ARCHITECTURE OF SYSTEM FOR CONFIGURABLE GIS DATA COMPRESSION

Jiri Komzak **Pavel Slavik**

Department of Computer Science and Engineering
Czech Technical University, Karlovo nam. 13,
121 35 Prague
Czech Republic
(komzaj1, slavik)@cslab.felk.cvut.cz http://www.cgg.cvut.cz/staff/komzak.php3

ABSTRACT

The paper deals with formal description of data transformation (compression and decompression process). An adaptive compression tool for different data types (both raster and vector), that is based on finite automata, is introduced. When using distributed geographic information system (GIS), data of different types has to be transmitted. The introduced tool enables flexible changes of compression method and selective loss control during compression process. Finite automata are discussed for several dictionary and other string-based compression methods. Application of one dimensional compression methods in the field of computer graphics is discussed.

Keywords: compression, geographic information system, flexible tool, finite automaton, selective compression, adaptive compression, multidimensional compression, lossy compression, LZW, RLC.

1. INTRODUCTION

The UK's Open University is one of the world's largest Universities, with over 160,000 currently enrolled distance-learning students distributed throughout the world. Requirements on development of a support tool for geographic position visualisation, off-line analysis and on-line presence and messaging arose, see [Komzak01]. A specialised GIS system is used to provide various information about OU students. Data used is stored in thematic layers and can be divided into several groups. Spatial data can be stored in raster or vector form. Since the system is distributed, the problem of low-cost data transmission from server to clients arises.

This problem is traditionally solved by data compression. The system transmits several different data types, which require specific compression methods. If each compression method was implemented separately, a special module would be needed for each data type or compression method. Thus a better way could be a general easy-configurable tool, compressor and decompressor, that allows usage of appropriate compression method after its configuration. At the same time, this approach brings possibility to use other compression methods without any substantial changes. Such a tool should also provide instruments for selective (adaptive) compression; it means selective loss in various areas of a picture.

Since the designed tool should be as flexible as possible, the formal description of compression process is needed for the tool construction and configuration. Finite automaton is one of the ways, how to describe in formal way the process of transformation between input information and output one. It reads input symbols and produces output symbols according to its internal state and the current input symbol.

Typical compression methods can be divided into three groups according to their principle [Salomon98]. Statistical compression methods, which create their models in dependence of probabilities of short parts (usually simple symbols), belong into the first group. The second group is constructed of compression methods, which replace repeated data with reference to its previous occurrence. The last group of methods predicts next symbol in dependence of precedent symbols and stores only the difference from this prediction. This paper deals

with the second group; the string-based compression methods and their application to graphics.

2. CODING-DECODING PROCESS ARCHITECTURES

As mentioned above, compression methods can be seen as a transformation of data from an original language to a resultant one, that both have a certain form. Suitable tool for unified description of string based compression methods is finite automaton. According to described method, the automaton can be static (data independent), semiadaptive (depends on data but remains unchanged during the compression and decompression process) or adaptive (constructed step by step as new states and transitions are added during processing).

The general compression process schema is shown in Fig. 3. Both compressor and decompressor execute translation by simulating finite automaton with its inserted transition and output functions. The system architecture contains features as adaptive change of transitions, linearization of input etc. (allows adaptive lossy compression of multidimensional data), that are discussed in the following chapters.

LOSSY VARIATIONS OF METHODS

Typical textual compressions are lossless. Another possibility of data reduction is to use lossy compression, which uses data loss to obtain better compression ratio. There is a possibility to transform a lossless compression principle to a lossy one. It replaces string by their more suitable representatives, which answer the condition of ”similarity“ to the original strings. For details see [Holub00].

When using a nondeterministic finite automaton for compression process description, it is necessary to prefer the most precise ending state of possible states with equal distance from the starting state. The typical part of system architecture for lossy variations of compression methods is inserting of additional lossy transitions into automaton (see Fig. 3). When using selective loss, activation of lossy transitions is controlled by currently processed symbols.

Lossy methods

The lossy methods are based on reduction of data (for example number of bits per pixel) and have no lossless variations. Thresholding (pure reduction of number of bits per symbol) and dithering (special techniques for global error reduction see [Slavik95]) are the basic ones.

The corresponding finite automata contain ending states for indices (representatives) reachable by inputting dictionary phrases (similar to LZW).

Extensions of lossless methods

The lossy RLE modifies symbols to create string of equal symbols as long as possible. The lossy variation of method LZW modifies the string of symbols to be equal to a phrase in the dictionary; it means already occurred sequences. Following example shows difference in string modification when coding *cdbdbdabdabc* by lossy RLE and LZW (allowed difference is one; single coded sequences are distinguished by underlining).

lossy RLE: ccccccaadaac
lossy LZW: cdcdcdacdaac

MULTIDIMENSIONAL DATA

The previously mentioned one-dimensional compression methods can be extended into more dimensions to use spatial context either by extension of methods to work directly with multidimensional data or by suitable data linearization (conversion of data from more dimensions into one) and usage of a known one-dimensional compression method. Not all one-dimensional methods can be extended to work in more dimensions (for example LZW). Special space filling curves are used for linearization preserving locality of data.

DATA FORMAT

Also other representations of a picture can be used as for example chain code, which represents only boundaries of objects. In such case, lossy compression changes directly shapes of objects. Modification of chain coded image by a variation of lossy RLE can be demonstrated by following example (for 8-directional chain-code).

2,1,2,3,2,3,2,2 -> 2,2,2,2,2,2,2,2

3. FINITE AUTOMATA

Finite automata for several compression methods (RLE, LZW, lossy RLE, thresholding, dithering and lossy LZW) were defined. We will describe one of them to illustrate the principle.

LOSSY TRANSITIONS

Lossy transitions can be used during the entire compression process or only for some specified areas. This can lead to a nondeterministic finite automaton, where the longest accepted string or for equally long strings the one with smaller error is used. This nondeterminism can be removed (see [Holub00]).

Lossy RLE

RLE method replaces strings of equal symbols by their number and value. Lossy variation of RLE

allows substitution of a symbol by a similar one to create longer string of equal symbols. Substitutions are limited by allowed difference in one symbol and maximal total error per string. The decompression is identical to the RLE decompression.

In following example an allowed difference per symbol equal to one is used.

compression

d/o	a	b	c
q_0	q_1 / ε *{n=1}*	q_2 / ε *{n=1}*	q_3 / ε *{n=1}*
q_1	q_1 / ε *{n++}*	**if err <= t** q_1 / **e** ***{n++}*** **else** fail / a n	fail / a n
q_2	**if err <= t** q_2 / **e** ***{n++}*** **else** fail / b n	q_2 / ε *{n++}*	**if err <= t** q_2 / **e** ***{n++}*** **else** fail / b n
q_3	fail / c n	**if err <= t** q_3 / **e** ***{n++}*** **else** fail / c n	q_3 / ε *{n++}*

In starting state q_0, the first symbol of equal sequence is read. Following equal or similar (bold underlined) symbols are counted in state for appropriate symbol. Similar symbols are accepted until the total error in sequence (stored in automaton's attribute *err*) is not too large (i.e. bigger then threshold *t*). When difference of symbol or total error are too big the fail function is called and symbol and its count are output. The fail function returns automaton to the starting state without reading input symbol. Special character '#' is used to indicate the end of input. After its reading, the fail function is called and symbol and its count are output.

Following example shows coding of string *baaaabc* to *b3a3c1* (used threshold is 2; configuration in form (state, input, output, number attribute, error attribute)).

(q_0, **baaaabc**#, ε, n=?, err=0) |-
(q_2, aaaabc#, ε, n=1, err=0) |-
(q_2, aaabc#, ε, n=2, err=1) |-
(q_2, aabc#, ε, n=3, err=2) |-
(q_0, aabc#, b3, n=3, err=2) |-
(q_1, abc#, b3, n=1, err=0) |-
(q_1, bc#, b3, n=2, err=0) |-
(q_1, c#, b3, n=3, err=1) |-
(q_0, c#, b3a3, n=3, err=1) |-
(q_3, #, b3a3, n=1, err=0) |-
(q_0, #, **b3a3c1**, n=1, err=0)

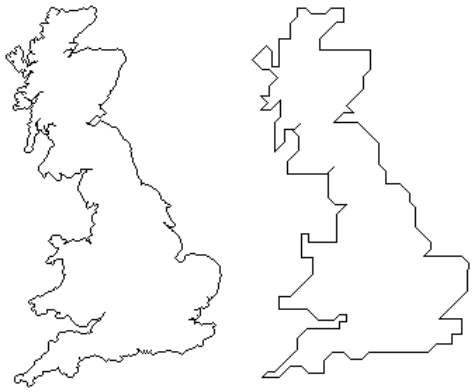
Other data types

Chain code representing a contour (for example a coast contour) can be taken as an example of other data type. Then, a variation of RLE with special error evaluation can be used for its compression. The finite automaton is the same except error attribute function, which in this case evaluates spatial distance between original and compressed contour.

4. TESTS

LOSSY CHAIN-CODE COMPRESSION

The compression of chain code was implemented by a variation of lossy RLE. The threshold attribute is computed as the distance of compressed point from the original one.



Lossy RLE of chain coded coast of British island (original and coded with compression ratio 8.04 and error/pixel 0.98).
Figure 1

LINEARIZATION

The mentioned linearizations using pass through rows and Peano and Hilbert curve were implemented. When using a lossy compression method a compressed data modification takes place and that is why so called artefacts appear. These depend on used curve and compression method.

LOSS CONTROL BASED ON POSITION IN DATA

In mobile computing, the available bandwidth is often a limiting factor, especially when images or multimedia data are used to communicate information. This leads to selective compression which allows small or none data loss inside some regions and bigger outside these regions.

The implemented compression methods allow loss control based on position in data. The data space is divided into parts so called regions of interest. Each region has a loss joint with it.

The Fig. 2 shows a compressed image divided into three regions with different data losses. The inner region has no loss. The one in the middle has 10% loss and the surround has allowed loss 20%. The size of this way selectively compressed image is one forth of original size.

Image compressed by 2D RLE with controlled loss.
Figure 2

5. CONCLUSION

A general adaptive tool usable for compression and decompression of different types of data in GIS was described. The tool uses finite automata for description of compression and decompression process for different types of picture data (raster and vector form).

Due to this approach, there is possibility to easy configure compressor and decompressor and use different compression methods. Another very important possibility is to optionally turn on and turn off lossy transitions during the compression process and in such a way to change the data loss even for each symbol. On the other side, the decompressor is universal; it means it can decompress data compressed by both lossless and lossy variation the same method without reconfiguration. This makes the tool very flexible. After linearization also multidimensional data can be compressed as well as other linear representations of image, for example chain code.

ACKNOWLEDGEMENTS

Jan Holub provided feedback and advice on the formal description.

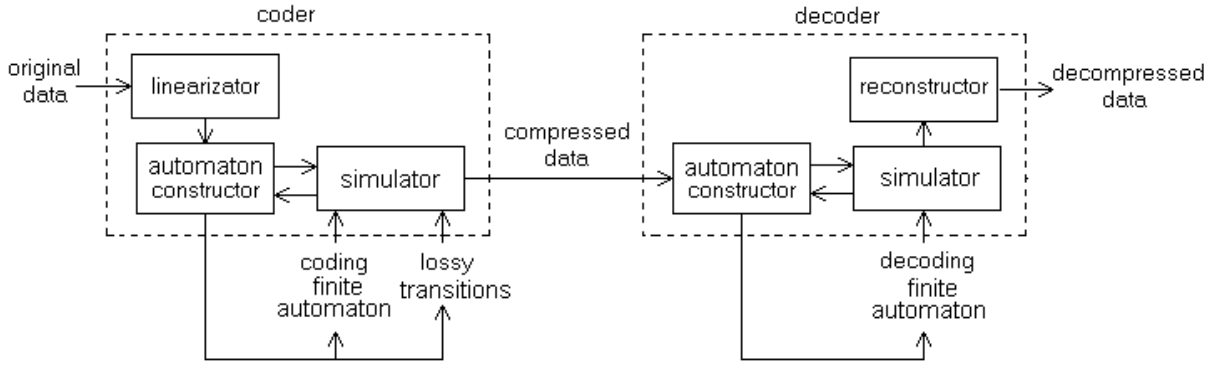
REFERENCES

[Holub00] Holub, J.: *Simulation of Nondeterministic Finite Automata in Pattern Matching*. Ph.D. thesis, Czech Technical University, Prague, February 2000, p. 118.

[Komzak01] Komzak, J. and Eisenstadt, M.: Visualisation of entity distribution in very large scale spatial and geographic information systems. KMI-TR-113, Knowledge Media Institute, Open University, Milton Keynes, UK, June 2001.

[Salomon98] Salomon, D.: *Data Compression*, Springer-Verlag New York, 1998

[Slavik95] Slavik, P. and Prikryl, J.: Dithering as a method for image data compression. *Proc. WSCG95*, Vol. II, pp. 283-288.



General system architecture.
Figure 3