Distance Field Compression

M. W. Jones
Department of Computer Science
University of Wales Swansea
Singleton Park, Swansea SA2 8PP, UK
m.w.jones@swan.ac.uk

ABSTRACT

This paper compares various techniques for compressing floating point distance fields. Both lossless and lossy techniques are compared against a new lossless technique. The new Vector Transform technique creates a predictor based upon a Vector Distance Transform and its suitability for distance field data sets is reported. The new technique produces a lossless encoding at a third of the file size of entropy encoders, and equivalent to lossy wavelet transforms, where around 75% of the coefficients have been set to zero. The algorithm predicts each voxel value linearly based upon two previous voxels chosen from one of 13 directions which have been previously computed. Those that cannot be predicted are explicitly stored.

Keywords
Compression, distance field, distance transform, vector distance transform, lossless

1 INTRODUCTION

Although there has been much work on the topic of compressing volume data [Mur93, FY94, IP98, ILRS03], little or no research has been published on the compression of distance fields. Distance fields have gained a wide appreciation in the graphics community recently as they have been found useful for many applications, from contour connection [JC94], object representation [Jon96, FPRJ00, JS01], object reconstruction [CL96], interactive modelling [BC02], skipping over empty space during ray-tracing [SK00] to alternative data structures for the geometry simplification process. The common element for distance fields is that they should enable the query of distance in three-dimensional space to an object contained within that space. The main approaches for achieving this are to calculate the distance for each requested point, the full distance field [JS01], an adaptive distance field [FPRJ00] or a piecewise linear approximation [WK03]. Broadly, the first involves a costly query of the object to calculate the closest distance and is only suitable for objects with low primitive count, or cases where few queries are required. The full distance field calculates a discrete grid of distances, and the distance to query points are calculated using interpolation from known points. The adaptive distance field provide distances in a similar manner, but these are interpolated from octree nodes. Finally the piecewise linear approximation method approximates the distance field with linear functions, and distance queries are handled by determining and evaluating the appropriate function.

Each approach has its own benefits and disadvantages (most notably a trade-off between accuracy, storage requirements and computational effort), and are therefore suited to different applications. The analysis of the most appropriate approach is not the topic of this paper, but rather methods for the compression of full distance fields will be considered and evaluated.

Section 2 will describe some of the methods used for compression of volume data. Their use on distance fields will be evaluated and the compression ratios will be reported.

Section 3 will show how a vector distance transform can be used as a predictor of voxel values in a distance field in order to aid the compression process. It will derive the predictor, and give details of the implementation, including consideration to numerical accuracy and space required.

Section 4 will examine and compare the various lossless and lossy techniques with the new Vector Transform method, and Section 5 will conclude the paper

Journal of WSCG, Vol.11, No.1., ISSN 1213-6972
WSCG'2003, February 3-7, 2003, Plzen, Czech Republic.
Copyright UNION Agency–Science Press

and suggest some future work.

2 PREVIOUS WORK

The compression of distance fields offers its own challenges when compared to image compression or standard volume compression. Images or volumes usually represent data using an integer format of a certain bit length - for example the hydrogen data (from AVS) uses an 8 bit unsigned integer format, and the UNC CThead uses a 12 bit signed integer format. Lossless compression techniques can take advantage of some values appearing more frequently than others (for example, bone in CT data sets), and employ a codebook approach to compression. Lossy techniques can quantize the data further, use the DCT, or wavelets.
Distance fields, D, represent the distances as *n* bit floating point numbers, where usually *n=16, 32,* or *64*. Each value represents the minimum distance from that voxel, p, to the surface S:

$$D(p) = sgn(p) \cdot min\{|p-q| : q \in S\}$$
$$sgn(p) = \begin{cases} -1 & \text{if p inside} \\ +1 & \text{if p outside} \end{cases} \quad (1)$$
where || is the Euclidean norm

Simple lossless compression may be achieved by entropy encoding, or lossy compression can be achieved by reducing accuracy in the form of quantizing the original values, and then a lossless encoding may be applied to achieve further compression. The next sections look at applying existing methods for lossless and lossy compression to floating point distance fields.

2.1 Lossless techniques

Entropy encoding techniques such as Huffman coding, run-length-encoding and arithmetic coding [Sal01] have been developed with regard to text or integer based data representations. Such methods perform poorly on representations based upon real numbers because the bit patterns are too random for entropy encoding to work well (Table 1). Matters can be improved by observing that the 64 bit IEEE floating point representation is divided into 1 bit for sign, followed by 11 bits for the exponent, and 53 bits for the mantissa (in practise 52 are stored, and the most significant is implied to be 1). By rearranging the bits such that all the sign bits occur first, followed by all the exponent bits, and finally followed by the mantissa bits, it is possible for entropy coding to take advantage of the coherency in the sign, exponent, and the first few bits of the mantissa (Table 2) (previously applied to image data in [CM95] amongst others).
The lossless Lorenzo predictor [ILRS03] (which was developed for scalar data) is not suited to floating point distance field data sets (exactly predicting only 0.6% of voxel values in the CThead distance field). Another predictor [FY94] performs similarly. Both predictors have been designed with scalar data in mind, and work well with such data.
Existing lossless techniques do not offer large compression ratios for floating point distance fields, and so lossy techniques, as discussed in the next section, may be used to improve the compression ratio, but at the expense of introducing variable amounts of error. As shall be seen in Section 3, the method presented in this paper offers a lossless encoding of floating point distance fields at a substantially improved ratio compared to existing lossless techniques, and at a ratio comparable to lossy compression methods.

Distance Field 64 bit IEEE f.p.	Original size in bytes	Entropy encoding
AVS Hydrogen $128 \times 128 \times 128$	16777216 (100%)	13156273 (78.42%)
CThead $256 \times 256 \times 128$	67108864 (100%)	61014252 (90.92%)

Table 1: Entropy encoding of 64 bit floating-point distance fields.

Distance Field 64 bit IEEE f.p.	Original size in bytes	Entropy encoding
AVS Hydrogen $128 \times 128 \times 128$	16777216 (100%)	13220415 (78.80%)
CThead $256 \times 256 \times 128$	67108864 (100%)	53505457 (79.73%)

Table 2: Entropy encoding of 64 bit floating-point distance fields after rearrangement of the bit-planes.

2.2 Lossy Techniques

Wavelet transforms have been investigated extensively for the compression of volume data [Mur93][IP98] [Rod99][KS99]. Wavelets implement a hierarchical approach to compression whereby each level in the hierarchy is an interpolation of previous levels. Detail coefficients are stored and, for lossy encoding, are set to zero where they are less than a predefined threshold. The coefficients are quantized and then entropy encoding or zero-trees [Sha93] are employed for further compression. For lossless encoding all detail coefficients are retained, and no quantization takes place. Wavelet transforms can be suited to floating point distance field data by not using the quantization step. Simple transforms, such as Haar, offer a means of generating good compression ratios whilst controlling the overall error of the data set. Table 3 shows the Haar transform used in both the lossless and lossy modes, and the resulting error and data sizes for the CThead

Wavelet Transform	Coefficients set to zero	Max Error
59858717 (89%)	0%	0.000
33667571 (50%)	49%	0.046
10863320 (16%)	85%	0.540

Table 3: Wavelet compression using Haar transform.

data set (original size 67108864 bytes). Entropy encoding rather than zero-tree encoding was used on the detail coefficients. It can be seen from the table that substantial savings in storage can be achieved if some error is allowed in the data set. Approximating distance fields using piecewise linear functions has enabled a large compression at the expense of introducing some inaccuracy into the distance values [WK03]. This compression is particularly suited to situations where distance values at, or around, the surface are more important – e.g. during geometry processing. The lossless method presented in this paper is suited to situations where error is undesirable.

3 METHOD

The lossless method presented in this paper relies upon the observation that Distance Transforms generally produce good approximations to true distances. A more accurate Vector Distance Transform will be used as the basis for the predictor used in this method.

3.1 Vector Distance Transforms

The $3 \times 3 \times 3$ Chamfer Distance Transform (CDT) [Dan80] propagates distances throughout a volume, D, by calculating the distance for the current voxel by adding unit distance to each face neighbouring voxel, $\sqrt{2}$ distance to each edge neighbour, and $\sqrt{3}$ distance to each vertex neighbour, and then taking the minimum of all those distances:

$$\begin{aligned}&D(x,y,z) = \\ &min(D(x+i,y+j,z+k) + \sqrt{i^2+j^2+k^2}) \\ &\forall i,j,k \in \{-1,0,1\}\end{aligned} \tag{2}$$

For implementation, the CDT is carried out in two passes – forwards using previously calculated neighbours, and backwards using the remaining neighbours. CDTs suffer from poor accuracy as the distance from the surface increases. This problem is overcome by using Vector Distance Transform (VDT) techniques [Mul92, SJ01] which enable accurate distance fields to be generated quickly. VDTs store the vector $\mathbf{v}$ to the closest point $\mathbf{p}$ to the surface at each voxel in a volume V such that $V(\mathbf{p}) = \mathbf{v}$, and add (or subtract) the appropriate unit distance from each dimension based upon the direction of the neighbour (again taking the minimum):

$$\begin{aligned}&D(x,y,z) = \\ &min(|V(x+i,y+j,z+k) - (i,j,k)|) \\ &\forall i,j,k \in \{-1,0,1\}\end{aligned} \tag{3}$$

VDTs propagate distances more accurately than CDTs, but there are certain configurations which prevent VDTs from being completely accurate. As these cases are the exception rather than the majority case, the VDT can be used as a predictor for a voxel value in a distance field data set, and hence may be used as a basis for a compression technique.

3.2 Derivation of the Predictor

In this section, a distance predictor will be derived, which uses the known distances (d_2 and d_1) from 2 previous voxels, to calculate the distance for the current voxel (d_0). For derivation purposes, assume three consecutive voxels v_2, v_1 and v_0 in the x-axis with v_2 being the leftmost and distances to the surface at each voxel d_2, d_1 and d_0 respectively, with d_2 and d_1 known. Assuming the vector to the surface at v_2 is (x, y, z), then using a vector transform in the forward x direction would result in v_1 having a vector $(x+1, y, z)$ to the surface, and v_0 would have $(x+2, y, z)$.
We have:

$$x^2 + y^2 + z^2 = d_2^2 \tag{4}$$

$$(x+1)^2 + y^2 + z^2 = d_1^2 \tag{5}$$

and

$$(x+2)^2 + y^2 + z^2 = d_0^2 \tag{6}$$

From Equations 4 and 5 we obtain:

$$y^2 + z^2 = d_2^2 - x^2 = d_1^2 - (x+1)^2 \tag{7}$$

so

$$d_2^2 - x^2 = d_1^2 - x^2 - 2x - 1 \tag{8}$$

giving

$$x = \frac{d_1^2 - d_2^2 - 1}{2} \tag{9}$$

Substituting into Equation 6 and using Equation 5 we obtain

$$d_0^2 = 2(d_1^2 + 1) - d_2^2 \tag{10}$$

Equation 10 also can be derived in the y and z axes. This gives 3 forward directions from which a voxel distance may be calculated (the already computed face neighbours of the current voxel).
In one case where v_2, v_1, and v_0 form a diagonal within the xy-plane we have

$$x^2 + y^2 + z^2 = d_2^2 \tag{11}$$

$$(x+1)^2 + (y+1)^2 + z^2 = d_1^2 \tag{12}$$

and

$$(x+2)^2 + (y+2)^2 + z^2 = d_0^2 \quad (13)$$

From Equations 11 and 12 we obtain:

$$z^2 = d_2^2 - x^2 - y^2 = d_1^2 - (x+1)^2 - (y+1)^2 \quad (14)$$

which gives

$$x + y = \frac{d_1^2 - d_2^2 - 2}{2} \quad (15)$$

Similarly, substituting into Equation 13 and using Equation 12 we obtain

$$d_0^2 = 2(d_1^2 + 2) - d_2^2 \quad (16)$$

Equation 16 also can be derived in the yz-plane and xz-plane. This gives 6 forward directions from which a voxel distance may be calculated (the already computed edge neighbours of the current voxel).
Similarly the 4 already computed vertex neighbours of the current voxel lead to 4 forward directions from which a voxel distance may be calculated using Equation 17 which is derived in a similar manner to above.

$$d_0^2 = 2(d_1^2 + 3) - d_2^2 \quad (17)$$

In summary, the distance at the current voxel may be predicted from the distances at two previous voxels, in any of 13 already considered (previously calculated) directions. Each direction can be tested in turn until the predicted distance is equal to the distance stored at the voxel. In the case that a prediction succeeds, a 4 bit code can indicate the direction. If the prediction is not successful, the 4 bit code can indicate this, and the correct distance may be stored. The sign of d_0 is predicted using the sign of d_1. All other distances are removed. Entropy encoding may be applied to the 4 bit codes, and the stored distances to further compress the data.

3.3 Implementation

The implementation uses a distance predictor which requires the distance at the current (d_0) and two previous voxels (d_1 and d_2), the number of axes, n, the distances are from ($n = 1$, 2 or 3), and the bit code to be stored if the distance can be predicted:

```
$Predictable(d_2, d_1, d_0, n, dir)$
if $(d_0^2 = 2(d_1^2 + n) - d_2^2) \wedge (sgn(d_0) = sgn(d_1))$
then
    $addbits(dir)$
    return $true$
else
    return $false$
```

To compress the distances of a distance field D, the predictor is called with distances from each direction until one succeeds. If all fail, the distance is stored explicitly, and an appropriate bit code is added. Only four calls are shown, and a test to check the voxels are inside the distance volume D is omitted for clarity

```
$CompressDistances(D)$
if $Predictable(D_{i-2,j,k}, D_{i-1,j,k}, D_{i,j,k}, 1, 0)$
    return
if $Predictable(D_{i,j-2,k}, D_{i,j-1,k}, D_{i,j,k}, 1, 1)$
    return
if $Predictable(D_{i,j,k-2}, D_{i,j,k-1}, D_{i,j,k}, 1, 2)$
    return
if $Predictable(D_{i-2,j-2,k}, D_{i-1,j-1,k}, D_{i,j,k}, 2, 3)$
    return
$\vdots$
$addbits(stored)$
$store(d_0)$
```

Once the distance predictor has been carried out on the distance field, entropy encoding is carried out on the bit store. The distance store is transformed using bit-plane encoding before further compression using entropy encoding.

3.4 Further Considerations

Numerical accuracy Due to the fact that distances may be predicted over large numbers of voxels during the vector transform, and each prediction is based on a previous prediction, some errors may be introduced due to numerical inaccuracies in the internal representation of floating point numbers. This has been overcome in the implementation by using 128-bit doubles for some of the intermediate calculations and by using a separate quality check. The quality check tests uncompression during the prediction stage, using the previously predicted voxel values. If the quality check indicates the uncompressed value will differ from the original value due to numerical accuracy problems, the predictor is said to have failed, and the original distance value is stored. This solves the problem, but reduces the number of predictable voxels by around 10% from the number that could be theoretically predicted. Future work will investigate the possibility of minimising the prediction path in an attempt to reduce the numbers of voxels that could be predicted, but cannot, due to numerical accuracy problems.
Time and Space Distance field data sets can be quite large – the CThead distance field example is around 64MB. Using wavelet compression would result in the need to have the data set present in memory during the whole operation, as the wavelet transform makes several passes through the voxel values on each axis. The algorithm presented here is linear, and makes just one forward pass through the data during both compression and uncompression. As a result the algorithm needs at most just three slices of the data set in

memory at anyone time (the furthest voxel used during prediction will be two slices above the current voxel). This linear traversal of the data rather than the several passes used to traverse the hierarchical data resulting from the wavelet transform also results in this method being faster than wavelet transforms – the Haar compression takes about 12 seconds, and this method takes about 3 seconds (on a P4 2.5GHz).

4 RESULTS

Two distance field data sets were used for the comparison. The first is a distance field data set produced from the UNC CThead (Figure 1). Extra slices were added at the top and bottom to make the distance field $256 \times 256 \times 128$ (to make the size more convenient for the wavelet transform). Distances were calculated at sub-voxel accuracy to the skull. The second is a distance field data set produced from the AVS Hydrogen data set (Figure 2). Extra slices were added around the data set to make the distance field $128 \times 128 \times 128$. Distances were calculated at sub-voxel accuracy to a value of 128 (the original data is 0 to 255). Lossless and lossy compression techniques were carried out on both data sets. Entropy encoding, this vector transform, and entropy encoding after rearranging the bitplanes were used as lossless methods. Tables 4 and 5 show that this vector transform method produces substantially better compression than entropy encoding methods, resulting in a file which is a third of the size of the next best method. It is also interesting to note the size of error introduced by using a lossy method (Haar transform) to create a file of the same size. In the case of the AVS Hydrogen data set, the low error produced by the Haar transform is due to the fact that voxels have only 0–255 values in the original data set, and therefore the distance field for 128 contains distances to integer voxel positions or a restricted number of positions within the voxel, which compresses well.

5 CONCLUSION

This paper has compared various techniques for compressing floating point distance fields. Both lossless and lossy techniques were compared against a new lossless technique. The new Vector Transform technique creates a predictor based upon a Vector Distance Transform which was demonstrated to be most suitable for distance field data sets. The new technique produces a lossless encoding at a third of the file size of entropy encoders, and equivalent to lossy wavelet transforms, where around 75% of the coefficients have been set to zero. The error introduced by the wavelet transforms was reported, although the lossless wavelet transform should be used as the main comparison as this Vector Transform technique is lossless.

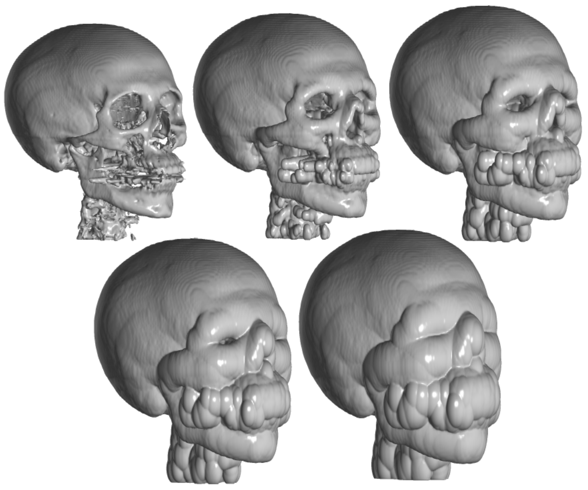
Figure 1: CThead distance field (various distance offsets).

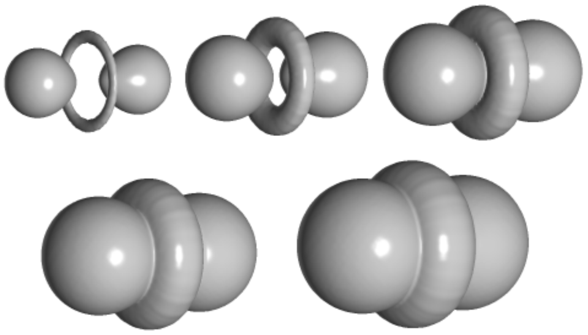
Figure 2: AVS hydrogen distance field (various distance offsets).

The algorithm predicts voxel values based upon two previous values. The algorithm is memory efficient as only three slices are necessary in main memory for the algorithm to operate. It is time efficient as each voxel is computed once (unlike hierarchical wavelet methods).
Future work will concentrate on increasing the number of voxels that can be predicted by careful consideration of the prediction path, and by examining if a backward pass may be introduced. It may also be possible to create a hierarchical method which would allow lossy encoding and hence provide even better compression ratios, but with controlled loss.

References

[BC02] A. Baerentzen and N. J. Christensen. Interactive modelling of shapes using the level-set method. *International Journal of Shape Modelling*, 8(2):79–97, 2002.

[CL96] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings SIGGRAPH '96*, pages 303–312, 1996.

Method	Size in bytes (and % of original)	Max Error	Average Error	Type
Entropy Encoding	61014252 (91%)	0.000	0.000	lossless
Haar Transform	59858717 (89%)	0.000	0.000	lossless
Bit-plane Encoding	53505457 (80%)	0.000	0.000	lossless
Haar Transform	33667571 (50%)	0.046	0.004	lossy
Haar Transform	17538888 (26%)	0.300	0.022	lossy
Vector Transform	17459421 (26%)	0.000	0.000	lossless
Haar Transform	10863320 (16%)	0.540	0.053	lossy

Table 4: Comparison on 67108864 byte CThead distance field using lossy and lossless compression techniques.

Method	Size in bytes (and % of original)	Max Error	Average Error	Type
Bit-plane Encoding	13220415 (79%)	0.000	0.000	lossless
Entropy Encoding	13156273 (78%)	0.000	0.000	lossless
Haar Transform	12872603 (78%)	0.000	0.000	lossless
Haar Transform	8726198 (52%)	0.003	0.0003	lossy
Vector Transform	3922737 (23%)	0.000	0.000	lossless
Haar Transform	3883749 (23%)	0.025	0.003	lossy

Table 5: Comparison on 16777216 byte AVS hydrogen distance field using lossy and lossless compression techniques.

[CM95] H. Cai and G. Mirchandani. Wavelet transform and bit-plane encoding. In *International Conference on Image Processing*, volume 1, pages 578–581. IEEE Computer Society Press, 1995.

[Dan80] P-E. Danielsson. Euclidean distance mapping. *Computer Graphics and Image Processing*, 14:227–248, 1980.

[FPRJ00] S. Frisken, R. N. Perry, A. P. Rockwood, and T. R. Jones. Adaptively sampled distance fields: A general representation of shape for computer graphics. In *SIGGRAPH Proceedings on Computer Graphics*, pages 249–254, July 2000.

[FY94] J. Fowler and R. Yagel. Lossless compression of volume data. In *1994 Symposium on Volume Visualization*, pages 43–50, 1994.

[ILRS03] L. Ibarria, P. Lindstrom, J. Rossignac, and A. Szymczak. Out-of-core compression and decompression of large n-dimensional scalar fields. *Computer Graphics Forum*, 22(3):343–448, 2003.

[IP98] I. Ihm and S. Park. Wavelet-based 3D compression scheme for very large volume data. In *Graphics Interface*, pages 107–116, 1998.

[JC94] M. W. Jones and M. Chen. A new approach to the construction of surfaces from contour data. *Computer Graphics Forum*, 13(3):75–84, 1994.

[Jon96] M. W. Jones. The production of volume data from triangular meshes using voxelisation. *Computer Graphics Forum*, 15(5):311–318, December 1996.

[JS01] M. W. Jones and R. Satherley. Shape representation using space filled sub-voxel distance fields. In *Shape Modelling and Applications*, pages 316–325. IEEE Computer Society Press, 2001.

[KS99] T. Kim and Y. Shin. An efficient wavelet-based compression method for volume rendering. In *Proceedings Pacific Graphics*, pages 147–157, 1999.

[Mul92] J. C. Mullikin. The vector distance transform in two and three dimensions. *CVGIP: Graphical Models and Image Processing*, 54(6):526–535, 1992.

[Mur93] S. Muraki. Volume data and wavelet transforms. *Computer Graphics and Applications*, 13(4):50–56, July 1993.

[Rod99] F. Rodler. Wavelet based 3D compression with fast random access for very large volume data. In *Proceedings Pacific Graphics*, pages 108–117, 1999.

[Sal01] D. Salomon. *A Guide to Data Compression Methods*. Springer, 2001.

[Sha93] J. M. Shapiro. An embedded hierarchical image coder using zerotrees of wavelet coefficients. In *Proceedings DCC'93 (IEEE Data Compression Conference)*, pages 214–233, 1993.

[SJ01] R. Satherley and M. W. Jones. Vector-city vector distance transform. *Computer Vision and Image Understanding*, 82:238–254, 2001.

[SK00] M. Sramek and A. Kaufman. Fast ray-tracing of rectilinear volume data using distance transforms. *IEEE Transactions on Visualization and Computer Graphics*, 3(6):236–252, 2000.

[WK03] J. Wu and L. Kobbelt. Piecewise linear approximation of signed distance fields. In *Vision, Modeling and Visualization*, page to be published. IOS Press, 2003.