An easy to use algorithm for the recalculation on a personal computer of imagery data in a generalised database of vectors.

Roose Hans
Laboratory of Cartography
Institute of Earth Sciences
Catholic University of Leuven
Redingenstraat 16b
3000 Leuven
Belgium
Commission number: WG IV/3

INTRODUCTION

Due to recent research and developments in micro-informatics, one is confronted with "maps" produced by lineprinters or by inkjetprinters, in a raster format.
Although the well known effect that an unexperienced map user does have problems with a "pixelised" image to recognize the individual elements, researchers in Remote Sensing and other cartographic oriented branches are often satisfied with receiving a printout or hardcopy of their matrix of data.
Next to this, most of the translations from raster information into plotable vector information is done on large, and expensive minicomputers or mainframe computers.

This paper describes an easy to use algorithm to translate raster oriented data (e.g. space borne, or air borne imagery data), into a dataset of vectors, using a simple microcomputer.
Starting from this algorithm, automated techniques for generalisation of the information were developed, so that the time-consuming process of interactive editing of large datasets can be avoided.
An attempt was also made to make an extension towards automated editing and removal of noise, gaps, unexisting connections,...

BACKGROUND AND INSTRUMENTS.

Peuquet (1981) has divided the raster to vector translation processes into two major groups: the line following process and the scan-line approach.
On the one hand, the line following process, in which each individual line is followed from pixel to pixel, in any direction, or with preferred directions, until the end of the line is reached, is found to be very simple, but timeconsuming (each new line needs a complete new scan of the information), even for mainframe computers, and needs large temporary datasets for holding the information.

The scanning of the image, on the other hand, takes on each data line the elements of the lines and expands the existing lines in the memory. This approach is less simple to program but still gives better and faster results than the line following process, although the computing time is increasing very fast, with an increasing number of unfinished lines.

This paper explains an adapted method of the scan-line approach.

For the research, we used a PC-XT (IBM-compatible) with 576 Kb RAM memory and a plotter (HP A1-format).

The complete program has been written in FORTRAN, knowing very well that this is not the fastest programming language, but it allowed fast and simple adaptions to run the program under mainframe conditions.

DESCRIPTION OF THE ALGORITHM.

To explain the principles of the processing, we take an example of the translation of a satellite image, made by SPOT, into a plotable dataset.

We have to consider that each image element, or pixel, has a surface in reality of 20 m by 20 m (for the multispectral images of SPOT). This means that we have to deal with four vectors per pixel, to surround it completely.

The whole algorithm is based on two working principles: the storage of the vectors and the processing of the new pixels.

Considering the processing of the new pixels, one can think of 10 different situations in which a pixel is added, or not, to an existing line or group (See Fig. 1.)

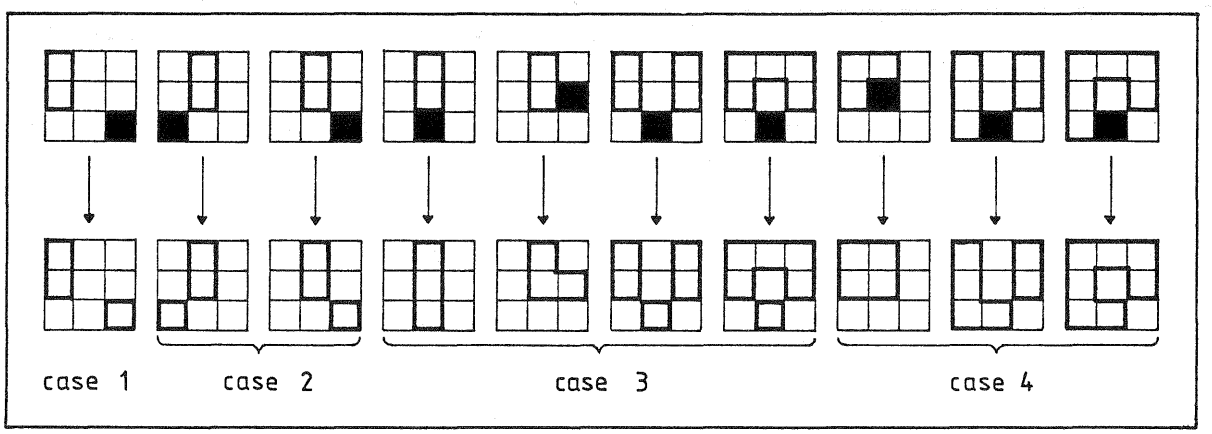

Fig. 1. Ten possible situations.

In the first situation, the new pixel has no connections with existing lines or groups. We consider it as a start of a new group, and we add four vectors.

In a second case, only one corner of the new pixel is already known to the processed lines. That certain line, to which the pixel belongs, is then expanded with four vectors.
A third possibility exists when there are two pairs of co-ordinates of the new pixel that are already known. Two subcases are here possible. On the one hand, the new pixel is only a extension of an existing line, and then we add two new vectors to that line. On the other hand, it's possible that the new pixel connects two existing lines or groups. Depending on the situation, a new group is added or an old one is absorbed by the "main" group. In both cases four new vectors are added.
In the fourth and final case, we know already three pairs of co-ordinates of the new pixel. Here we have again several possibilities. If the pixel lays simply next to a known group, only one pair of co-ordinates is changed, and nothing is added. Or, the new pixel makes a connection between two groups. Then, depending on the situation, a new line is added, or removed. In both cases two vectors are added.
Table 1 gives a small review of the actions to be taken in each case.

case	situation	n° of known vectors	n° of vectors added	n° of lines added or removed
1	1	0	4	+1
2	1	1	4	0
	2	1	4	0
3	1	2	2	0
	2	2	2	0
	3	2	4	-1
	4	2	4	+1
4	1	3	0	0
	2	3	2	-1
	3	3	2	+1

Table 1. Actions to be taken, depending on several situations.

The speed of the algorithm is mainly due to the fact that it uses random access files, with records which are keeping track of not only the co-ordinates of the vector, but also the record number of the preceding and the next vector so that a modification of a certain line only affects the corresponding vectors.
Next to that, due to the scanning process, the program only keeps track of the last processed line (this to know how to connect a new pixel) and of the line being processed.
Once a line has been completed or a group has been closed, this is, the last modification has taken place more than a line before the new pixel, the program translates the random access records into sequential plotable vectors, in order to reduce jumping in the vectorfile.

Some advantages of this way of working:
- vectors are quite a lot easier to deal with than the raster information when it comes to rotations, scale adaptions, changes in map projections,...
- the algorithm is independent of the size of the input. The only restriction is then the storage capacity of the computer.

GENERALISATION PROCESS

Although the algorithm does give a number of plotable vectors, one is still confronted with a "blocked appearance" of the plotted information. It is thus necessary to "generalise" this information, in an adequate way. We may assume that a satellite detects a road as a number of pixels, connected in a certain way (see Fig. 2A,B).

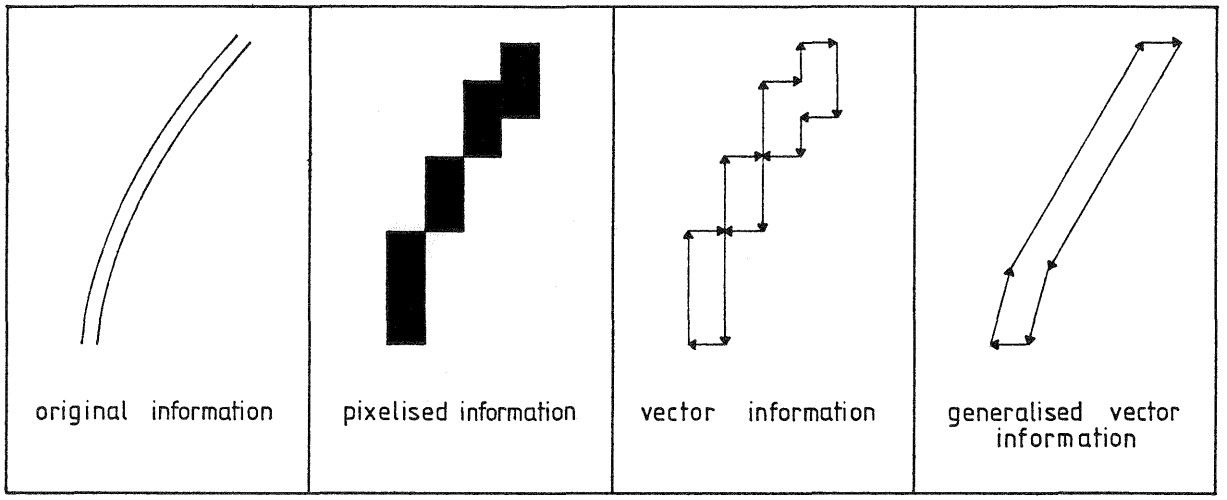

Fig. 2. Differences between reality, pixelised information, vector information, and generalised vector information.

If we use the algorithm without modifications, the output results in Fig. 2C. This is still not a good representation of the reality. It's better to find out how to generalise the information, so that an output as in Fig. 2D may be obtained.

One can think of some major problems. First, we have to keep the surface of the original blocked line equal to the surface of the generalised image.
Secondly, when different groups are laying next to each other, the generalisation has to work in both cases.
Thirdly, the algorithm has to find the right solutions in all possible situations. This causes large extensions of the algorithm.

It's not possible and useful to explain in detail all cases for generalisation. Only one is worked out here, to show the idea behind the algorithm.

If we take the situation as in Fig. 3A., one can see that the new pixel is a connection between two groupes and that we already know two of the four co-ordinate pairs of that pixel. Without generalisation, the algorithm woould solve this problem as in Fig. 3B., with adding four new vectors to the dataset and absorbing the right line into the left.
If we want to generalise this situation, the algorithm is going to check the values around the pixel, that is being processed, in the directions given on Fig. 3B.
If none of the surrounding pixels is of the same class as the central pixel, then we may use the generalisation process. This results in Fig. 3C.

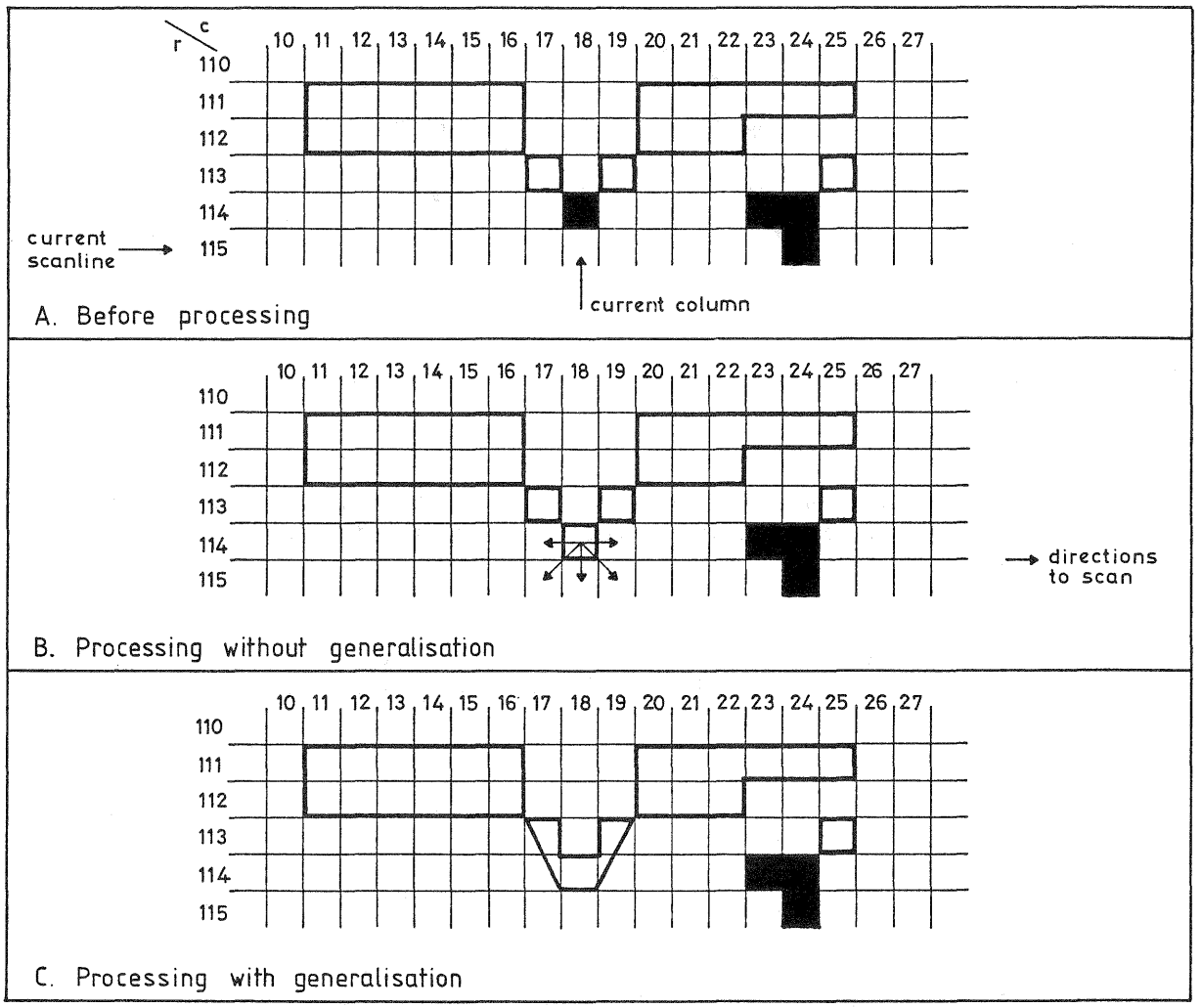

Fig. 3. Processing of a new pixel, connecting two groups.

The whole process of changes in the different records is stated in Table 2., using the same co-ordinates as in Fig. 3.

A. Before processing.						B. After processing.					
recn°	X	Y	linen°	prec.pt	foll.pt.	recn°	X	Y	linen°	prec.pt.	foll.pt.
1001	11	110	5	1010	1002	1001	11	110	5	1010	1002
1002	17	110	5	1001	1003	1002	17	110	5	1001	1003
1003	17	111	5	1002	1004	1003	17	111	5	1002	1004
1004	17	112	5	1003	1005	1004	17	112	5	1003	1005
1005	18	112	5	1004	1006	1005	18	112	5	1004	1006
1006	18	113	5	1005	1007	1006	18	113	5	1005	1028
1007	17	113	5	1006	1008	1007					
1008	17	112	5	1007	1009	1008	17	112	5	1041	1009
1009	11	112	5	1008	1010	1009	11	112	5	1008	1010
1010	11	111	5	1009	1001	1010	11	111	5	1009	1001
1021	20	110	6	1030	1022	1021	20	110	5	1030	1022
1022	26	110	6	1021	1023	1022	26	110	5	1021	1023
1023	26	111	6	1022	1024	1023	26	111	5	1022	1024
1024	23	111	6	1023	1025	1024	23	111	5	1023	1025
1025	23	112	6	1024	1025	1025	23	112	5	1024	1026
1026	20	112	6	1025	1027	1026	20	112	5	1025	1040
1027	20	113	6	1026	1028	1027					
1028	19	113	6	1027	1029	1028	19	113	5	1006	1029
1029	19	112	6	1028	1030	1029	19	112	5	1028	1030
1030	20	112	5	1029	1021	1030	20	112	5	1029	1021
						1040	19	114	5	1026	1041
						1041	18	114	5	1040	1008

Table 2. Process of connecting two lines.

The main effect of this large extensions of the algorithm is the improved readability of the map, that has been produced. Another advantage is that the resulting dataset of vectors is minimalised, since only the real begin- and endpoints of the linesections are withdrawn. This saves a lot of space to store the information.

Disadvantages are that the algorithm has to test more situations, which results in a slower program execution (although it's still far more preciser and faster than interactive editing).

FUTURE EXTENSIONS

Of course, this research is not finished yet. The algorithm is going to be extended towards processes of automated editing and removal of gaps and noise.
This implies a lot of decision rules for the algorithm to know in all sort of situations how to deal with certain information.

BIBLIOGRAPHY

Greenlee, D.D., 1987, Raster and Vector Processing for Scanned Linework. Photogrammetric Engineering and Remote Sensing, vol. 53, no. 10, pp. 1383-1387.

Douglas, D.H., and Peucker, T.K., 1973, Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. The Canadian Cartographer, Vol. 10, no. 2, pp. 112-122.

Peuquet, D.J., 1981, An examination of techniques for reformatting digital cartographic data, part 1: the raster-to-vector process. Cartographica 18, vol. 1, pp. 34-48.