Frank, A. U. "Canonical Geometric Representations (Draft)." 30. Orono, Maine, USA: Department of Civil Engineering, 1984.

Department of Civil Engineering

SURVEYING ENGINEERING

DRAFT

Canonical Geometric Representations

Andrew U. Frank

Report No. 42

This manuscript is in a preliminary form and comments from readers are encouraged and very welcome.

University of Maine at Orono
103 Boardman Hall
Orono, ME 04469

DRAFT

Canonical geometric representations

Andrew U. Frank
University of Maine at Orono
Orono, Maine 04469

1. Introduction

In applications like civil engineering, surveying, cartography and many others, geometric models of real world situations must be formed. These models are then used for detecting and understanding changes and for planning future actions (Chevallier 1984). Traditionally such models were formed and represented as maps and all operations carried out on them graphically. The inheritent limitations in the possibilities for graphical representations made it desirable to use computers for modeling. It is expected that powerful operations can be programmed to manipulate these models and advanced graphical output techniques used to produce maps and charts easy for humans to understand.

This paper deals with general requirements for selecting internal representation of geometric properties; if the form of the internal representation in the computer is well chosen, the programs to perform the necessary actions become conceptually simpler (Wirth 1978). All concerns for modeling other than geometric aspects of the objects are excluded here; the concepts discussed should be applied to the models of these other properties as well, but this is not within the scope of this paper.

This paper advocates applying the idea of canonical representation, as used in mathematics, to geometric representation. Rules should be set up in order that for each geometric configuration in real world there is exactly one (and not many equivalent) internal representation. Additional decisions have to be made to formulate such rules; rationales for such rules are not included here and will appear somewhere else (Frank 1984).

The organization of the presentation is as follows: In paragraph 2 we generally outline the areas of applications where we see use for geometric models of real world situation, and indicate the problems typically related to such models. Paragraph 3 will explain what we understand by model, reality, and information product and the processes which relate them. Paragraph 4 introduces our understanding of geometric properties and paragraph 5 discusses the limitation in precision inheritent in all measurement processes and the ensuing uncertainty about location of geometric objects in space. After this preparation we formulate in paragraph 6 the concept of canonical representations, the requirements for canonical representation and the advantages of their use. As an example we present in paragraph 7 and 8 a set of rules for a canonical geometrical representation. In the closing paragraph we will show some applications for these rules and discuss how these rules fit in the larger context of selecting representation for geometric properties in a data structure.

2. Areas of Application

For many areas of studies in Civil Engineering, urban planning, and environmental studies, geometric descriptions of certain aspects of the real world are necessary. For planning of military operations knowledge of terrain properties is extremely valuable and for many scientific studies in geography, geology, and forestry, geometric information is indispensable. Traditionally, such geometric descriptions of the world are collected, distributed,and used in the form of charts and maps of different kind and scales. In most operations, where maps are currently used, computer assisted operations might be applied potentially (not considering other restrictions, like availability, size, cost, etc.).

The concepts in this paper are explained using two dimensional examples from the cartography area. However, the same concepts must be applied to three dimensional models as well. In the earth science we primarily see applications for geometric models for geological formations used to support prospecting systems, but the same problems can also be observed in other applications where three dimensional models of real world situations are used; further, geometric models may be applied in medicine, where currently efforts to model brain tissue are undertaken). (.....).

This paper is less concerned with areas of applications where models of planned objects are produced (e.g. mechanical engineering, CAD, automobile and aerospace industry). Most systems in these applications use unique representations already, as these are the output from the modeling procedures, (not from data collecting in the real world).

3. Reality, model and information product

To avoid confusion, we shall briefly define how we use these terms. A more elaborate discussion was given in (Frank-Rom 1982) (Frank-Diss 1983). We name 'reality' the physical world as it exists independently of human observer. Human observers select certain aspects they perceive of this reality, the aspects they judge relevant for their plans, and form mental models of reality. In order to communicate about reality (more accurately, about the mental models), humans use conventional rules to represent their mental models in a form understandable to others (e.g. as a text for drawing). Such encoded data about reality can be treated in an electronic data processing system and the outputs, again represented in a form understandable to humans, can be used for information of human users.

Data on different levels of abstraction can be stored in a computer. The difference relevant here is the storage of a primary model, directly representing observable traits of the objects in real world and secondary or derived models (Hake 1970), which are indirect representations of the real world objects.

In computer assisted cartography manually prepared maps are often digitized and the data stored in computer-readable form; in these cases the computer treats a model of the map, not of reality. The treatment in the computer is then restricted to operations on the cartographic symbols, but the relationships between real world objects are not directly accessible, but only present, when human users interprete the map - a task impossible for a program.

This difference is not obvious as long as the computer system is used for storing the map and reproducing it, perhaps after some minor changes are made to update it. The computer is then used as a map editing system comparable to a word processor with which one can edit texts (Frank cis 1984).

Often users would like to use the map data for other tasks which seem easy - at least from a human user's point of view, when the data are interpreted (e.g. calculate the area of built surfaces or the shortest path along roadways etc.). As to the analogy with the text editor, nobody expects that a text editor system can automatically produce e.g. an abstract (although it would be useful). Regular text editors work on the level of understanding characters, lines, pages, but do not interprete the contents of the text. Similarly, map editors work on the level of graphics and should not be expected to understand anything about roadways.

However, if we store directly facts describing the observable reality and the relationships between them, we can create computer models of reality. If the relationships between facts we are interested in are expressed in the data stored, we can use programs to deduce from the stored data other facts or use the models to simulate potential changes to observe their effects (Johnson.....Urban dynamics). It is obvious that a computer model is much more useful for application than a stored form of map graphics.

We felt it important to clarify this point, as in the sequel we will concentrate on systems to <u>model reality</u> directly. The following discussion may also be useful for constructing graphic editors, but we do intend to address

mainly the problems of geometric modeling of two dimensional phenomena as we are usually dealing with them in engineering, planning and surveying.

From these computer stored models information products for users may be produced in a great variety, more than using a map editing system. Map output, in different scales and with contents and symbols suitable for the user's needs can be produced together with lists of interesting facts. The programs to output such human understandable products, however, are more complicated than the output procedures in a map editor: they must embody the basic rules of cartography and construct the map graphics from the primary facts much like a human cartographer.

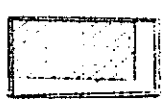
4. Geometric properties

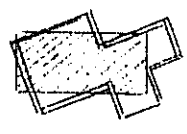
The description of geometric objects of reality includes not only a description of their position, extension, and situation relative to other objects, but also information about other properties of the objects, i.e. type of the object (house, road, etc.), specific properties (number of stories of house, class of road, etc.) and so on.

This paper addresses only the problem of describing the geometric properties independent of any other. It is attempted to reach a clearly layered design of programs where one set of modules can handle all geometric treatments independent of the meaning of the objects (Frank-Zurich 1984).

We assume that geometric properties of objects can be described in a vector representation using lines of defined form to delimit them from their surroundings, and points to indicate beginning and ending of lines and that other geometric properties can also be expressed using the concepts of points and lines.

The Alternative methods using rasters can also be subject to the concept of canonical representation. On a first attempt, however, it seems difficult to achieve, as a raster representation of an object depends on the axis selected for the raster.





It is not easy to see how this could be overcome, even using advanced methods of raster representations like quad trees ( ).

The vector representation is applicable in all cases where complete geometric descriptions (of arbitrary precision) for the objects of interest are available and these descriptions can be transformed into sets of points (given by coordinates) and lines between them. Traditional surveying as a profession is typically concerned with gathering such data.

However, it is thinkable - but no practical at the present time - that arbitrary descriptions of a geometry, not always containing sufficient data to deduce a fully coordinated geometric description, can be stored and treated in an information system. Manual systems of this kind exist, e.g. the mete and boundary descriptions of real estate property in the U.S. Such descriptions are very difficult to check for completeness and absence of contradictions. The complex body of rules to solve conflicts between contradictory real estate boundary descriptions or to infere missing information can be offered as a hint to the logical complexity of programs operating on such data. The type of computer system and programming language commonly called fifth generation (McGorduck 1983) may allow to treat such data in the future; they are however, clearly excluded from the treatment in this paper.

5. Limited precision of measuring

No measuring procedures can achieve complete precision but all methods yield results with a certain error (Leinfellner 1965). With increased effort and cost measurements can be made more precise, but there are definite physical limits (Heisenberg 1926). Current practice in surveying and geodetic measurements result in a precision of 10^{-5} to 10^{-6}; better results involve very elaborate advanced methods.

Whatever method used, relative positions of points in space can only be determined with limited precision. Measuring the position of a point with respect to some given points twice results necessarily in two different values for the position; the distribution of the resulting values can be described by statistical properties (assuming probabilistic properties of the measuring process).

All measuring methods yield relative positions of points, for the sake of simplicity, however, we prefer to express positions in a coordinate system. Coordinates are dependent on defining points, often called control points. Mathematic adjustment methods permit to calculate best estimates for the coordinate values of points; with certain assumption about statistic properties of resulting coordinate values can be computed, typically expressed as 'standard deviation values'. The results do not only depend on the relative position measurements but also on the choice of control points used. A formal treatment can be found in (Baarda 1976) (Grafarend .....), but it seems that methods suitable for practical application are not yet found.

6. The concept of canonical geometric representation

To form a computer internal model, the geometric properties - together with other properties - of real world objects must be stored in a form readable for the computer.

Several different methods are conceptually possible to describe geometric properties (Cox .....) (Alder .....) and they can be mapped onto different computer-internal representations. The latter depend heavily on the programming language used, the hardware available, etc. The discussion in this paper will be limited to using examples for the conceptual level of geometric modeling and not describe any internal realizations; the concept of canonical geometric representation must be applied on both levels equally.

Unfortunately, even within a selected method to describe geometric properties, a given geometric situation can be correctly described in different ways - mathematically spoken there is not a one-to-one mapping between geometry and description (an example follows).

If our concern is limited to producing graphical output from the internal representation, and the output from those different equivalent internal representations is the same because e.g. the sequence of the single lines drawn is not visible to the user, there is no need to worry about different representations.

If more complex operations should be carried out on the models to deduce new forms of information from the stored data, then the different but equivalent internal representations must be recognized by all algorithms, making these substantially more complex.

> The concept of canonical geometric representations says that internal representation of equivalent geometric situations in reality must be grouped in equivalence classes and one member in these classes selected as the canonical representation .
>
> This results in a one-to-one mapping from geometric situation to canonical representation, reducing generally complexity of treatment.

This is an application of ideas from the set theory:

> Relations which are reflective (a=a), symmetric (a=b ⇒ b=a) and transitive (a=b and b=c ⇒ a=c) are called equivalence relations.
>
> If for a set S an equivalence relation R is defined, S can be partitioned in disjoint classes, where x, y in S are exactly in the same class if x R y. The set of classes in S induced by R is often called the quotient set S/R. Mathematicians call the mapping from S to S/R the canonical mapping and a set C, which contains exactly one element from each class in S/R, a (canonical) representation system of S/R.

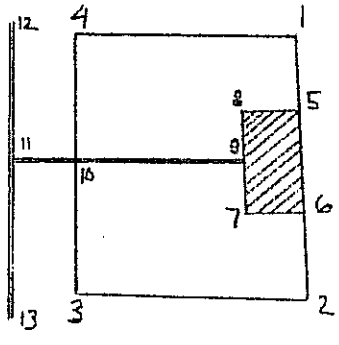
An example:

S is the set of rationale numbers, expressed as quotients (1/2, 2/4, 3/7, etc.). R is = denoting equal value ("=" is clearly an equivalence relation).

Now S can be partitioned in classes of different rational numbers with equal value e.g. (1/2, 2/4, 3/6, 5/10, etc.) or (3/4, 9/12, 12/16 etc.). Naturally we select from each class the rational number with the smallest denominator as the representant of the class. The set of reduced rational numbers forms a canonical representation system for the set of rational numbers.

From the set of internal representations of geometric properties we propose to form equivalence classes of representations describing the same geometric configurations. Rules must be defined to indicate which of these potential descriptions must be chosen as the canonical representation.

Before further exploring this idea, we present a simplified example. Consider the



situation in figure 1, showing a house on a small parcel and a utility line servicing this house. All lines are for simplicity straight and we have numbered all points for further reference by arabic numerals.

From the different possible methods to describe geometric situations of this kind, we select a description based on nodes (points) and edges (lines connecting nodes). We will denote a line by enclosing its two nodes in parenthesis e.g. (1, 2).

The method we use is not part of the concept of the canonical geometric representation, but convenient to describe the examples we will use. The concept of canonical geometric representation is independent of the method used and can be applied for different methods; a method is then said to fulfull the requirements of the concept.

Possible description of figure 1 are, among many others

a) (1,2) (2,3) (3,4) (4,1)
(5,6) (6,7) (7,8) (8,5)
(9,11) (12,13)

b) (1,5) (5,6) (6,2) (2,3) (3,10) (10,4) (4,1)
(6,7) (7,9) (9,8) (8,5)
(9,10) (10,11) (12,11) (11,13)

c) (1,5) (5,8) (8,9) (7,6) (6,2) (2,3) (3,4) (4,1)
(5,8) (8,7) (7,6) (6,5)
(9,11) (12,13)

As regards the graphical output alone, all of them represent faithfully the graphical aspects and can be used to reproduce the original figure 1 easily. (Obviously, slightly different semantics were used to produce these representations).

To produce other output or to answer other questions, the suitabilities of these representations vary:

Examples:

i - draw only parcel (suitable representation a and b)

ii - compute surface of house (suitable representation a and c)

iii - list all houses built on the boundaries of their parcel (suitable representation a)

iv - decide which houses are serviced by the utilities (suitable representation b)

In the setting of this example, algorithms can be devised to transform a representation in another, more suitable one for a given task. This can either be done for the whole data collection at once or for a limited area, relevant for the task. Such transformations are most often built into the program to deduce the question asked.

Many of such transformations must recur to comparison of coordinates of points to identify points or to decide if a point lies on a line or not (Example: question iv above, using representation a: it is necessary to decide that point 9 lies on the line of the house (7,8). For large collections of data, such operations are time consuming and, due to the inheritent errors in coordinates, can not be done reliably (cf next paragraph).

(Note the following might be left out)

A program can be considered as an executable formulation of an algorithm. Here we are only considering the abstract properties of an algorithm and do therefore not deal with the practical problems of programming languages, computers and the like. An algorithm is considered to be a transformation (mapping) of an input into an output (more precisely an input state into an output state (Dijkstra 19 ....). Useful algorithms produce certain output states exactly for certain inputs, whereas other inputs lead to other results and they terminate for whatever kind of input, producing either a result or an error indication.

For an algorithm and one of its results, we can establish a condition the input must fulfill for the algorithm to produce this result. We will call the description of the input the precondition for a given output (the postcondition).

Applied to a simple algorithm F, to calculate the prime factors of a positive integer, i.e. an algorithm that transforms positive integers to their prime factors, e.g.

-1	error	6	2·3
0	error	7	7
1	1	8	2·2·2
2	2	9	3·3
3	3	10	2·5
4	2 · 2		
5	5		

etc.

the precondition for a result $f_1 \cdot f_2 \cdot f_3 \ldots (\pi f_i)$ is $a = \pi f_i$, e.g. the precondition for the result 2·5·3 is the input value 30. This algorithm is simple as the number of inputs and outputs (at least in the useful range of the algorithm) is the same and output is a mapping of only one input value (the map is one-one).

An algorithm A to add positive integer numbers to produce a result of s has a precondition that a + b = x (a, b being the inputs), many different combinations of input values resulting in the same output. We can say that the algorithm divides the space of possible input values into equivalence classes yielding the same result.

(end of possible cut)

Formal expression of algorithm applicable to geometric data seems difficult; they are usually expressed as sequences of operations on a specific data structure ( ) or as logical assertions of qualities of data structures.

Only exceptionally operations are expressed geometrically, basically because the unbounded objects treated by Euclidian geometry (e.g. lines) are very different from the limited objects we are concerned with (line segments, areas, etc.).

This is unfortunate since it makes published algorithms applicable only if the datastructure they are based on and the datastructure used coincide or, at least, transformation routines between the two representations may easily be written. Too often a clear description of the underlying assumption for the algorithm is missing.

On the other hand, it is difficult to write algorithms to handle different representations of geometric data (e.g. write an algorithm to compute the area of the house which works on all representations above). A data-structure for an algorithm or an application is usually selected in order to make the programming easy.

The requirements of the canonical form can be considered as a help to select a representation. If an algorithm has only to deal with the selected representatives instead of all the different inputs of the equivalence classes, it is certainly easier to design and prove the algorithm correct. This becomes more apparent when we have to design datastructures and algorithms for a large application where we have to settle on one representation, even when it is not ideal for all algorithms we will need.

The rest of this paper will give an example of a method to select the representatives of equivalence classes and give a suitable datastructure. This is understood as an example of a canonical geometric representation and is not intended to describe the canonical geometric representation.

7. Canonical representation of points

Here we consider the presentation of a set of point, postponing the discussion of more involved geometric figures to the next paragraph.

The position of a point is easiest to represent by giving its coordinate values (for the required number of axes). The meaning of the point with respect to the real world object is here considered to be a point identifier (typically a point number) but more data about the point may be necessary for the application.

It is obvious that all points with the same coordinate values form an equivalence class (with respect to geometric position). Such an equivalence class is represented by the node with the respective position. It may or may not be possible to reduce the number of points with the same position, depending on the meaning of the points. Some applications use points with the same position but different meaning (e.g. different point numbers). This results in the following rules for a canonical representation of points:

P1 The position of a point is represented by one node. (A node may represent more than one point)

P2 No two nodes have the same position.

Under the assumption that the coordinate values are free of errors, rule P2 reads:

P2' No two nodes have for all coordinates the same values.

However, in the more realistic assumption of coordinate values given with errors due to e.g. inevitable measuring errors and/or rounding errors in computations with finite precision. In this case the decision whether nodes have the same position or not can not be answered using a simple comparison of their coordinate values. Even two nodes having different coordinate values can identify the same position; the difference in the coordinates can be fully explained by the expected errors in the values.

Needed, therefore, is a function "Same Position (p1, p2 position): Boolean;" which yields "true" if p1 and p2 are the same position and "false" otherwise. This function is then used to compare all points and to form equivalence classes each represented by a node with a representative position (e.g. the weighted average of the joint coordinates).

The following rules should be valid for this function Same_Position and the equivalence classes formed:

E1 Same-Position for a point with itself is always true. For any two points (including the representative node) in an equivalence class Same-Position is true).

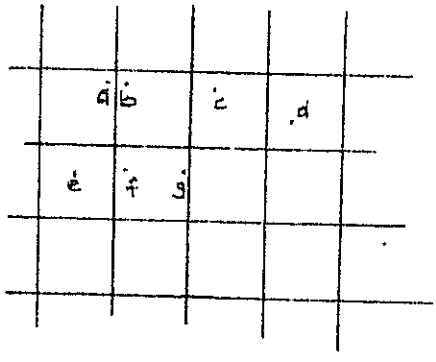
E2 no point is in two equivalence classes (i.e. for no two points in different equivalence classes Same_Position is true)

(This is the applied formulation of the abstract mathematical idea of equivalence classes induced on a set S of objects a, b, c... formed by an equivalence operation f (e.g. '=')

reflexivity:	f (a,a)	(e.g. a=a)
Symmetry:	f (a,b) = f(b,a)	(e.g. a=b = b=a)
transitivity:	f (a,b) and f(b,c) = f (a,c)	(e.g. a=b and b=c = a=c)

There is a trivial solution for the function Same_Position: the coordinate space is divided into areas for which a representative node is selected (e.g. as a regular grid, figure). Same Position is then true if both positions are within the same area and false otherwise.

Unfortunately, this solution does not reflect the meaning most applications attribute to point coordinates and the errors on coordinate values.



The two points a and b in figure 2 are represented by two different nodes, whereas the two points f and g, which are much more distant from each other given their original coordinates, are represented by the same node. This undesirable result is independent of the special methods we use to define the areas, and is true for all methods which define the equivalence classes independent of distances between the points to be stored. Whatever the method to form the areas for the equivalence classes, there must be limits between them. Two points just separated by this limit with an

arbitrary small distance between them will be represented by two different nodes when we would expect them to be represented by one, whereas other points sufficiently distant from each other are merged to one node.

Clearly Same_Position should base its decision on the distance between the two positions (and probably use some indications about the precision of measurements too). This seems to reflect ones understanding of measuring errors etc. better.

Unfortunately, a function Same_Position with these properties is not an equivalence relation since it is not transitive.

Assume: Same_Point (a, b) = true

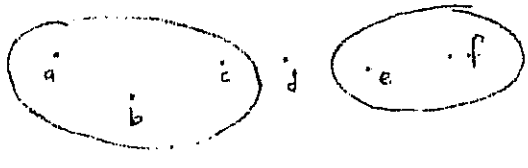
Same_Point (b, c) = true

it does <u>not</u> follow

Same_Point (a, c) = true

a b c
o o o

or, if we make it transitive, very large classes of equivalent points may be formed.

Assume 

a, b, c and d, e, f are already stored and equivalent under Same_Point.
Add a point g, for which Same_Point (b, g) = true

and Same_Point (f, g) = true.

Because we constructed Same_Point to be transitive, it follows that now all a, b, c, d, e, f, g form one equivalence class. In a special case this may lead to a situation where all points form one single equivalence class and are represented by the same node. This is obviously not useful.

It follows that a function Same_Point, based only on position information is not possible. (This is an effect of coordinate space being a connected topological space, a property very much needed otherwise). A distance-based function can be used to identify points which potentially may have the same position and then object semantics is used for the decision whether the two points must be represented by one node or not. Indications like same point identifier or similar may be used, and in many cases, a human operator must apply his judgement.

For normal applications only few points will have similar positions and will require human interaction. Special cases, however, are encountered when joining geometric data from different sources (e.g. digitalization of different maps). Rules to arrive at a semantics-based decision are needed and future systems for a logical inference can be used to introduce this expert knowledge.

8. Canonical representation of lines

In this paragraph we consider the representation of lines. We assume that the points required are already represented as nodes, according to the rules P1 and P2. Further, we will only discuss cases in which all lines are straight although similar arguments apply to curves.

The lines currently used to describe geometric properties of objects are represented by edges. The following rules are intended to guarantee that edges form a canonical representation.

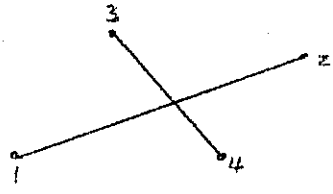
L1: Between two points only one (straight) edge is allowed.

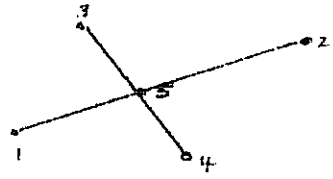
This excludes storage of the edge (a, b) and (b, a) as two distinct entites.

(In case of curved lines, a stricter formulation "edge of the same form..." is appropriate.)

The decision whether the edge (a, b) or (b,A) is chosen as the canonical representant can be either based on a logical predicate using the properties of the nodes a, b (e.g. the westernmost, southernmost is the the start node) or just the form first stored is used. A rule based decision results in an invariant for lines (the azimuth of a stored line is in the range (0, π, 0 included, π excluded) which can then be exploited to simplify other algorithms.

This rule alone is not sufficient to deal with the case in figure 3a and 3b:





The same geometric configuration, one time represented with (1, 2) (3, 4) and another time with (1,5) (5,2) (3,5) (5,4).

Both can be designed as canonical representations and transformation into this form must be performed. Other considerations lead us to select the second form (figure 3b). We call it the principle of explicite topological relations (Frank, Diss). It requires that relations, like incidence of a node with an edge or crossing of two edges, be represented explicitly as node-edge relations.

L2 If two edges cross, the crossing point is a node and the edges are split.

The same principle leads us not to select 4a (1,3) and to designate

(1,2) (2,3) as the canonical representation.

L3 If a node is incident with an edge, the node is splitting the edge and becomes the beginning point of one and endpoint of the other new edge.

or

L3 A node which is incident with an edge is either its beginning or ending node.

Using these rules, the representation b of the initial example in figure 1 is the canonical representation.

9. Conclusions

A canonical representation of geometric properties in a datastructure is possible. The concept of a canonical representation is to delineate a specific geometric configuration thru one specific internal representation and to avoid the situation where many different internal representations stand for the same geometric configuration. Different sets of rules can be established and other considerations will guide their selection.

In database oriented programming, we are creating datastructures independently of the (later) designs of algorithm using these data - this is very different from traditional programming where design of datastructure and algorithm is carried out in parallel (Wirth 1968).

Algorithm design is typically - sometimes consciously, sometimes unconsciously - geared by some properties of the data the algorithm works on. These properties, called usually invariants of the algorithm, indicating that the algorithm will not change them. If the truth of the invariants is established at the beginning of the execution of the algorithm, they are asserted at its end.

In database design, we must design these properties of the datastructure beforehand and independently of the algorithms later used. To guide our selection of properties, we may generally observe that algorithm design is easier the stricter the invariants are: meaning the algorithm has to deal with less different representation for the same input.

The concept of canonical representation for geometric properties is a maximal invariant; it requires that a set of rules is given, so that a simple geometric situation is represented in exactly one internal form. In this paper, examples for rules leading to a canonical form are given. However, when designing such a set of rules, other considerations come into play which we will elaborate on in a future paper.

The advantages of the use of canonical geometric presentation are most obvious in all applications, where a geometric pattern must be detected. This is necessary e.g. when a given situation has to be found on a large map (e.g. in a high level map query language where geometric patterns are acceptable for query description) or when comparing two descriptions of the same area from two different time epoches and changes must be detected. Based on our own experience, we believe that most geometrical algorithms are easier to formulate if operating on a canonical representation.

REPORT: Surveying Engineering Publications

1. "Defining the Celestial Pole," A. Leick, Manuscripta Geodetica, Vol. 4, No. 2

2. "A New Generation of Surveying Instrumentation," A. Leick, The Maine Land Surveyor, Vol. 79, No. 3

3. Article on the teaching of Adjustment Computations at UMO, A. Leick The Maine Land Surveyor, Vol. 79, No. 3

4. "Spaceborne Ranging Systems - A useful tool for network densification," A. Leick, The Maine Land Surveyor, Vol. 80, No. 1.

5. "Potentiality of Lunar Laser Range - Differencing for Measuring the Earth's Orientation," A. Leick, Bulletin Geodesique

6. Crustal Subsidence in Eastern Maine, D. Tyler, J. Ladd and H. Borns; NUREG/CR-0887, Maine Geological Survey, June 1979.

7. Land Information Systems for the Twenty-First Century, E. Epstein and W. Chatterton, Real Property, Probate and Trust Journal, American Bar Association, Vol. 15, No. 4, 890-900 (1980).

8. "Analysis of Land Data Resources and Requirements for the City of Boston," Epstein, E.F., L.T. Fisher, A. Leick and D.A. Tyler, Technical Report, Office of Property Equalization, City of Boston, December 1980.

9. Legal Studies for Students of Surveying Engineering, E. Epstein and J. McLaughlin, Proceedings, 41st Annual Meeting, American Congress on Surveying and Mapping, Feb. 22-27, 1981, Washington, D.C.

10. "Record of Boundary: A Surveying Analog to the Record of Title", E. Epstein, ACSM Fall Technical Meeting, San Francisco. Sept. 9, 1981.

11. "The Geodetic Component of Surveying Engineering at UMO," A. Leick, Proceedings of 41st Annual Meeting of ACSM, Feb. 22-24, 1981.

12. "Use of Microcomputers in Network Adjustments," A. Leick, Technical Papers, American Congress on Surveying and Mapping, September 9-11, 1981 (co-author: Wayne Welton, Senior in Surveying Engineering)

13. "Vertical Crustal Movement in Maine," Tyler, D.A. and J. Ladd, Maine Geological Survey, Augusta, Maine, January 1981

14. "Minimal Constraints in Two-Dimensional Networks," A. Leick, Journal of the Surveying and Mapping Division (renamed to Journal of Surveying Engineering), American Society of Civil Engineers, Vol. 108, No. SU2, August 1982 -

15. "Storage Methods for Space Related Data: The FIELD TREE", Frank in MacDonald Barr (Ed.) Spatial Algorithms for Processing Land Data with a Minicomputer. Lincoln Institute of Land Policy 1983.

16. "Structure des donnies pous les systemes d'information du territoire", (Date Structures for Land Information Systems) Frank in Proceedings 'Gestion du territoire assistee pas ordinateus', November 1983, Montreal.

17. "Semantische, topologische und raumliche Datenstruckturen in Landinformation-system (Semantic, topological and spatial data structures in Land Information Systems) Frank, B. Studenman in FIG XVII Congress Sofia, June 1983. Papers 301.1

18. Adjustment Computations, A. Leick, 250 pages; available from the Department of Civil Engineering.

19. Geometric Geodesy, 3D-Geodesy, Conformal Mapping, A. Leick, 380 pages; available from the Department of Civil Engineering.

20. Text for the First Winter Institute in Surveying Engineering, Leick & Tyler, 340 pages.

21. Adjustment Computations for the Surveying Practitioner, A. Leick, These notes are being typed (co-author: D. Humphrey, Senior in Surveying Engineering).

22. Advanced Survey Computations, A. Leick, 320 pages; available from the Department of Civil Engineering.

23. Surveying Engineering Annual Report, 1983-84.

24. "Macrometer Satellite Surveying," A. Leick, ASCE Journal of Surveying Engineering, August, 1984.

25. "Geodetic Program Library at UMO," A. Leick, Proceedings, ACSM Fall Convention, San Antonio, October, 1984.

26. "GPS Surveying and Data Management," URISA Proceedings, Seatle, August, 1984.

27. Adjustments with Examples, A. Leick, 450 pages; available from the

28. Geodetic Programs Library, A. Leick, University of Maine Technical Report, 1984.

29. Data Analysis of Montgomery County (Penn) GPS Satellite Survey, A. Leick, Technical Report, August, 1984.

30. Macintosh: Rethinking Computer Education for Engineering Students, Andrew Frank, August, 1984.

31. "Surveying Engineering at the University of Maine (Towards a Center of Excellence)," David Tyler and Earl Epstein, Proceedings, MOLDS Session, ACSM Annual Meeting, Washington, March, 1984.

32. "Innovations in Land Data Systems," David Tyler, Proceedings, Association of State Flood Plain Managers, Annual Meeting, Portland, Maine, June, 1984.

33. "Crustal Warping in Coastal Maine," David Tyler et. al., Geology, August, 1984.

34. "St. Croix Region Crustal Strain Study," David Tyler and Alfred Leick, Technical Report submitted to the Maine Geological Survey, June, 1984.

35. Applications of DBMS to Land Information Systems, Andrew Frank, in C. Zaniolo, C. Delobel (Ed.), Proceedings, Seventh International Conference on Very Large Databases, Carnes (France), September, 1981.

36. MAPQUERY: Database Query Language for Retrieval of Geometric Data and Their Graphical Representation, Andre Frank, Computer Graphics Vol. 16, No. 3, July 1982, p. 199 (Proceedings of SIGGRAPH '82, Boston).

37. PANDA: A Pascal Network Data Base Management System, Andre Frank, in G.W. Gorsline (Ed.), Proceedings of the Fifth Symposium on Small Systems, (ACM SIGSMALL), Colorado Springs (CO), August, 1982.

38. Conceptual Framework for Land Information Systems - A First Approach, Andre Frank, paper presented to the 1982 Meeting of Commission 3 of the FIG in Rome (Italy) in March 1982.

39. Requirements for Database Systems Suitable to Manage Large Spatial Databases, Andrew U. Frank, in Duane F. Marble, et. al., Proceedings of the International Symposium of Spatial Data Handling, August, 1984, Zurich, Switzerland.

40. Extending a Network Database with Prolog, Andrew U. Frank, in First International Workshop on Expert Database Systems, October, 1984, Kiawak Island, SC.

41. The Influence of the Model Underlying the User Interface: A Case Study in 2D Geometric Construction, Werner Kuhn and Andrew U. Frank.