used in data fusion, in particular, in data fusion of geospatial images. The traditional use of fuzzy techniques is usually based on fuzzy logic operations, operations that have been designed to combine ("fuse") two sources of data. It is, in principle, possible to apply this fusion several times and thus get a fusion of multiple data sources. It is desirable to use more recent fuzzy techniques specifically designed for fusing multiple sources–techniques based on fuzzy (non-additive) measures–the fuzzy generalizations of the traditional mathematical notion of measure.

2. Case Study: Geoinformatics Motivation for the Problem

Geospatial databases: general description. In many application areas, researchers and practitioners have collected a large amount of geospatial data. For example, geophysicists measure values d of the gravity and magnetic fields, elevation, and reflectivity of electromagnetic energy for a broad range of wavelengths (visible, infrared, and radar) at different geographical points (x, y); see, e.g., [18]. Each type of data is usually stored in a large geospatial database that contains corresponding records (x_i, y_i, d_i). Based on these measurements, geophysicists generate maps and im-

ages and derive geophysical models that fit these measurements.

Gravity measurements: case study. In particular, gravity measurements are one of the most important sources of information about subsurface structure and physical conditions. There are two reasons for this importance. First, in contrast to more widely used geophysical data like remote sensing images, that mainly reflect the conditions of the Earth's surface, gravitation comes from the whole Earth (e.g., [9, 10]). Thus gravity data contain valuable information about much deeper geophysical structures. Second, in contrast to many types of geophysical data, which usually cover a reasonably local area, gravity measurements cover broad areas and thus provide important regional information.

The accumulated gravity measurement data are stored at several research centers around the world. One of these data storage centers is located at the University of Texas at El Paso (UTEP). This center contains gravity measurements collected throughout the United States and Mexico and parts of Africa.

The geophysical use of gravity database compiled at UTEP is illustrated for a variety of scales in [1, 3, 6, 8, 11, 16, 19, 20].

Duplicates: where they come from. One of the main problems with the existing geospatial databases is that they are known to contain many duplicate points (e.g., [7, 13, 17]). The main reason why geospatial databases contain duplicates is that the databases are rarely formed completely "from scratch", and instead are built by combining measurements from numerous sources. Since some measurements are represented in the data from several of the sources, we get duplicate records.

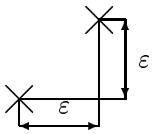
Why duplicates are a problem. Duplicate values can corrupt the results of statistical data processing and analysis. For example, when instead of a single (actual) measurement result, we see several measurement results confirming each other, and we may get an erroneous impression that this measurement result is more reliable that it actually is. Detecting and eliminating duplicates is therefore an important part of assuring and improving the quality of geospatial data, as recommended by the US Federal Standard [5].

Description in terms of a fuzzy (non-additive) measure. It is difficult to describe duplicates in probabilistic terms. A natural description is as follows: for any set of records S, we define $D(S)$ as 1 if S contains duplicates, and 0 if it does not. When we combine two duplicate-free databases S_1 and S_2 (with $D(S_1) = D(S_2) = 0$) into a single database $S_1 \cup S_2$, then, if S_1 and S_2 share a record, will have

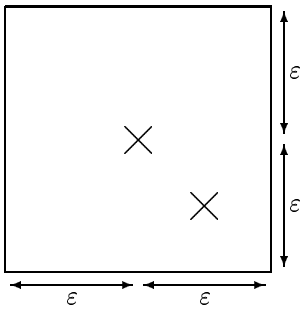
$$D(S_1 \cup S_2) = 1 > D(S_1) + D(S_2) = 0,$$

i.e., D is indeed a non-additive (fuzzy) measure.

Duplicates correspond to interval uncertainty. In the ideal case, when measurement results are simply stored in their original form, duplicates are identical records, so they are easy to detect and to delete. In reality, however, different databases may use different formats and units to store the same data: e.g., the latitude can be stored in degrees (as 32.1345) or in degrees, minutes, and seconds. As a result, when a record (x_i, y_i, d_i) is placed in a database, it is transformed into this database's format. When we combine databases, we may need to transform these records into a new format – the format of the resulting database. Each transformation is approximate, so the records representing the same measurement in different formats get transformed into values which correspond to close but not identical points $(x_i, y_i) \neq (x_j, y_j)$. Usually, geophysicists can produce a threshold $\varepsilon > 0$ such that if the points (x_i, y_i) and (x_j, y_j) are ε-close – i.e., if $|x_i - x_j| \leq \varepsilon$ and $|y_i - y_j| \leq \varepsilon$ – then these two points are duplicates.



In other words, if a new point (x_j, y_j) is within a 2D *interval* $[x_i - \varepsilon, x_i + \varepsilon] \times [y_i - \varepsilon, y_i + \varepsilon]$ centered at one of the existing points (x_i, y_i), then this new point is a duplicate:



If the two points are duplicates, we should delete one of these two points from the database. Since the difference between the two points is small, it does not matter much which of the two points we delete. In other words, we want to continue deleting duplicates until we arrive at a "duplicate-free" database. There may be several such duplicate-free databases, all we need is one of them.

Duplicates are not easy to detect and delete. At present, the detection and deletion of duplicates is done mainly "by hand", by a professional geophysicist looking at the raw measurement results (and at the preliminary results of processing these raw data). This manual cleaning is very *time-consuming*. It is therefore necessary to design *automated* methods for detecting duplicates.

If the database was small, we could simply compare every record with every other record. This comparison would require $n(n-1)/2 \sim n^2/2$ steps. Alas, real-life geospatial databases are often large, they may contain up to 10^6 or more records; for such databases, $n^2/2$ steps is too long. We need faster methods for deleting duplicates.

From interval to fuzzy uncertainty. Sometimes, instead of a single threshold value ε, geophysicists provide us with several possible threshold values $\varepsilon_1 < \varepsilon_2 < \ldots < \varepsilon_m$ that correspond to decreasing levels of their certainty:

- if two measurements are within ε_1 from each other, then we are 100% certain that they are duplicates;
- if two measurements are within ε_2 from each other, then with some degree of certainty, we can claim them to be duplicates,
- if two measurements are within ε_2 from each other, then with an even smaller degree of certainty, we can claim them to be duplicates,
- etc.

In this case, we must eliminate *certain* duplicates, and mark *possible* duplicates (about which are not 100% certain) with the corresponding degree of certainty.

In this case, for each of the coordinates x and y, instead of a single interval $[x_i - \varepsilon, x_i + \varepsilon]$, we have a nested family of intervals $[x_i - \varepsilon_j, x_i + \varepsilon_j]$ corresponding to different degrees of certainty. Such a nested family of intervals is also called a *fuzzy set*, because it turns out to be equivalent to a more traditional definition of fuzzy set [2, 12, 14, 15] (if a traditional fuzzy set is given, then different intervals from the nested family can be viewed as α-cuts corresponding to different levels of uncertainty α).

In these terms, in addition to detecting and deleting duplicates under interval uncertainty, we must also detect and delete them under fuzzy uncertainty.

What we are planning to do. In this paper, we propose methods for detecting and deleting duplicates under interval and fuzzy uncertainty, and test these methods on our database of measurements of the Earth's gravity field.

3. Geospatial Databases: Brief Introduction

Geospatial databases: formal description. In accordance with our description, a *geospatial database* can be described as a finite set of *records* $r_1, \ldots, r_n$, each of which is a triple $r_i = (x_i, y_i, d_i)$ consisting of two rational numbers x_i and y_i that describe coordinates and some additional data d_i.

The need for sorting. One of the main objectives of a geospatial database is to make it easy to find the information corresponding to a given geographical area. In other words, we must be able, given one or two coordinates (x and/or y) of a geographical point (center of the area of interest), to easily find the data corresponding to this point and its vicinity.

It is well known that if the records in a database are not sorted by a parameter a, then in order to find a record with a given value of a, there is no faster way than linear (exhaustive) search, in which we check the records one by one until we find the desired one. In the worst case, linear search requires searching over all n records; on average, we need to search through $n/2$ records. For a large database with thousands and millions of record, this takes too much times.

To speed up search, it is therefore desirable to *sort* the records by the values of a, i.e., to reorder the records in such a way that the corresponding values of a are increasing: $a_1 \le a_2 \le \ldots \le a_n$.

Once the records are sorted, instead of the time-consuming linear search, we can use a much faster *binary search* (also known as *bisection*). At each step of the binary search, we have an interval $a_l \le a \le a_u$. We start with $l = 1$ and $u = n$. On each step, we take a midpoint $m = \lfloor (l+u)/2 \rfloor$ and check whether $a < a_m$. If $a < a_m$, then we have a new half-size interval $[a_l, a_{m-1}]$; otherwise, we have a half-size interval $[a_m, a_u]$ containing a. In $\log_2(n)$ steps, we can thus locates the record corresponding to the desired value of a.

How to Sort: Mergesort Algorithm. Sorting can be done, e.g., by *mergesort* – an asymptotically optimal sorting algorithm that sorts in $O(n \cdot \log(n))$ computational steps (see, e.g., [4]).

4. The Problem of Deleting Duplicates: Ideal Case of No Uncertainty

To come up with a good algorithm for detecting and eliminating duplicates in case of interval uncertainty, let us first consider an ideal case when there is no uncertainty, i.e., when duplicate records $r_i = (x_i, y_i, d_i)$ and

$r_j = (x_j, y_j, d_j)$ mean that the corresponding coordinates are equal: $x_i = x_j$ and $y_i = y_j$.

In this case, to eliminate duplicates, we can do the following. We first sort the records in lexicographic order, so that r_i goes before r_j if either $x_i < x_j$, or $x_i = x_j$ and $y_i \le y_j$. In this order, duplicates are next to each other.

So, we first compare r_1 with r_2. If coordinates in r_2 are identical to coordinates in r_1, we eliminate r_2 as a duplicate, and compare r_1 with r_3, etc. After the next element is no longer a duplicate, we take the next record after r_1 and do the same for it, etc.

After each comparison, we either eliminate a record as a duplicate, or move to a next record. Since we only have n records in the original database, we can move only n steps to the right, and we can eliminate no more than n records. Thus, totally, we need no more than $2n$ comparison steps to complete our procedure.

Since $2n$ is asymptotically smaller than the time $n \cdot \log(n)$ needed to sort the record, the total time for sorting and deleting duplicates is $n \cdot \log(n) + 2n \sim n \cdot \log(n)$. Since we want a sorted database as a result, and sorting requires at least $n \cdot \log(n)$ steps, this algorithm is asymptotically optimal.

5 Interval Modification of the Above Algorithm: Description, Practicality, Worst-Case Complexity

In the previous section, we described how to eliminate duplicates in the ideal case when there is no uncertainty.

In real life, as we have mentioned, there is an interval uncertainty. A natural idea is therefore to modify the above algorithm so that it detects not only exact duplicate records but also records that are within ε of each other.

In precise terms, we have a geospatial database $\langle r_1, \ldots, r_n \rangle$, where $r_i = (x_i, y_i, d_i)$, and we are also given a positive rational number ε. We say that records $r_i = (x_i, y_i, d_i)$ and $r_j = (x_j, y_j, d_j)$ are *duplicates* (and denote it by $r_i \sim r_j$) if $|x_i - x_j| \le \varepsilon$ and $|y_i - y_j| \le \varepsilon$.

We say that a subset of the database is obtained by a *cleaning step* if:

- it is obtained from the original database by selecting one or several different pairs of duplicates and deleting one duplicate from each pair, and
- from each *duplicate chain* $r_i \sim r_j \sim \ldots \sim r_k$, at least record remains in the database after deletion.

A sequence of cleaning steps after which the resulting subset is duplicate-free (i.e., does not contain any duplicates) is called *deleting duplicates*.

The goal is to produce a (duplicate-free) subset of the original database obtained by deleting duplicates – and to produce it sorted by x_i (and double-sorted by y).

Similarly to the ideal case of no uncertainty, to avoid comparing all pairs (r_i, r_j) – and since we need to sort by x_i anyway – we first sort the records by x, so that $x_1 \le x_2 \le \ldots \le x_n$. Then, first we detect and delete all duplicates of r_1, then we detect and delete all duplicates of r_2 (r_1 is no longer considered since its duplicates have already been deleted), then duplicates of r_3 (r_1 and r_2 are no longer considered), etc.

For each i, to detect all duplicates of r_i, we check r_j for the values $j = i+1, i+2, \ldots$ while $x_j \le x_i + \varepsilon$. Once we have reached the value j for which $x_j > x_i + \varepsilon$, then we can be sure (since the sequence x_i is sorted by x) that $x_k > x_i + \varepsilon$ for all $k \ge j$ and hence, none of the corresponding records r_k can be duplicates of r_i.

While $x_j \le x_i + \varepsilon$, we have $x_i \le x_j \le x_i + \varepsilon$ hence $|x_i - x_j| \le \varepsilon$. So, for these j, to check whether r_i and r_j are duplicates, it is sufficient to check whether $|y_i - y_j| \le \varepsilon$.

Thus, the following algorithm solves the problem of deleting duplicates:

Algorithm 1.

1. Sort the records by x_i, so that $x_1 \le x_2 \le \ldots \le x_n$.
2. For i from 1 to $n-1$, do the following:

   for $j = i+1, i+2, \ldots$, while $x_j \le x_i + \varepsilon$

   if $|y_j - y_i| \le \varepsilon$, delete r_j.

For the gravity database, this algorithm works reasonably well, but we cannot be sure that it will always work well, because its worst-case complexity is still $n(n-1)/2$. Indeed, if all n records have the same value of x_i, and all the values y_i are drastically different (e.g., $y_i = y_1 + 2 \cdot (i-1) \cdot \varepsilon$), then the database is duplicate-free, but the above algorithm requires that we compare all the pairs.

For gravity measurements, this is, alas, a very realistic situation, because measurements are sometimes made when a researcher travels along a road and makes measurements along the way – and if the road happens to be vertical ($x \approx \text{const}$), we end up with a lot of measurements corresponding to very close values of x.

We therefore need a faster algorithm for deleting duplicates.

6. New Algorithm: Description, Complexity

The following algorithm starts with a database of records $r_i = (x_i, \ldots, y_i, d_i)$ (not necessarily 2-dimensional) and a number $\varepsilon > 0$ and deletes duplicates faster:

Algorithm 2.

1. For each record, compute the indices $p_i = \lfloor x_i/\varepsilon \rfloor, \ldots, q_i = \lfloor x_i/\varepsilon \rfloor$.
2. Sort the records in lexicographic order $\le$ by their index vector $\vec{p}_i = (p_i, \ldots, q_i)$. If several records have the same index vector, keep only one of these records and delete others as duplicates. As a result, we get an index-lexicographically list of records: $r_{(1)} \le \ldots \le r_{(m)}$, where $m \le n$.
3. For i from 1 to n, we compare the record $r_{(i)}$ with its immediate neighbors; if one of the immediate neighbors is a duplicate to $r_{(i)}$, then we delete this neighbor.

Let us describe Part 3 in more detail. By an *immediate neighbor* to a record r_i with an index vector $(p_i, \ldots, q_i)$, we mean a record r_j for which the index vector $\vec{p}_j \ne \vec{p}_i$ has the following two properties:

- $\vec{p}_i \le \vec{p}_j$, and
- for each index, $p_j \in \{p_i - 1, p_i, p_i + 1\}, \ldots$, and $q_j \in \{q_i - 1, q_i, q_i + 1\}$.

It is easy to check that if two records are duplicates, then indeed their indices can differ by no more than 1, i.e., the differences $\Delta p \stackrel{\text{def}}{=} p_j - p_i, \ldots, \Delta q \stackrel{\text{def}}{=} q_j - q_i$ between the indices can only take values -1, 0, and 1. To guarantee that $\vec{p}_j \ge \vec{p}_i$ in lexicographic order, we must make sure that the first non-zero term of the sequence $(\Delta p, \ldots, \Delta q)$ is 1.

Overall, there are 3^d sequences of -1, 0, and 1, where by d, we denoted the dimension of the vector $(x, \ldots, y)$. Out of these vectors, one is $(0, \ldots, 0)$, and half of the rest – to be more precise, $N_d \stackrel{\text{def}}{=} (3^d - 1)/2$ of them – correspond to immediate neighbors.

To describe all immediate neighbors, during Step 3, for each i and for each of N_d difference vectors $\vec{d} = (\Delta p, \ldots, \Delta q)$, we keep the index $j(\vec{d}, i)$ of the first record $r_{(j)}$ for which $\vec{p}_{(j)} \ge \vec{p}_{(i)} + \vec{d}$ (here, $\ge$ means lexicographic order). Then:

- If $\vec{p}_{(j)} = \vec{p}_{(i)} + \vec{d}$, then the corresponding record $r_{(j)}$ is indeed an immediate neighbor of $r_{(i)}$, so must check whether it is a duplicate.
- If $\vec{p}_{(j)} > \vec{p}_{(i)} + \vec{d}$, then the corresponding record $r_{(j)}$ is not an immediate neighbor of $r_{(i)}$, so no duplicate check is needed.

We start with $j(\vec{d}, 0) = 1$ corresponding to $i = 0$. When we move from i-th iteration to the next $(i+1)$-th iteration, then, since the records $r_{(k)}$ are lexicographically ordered, for each of N_d vectors $\vec{d}$, we have $j(\vec{d}, i+1) \ge j(\vec{d}, i)$. Therefore, to find $j(\vec{d}, i+1)$, it is sufficient to start with $j(\vec{d}, i)$ and add 1 until we get the first record $r_{(j)}$ for which $\vec{p}_{(j)} \ge \vec{p}_{(i+1)} + \vec{d}$.

The following result show that this algorithm is indeed asymptotically optimal:

Proposition 1. *Algorithm 2 requires $O(n \cdot \log(n))$ steps in the worst case, and no algorithm with asymptotically smaller worst-case complexity is possible.*

Proof. Algorithm 2 consists of a sorting–which requires $O(n \cdot \log(n))$ steps–and the main Part 3. During this part, for each of N_d vectors $\vec{d}$, we move the corresponding index j one by one from 1 to $m \le n$; for each value of the index, we make one or two comparisons. Thus, for each vector $\vec{d}$, we need $O(n)$ comparisons.

For a fixed dimension d, there is a fixed number N_d of vectors $\vec{d}$, so we need the total of $N_d \cdot O(n) = O(n)$ computational steps. Thus, the total time of Algorithm 2 is $O(n) + O(n \cdot \log(n)) = O(n \cdot \log(n))$.

On the other hand, since our problem requires sorting, we cannot solve it faster than in $O(n \cdot \log(n))$ steps that are needed for sorting [4]. Proposition is proven.

7. Deleting Duplicates Under Fuzzy Uncertainty

As we have mentioned, in some real-life situations, in addition to the threshold ε that guarantees that vε-close data are duplicates, the experts also provide us with additional threshold values $\varepsilon_i > \varepsilon$ for which ε_i-closeness of two data points means that we can only conclude with a certain degree of certainty that one of these data points is a duplicate. The corresponding degree of certainty decreases as the value ε_i increases.

In this case, in addition to deleting records that are absolutely certainly duplicates, it is desirable to mark possible duplicates – so that a professional geophysicist can make the final decision on whether these records are indeed duplicates.

A natural way to do this is as follows:

- First, we use the above algorithm to delete all the certain duplicates (corresponding to ε).
- Then, we use the same algorithm to the remaining records and mark (but not actually delete) all the duplicates corresponding to the next value ε_2. The resulting marked records are duplicates with the degree of confidence corresponding to ε_2.
- After that, we apply the same algorithm with the value ε_3 to all unmarked records, and mark those which the algorithm detects as duplicates with the degree of certainty corresponding to ε_3,

- etc.

In other words, to solve a fuzzy problem, we solve several interval problems corresponding to different levels of uncertainty. It is worth mentioning that this "interval" approach to solving a fuzzy problem is in line with many other algorithms for processing fuzzy data; see, e.g., [2, 12, 14, 15].

Acknowledgments

This work was supported in part by NASA grants NCC5-209 and NCC2-1232, by the Air Force Office of Scientific Research grant F49620-00-1-0365, and by NSF grants CDA-9522207, EAR-0112968, EAR-0225670, and 9710940 Mexico/Conacyt.

This research was partly done when V.K. was a Visiting Faculty Member at the Fields Institute for Research in Mathematical Sciences (Toronto, Canada).

References

[1] D. C. Adams, G. R. Keller, Precambrian basement geology of the Permian Basin region of West Texas and eastern New Mexico: A geophysical perspective, *American Association of Petroleum Geologists Bulletin* 80:410–431, 1996.

[2] G. Bojadziev, and M. Bojadziev, *Fuzzy sets, fuzzy logic, applications*, World Scientific, Singapore, 1995.

[3] L. Cordell, G. R. Keller, Bouguer Gravity Map of the Rio Grande Rift, Colorado, New Mexico, and Texas Geophysical investigations series, U.S. Geological Survey, 1982.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, C., *Introduction to Algorithms*, MIT Press, Cambridge, MA, and Mc-Graw Hill Co., N.Y., 2001.

[5] FGDC Federal Geographic Data Committee, FGDC-STD-001-1998. Content standard for digital geospatial metadata (revised June 1998), Federal Geographic Data Committee, Washington, D.C., http://www.fgdc.gov/metadata/contstan.html

[6] M. M. Fliedner, S. D. Ruppert, P. E. Malin, S. K. Park, G. R. Keller, and K. C. Miller, Three-dimensional crustal structure of the southern Sierra Nevada from seismic fan profiles and gravity modeling, *Geology* 24:367–370, 1996.

[7] M. Goodchild and S. Gopal (Eds.), *Accuracy of Spatial Databases*, Taylor & Francis, London, 1989.

[8] V. J. S. Grauch, C. L. Gillespie, and G. R. Keller, Discussion of new gravity maps of the Albuquerque basin, *New Mexico Geol. Soc. Guidebook* 50:119–124, 1999.

[9] W. A. Heiskanen and F. A. Meinesz, *The Earth and its gravity field*, McGraw-Hill, New York, 1958.

[10] W. A. Heiskanen and H. Moritz, *Physical Geodesy*, W.H. Freeman and Company, San Francisco, California, 1967.

[11] G. R. Keller, Gravitational Imaging, In: *The Encyclopedia of Imaging Science and Technology*, John Wiley, New York, 2001.

[12] G. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic: Theory and Applications*, Prentice Hall, Upper Saddle River, NJ, 1995.

[13] M. McCain and C. William, Integrating Quality Assurance into the GIS Project Life Cycle, Proceedings of the 1998 ESRI Users Conference. http://www.dogcreek.com/html/documents.html

[14] H. T. Nguyen and V. Kreinovich, "Nested Intervals and Sets: Concepts, Relations to Fuzzy Sets, and Applications", In: R. B. Kearfott et al (eds.), *Applications of Interval Computations*, Kluwer, Dordrecht, 1996, 245–290.

[15] H. T. Nguyen and E. A. Walker, *First Course in Fuzzy Logic*, CRC Press, Boca Raton, FL, 1999.

[16] J. A. Rodriguez-Pineda, N. E. Pingitore, G. R. Keller, A. Perez, An integrated gravity and remote sensing assessment of basin structure and hydrologic resources in the Chihuahua City region, Mexico, *Engineering and Environ. Geoscience* 5:73–85, 1999.

[17] L. Scott, Identification of GIS Attribute Error Using Exploratory Data Analysis, *Professional Geographer* 46(3):378–386, 1994.

[18] P. Sharma, *Environmental and Engineering Geophysics*, Cambridge University Press, Cambridge, U.K., 1997.

[19] S. M. Simiyu, G. R. Keller, An integrated analysis of lithospheric structure across the East African Plateau based on gravity anomalies and recent seismic studies, Tectonophysics 278:291–313, 1997.

[20] A. L. Tesha, A. A. Nyblade, G. R. Keller, D. I. Doser, Rift localization in suture-thickened crust: Evidence from Bouguer gravity anomalies in northeastern Tanzania, East Africa, *Tectonophysics*, 278:315–328, 1997.