Chapter 3

Generation of Octrees from raster scan with reduced information loss

Abstract

Having addressed the issue of generating octrees from raster scan efficiently using parallel computation, it is worthwhile to ensure that the encoding is accomplished without much loss of information. Knowing that each cube usually stores an average of the intensity value in the cubical subspace, the encoding could lead to a "blocky" (as against "patchy") solid during regeneration. Retaining the entire intensity distribution in three dimensions may defeat the whole purpose of encoding. The present Chapter addresses the technique for octree encoding with minimum information loss. After considering the techniques based on vector quantization [183Gary84] [152Nasrabadi88] an efficient and easy to implement approach based on wavelet transform [184Gross96] is introduced.

Often the reconstructed data from octrees results in a blocking effect because of the average density information assumed in the cubical space represented by the cubes defining the solid. The average density along with the variance in the cubical space also does not improve the situation. Recently, Wavelet transforms have been shown to be efficient in giving a compressed representation of square and cubical region [184Gross96] and the inverse wavelet transform, reconstructs the cubical space without much loss of information. The cubical region, which is represented by an octree leaf node along with the wavelet transform of the cubical region, can be thus
stored in a very compact way. During reconstruction, the intensity distribution in the cubical space, as decided by the octree leaf node, can be recovered. It is shown that the wavelet transform can be progressively constructed during the raster to octree conversion [Sojan98]. Starting at the voxel level, the 8 cubical blocks, from a pair of adjacent planes, are progressively combined depending upon the homogeneity of the density distribution among the eight voxels (condensation). During the condensation of the voxels (larger sized cubes at higher levels), the wavelet transform of the cubical space spanned by the eight voxels (or eight larger sized cubes) can also be computed and combined.

3.1 Reconstruction of solids from octrees

The common problem faced during the reconstruction process is to assign the intensity distribution to all voxels in the solid space, as the octree representation usually stores the average intensity in the cubical subspace. When cubes represented by a leaf element are large, the reconstructed solid becomes "patchy". Vector quantization can improve the reconstruction, wherein we attach a certain number of code words along with each cube. During reconstruction, the code words are replaced by the actual vectors, selected using code words [Amir96]. In this Chapter, we propose that the wavelet transform can be used efficiently to encode the solid space. By attaching a certain amount of detail along with the approximation computed at some level, a good amount of reduction in the cubical sub-space is also achieved. We also demonstrate that the wavelet transform can be easily computed during the raster to octree construction, using a class of bi-orthogonal low pass and high pass filters. During reconstruction of the solid from the octree, the intensity distribution in the
cubical sub-space encoded by the o-code is regenerated from the stored wavelet transforms.

3.2 Vector Quantization

A vector quantizer is a system for mapping a sequence of continuous or discrete vectors into a digital sequence suitable for communication over storage in a digital channel. The goal of such a system is data compression: to reduce the bit rate so as to minimize communication channel capacity or digital storage memory requirements while maintaining the necessary fidelity of the data.

Vector Quantization (VQ) is an effective data compression technique for speech and images where high compression ratio is desired. An important problem in VQ is how to design a codebook that is good for a source [186Gersho92] [187Linde80], and another important issue naturally associated with the problem is testing designed codebooks. In testing a designed codebook, we may consider two aspects: the optimality for a given source and the robustness for various other sources [188Neuhoof87]. The former has to do with good design of a codebook for a particular source, while the latter implies goodness of a designed codebook for other sources that cannot be targeted in the design process. Experimental observations on robustness of codebooks are introduced under the title of the quantizer mismatch problems in [189Gray75] [190Mauer79] [191Yamada84]. However, most codebook design problems are concerned with a faithful design of a codebook for a given source [186Gersho92]. Thus it is important in judging a codebook to validate its optimality for a particular source.
Kim et al. [192Kim97] discusses a criterion for testing a vector quantizer codebook that is obtained by "training". VQ codebook is designed by a clustering algorithm using a training-set-distortion (TSD). The algorithm stops when TSD significantly decrease. In order to test the resultant codebook, validating-set-distortion (VSD) is calculated on a separate validating set (VS). Codebooks that yield small difference between the TSD and the VSD are regarded as good ones. However, the difference in VSD-TSD is not necessarily a desirable criterion for testing a trained codebook unless certain conditions are satisfied. A condition that is previously assumed to be important is that the VS has to be quite large to well approximate the source of distribution. This condition implies greater computational burden of testing a codebook. They discussed the condition under which the difference VSD-TSD is a meaningful codebook-testing criterion. Then, convergence properties of the VSD, a time-average quantity, are investigated. Finally they showed that for larger codebooks, a VS size as small as the size of the codebook is sufficient to evaluate the VSD. Kim et al consequently presents a simple method to test trained codebooks for VQ's.

It is worthwhile to note that the codebook gives a code representing a data distribution. But the dataset itself will have to be stored somewhere to be used for filling in the solid space during reconstruction. On the other hand, techniques based on wavelet transform facilitate the representation of the density distribution in a cubical space with a smaller dataset (called approximation and detail) from which the data distribution in the cubical space can be easily reconstructed during decoding. We
therefore propose an approach to incorporate the wavelet compression, which combines elegantly with the synthesis of octrees from CAT scan slices.

3.3 Wavelet transforms

Wavelets are a mathematical tool for hierarchically decomposing functions. They allow a function to be described in terms of a coarse overall shape, plus details that range from broad to narrow. Regardless of whether the function of interest is an image, a curve, or a surface, wavelets offer an elegant technique for representing the levels of detail present. Authors [193Stollnitz95a] [194Stollnitz95b] were intended to provide people working in computer graphics with some intuition for what wavelets are, as well as to present the mathematical foundations necessary for studying and using them.

Although wavelets have their roots in approximation theory [195Daubechies88], and signal processing [196Mallat89], they have recently been applied to many problems in computer graphics. These graphical applications include image editing [197Berman94], image compression [198DeVore92], and image querying [199Jacobs95]; automatic level of detail control for editing and rendering curves and surfaces [200Finkelstein94] [201Gortler95] [202Lounsbery93], surface reconstruction from contours [203Meyers94], and fast method for solving simulation problems in animation [204Liu94], and global illumination [205Christensen95] [206Christensen94] [207Gortler93] [208Schroder93].

The Haar basis is the simplest wavelet basis. One-dimensional Haar wavelet is illustrated as follows.
3.3.1 The one-dimensional Haar wavelet Transform.

Suppose we are given a one-dimensional "image" with a resolution of four pixels, having the values

9 7 3 5

We can represent this image in the Haar basis by computing a wavelet transform. To do this, first average the pixels together, pair-wise, to get the new lower resolution image with pixel values

8 4

Clearly some information has been lost in this averaging process. To recover the original four pixel values from the two averaged values, we need to store some detail coefficients, which capture the missing information. In the above example, we will choose 1 for the first detail coefficient, since the average we computed is 1 less than 9 and 1 more than 7. This single number allows us to recover the first two pixels of our original four-pixel image. Similarly the second detail coefficient is -1, since 4 + (-1) = 3 and 4 - (-1) = 5.

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Averages</th>
<th>Detail coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>9 7 3 5</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>8 4</td>
<td>1 -1</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 3.1 Decomposition of four-pixel image.

Thus, the original image has decomposed into a lower resolution (two-pixel) version and a pair of detail coefficients. Repeating this process recursively on the averages gives the full decomposition. Finally, we will define the wavelet transform (also...
called the wavelet decomposition) of the original four pixel image to be the single
coefficient representing the overall average of the original image, followed by the
detail coefficients in order of increasing resolution. Thus, for one-dimensional Haar
basis, the wavelet transform of our original four-pixel image is given by

\[ 6 \ 2 \ 1 \ -1 \]

Storing the image wavelet transform, rather than the image itself, have a number of
advantages. One advantage of the wavelet transform is that often a large number of
the detail coefficients turn out to be very small in magnitude, as in the example of
Table 3.1 above. Truncating, or removing, these small coefficients from the
representation introduce only small errors in the reconstructed image, giving a form
of "lossy" image compression. More details of this can be referred in
\[^{193}Stollnitz95a\].

3.3.2 One-Dimensional Haar wavelet basic functions

We have shown that one-dimensional images can be treated as sequence of
coefficients. Alternatively, we can think of images as piece wise-constant functions
on the half-open interval \([0,1)\). To do so, we will use the concept of vector space
from linear algebra. A one-pixel image is just a function that is constant over the
entire interval \([0,1)\). We'll let \(V^0\) be the vector space of all these functions. A two-
pixel image has two constant pieces over the intervals \([0, 1/2)\) and \([1/2,1)\). We'll call
the space containing all these functions \(V^1\). If we continue in this manner, the space
\(V^j\) will include all piece wise - constant functions defined on the interval \([0,1)\) with
constant pieces over each of \(2^j\) equal subintervals.
We can now think of every one-dimensional image with $2^J$ pixels as an element, or vector, in $V_j$. Note that because these vectors are all functions defined on the unit interval, every vector in $V_j$ is also contained in $V_{j+1}$.

3.3.3 Two-dimensional Haar wavelet transforms

There are two common ways in which wavelets can be used to transform the pixel values within an image. Each of these transformations is a two-dimensional generalization of the one-dimensional wavelet transform described in section 3.3.2.

The first transform is called the standard decomposition [209Beylkin91]. To obtain the standard decomposition of an image, we first apply the one-dimensional wavelet transform to each row of pixel values. This operation gives us an average value along with detail coefficients for each row. Next, we treat these transformed rows as if they were themselves an image and apply the one-dimensional transform to each column. The resulting values are all detail coefficients except for single overall average coefficients. An algorithm to compute the standard decomposition is given by [210Stollnitz96].

![Fig. 3.1 Standard decomposition of image.](image)
The second type of two-dimensional wavelet transform, called the non-standard decomposition [same as standard decomposition], alternates between operations on rows and columns. First, we perform one step of horizontal pair-wise averaging and differentiating to each column of the result. To complete the transformation, we repeat this process recursively only on the quadrant containing averages in both directions. Fig 3.2 shows all the steps involved in the non-standard decomposition.

![Diagram of non-standard decomposition](image)

Fig.3.2 Non-standard decomposition of image

Bi-orthogonal Discrete Wavelet Transforms are computed by applying 1-D symmetric low pass and high pass filters on the given image to generate the signal approximation and detail \cite{Thomas96}. Both the filter outputs are decimated by a factor of two. Thus the decomposition is space preserving, meaning that total number of points at the output of the filtering is equal to the number of points in the input. The inverse transform is calculated by up-sampling the transform by adding zeroes, applying the inverse filters and then combining the filter outputs. Fig.3.5 shows the computational scheme.
We have used the Haar wavelets for implementing the wavelet transform.

We use two types of filters viz.

(i) Low pass $[1/\sqrt{2}, 1/\sqrt{2}]$ and

(ii) High pass $[1/\sqrt{2}, -1/\sqrt{2}]$

The resulting octave decomposition in three dimensions results in cubical subspaces of lower dimension computed recursively. As given by Muraki [212Muraki93], we compute the wavelet transforms along a row, then along the column and then along depth, using QMF filters mentioned above.

3.4 Computation of wavelet transform for the o-codes

Fig. 3.3 shows the progressive generation of approximation and details at resolution $(N/4 \times N/4)$, from an image of size $N \times N$. Assuming that the details at higher resolution can be neglected, one can reconstruct the original image (with certain amount of loss) from the approximation $(A)$ and detail $(D_1,D_2,D_3)$ using the inverse wavelet transform. On the other hand, look at the octree construction from raster. For ease of explanation we illustrate the process of generating quadtrees from...
raster image. Four pixels positioned as in Fig.3.4 are combined to form a q-code Q1 at level 1 (representing a size of 2x2). Let us attach the approximation and details from the four pixels along with Q1. Similarly q-codes Q2, Q3 and Q4 shall be generated in the process of looking for larger homogeneous spaces. As Q1, Q2, Q3 and Q4 condense to produce a q-code of size 4x4, the approximation and the details computed along with Q1, Q2, Q3 and Q4 are used to find the approximation and details at resolution 2x2. (We neglect the detail and use only the approximation.) Now we have a q-code representing a size of 4x4 along with approximation and details corresponding to a size of 2x2. The idea of computing the approximation and details, to q-codes representing larger size can be easily extended. The idea of condensation to q-codes representing larger square can also be easily extended to cubical blocks. The wavelet transform is computed on 8 voxels (at level 0), 4 each from adjacent planes. As the o-codes corresponding to cubes of larger size are generated, the approximation and details are computed recursively.

The computation of wavelet transform thus progresses from pixels (as shown in Fig.3.5) and produces the approximation and details for each condensation. Corresponding to some lower level of resolution, we propose to retain the details also. It is then possible to reconstruct the intensity distribution in the cubical space by suitably interpolating the details [184Gross96].

The explanation given above applies equally well to cubical blocks generated during the synthesis of octrees. The approximation and details are constructed by considering x,y,z directions. Finally each o-code shall have a set of approximations
Fig. 3.5 Computation of Wavelet Transform for a 2D image
and details (depending upon where we stop the computation of approximations). The increase in storage space is thus decided by the level to which the approximations and details are to be computed.

3.5 Reconstruction of the solid

As in the case of octree decoding each cubical block is positioned depending upon the co-ordinate information contained in the o-code. The corresponding cubical space is filled using the density distribution constructed from the approximation and detail. Fig.3.6 shows the scheme of inverse wavelet transform [193Stollnitz95] resulting in the generation of data distribution. Depending upon the details dropped out during the computation of wavelet transform, the generated dataset from inverse wavelet transform may have to be interpolated.

Fig.3.6 Computation of inverse Wavelet Transform
3.6 Conclusion

This Chapter summarizes how the wavelet transform can be computed during the condensation process encountered in the raster to octree conversion. The wavelet transform corresponding to a certain amount of resolution along with some detail is stored for every o-code. During re-construction, the o-code specifies the size and position of the cubical space, while the wavelet transform helps to reconstruct the intensity distribution.