REPRINTED FROM:

# PATTERN RECOGNITION IN PRACTICE

Proceedings of an International Workshop
held in Amsterdam, May 21-23, 1980

edited by

**Edzard S. GELSEMA**
Department of Medical Informatics
Free University, Amsterdam

and

**Laveen N. KANAL**
Department of Computer Science
University of Maryland, College Park, Md.

1980

NORTH-HOLLAND PUBLISHING COMPANY – AMSTERDAM • NEW YORK • OXFOR

ARTICULO 43

Pattern Recognition in Practice
E.S. Gelsema and L.N. Kanal (eds.)


# RECONFIGURABLE GEOGRAPHIC DATA BASES

Adolfo Guzmán

Computing Systems Dept., IIMAS
National University of Mexico

Information systems that handle geographic information usually store the data in one of these manners: (1) as a matrix of adjacent PIXELS, much as the LANDSAT Systems: (2) as polygons surrounding areas with uniform properties; (3) as shape descriptions with neighborhood relations among them. The time and difficulty used to retrieve, update, locate and transform the data, depend on the format it is stored.

In this paper a configurable data base is proposed, that dynamically changes its data representation, trying to provide best performance in view of current and past uses of it. This data base contains a cost function to be minimized, a statistics module that collects information of past usages of the data base, and a predictor that extrapolates the current user's behavior into future (possible) users behavior. Finally, according to the prediction, an reconfigurator module actually changes the format of the stored data to the most suitable format.

## INTRODUCTION

Geographic information is obtained in large quantities through the world. Airplanes and satellites such as LANDSAT and SKYLAB obtain each day vast amounts of pictorial data.

In order to efficiently use this information (see "The need for electronic paper"), it is necessary to improve the acquisition, distribution, storage and access to relevant parts of this data.

This article addresses the last two aspects of the problem: how to store the information in an economic way, and how to get it back efficiently when needed.

The first part of the paper describes and delimits the problem; the second part reviews actually available solutions. The main thrust of our proposition is in the third part, where a system is proposed that stores geographic information in several ways, and dynamically changes these storage formats, according to an efficiency criterion. The last part of the article describes other reconfigurable systems, one of which is already implemented at the University of Mexico.

The purpose of the paper is to invite criticisms and suggestions, since the proposed system will soon be implemented at the Computing Systems Department.

## LIMITATIONS OF GEOGRAPHIC INFORMATION SYSTEMS.

Memory. Pictorial and graphical information systems require large amounts of storage areas and fast access to it.

Fortunately, memory costs go down each year. Also, new memory types, between central memory and disk, have appeared (ccd's, bubble memories).

Also, minicomputers and recent microcomputers posses a large address space, which simplifies software writing.

In addition, each information level (each type of picture, map, detail and precision) has its optimum way of storing it. These ways have not been systematically combined in the same package.

Moreover, different uses of the same information have different retrieval speeds, according again to the storage format.

THE NEED FOR "ELECTRONIC PAPER"

Most of the geographic information is recorded in maps and pictures, and it is accessed with the help of ruler and compass, copied by hand and ink, much in the same manner of Ptolemaic times.

These archaic methods of storing and accessing information create a strong bottleneck that delimits its uses.

Fortunately, the computer and visual/geographic information systems [ 10 ] provide new ways to store and use this information. Each way of storage has advantages and drawbacks. Why not combine these different manners of storage through an "intelligent" system that dynamically choses the best for a given application or pattern of use?

CPU Time. To massage large quantities of data, a great amount of computer's time is required. Processing is slow.

Fortunately, computer prices are also going down, at the same time that their speed goes up, because of faster components and better architectures (pipelines, asynchronous computers, image machines [5,12, 13 ], heterarchical architectures [11 ]).

Data Acquisition. To acquire map information is expensive, as well as to digitize it. The same is true for image data.

Fortunately, a great part of the information is already in digital form, because it is acquired in that way, or it can be cheaply digitized if this is done at the early stages of its acquisition.

Languages. There is a need to design and use better picture-manipulation/picture-processing languages.

Some new designs [ 1 ] possess specialized data-types, and can be executed in parallel machines.

Man-machine interaction. The mutual collaboration between man and machine must be such that each part carries on the tasks best suited for it. It is also required that the common "language" for interaction should be picture related (application related).

It is relevant here to mention the efforts of Gaillat [ 6 ]to marry Detenal's geographic data bank [ 2 ] with the P.R. System [10], in order to produce a smart assistant to the photointerpreter.

PROBLEM DESCRIPTION

A bidimensional image or map is a set of geometric elements possesing attributes (properties). There are three types of geometric elements: points (well, ruins, airport), lines (rivers, boundaries, highways) and regions (lake, city, swamp). Examples of attributes are: population, altitude, chemical composition, slope. In addition, the adjective "geometric" indicates that each element has a position (x,y coordinates) in the plane.

A geographic data base is a data base containning items that are geometric elements. Since a set of geometric elements forms an image or map, it is appropriate to say that a geographic data base manipulates maps and images. This article deals with the problem "how to design geographic data bases that exhibit reconfigurability"

A data base is reconfigurable [ 7 ] if its contents can be stored in a different manner, changing the form or format in which they are kept, but keeping invariant the information contained. For instance, we can represent Lake Chapala [Figure "Lake Chapala"] (a) by its boundary; (b) by a collection of pixels; (c) by its shape number [ 4 ]. From the point of view of the user, (a), (b) and (c) represent the same lake in different formats. Now, if the data base automatically changes (due to some convenient reason) the storage format of Lake Chapala from (a) to (b), it is a reconfigurable data base. The user did not order the change; he is unaware of it.

Reconfigurable data bases are able to choose the best format for the information they keep, according to the patterns of use of the data. They need to have, for each format, an access mechanism; conversion routines (from one format to another) are also needed.

An attribute is inheritable if the subelements of an element also posses the attribute. Thus, "it is a pine forest" is an inheritable attribute, since each part of an element that "is a pine forest" is also a pine forest. The property (this town) "contains 2000 inhabitants" is not inheritable.

Inheritable attributes allow savings in memory, because it is not necessary to repeat the attribute for each subelement; the attribute can be placed at a "high level" element. This will become clear later.

## ACTUAL SOLUTIONS FOR DATA REPRESENTATION

A reconfigurable geographic data base has several manners to represent and store its data. It is now convenient to study them, in terms of occupied memory and also in terms of access methods and access time.

### Representation of points.

x,y representation. A point is represented by a pair of coordinates. The attributes associated to that point are represented by a collection of pairs property (attribute)-value.

### Representation of lines.

Freeman chains. A line is represented by the relative coordinates of sequential points. The coordinates of each point are expressed relative to the preceeding point in the line.

The inheritable attributes are associated with the line, and the non-inheritable attributes are defined along the line.

Storage of Freeman chains is compact. Reconstruction of the line is easy, but the "geometric" properties of the represented line are difficult to obtain; for instance "what is the shortest distance between two lines?".

Poligonal lines. A line is represented by the absolute (x,y) coordinates of sequential points. That is, the line is approximated by straight-line segments, and each vertex is recorded.

The inheritable attributes are associated with the line, and the non-inheritable attributes are defined along the line. For instance, "elevation" is defined along

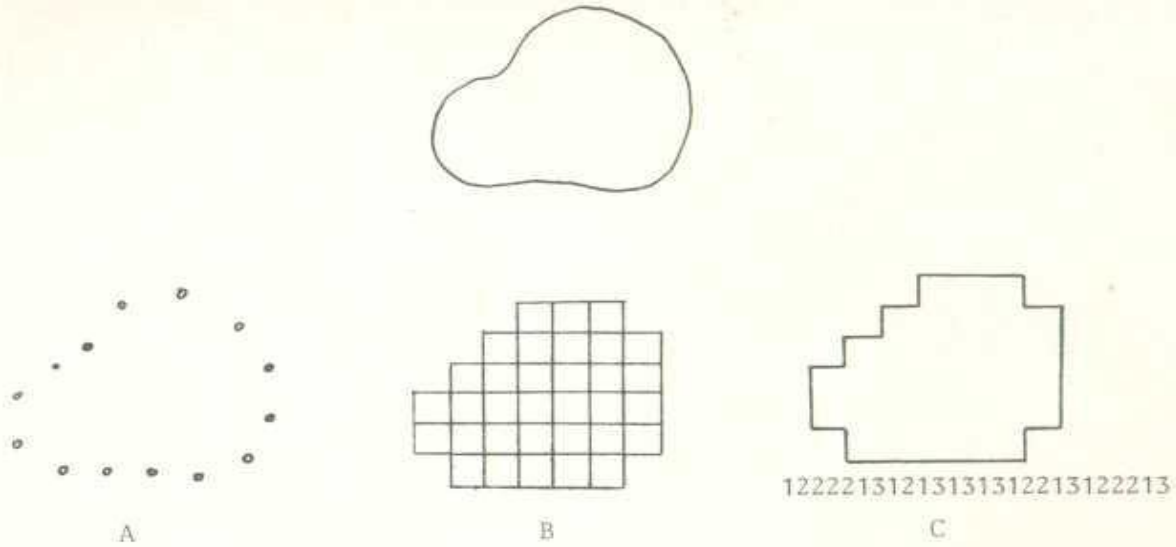



Figure "LAKE CHAPALA"

(A) Stored as a collection of boundary points: (3.5, 4.2), (3.6, 4.7),...
(B) Stored as a collection of pixels:
(C) Stored as a shape number: 12222131213131312213122213

In a reconfigurable geographic data base, you could store Lake Chapala as a collection of pixels and later discover that the information system now contains it as a shape number. Under what conditions is such change desirable? Read the section "The proposed system".

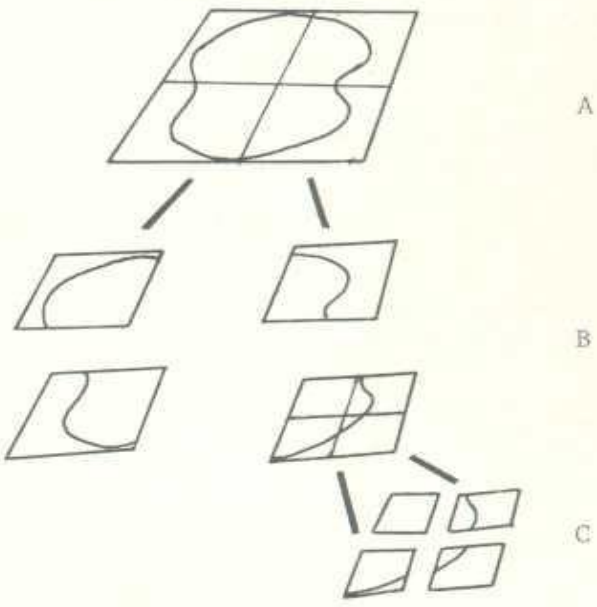



Figure "BRIBIESCA'S REPRESENTATION"

The region (A) begins to be subdivided (B) into a hierarchy of pixels (quad tree), but this subdivision is stopped prematurely (C). Each final leave (C) then carries as attribute the Freeman chain describing its partion of the boundary. This increases the precision with which final graphs can be obtained.

the line, v. gr., at fixed intervals.

This is a popular representation. It demands more memory than the Freeman chain. Reconstruction is easy.

Representation of regions

Pixels. Each region is divided in squares (pixels) of size dictated by the desired precision. Attributes are assigned to each pixel, as if it were a point. For this reason, it is not possible to distribute further non-inheritable attributes within a pixel. For example, if a pixel contains "5000 inhabitants", we can not know from this information if they are located near its center, or in its SE corner. Should this information become necessary, the picture has to be resampled (in order to measure the attributes again) with a finer pixel size.

Pixel representation avoids storing the x,y coordinates of each pixel; these coordinates are implicit properties [ 7 ], because from the position of the pixel information within the record, it is possible to deduce the coordinates.

Access to the information is fast, since it is possible to compute in advance the record position of the needed information. Geometric properties are easily obtained, too, because each pixel has known coordinates.

Storage of properties associated with each pixel follows one of these two ways:

(A) Access through the property. See figure 'Pixel Representation, (a)'. This is in widespread use. Each property is represented by a 2-d matrix of values. For instance, LANDSAT data consists of a "green image" stored in a large matrix containing numbers from 0 to 127 (the intensity in the green range), plus a "red image", plus an "infrared image", etc.

Other example: a matrix of altitudes.

This form of storage is advantageous when the values of the stored attribute exist everywhere (v.gr., altitudes). But it will be expensive to use for properties that show up only in a few places (copper deposits), since most of the data will consist of zeroes.

If the data base is going to frequently answer questions that combine several properties ("give me all the places with green>80 and altitude between 1500 and 2000"), it is disadvantageous to have the properties of the same pixel stored in different planes (matrices), since several records --one for each property-- need to be read from disk.

(B) Access through the pixel. The properties or attributes of each pixel are placed in a list associated to the pixel. Stored in this format, a LANDSAT image will be a matrix of 4-tuples of numbers, each 4-tuple being of the form 64-15-119-23, meaning "green=63, red=15, infrared=119, etc." See Figure 'Pixel Representation,(b)'.

For attributes that exist everywhere, storage size is the same for both (A) and (B). But now supposse that it is necessary to store "rare" properties, that show up only in a few places. Then we could use (B) with explicit properties [ 7 ], meaning that together with the attribute value it is necessary to store the name (which is generally a number) of the attribute.

That is, a pixel could have the follwoing property list:

GREEN - 64
COPPER - 30%
ALTITUDE - 1500

while another pixel could have the follwoing property list:

GREEN - 39
CACTUS - 22%
ALTITUDE - 1500
ARCHAEOLOGICAL MONUMENTS - 3
AIRPORTS - 1

Very well. Do not store inexistent attributes. Store the existing attributes as explicit properties. Nevertheless, these rules give rise to property lists of unequal length (See Figure 'Pixel Representation, (c)') which is fatal for quick access to a given pixel. It is necessary to keep all the list equally long. If this is accomplished, we may as well use arrays instead of lists. I have found two ways to keep property lists of equal length:

(1) In Detenal's geographic data bank [ 2 ], a "reasonable maximum length" is computed, and this size is assigned to each "list" (an array, in fact). If a pixel has fewer properties, space on its property list is left unused. If it has more, some properties are lost (not stored).

(2) In the Mexican Child data base [ 9 ]popular properties (those existing everywhere) are stored implicitly [ 7 ], but "rare" properties are stored in a list of fixed size, like Detenal's. Both (1) and (2) achieve the use of fixed size property lists.

Victor Germán Sánchez [ 7 ] abounds in the advantages of implicit vs. explicit properties, when to use which, and when to reconfigurate from one to another.

Quad trees. [14]. Each region is divided in an initial number of (big) pixels; to each of them properties are associated as explained under "pixels" above. Now, each of these pixels is considered under some homogeneity criterion (v. gr., that the gray level maximum difference between subportions of the pixel should not exceed 10% of the average; or that max height - min height < 10 m.). If it is homogeneous, it is left alone and it does not generate sons. If not homogeneous, it produces a fixed number (four, for the quad-trees of U. of Maryland) of children: These are pixels of smaller size. All of them collectively cover the same region as their father. Attributes are also associated to these smaller pixels, thus allowing greater accuracy in the representation. Each pixel is further considered for subdivision, and so on; the process stops when all the (final) leaves of the tree satisfy the homogeneity criterion.

The pixels need not be square; the brothers need not be of the same size or shape: Dora Gómez [ 8 ] uses, for representation of terrain and 3-d surfaces (elevations), quad trees of irregularly shaped triangles of different size.

Each pixel does not need to have exactly four sons. Bribiesca [ 2 ] uses, in Detenal's geographic data bank, a different number of sons for each level, in order to have layer of pixels representing a given map scale. See Figure "Tree of Pixels in Detenal's Geographic Data Bank".

Quad trees allow easy access to the information. If we only keep the attributes of the final leaves (those pixels having no descendants), storage shrinks.

Quad trees keep fairly well most geometric and neighborhood properties.

Polygon representation. Describe each region by its boundary; describe this boundary by a polygon (see "poligonal lines" above); that is, by a number of perimeter points.

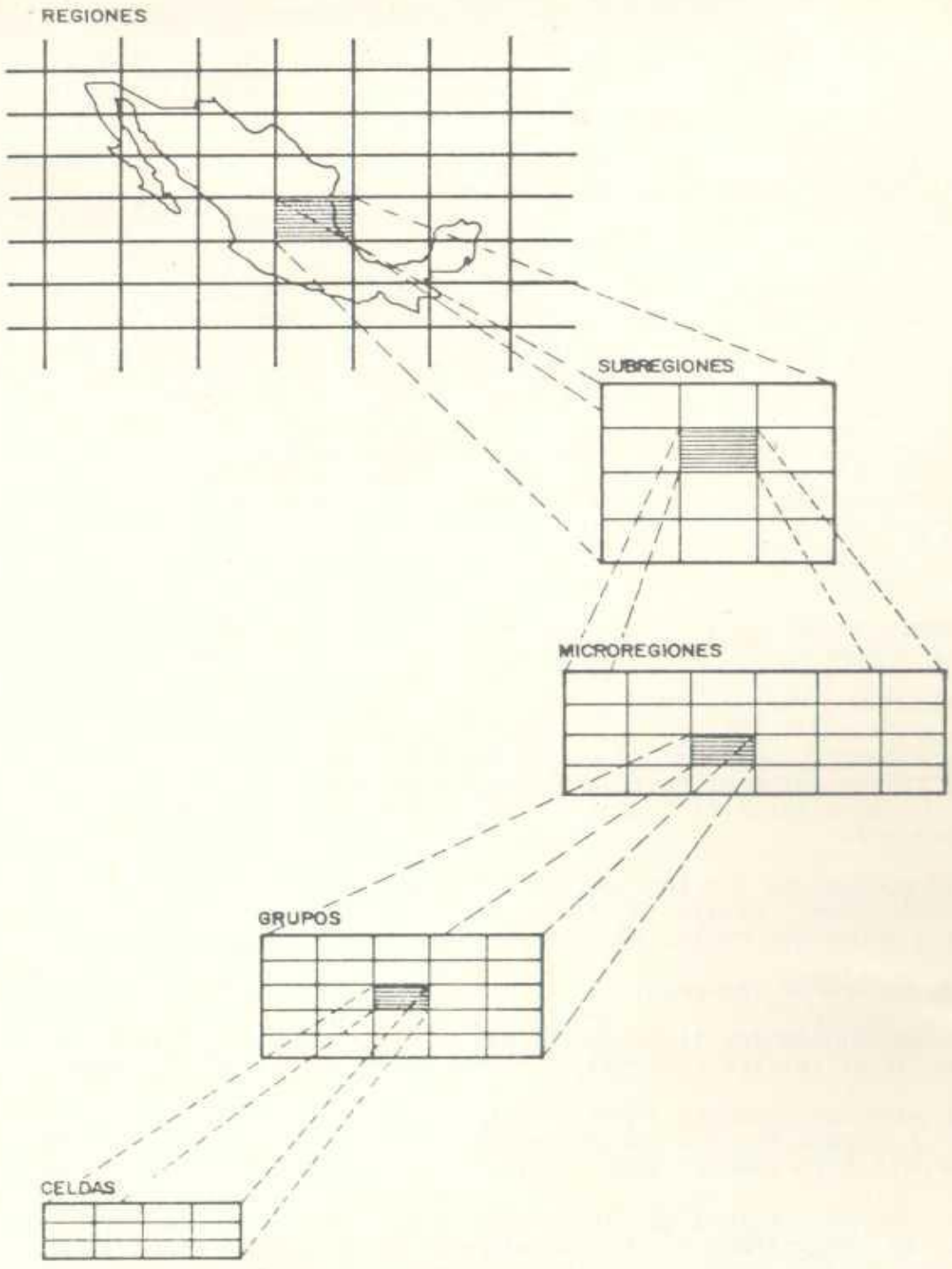



Figure "TREE OF PIXELS IN DETENAL'S GEOGRAPHIC DATA BANK"

Each layer is formed by pixels that represents maps at different scales. These scales determine the number of sons each pixel must have. At each level, all pixels have the same number of sons.

Properties are attached to each polygon.

Polygon representation occupies little memory. Nevertheless, to final region intersections ("give me all the deciduos forests that grow on top of sandy soil") is slow.

First pixels, then polygons. In a recent implementation of Detenal's geographic data bank, Bribiesca uses quad-trees up to a certain level, then stops. But at these leaves of the tree he then stores the Freeman chain of the part of the boundary of the region crossing that pixel. See Figure "Bribiesca's Representation".

This is useful when most questions asked to the data base do not need high precision in their answers (they are answered with the help of the pixels only), but the final maps need to be drawn precisely. Instead of a map with staircase-like boundaries, each step the size of a pixel, he uses the freeman chain within a pixel and obtains a much smoother boundary. And he avoids a lot of polygon intersections.

Shape numbers. When a region is considered disregarding ist size, orientation and position, it is seen as a "shape". From each shape, it is possible [ 3 , 4 ] to deduce a unique shape number, that allows easy comparisons between shapes. With the help of the shape numbers it is possible not only to decide whether two shapes are identical, but (if not identical), how close in shape they are. They give a metric or distance that measures the similarity in shape.

Shape numbers occupy very little memory. Unfortunately, operations other than shape comparison become difficult on them: intersection of two shapes, for instance.

## ACTUAL SOLUTIONS FOR DATA ACCESS

Geometric elements can be accesed (a) according to their geographic location; (b) according to the value of their attributes. It is very common to combine both mechanisms: "give me all the places within 10 kilometers of Popocatépetl Volcano having lava soil".

The access mechanisms for the attributes are precisely those used in general information systems: arrays, records, inverted lists, access vectors, hash cording, multilists, index sequential files, auxiliary keys, etc. [ 7 ].

The access mechanism for points is through their coordinates.

The access mechanism for lines is through the coordinates (or coordinates interval) of some point of it: its centroid, or a rectangle just enclosing the line.

For pixel access, to access a pixel with given x,y coordinates, these coordinates are used to compute the row (of the matrix) containing it. This record is then read from disk into memory. Then x gives us the position within the row .

To access a pixel within a quad tree, the tree is penetrated always from its root. Some interval comparisons of the coordinates of the needed pixel will reveal which of its sons contains such point. In this manner we descend towards the inner levels. To go down from one level to the next, some comparisons are required. Access is fast.

## WHY ARE RECONFIGURABLE DATA BASES NEEDED?

The data access mechanisms described above for geographic information (that is, for purely geometric elements) greatly differ from those used in conventional information systems. When the first methods are combined with the last, a great variety results. Each combination has its echological niche.

Nevertheless,

(1) The needs for accessing and the subsequent modification of the information may vary, as time changes, and a different form of data representation may become more attracrive.

(2) Even if each image or map is represented in its most efficient format, it is quite common for the data base to have to answer questions that need to access information from several images, hence to have to manipulate several formats.

(3) To transform each format to a canonical form each time a question is answered, in order to simplify the obtention of the answer, may become inefficient.

(4) To save the same map information in different formats is very expensive in memory, as well as in updating time.

On the other hand, conventional (i.e., not geographic) information systems exhibit some diversity of formats for storing their data. Recently, there has been some interest in the implementation of reconfigurable systems [ 7 ], even in the form of reconfigurable machines [ 12].

## THE PROPOSED SYSTEM

It is hereby proposed to design and to implement a geographic reconfigurable data base that:

(1) Accepts and uses the main formats employed in cartographic/geographic/image processing systems.

(2) Answers arbitrary questions about the information it contains, irrespective of the formats used to store it.

(3) Obtains these answers in an optimal or near-optimal manner, with respect to the formats of the involved information. That is, "does its best" when it is required to agglutinate information kept in dissimilar forms.

(4) Collects and extrapolates statistics about questions and updates, so as to decide to change the format of some portion of the information, if the trends show this to be desirable.

### The parts of the system

The reconfigurable geographic data base will consist of the following models:

- A syntactic analyzer/parser for queries/updates.
- An optimizer.*
- Access routines for different representations.
- Utility library to perform the operations.
- Reporter/graph maker.
- Statistics module.*
- Predictor.*
- Cost function.*
- Reconfigurator.*

With asterisks are signaled the modules needed to make to data base reconfigurable.

### How the system works

The syntactic analyzer converts input from the user into calls to the appropriate access routines and utility subprograms. The optimizer is needed before these calls are made, because the richness of formats makes more desirable to chose a good way

to perform the operations. Then the access routines bring the data into memory, and the utility subprograms perform the desired operations. These operations usually end with a report or map being produced: this is the job of the report/graph maker.

While all this is being done, the statistics module counts the number of access and updates. The predictor extrapolates past use into future behavior: it predicts what future use of the data base will arise. From time to time, a cost function is invoked to ascertain whether the current formats are cost-effective. If not, this cost function suggests a reconfiguration. This reconfiguration is performed by the Reconfigurator.

We now proceed to describe each of the starred parts: those modules needed to make the data base reconfigurable.

The optimizer. When an execution command is received, the form and order of execution of the operations is important. For instance, if the command makes necessary to perform operations between a pixel image and a map stored in polygon representation (as it was the case with Gaillat [ 6 ]), the optimizer decides whether to transform the polygon to pixel representation and perform the operations in that format, or, instead, to do everything in polygon representation.

The statistics module. This part of the system keeps several counters, for retrievals, access, updates, searches and the like. This information pertains to the "recent past" behavior of the data base.

The predictor. By extrapolation of the behavior of the data base in the recent past, the predictor calculates its behavior in the immediate future, taking notice of trends.

The cost function. Each form of storage has different costs in terms of speed of access, storage required, and cpu time needed to perform several operations. The cost functions contains all these parameters. From time to time, the system uses this function to compute the current cost (of the use of the data base in the immediate future). Specifically, it computes the cost of continuing using the data base with the current formats, as well as the cost of use with alternate formats. It then selects the best (least expensive) alternative, and indicates to the reconfigurator how it wants the data base to be changed.

The reconfigurator. Transforms the data base, changing the formats of storage of some of the items, but keeping unaltered its information. It basically consists of several transformation routines that go from one format type to another.

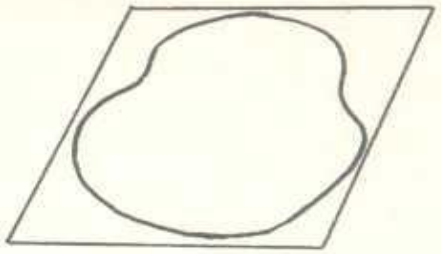
The reconfigurator works under request from the cost function. Since the task of reconfiguring also has a (high) cost, it should be taken into account in the cost function, at least in an approximate manner.

## CONCLUSIONS

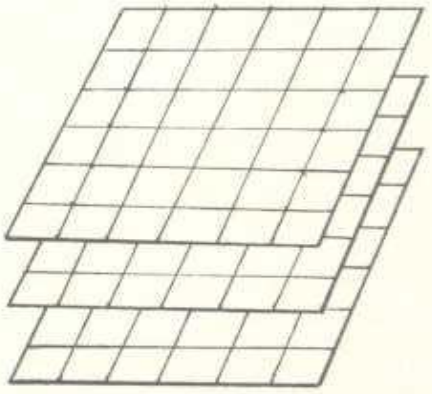
Geographic information systems perform useful and necessary tasks, in view of the large amounts of data and processing required. The formats that the information systems use for storing its data have several costs in terms of speed and memory requirements. It is therefore advisable to design a reconfigurable geographic data base, which always keeps its information stored in a best or near-best set or formats.

Drawbacks. The proposed system is larger than fixed-format systems, since it handles richer varieties of formats. The cost of the Reconfigurator may be large in terms of programming effort and running time.

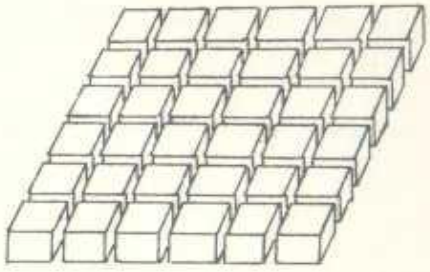
Advantages. The user needs not worry about how to capture its data, because it is internally converted, if necessary, into a near-optimum format. If the evolution



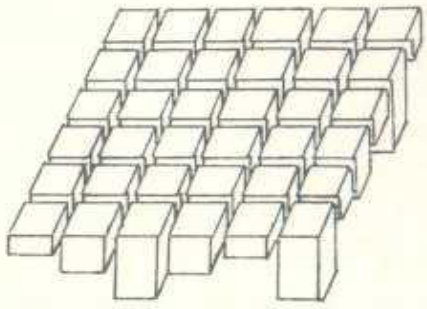
1



A



B



C

Figure "PIXEL REPRESENTATION"

(1) The region
(A) Access through the property: each attribute forms a matrix of values
(B) Access through the pixel: each pixel has a small list of attributes
(c) Access through the pixel: same as (B), but the lists are of unequal length

function is properly chosen, the overall performance of the system will surpass that of a fixed-format system.

Future work. A reconfigurable data base in already implemented [7 ] at the University of Mexico. We propose to implement the system described in this paper at the Computing Systems Dept. of the University. Main difficulties felt are: the evaluation function, and the optimizer.

ACKNOWLEDGEMENTS

My appreciation to the members of the Computing Systems Department, and particularly to Renato Barrera, who is coauthor of several ideas.

The AHR Project [12], with its reconfigurable computer, has provided a good place to work and think.

Work hereby described was partially supported by a grant (#1632) from the Consejo Nacional de Ciencia y Tecnología (México).

REFERENCES

1 Barrera, R., Guzmán, A., Jinich, J. Radhakrishan, T. Design of a High Level Language for Image Processing. IIMAS, 1979, PR-78-22. University of Mexico.

2 Bribiesca, E., Guzmán, A. User's manual to Exploit a Geographic Data Bank. CCAL-74-17, Scientific Center, IBM de México. Dec. 74. (in Spanish).

3 Bribiesca, E., and Guzmán, A., Shape Description and Shape Similarity measurement for two-dimensional regions. Proceedings of the 4th International Conference on Pattern Recognition Kyoto, Japan, 1978, 608-612. Also available as Comunicaciones Técnicas IIMAS-UNAM, 1978: 9, 166 (PR-78-18). IIMAS, National University of México. Accepted for Publication in Journal of Geoprocessing.

4 Bribiesca, E., A. How to describe pure form and how to measure differences in shapes using shape numbers. 1979 IEEE Conference on Pattern Recognition and Image Processing. Chicago, U.S.A. Accepted for publication in Pattern Recognition Journal. (1980).

5 Briggs, F.A., Fu, K.S., Hwang, K., Patel, J.H. $PM^4$: a reconfigurable multiprocessor system for pattern recognition and image processing. TR-EE-79-11, March. 79. School of Electrical Engineering, Purdue University. U.S.A.

6 Gaillat, J.P., Méndez, R. An intelligent assistant to the photointerpreter. Comunicaciones Técnicas IIMAS, Univ. of Mexico. 1979 (in Spanish).

7 Germán-Sánchez, Víctor. A general reconfigurable information system. M. Sc. Thesis, IIMAS, National Univ. of Mexico. To appear in 1980 (in Spanish)

8 Gómez, Dora and Guzmán, A., Digital Model for three-dimensional surface representation. Journal of Geoprocessing 1 (1979) 53-70. Elsevier Publishing Co.

9 Guzmán, A., et al. Banco de datos del niño Mexicano. Comisión Nal. para el año Internacional del niño. DIF. 1979.(in Spanish)

10 Guzmán, A., Seco, Rosa and Sánchez, Victor-Germán. Computer Analysis of LANDSAT images for crop identification in Mexico. Proceedings of the International Conference on Information Sciences and Systems, August 19-24, 1976. University of Patras. Patras, Greece.

11 Guzmán, A. Heterarchical architectures for parallel processing of digital images. IIMAS, 1979, AHR-79-3 (PR-79-23), Comunicaciones Técnicas, Serie Naranja. Univ.