Lossless Compression of Surfaces Described as Points

Juan Ramón Rico-Juan, Jorge Calera-Rubio, and Rafael C. Carrasco*

Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante, E-03071 Alicante, Spain,
{juanra,calera,carrasco}@dlsi.ua.es

Abstract. In many applications, objects are represented by a collection of unorganized points that scan the surface of the object. In such cases, an efficient way of storing this information is of interest. In this paper we present an arithmetic compression scheme that uses a tree representation of the data set and allows for better compression rates than general-purpose methods.

1 Introduction

Effective compression of images is a major research area, especially two dimensional images and video images. Some work has also been done on three dimensional image compression [CHF96]. A number of these works deal with medical applications [IP98] where all information, including that related to inner points, is relevant. However, there exist cases (for instance, most applications in industrial design), where the only relevant information is conveyed by the surface delimiting the object. In such cases, data are usually provided by a scanner that scans the surface and outputs a collection of points (given as vectors in the Euclidean space). Previous work has been done in order to build models that allow for a suitable geometric description of surfaces of this type so that a highly efficient codification of the image is achieved [HDD+94]. The cost of such algorithms is always a crucial issue.

In the simpler case of two dimensional images, the contour of a simply connected shape can be efficiently coded as a string of symbols. For this purpose it is enough to choose the direction in which the contour is covered (i.e., clockwise or counter-clockwise). Then, every point has two neighbors (a predecessor a successor) and the string representation is simply generated by writing the relative position between consecutive points. Usually, 2 dimensional figures are described as a collection of pixels. Because every pixel has 8 possible neighbors, one byte is enough to describe the relative position of a pixel with respect to the previous one. Therefore, if the shape is simply connected, one string contains all the information needed to reconstruct the shape, except for a global translation that is not relevant for most purposes. With this method a considerable reduction in the file size needed to store the information has been reported [GEJ96].

* Work partially supported by the Spanish CICyT under grant TIC97-0941.

F.J. Ferri et al. (Eds.): SSPR&SPR 2000, LNCS 1876, pp. 457–461, 2000.

The method described above allows for simple and fast coding and decoding. However, in the case of 3D images, some important differences arise:

1. The object is limited by a surface and the number of possible neighbors in a grid increases to 26.
2. In the usual setting, the scanner precision is much higher than the minimum distance between points, so that they are no longer adjacent.
3. Because a point in a surface may have more than two neighbors, a priority has to be established in order to define a path. Even so, it may be impossible to scan the object surface with a single path that goes only once through every point.

The last fact suggests that a natural way to describe the surface is using a tree representation, as we describe in the following section.

2 Data Representation and Modeling

Given a sample of 3D vectors $S = \{\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_{|S|}\}$, we may define the fully connected graph G where the node set is S and the weight of edge $(\mathbf{r}_i, \mathbf{r}_j)$ is the Euclidean distance between $\mathbf{r}_i$ and $\mathbf{r}_j$. Then, we may build T, the minimum spanning tree (MST) of G, with any of the standard algorithms (see, for instance [CLR90]). Due to the geometric properties[1] of the Euclidean space, the maximal number of neighbors is 12 (12 having probability 0). However, due to the fact that, in our case, the points are distributed on a surface (that is, locally on a plane), in practice less than 6 neighbors are always found and the tree width of T is at most five.

Every node n in the MST with father m is labelled with a vector giving the difference $\mathbf{d}(n) = \mathbf{r}_m - \mathbf{r}_n$. For the root node p, we take $d(p) = \mathbf{r}_p$. In our approach, we will compress the information using arithmetic coding [WNC87, CT91] of the input. For this purpose we need a stochastic models describing the tree structure and the vector components contained in T:

1. In order to model the structure of the tree we will compute the probabilities p_k that a node expands in a given number k of siblings. This probability is estimated as the relative number of subtrees in T having k siblings, where the implicit assumption is done that it does not depend on the position of the node.
2. In order to code the vector $\mathbf{d}(n)$ components, which are floating-point numbers, we use three probability distributions $F_1(d_1), F_2(d_2), F_3(d_3)$, one for each component. An efficient coding requires that these distributions should be analytically invertible. Therefore, rather than using a normal distribution we rather use logistic sigmoid functions

$$F_i(x) = \frac{1}{1 + \exp(-\lambda_i(x - \mu_i))} \tag{1}$$

[1] This question is related to that of packing spheres with optimal density.

whose density function is $\lambda F_i(x)(1 - F_i(x))$. The parameters μ_i and λ_i are evaluated from the average and standard deviation σ_i [2] of the components d_i of the nodes in T.

3 Arithmetic Coding of the Data

The arithmetic compression is performed by following a preorder traversal of T and coding at each node n the vector $\mathbf{d}(n)$ and the number of siblings of node n. For instance, the tree plotted in the figure the traversal will produce as input for the encoder the sequence

$$\mathbf{d}(1)\ 3\ \mathbf{d}(2)\ 0\ \mathbf{d}(3)\ 2\ \mathbf{d}(5)\ 0\ \mathbf{d}(6)\ 1\ \mathbf{d}(7)\ 0\ \mathbf{d}(4)\ 0$$

Note that the sequence above consists of 28 input symbols as every vector $\mathbf{d}(n)$

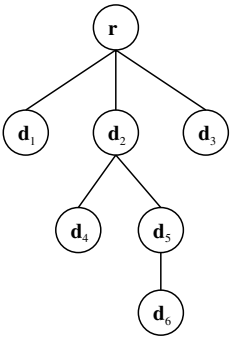

Fig. 1. Sample tree.

contains three numbers ($d_1(n)$, $d_2(n)$ and $d_3(n)$) and every number is considered a different symbol by the arithmetic encoder. The output file contains the parameter set as a header (the parameters μ_i and λ_i together with the probabilities p_k) with a negligible increase of the file size. Then, arithmetic coding is performed using alternatively the four models contained in the header. In order to avoid rounding errors during decoding (due to the 32 bit arithmetics), the tails of the distribution F are treated in a special way. Given a certain scanner precision ϵ, there exists $x_t > 0$ such that

$$F(x_t + \frac{\epsilon}{2}) - F(x_t - \frac{\epsilon}{2}) = 2^{-32} \tag{2}$$

All x such that $|x| \geq x_t$ are considered outliers. In such cases (which are highly improbable), the code for the region $|x| > x_t$ is generated followed by the number x uncoded.

[2] The parameter λ_i is related to σ_i by $\lambda_i \sigma_i = \pi\sqrt{3}$.

4 Results and Discussion

A collection of five different files with a variable number of data (between 4 and 19 thousand points) were compressed with the previously described method. For the sake of comparison, the raw data were also compressed using 1) a Lempel-Ziv compressor (`gzip`); 2) a Huffman coder (`bzip2`, which uses a Barows-Wheeler transform) and; 3) an arithmetic coder based on tree-grams described in [Nel91]. The results are presented in table 4. Compression rate are defined as the quotient between file compressed file and file original size.

dataset #	`gzip`	3-gram	`bzip2`	ours	size (bytes)
1	0.36	0.30	0.29	0.22	108758
2	0.31	0.29	0.27	0.22	253615
3	0.38	0.32	0.34	0.21	339859
4	0.36	0.31	0.34	0.20	449768
5	0.21	0.16	0.11	0.17	484356

Table 1. Compression rate for 5 different data sets.

As shown in the table, our method favorably compares with the existing general purpose methods and allows for compression rates about 5 which are in consistently better than those obtained with the other methods.

5 Conclusion

We have implemented an arithmetic coder that compresses files of data containing points describing the surface of three dimensional objects. The method uses a simple representation of the surface consisting on a tree of relative positions between points and modelizes this structure and the components. The time complexity coincides with that of the standard MST construction algorithms. The compression rates are comparable or significantly better than those obtained with standard methods. Further refinements on the data modeling may lead to improved results.

References

[CHF96] Wayne O. Cochran, John C. Hart, and Patrick J. Flynn. Fractal volume compression. *IEEE Transactions on Visualization and Computer Graphics*, 2(4):313–322, December 1996.

[CLR90] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1990.

[CT91] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley Series in Telecommunications. John Wiley & Sons, New York, NY, USA, 1991.

[GEJ96] M. Gokmen, I. Ersoy, and A. K. Jain. Compression of fingerprint images using hybrid image model. In *Int. Conf. on Image Processing*, volume III, pages 395–398, Lausanne, September 1996.

[HDD+94] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetz. Piecewise smooth surface reconstruction. *Computer Graphics*, 28((SIGGRAPH 1994 Proceedings)):295–302, 1994.

[IP98] Insung Ihm and Sanghun Park. Wavelet-based 3d compression scheme for very large volume data. In Kellogg Booth Wayne Davis and Alain Fourier (Editors), editors, *Proceedings of the 24th Graphics Interface*, pages 107–116, San Francisco, June 1998. Morgan Kaufmann.

[Nel91] Mark R. Nelson. Arithmetic coding and statistical modeling. *Dr. Dobb's Journal of Software Tools*, 16(2):16, February 1991.

[WNC87] I. H. Witten, R. M. Neal, and J. G. Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6), June 1987.