Spatial queries on compressed raster images

How to get the best of both worlds

Report

Author(s):
Pajarola, Renato; Widmayer, Peter

ETH

Eidgenössische
Technische Hochschule
Zürich

Departement Informatik
Institut für
Theoretische Informatik

Renato Pajarola
Peter Widmayer

**Spatial Queries on
Compressed Raster
Images: How to get the
Best of Both Worlds**

October 1995

240

Renato Pajarola
Computer Science Department
Federal Institute of Technology
CH-8092 Zürich, Switzerland
e-mail: pajarolas@inf.ethz.ch

Peter Widmayer
Computer Science Department
Federal Institute of Technology
CH-8092 Zürich, Switzerland
e-mail: widmayer@inf.ethz.ch

This report is also available via anonymous ftp from ftp.inf.ethz.ch
as doc/tech-reports/1995/240.ps.

Spatial Queries on Compressed Raster Images: How to get the Best of Both Worlds

Renato Pajarola   Peter Widmayer

Abstract

The maintenance of large raster images, such as satellite pictures in geographic information systems, is still a major performance bottleneck. For reasons of storage space, images in a collection are maintained in compressed form. An operation on an image can be performed by first decompressing the compressed version. This, however, can be a major source of inefficiency, because the entire image needs to be accessed even though a small part of it might suffice to answer the query.

We propose to perform spatial queries directly on the compressed version of the image. With current compression algorithms, this cannot be done efficiently. We therefore propose a new compression technique that allows for the subsequent use of a spatial data structure to guide a search. In response to a range query, our algorithm delivers a compressed partial image; this response may be sent over a network as it is, or it may be decompressed for further processing. We have implemented the new algorithm, and we have compared it with standard and sophisticated compression algorithms (that do not support spatial operations). Our performance evaluation with satellite images shows that the new algorithm not only supports spatial queries, but is also competitive in terms of the compression that it achieves.

1   Introduction

Geographic Information Systems (GIS) are penetrating a growing number of application domains, including cartography, urban planning, risk assessment, pollution control, and transport management systems. Since GIS users ask for complex operations with GIS data, efficiency is one of the main bottlenecks of today's GIS. In particular, we need data structures and algorithms for maintaining geometric objects on secondary storage in such a way that queries referring to the location of objects in space can be processed efficiently. Over the past decade, this problem has been attacked with significant effort, for geometric objects ranging from aligned rectangles to polygonal partitions, and for geometric operations that focus on range queries as a typical representative of geometric queries. From a practical point of view, we feel that it turned out that taking one out of a number of efficient spatial access structures is mandatory, but it does not matter so much which one.

The efficiency problem, however, is far more severe (in absolute terms) for raster data, because here, the data volume is far larger than for vector data. A single satellite image, for instance, with its multiple data channels and high resolution quality, is easily as large as several hundred megabytes. Unfortunately, spatial access to raster images has received relatively little attention. On the one hand, efficient access is possible by storing raster data in one of several data structures, for instance quad trees [12, 13], or by ordering the raster data according to expected access patterns [14]. On the other hand, efficient access alone is not what we need: due to the enormous data volume, we would like to compress the data as much as possible.

Over the years, quite a few data compression techniques have been developed that aim at eliminating redundancy from raster images [10]. For archiving a raster image or for sending an image over a network, compression is particularly useful. It has, however, an adverse effect on query processing: decompressing an image requires that the whole image is accessed on external storage; in addition, the decompression algorithm may take a lot of computation time. Whenever a query does not refer to the full image, some of this effort seems to be wasted. As raster images become larger and larger, partly because satellite snapshots can be glued together into one image easier and easier, partly because the resolution increases, it becomes less likely that even a significant part of an image is needed to answer a query. It seems crucial that in this setting, access to a small part of an image is supported on the compressed version of the image.

2   The Problem

In this paper, we consider the problem of answering a range query on a compressed version of a raster image, without decompressing the entire image. We restrict ourselves to lossless compression; our justification is theoretical interest as well as practical demands, such as the high cost in acquiring a satellite images. The range query serves as a particular, but proto-typical spatial query. Ideally, we would like to find a raster data structure and compression technique with the following properties:

1. the physical clusters of the pixels of the raster image reflect the pixels' locations in space; this implies that pixels in the same storage block are close in space;

2. a range query can be answered by accessing just those storage blocks that contain the compressed version of the data in the range;

3. the compression ratio is as good as that of any other well-known image compression algorithm.

Property 1 is interpreted to imply that the geometric shape of the union of all pixels in a block (or some container for the pixels, such as the bounding box) is compact and good for efficient spatial indexing; see, for instance, the criteria for efficiency of spatial access structures based on aligned rectangles ([2, 11, 1]).

These properties together call for a solution that combines the characteristics of the best available spatial data structure with those of the best available compression algorithm. The problem now lies in the fact that the compression technique must lend itself to spatial clustering. Any compression technique, for instance, that makes use of the statistics of the whole image will be unable to fulfill requirement 2. Unfortunately, this is just what the best compression techniques do. On the other hand, we can hope that by exploiting local (w.r.t. a single storage block) statistics for compression, the compression ratio might be quite good.

We propose a solution to the compressed raster image handling problem that satisfies all of the above requirements. In Section 3, we discuss the basic issues in the design of a suitable compression method. We propose a compression algorithm in Section 4. Section 5 sketches the spatial access to the compressed image. In Section 6, we present the experimental results that we gained from an implementation of various compression algorithms and experiments with satellite images. Section 7 concludes the paper.

3   Basics of Compressed Raster Image Handling

To combine compression with spatial access, two clustering strategies seem applicable at first sight: a partition of the data space based on the original pixels, or a clustering based on the compressed data. The former one may use any partition of the data space into rectangular regions, where each region serves as the two dimensional key to the data block(s) holding the compressed image data of the region. The second strategy builds its regions according to the amount of compressed data that fill one or more entire data blocks. However, any fixed partition of the space of original pixels leads to regions whose compressed data may not fill entire data blocks, because the compression ratio is not uniform all over the image (see Figure 1). Therefore, in a case in which the compressed data fit into one block, two blocks may need to be accessed. Some (but not all) of this inefficiency can be avoided by scanning the image along an appropriate *Space Filling Curve* (SFC) [4, 8] and creating index regions – bounding boxes of data blocks – which cover the area of the compressed data in one data block.
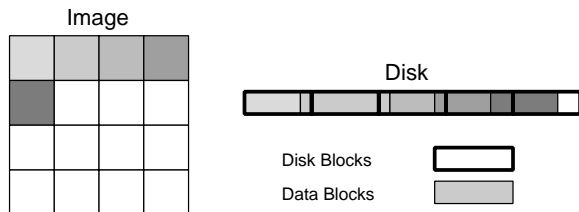

Figure 1: Logical Clustering

To permit efficient clustering, spatial access and lossless compression at the same time, an algorithm and data structure have to obey several restrictions. First, there should not be any free space in a data block, because this would affect the space usage and therefore the compression ratio negatively. Second, to keep the number of block accesses for (range-) queries low, every index region should smoothly map to a data block, and therefore the compressed data should not continue beyond data block boundaries. Third, the compression algorithm is not allowed to take the statistics of pixels encountered in other data blocks into account when coding the elements of the current block's region. The reason is that this would lead to some extra block accesses for reconstructing this part of the image. Therefore, we restrict the compression scheme to local operations and disallow the use of statistics or history of other image parts.

Unfortunately, none of the known compression algorithms fulfills our requirements, neither the highly sophisticated nor the more standard ones. The newly proposed (highly sophisticated) standard *CALIC* [17] achieves high compression ratio based on complex prediction functions and extensive context statistics; for our problem, this collides with the requirement of history-less operations. Another sophisticated method called *FELICS* [6] is also based on image statistics, and the current standard *JPEG* [15] has drawbacks in the need for prescanning the whole image to achieve good compression, and that a certain amount of statistics be maintained and stored with the compressed data. Furthermore, the prediction functions of CALIC, and more or less also those from JPEG and FELICS, depend on the knowledge of some particular neighboring pixels; this imposes restrictions on the traversal order of the pixels in the image, and it therefore makes the construction of well-shaped pixel regions impossible.

4 A Spatial Compression Algorithm

As discussed in [5], lossless image compression algorithms usually view a pixel value as a random variable. An algorithm then typically consists of four processing steps: first, select the location of the next pixel to be encoded (*pixel selection*); second, compute a prediction of the value of the selected pixel from the history of already encoded pixels (*pixel value prediction*); third, compute a prediction for the variance of that random variable (*variance estimation*); fourth, encode the difference between the pixel value prediction and the actual pixel value, making use of the variance estimate (*prediction error coding*). The requirement to do dynamic clustering based on the compressed data as mentioned in Section 3 leads to some restrictions on the pixel sequence. The scan-line sequence, as used in most compression algorithms, leads to poor spatial selectivity of the data blocks compared to more sophisticated space filling curves. Figure 2 shows the resulting data regions for three different pixel sequences with equal block capacities of 17 pixels; the first four resulting data regions (A, B, C and D) are shown in different colors. The Z-curve (Figure 2 a), which can have heavily overlapping regions (i.e. A and B), and scan-line (Figure 2 c) sequences may even produce data regions with multiple connected components. For the pixel sequence we propose to use the Hilbert-curve, since it generates quite compact data regions, and it also supports the prediction step quite well.
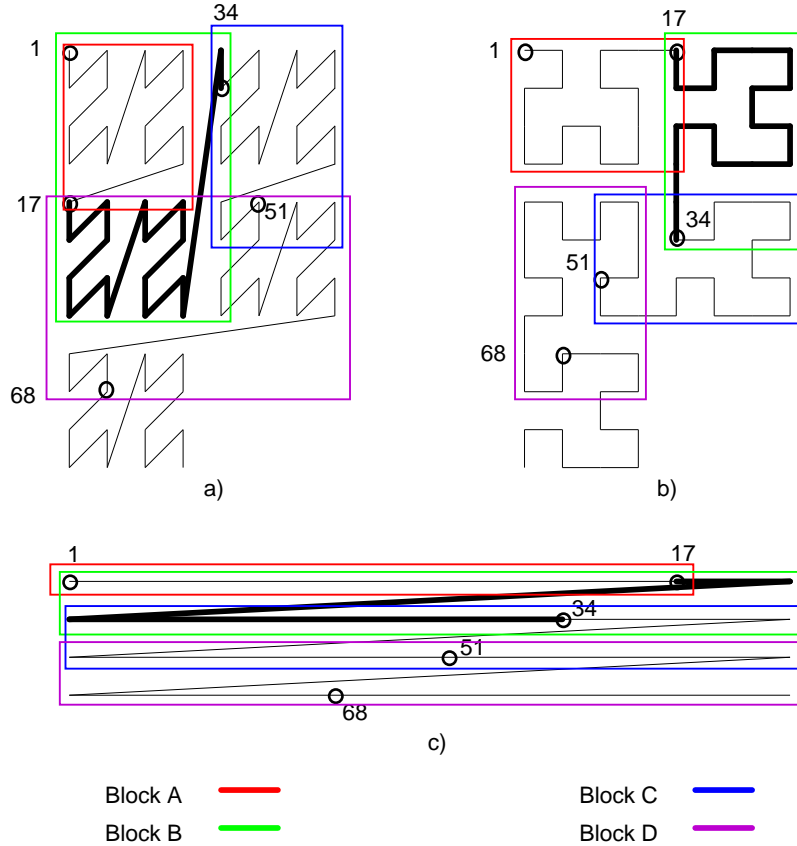

Figure 2: Clustering Sequences

Prediction is usually based on neighboring pixels in space. However, the pixels used as predictors for the current value have to be encoded before the current one, because the decompression process has to use the same predictor values. If the relative spatial positions of the neighbors always have the same relative index in the sequence, as it is the case for

scan-lines, sophisticated combinations can be used to derive a close prediction (CALIC makes use of this fact, e.g.). Space filling curves like the Z-curve or the Hilbert-curve don't conform to this behavior but are useful for effective clustering. We therefore propose to calculate prediction from a specific set of relative indices (in the pixel sequence) of previously encountered pixels. However, the relative spatial positions of these pixels are not the same throughout the sequence. Furthermore, their influence based on the Euclidean distance differs from one space filling curve to another. Let us call the set of pixels used to predict the current pixel's value the *context*.

We propose to choose the context C_v of a pixel v to be the ordered collection of the last k pixels visited before coding the current one. We now face the problem of computing a prediction that is good in all the geometrically different situations that the context describes. To get a close prediction $\hat{v}$, we propose the weighted sum of equation 1 over the context C_v; this incorporates neighboring pixels according to the space filling curve (SFC), but doesn't rely on exact relative locations. The weights w_j resemble an exponential function, decreasing with growing distance on the SFC, and depend on the size k of the context and on the index j of pixel v_j in the sequence. Thus the distribution of the weights w_j takes into account that pixels close in space which also have smaller relative indices on the SFC will also have a bigger influence on the prediction.

$$
\begin{aligned}
w_1 &= 1 \\
w_j &= 3\lfloor \frac{w_{j-1}}{2} \rfloor + 2 \\
\hat{v} &= (\sum_{j=1}^{k} w_j)^{-1} \sum_{v_j \in C_v} w_j v_j
\end{aligned}
\tag{1}
$$

To achieve good data reduction, an accurate probability distribution has to be selected for the current prediction error. The *Laplace* distribution of equation 2 is normally used as a basis for statistical coding of differential signals [10, page 225]. In this case, the best compression ratio for a value x is achieved with the Laplace distribution with variance x^2. This value, however, cannot be used for compression, because it makes decompression impossible, since the value x isn't known at decompression time. Therefore, we need a good estimate for the variance.

$$
L(x) = \frac{1}{\sqrt{2\sigma^2}} \exp(-\sqrt{\frac{2}{\sigma^2}} |x - \mu|)
\tag{2}
$$

Naturally, our variance estimation uses the same context C_v as the prediction step. Unlike other compression algorithms, however, we do not use this context as a key to retrieve information about earlier prediction errors within the same context. Thus no history or statistical information of occurrences of this context are used for variance estimation; we only use local operations on the values in C_v. We propose to calculate a variance approximation σ^2 according to equation 3 as a normalized sum of the quadratic prediction errors in the current context C_v.

$$
\sigma^2 = \frac{1}{|C_v|} \sum_{v_j \in C_v} (v_j - \hat{v})^2
\tag{3}
$$

The prediction errors $e = v - \hat{v}$ can now efficiently be coded with a statistical entropy coder, using the estimated variance σ^2 to calculate the probability distribution at the current

position. Note that *arithmetic* coding [3] would attain the theoretical entropy bound of the compression ratio for a given probability model. However, the lack of an exact link between the symbols of the input sequence and the bits of the output binary rational number would interfere with the division of the compressed data into many different data blocks. A *Huffman* entropy coder [7] generates minimum redundancy codes which have a one-to-one relationship with the input symbols, and it also approximates the entropy bound for a given model. To reduce coding complexity, the Huffman tables are only computed for a specific set of variances, as suggested e.g. in [5].

Figure 3 shows these processing steps in a flowchart. Prediction and variance estimation are performed on a sliding window over the input sequence, where only local operations on a limited support area around the current picture element are used. The entropy coder which encodes the prediction errors makes use of precalculated tables to save calculation time. The decoding process is the symmetrical inversion of the encoding process. For ease of reference, let us call the described algorithm *Hilbert Compression*.
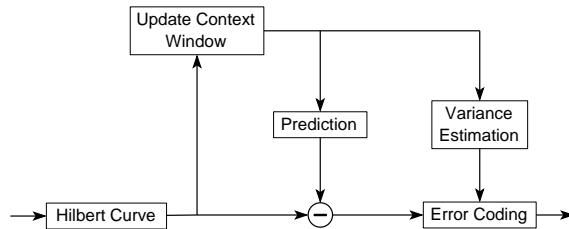
Figure 3: Coding a Pixel

The runtime of the algorithm is quite low, even without code optimization. To calculate the prediction and variance estimation, only a few computation steps are necessary ($O(k)$ for a context of k pixels), whereas other algorithms tend to spend more time on complex prediction computation. The main work of the entropy coder is done once in the initialization step. The probabilities of symbols are computed for a specific set of variances, and for each such variance the Huffman tree is constructed. Therefore, a pixel is coded in $O(m)$ steps by outputting the m bits from the path in the Huffman tree. Not only time complexity is low, also the space requirements are small. The main part of the coding routine just uses an image buffer of size k to compute the prediction and variance estimation, the precalculated Huffman trees need $O(nl)$ space, where n is the number of symbols in the alphabet and l is the number of variances used.

5   Spatial Access

Our spatial compression algorithm, Hilbert Compression, cuts the pixels of the raster image into sequences along the Hilbert curve, where not all sequences have the same length on the curve. As we have indicated in Figure 2, a bounding rectangle for each pixel sequence serves as geometric key for storage and retrieval. In the example shown in Figure 2 (b), rectangles A, B, C, and D are the geometric keys for maintaining the pixels from 1 up to 68. As a consequence of the properties of the Hilbert curve, the shapes of these bounding boxes tend to be almost square and to have very limited overlap. They are therefore a good basis for a spatial data structure that maintains a set of aligned rectangles, such as the R*-tree [2]. We simply advocate the use of any such data structure to maintain the compressed raster image. Note that operations on the compressed raster image then amount to the corresponding operations in the spatial data structure, with an additional mapping that needs to be computed between the position of a pixel in the compressed sequence and the location of the pixel in space.

8

6   Experimental Results

We have implemented Hilbert Compression (H.1.6 in Table 1) in C, based on the *Khoros* [1]
image processing system. The other compression algorithms used to compare the compression
efficiency are: *gzip* (gz) and *compress* (Z), two well known general purpose compression tools
available on UNIX systems; *CALIC* [17] which is a very sophisticated image compression
algorithm (we used an implementation of Xiaolin Wu et al. of the University of Western
Ontario); and the quasi standard *lossless JPEG* (ljpg) [16], where we used an implementation
from Cornell University.

Our Hilbert Compression algorithm has been tested with lots of data from satellite sources
such as *LANDSAT* and *SPOT* pictures. Several tests with derived RGB images and original
(7 channel) satellite images showed all the same relative behavior between the three image
compression techniques CALIC, lossless JPEG and Hilbert Compression, which used a context
size of 6 pixels. Table 1 shows a typical result for compression ratios *original size : compressed
size* for different channels of a LANDSAT satellite image. The scene is a multi spectral image
with seven different channels, each of 8 bit resolution, and it has a size of 5866×4782 elements;
this takes about 200 megabytes of data on disk. Figure 4 shows a small part of channel 5 of
this scene, which was hard to compress.

channel	gz	Z	calic	ljpg	H.1.6
1	1.70	1.66	2.30	2.08	2.11
2	2.00	2.02	2.72	2.46	2.44
3	1.80	1.79	2.44	2.18	2.18
4	1.27	1.18	1.70	1.56	1.52
5	1.23	1.15	1.68	1.54	1.51
6	16.02	4.77	4.59	6.58	3.27
7	1.54	1.50	2.09	1.91	1.87

Table 1: Compression Ratios for a LANDSAT Scene

The two compression algorithms gzip and compress are based on the *Lempel-Ziv* [9] coding
algorithm. They perform well only on highly redundant pictures, such as in the present test
for channel 6 of the satellite images, and sometimes the compression ratio falls below usable
values, such as for the channels 4 and 5. The CALIC algorithm achieves very good compression
ratios overall even for complex image data. JPEG and Hilbert Compression have quite similar
results and stay within about 10% of the CALIC results for the critical images.

Note that unfortunately, the CALIC and JPEG algorithms in their optimal form are not at
all a good basis for a modification towards spatial access. CALIC needs a scan-line like pixel
sequence to accomplish its complex prediction function. Therefore, clustering with respect to
filling entire data blocks with compressed data of one image region gets very complicated and
will reduce the compression ratio. More compression efficiency will be lost because CALIC
needs considerable history information – frequency counts – to achieve its data reduction
(a typical implementation needs counters for a few hundred specific contexts). This history
information has to be stored and retrieved to allow partial image reconstruction as mentioned
in Section 3; otherwise, the compression ratio will decrease significantly. JPEG also needs
information about image statistics and the scan-line sequence has the same drawbacks as
for the CALIC algorithm. Lossless JPEG could be adapted to use a better SFC and our

---
[1]available via anonymous ftp: `ftp.khoros.unm.edu`

prediction function, but the Huffman table would still have to be stored and precalculated from the complete image statistics.

Figure 4: Satellite Image of Central Switzerland

7   Conclusion

We considered the problem of range queries on a compressed raster image. With the goal of avoiding a decompression of the complete image before querying, we proposed a compression algorithm, Hilbert Compression, that is oriented towards spatial clustering and that permits decompression from local information alone. We have demonstrated experimentally that Hilbert compression ratio typically is competitive with well known compression algorithms such as lossless JPEG or CALIC. We feel that our proposal is a first step towards an investigation of how to perform spatial operations (or other complex operations) directly on the compressed data. Since the body of compressed raster image is growing steadily, and advanced applications call for increasingly complex operations (the IEEE Computer dedicated its September 1995 issue to this topic), further investigations in this direction appear to be useful.

Acknowledgement

References

[1] B. Becker, P. Franciosa, S. Gschwind, T. Ohler, G. Thiemt, and P. Widmayer. Enclosing many boxes by an optimal pair of boxes. In *Proc. of the 9th Annual Symposium on Theoretical Aspects of Computer Science STACS*, volume 577, pages 475–486, Cachan, 1992. Springer-Verlag. Lecture Nores in Computer Science.

[2] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: an efficient and robust access method for points and rectangles. In *Proc. of the ACM SIGMOD Int. Conf. on the Management of Data*, pages 322–331. ACM, 1990. Atlantic City, New Jersey.

[3] J. G. Cleary, R. M. Neal, and I. H. Witten. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, June 1987.

[4] C. Faloutsos and S. Roseman. Fractals for secondary key retrieval. In *Proc. of the ACM Conf. on the Principles of Database Systems*, pages 247–252. ACM, March 1989.

[5] P. G. Howard. *The Design and Analysis of Efficient Lossless Data Compression Systems*. PhD thesis, Department of Computer Science at Brown University, 1993.

[6] P. G. Howard and J. S. Vitter. Fast progressive lossless image compression. In *Image and Video Compression*, volume 2186, pages 98–109. SPIE, 1994.

[7] D. A. Huffman. A method for the construction of minimum redundancy codes. In *Proc. Inst. Electr. Radio Eng.*, pages 1098–1101, September 1952.

[8] H. V. Jagadish. Linear clustering of objects with multiple attributes. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 332–342. ACM, June 1990.

[9] A. Lempel and J. Ziv. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, pages 337–343, May 1977.

[10] A. N. Netravali and B. G. Haskell. *Digital Pictures: Representation, Compression and Standards*. Plenum Press, New York and London, 1995.

[11] B. Pagel, H. Six, H. Toben, and P. Widmayer. Towards an analysis of range query performance in spatial data structures. In *Proc. of the 12th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 214–221, Washington DC, 1993.

[12] H. Samet. *Applications of Spatial Data Structures: computer graphics, image processing, and GIS*. Addison Wesley Publ. Co., Massachusetts, 1989.

[13] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison Wesley Publ. Co., Massachusetts, 1989.

[14] S. Sarawagi and M. Stonebraker. Efficient organization of large multidimensional arrays. In *Proc. 10th Int. Conf. on Data Engineering '94*, pages 328–336, Houston, Texas, February 1994. IEEE.

[15] G. K. Wallace. Overview of the JPEG (ISO/CCITT) still image compression standard. In *Image Processing Algorithms and Techniques*, volume 1244, pages 220–233. SPIE, February 1990.

[16] G. K. Wallace. The JPEG still picture compression standard. *Communications of the ACM*, 34(4):30–44, April 1991.

[17] X. Wu, N. Memon, and K. Sayood. A context-based, adaptive, lossless/nearly-lossless coding scheme for continous-tone images, July 1995. Proposal for ISO compression standard.