**Chapter 3: Low Complexity Image Compression**

1. **Introduction**

Logically, low complexity methods are not as effective in rate distortion terms as other more complex compression versions. They are however very fast, simple to construct and can be very useful for testing image compression techniques. They are the simplest type of image compressor and the basic methods applied to them are used in all image compressors.

Low complexity methods are useful in high speed applications, where the low level of computation required by the system greatly enhances their usefulness. This is sometimes desirable in low cost systems, where a complicated compressor would double or triple the price of the product. Low complexity methods are also important for testing ideas for more complicated systems. Testing using low complexity compressors allows the method to be optimised without the added complications that can often obscure the problems.

In this chapter the basic low complexity compressors are discussed. Most of these compressors have been developed by others working in the field of fractal compression [13 - 19] but their individual performance has not been carefully looked at and this chapter attempts to do this. It first considers polynomial methods, then discrete cosine transforms (DCT’s) and finally enhancements that can be made to these methods. There are several new pieces of work in this chapter:

1. Comparing Polynomial approximations directly with DCT’s.
2. Finding the best number of coefficients to use with a transform series.
3. Limited searching in fractal methods [6].
4. Vector Quantisation for error correction in DCT’s.
5. **Basic Principles**

Low complexity methods produce image compressors that are very close to the model proposed in chapter 2. The transform stage is often lossy, since lossless methods are computationally heavy and require a more complex compression methods to exploit. The transform still benefits from being orthogonal and usually is, although this is not required. The transform is applied across the image using a fixed square block size (as shown in figure 3-1).



Figure 3-1. Diagram of fixed block size applied to an image.

This allows the use of linear quantisation with a quantisation step *q* as defined in equation 3.1, which is derived from equation 2.24 .

 (3.1)

where q is the quantisation step,

*Q* is the quantisation factor,

and *M* is the block size (the number of pixels in the block is order *M2*).

The same quantisation step is applied to each transform coefficient and produces roughly the same error in each coefficient. The quantised coefficients typically have an entropy of about 2-4 bits/symbol (observed over the course of this work) and therefore it is possible to use a Huffman coder without any problems.

The block size (*M*), the quantisation factor (*Q*), and the complexity of the transform are the only variables in a low complexity image compressor. It is necessary to decide how to change each of these variables to produce the optimal rate distortion (RD) performance curve for the system. The RD curve is the comparison of error (usually MSE or PSNR) vs. the compression of the system (usually in BPP or compression ratio). Varying the block size of the transform alters other factors, such as the quantisation step and entropy compression, therefore this can be used to produce rate distortion curves for images, if *Q* and the transform complexity remain constant. In this chapter an optimal value for *Q* will be found for each type of compressor by fixing the transform complexity at a marginal level and varying *Q* to give a set of rate distortion curves. The *Q* value with the best overall rate distortion performance can then be fixed as optimal for the given compression method. Given a fixed value of *Q,* it is possible to produce a set of rate distortion curves for different transform complexities, allowing the best transform complexity to be chosen. Although this progression does not give an exact solution to the three-variable problem, the results are practically acceptable.

All experiments in this chapter use linear quantisation (described in section 2.3) and Huffman lossless compression (described in section 2.4). The Huffman compressors use an optimised probability density function derived over a number of iterations of the image compressor.

1. **Polynomials**

Polynomial functions can be used to describe the pixel intensities of an image block. Polynomials can be quite basic, but they quickly become more complicated as more are added. The basic form of the polynomial transform is (equation 3.2):

 (3.2)

where  is the approximated image block,

and  is the transform coefficient *k,l.*

The basic form of equation 3.2 extends to an infinite number of polynomial functions, but it is common to use roughly six functions or transform coefficients to approximate an image block, as found in many fractal methods [13-19]. These polynomial functions are not orthogonal and are centred so that x and y are zero at the centre of the image block, as in figure 3-2.



Figure 3-2. Diagram of block centering for polynomials.

It is possible to calculate the coefficients of any set of polynomial functions for an image block by using a least squares fit method. The squared error between the original and the approximated image is shown in equation 3.3:

 (3.3)

where  is the pixel intensity map of the image block,

 is the approximated image block,

and .

Since  can be represented as a set of polynomial functions, we have:

 (3.4)

The least squared error method indicates that the best fit of the coefficient  occurs at the minimum error, when the differential with respect to  is zero, as demonstrated in equation 3.5.

 (3.5)

where .

It therefore follows that equation 3.6 is true for all transform coefficients  :

 (3.6)

Equation 3.6 is repeated for all transform coefficients giving N equations and N unknowns, which is a problem that can be solved by back-substitution. This gives a standard method to calculate the coefficients of a polynomial transform for any image block.

Although the above method can be used to fit polynomial functions to an image block, it is better to orthogonalise the polynomials found in equation 3.2, so that there are no relations between the transform coefficients . The Gram-Schmidt orthogonalised basis functions are shown in their one dimensional form in figure 3-3.



Figure 3-3. Diagram showing orthogonalisation of polynomial functions in 1 dimension.

As the order of the polynomial function increases, it becomes more important to decide in what order to apply the Gram-Schmidt method, since it will affect the final shape of the orthogonal function. As the indices of  increase, the size of the coefficients make less of a contribution to the approximation of the image. For this reason the order of orthogonalisation shown in figure 3-4 is used. This zigzag reordering generally arranges the coefficients in order of significance, with the most significant first, and it is used in JPEG for quantisation/run-length purposes. In this order the coefficients most affected by orthogonalisation are least important to the image block approximation.



Figure 3-4. Diagram showing zigzag ordering used to apply Gram-Schmidt orthogonalisation to polynomials.

The polynomials examined here are orthogonal, since they are easier to calculate and have proved to be more efficient than non-orthogonal functions. The experiments in the next two sections use orthogonal polynomials with linear quantisation (described in section 2.3) and huffman lossless compression (described in section 2.4).

**3.3.1 Experiment: Optimise the quantisation factor *Q* for polynomials.**

All the experiments in this chapter use very similar methods and this section will go into detail how each process is implemented. The details described below are carried out for all of the experiments but they are not mentioned to avoid needless repetition.

The aim of this experiment is to find the optimal value of the quantisation factor Q for the Goldhill test image. The quantisation factor controls the amount of quantisation applied to all the coefficients of each block. For a fixed value of Q the quantisation step applied to each coefficient will increase with the block size as described in equation 3.1. After the quantised transform coefficients were calculated they were losslessly coded by using a Huffman lossless compressor. The probability density distribution (PDD) that was used to generate the Huffman compressors symbols was produced by iterating the codec for each value of Q at different compressions until a stable PDD was obtained.

The number of polynomial functions was fixed to 6 as shown in equation 3.7, since this has been found previously in similar work [17] to be a reasonable number of functions to use.

 (3.7)

To produce a rate distortion curve for a particular value of Q the size of the image blocks was varied. Since the number of coefficients was fixed to 6 per block, increasing the size of the block also increased the compression of the codec. By varying the block size from 2x2 pixels to 20x20 pixels it allowed the compression to be changed by a factor of 100 (due to the areas involved).

The quantisation factor *Q* was varied over the range , to find its optimal value. A rate distortion curve was generated for each value of *Q* by varying the block size of the applied polynomial approximation.

Figure 3-5 shows the rate-distortion curves produced for various values of Q and it can be seen that *Q*=4 is the best value for *Q*. Not all the values of *Q* are shown in figure 3-5, so that the results are distinguishable, but the ones shown do demonstrate the general trend of the results. The quantisation factors slightly above *Q=4* are roughly as effective as *Q=4,* but since the higher levels of quantisations are less visually pleasing, the value of *Q=4* was chosen to be best (as shown in figure 3-6). *Q* can now be set for the following experiment, allowing the best number of functions to be found.



Figure 3-5. Rate distortion performance curve for a 6 coefficient polynomial function, with varying *Q* values.

Figure 3-6 shows a section of Goldhill coded at ~0.15 BPP with different quantisation factors (Q). The root mean squared error (RMSE) figures shown are for the whole Goldhill image, not just the fragment. Above Q=4/5 the quantisation error is more noticeable, but the RMSE does not reflect this and this is a serious failing of present error measures.





(a) Original (b) Q=2.0 (c) Q=3.0 (d) Q=4.0

RMSE 10.583 RMSE 10.524 RMSE 10.086

0.156 BPP 0.164 BPP 0.167 BPP



(e) Q=5.0 (f) Q=6.0 (g) Q=7.0 (h) Q=8.0

RMSE 10.257 RMSE 10.452 RMSE 10.0547 RMSE 10.273

0.152 BPP 0.140 BPP 0.158 BPP 0.150 BPP

Figure 3-6. 128 x 128 Sections from the Goldhill test image compressed to 0.15BPP.

**3.3.2 Experiment: Find the best number of polynomial functions to use for compression.**

The quantisation factor was fixed to *Q*=4, as found in the previous experiment, and the number of polynomial functions used to describe the image blocks was varied. Rate distortion curves were produced by varying the block size that the polynomial approximation was applied to, while keeping the order of the polynomial constant. Figure 3-7 shows the variation of the polynomial functions that were assessed. Although the order of the polynomial was varied, as shown in figure 3-7, the functions were kept symmetrical, since most images do not favour any particular orientation and this makes the compressed images more visually pleasing.



Figure 3-7. Diagram of the polynomial functions showing the variation of the orders tested.

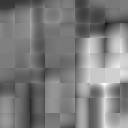
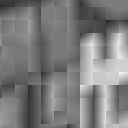
The Rate distortion curves shown in figure 3-8 indicate that increasing the complexity of the polynomial functions improves the performance of the system. It is clear from figure 3-8 that there is increasingly less to gain from using additional polynomial functions, but the order 12 polynomial does perform the best.



Figure 3-8. Rate distortion performance for *Q=4* varying the order of the polynomial.

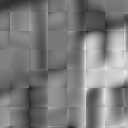
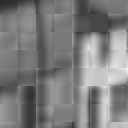
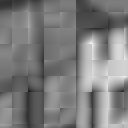
Not all the orders are represented in figure 3-8 for convenience, but they follow the same general trend as shown. Since order 12 must use a 5x5 block to work effectively and 4x4 blocks are a major part of low complexity image compressors, it is not desirable to use order 12 transforms. The cross terms used in order 6 and above (,  and ) are very computationally intensive and so this makes order 10 undesirable, since it has two additional cross terms for no major gain. The best orders to use seem to be order 6 and 8, since they perform well without being ruled out on other grounds, this is illustrated by figure 3-9. Order 1, 3 and 5 do not perform well, when compared with orders 6, 8 and 10, which have roughly the same RMS error. In summary, order 12 performs best, but it is too complex to be useful and in further work it would be best to use order 6 polynomials giving the best performance with operating efficiency.





(a) Original (b) Order 1 (c) Order 3 (d) Order 5

RMSE 16.249 RMSE 13.607 RMSE 12.804



(e) Order 6 (f) Order 8 (g) Order 10 (h) Order 12

RMSE 12.346 RMSE 12.075 RMSE 12.142 RMSE 11.794

Figure 3-9. Sections from the Goldhill test image compressed to 0.07BPP

1. **Limited Coefficient DCT**

The Discrete Cosine Transform [28] (DCT) is an orthonormal transform which is often used in image compression. The DCT is a lossless transform and it is generally used in complex image compressors, but if some of its coefficients are ignored, it can be used in a more simple way. The transform is applied to a square blocking of the image as described previously, but works best, when applied to 2n sized blocks (where n is an integer), since a Fast Cosine Transform [51] (FCT) can be used. The DCT has the general form shown in equation 3.8 and its coefficients are calculated using an inner product method shown in equation 3.9 :

** (3.8)

 (3.9)

where ** is the pixel intensity at *i,j*

 is the transform coefficient at *k,l*

and M and N are block side lengths.

The one dimensional representations of the DCT functions are shown in figure 3-10. The edges of the blocks are always flat and this ensures that the DCT has minimum discontinues at the edges of blocks. However when the coefficients are quantised, there tends to be a difference in grey-scale level between adjacent blocks in DCT methods which produced blocking artefact. This artefact is worse for larger blocks and coarse quantisation, but its visual effect is not manifested in the rate distortion performance curves, which are used to measure performance.



Figure 3-10. Diagram showing 4 discrete cosine functions

1. **Experiment: Optimise the quantisation factor *Q* for limited coefficient DCTs.**

Limited coefficient DCT’s do not use all the available coefficients provided by the lossless DCT with most of the high frequency transform coefficients being truncated. The number of DCT functions was fixed to 6 as shown in equation 3.10. The quantisation factor *Q* was then varied over the range . A rate distortion curve was generated for each value of *Q* by varying the size of the blocks that the polynomial approximation was applied to :

 (3.10)

where ** is the pixel intensity at *i,j*

 is the transform coefficient at *k,l*

and M and N are block side lengths.

Figure 3-11 shows the rate distortion curves produced for various values of Q and it can be seen that *Q*=4 is the best value for *Q*. Higher values of *Q* are not as effective, and they quantise the image more coarsely, which can be visually displeasing. *Q* can now be set for the following experiment, allowing the best number of functions to be found.



Figure 3-11. Rate distortion performance curve for a 6 coefficient DCT function, with varying *Q* values.

.



(a) Original (b) Q=2.0 (c) Q=3.0 (d) Q=4.0

RMSE 10.538 RMSE 10.274 RMSE 9.918

0.156 BPP 0.165 BPP 0.168 BPP



1. Q=5.0 (f) Q=6.0 (g) Q=7.0 (h) Q=8.0

RMSE 10.090 RMSE 10.297 RMSE 9.979 RMSE 10.204

1. BPP 0.141 BPP 0.158 BPP 0.150 BP

Figure 3-12. Sections from the Goldhill test image compressed to 0.15BPP

**3.4.2 Experiment: Find the best number of DCT basis functions to use for compression.**

The quantisation factor was fixed to *Q*=4, as found in the previous experiment, and the number of DCT basis functions used to describe the image blocks was varied. Rate distortion curves were produced by varying the block size that the DCT approximation was applied to, while keeping the order of the DCT constant. Figure 3-7 shows the variation of the polynomial function that were assessed in graphical form and this is exactly the same with the DCT, but the DCT functions replace the polynomial ones. The choice of DCT coefficients was also kept symmetrical as with the polynomial.

The Rate distortion curves shown in figure 3-13 indicate that increasing the complexity of the DCT basis functions improves the performance of the system. It is clear from figure 3-13 that there is increasingly less to gain from using additional DCT functions, but the order 12 DCT does perform the best. Not all the orders are represented in figure 3-13 for clarity but they follow the same general trend. For the same reason, as stated in section 3.3.2 the order 6 DCT is probably the best, since the computational increase for order 12 is much higher than the improvement in rate distortion performance. Figure 3-14 shows that there is virtually no visual difference between the higher order approximation and hence order 6 is an effective choice.



Figure 3-13. Rate distortion performance for *Q=4* varying the order of the DCT.



(a)Original (b) Order 1 (c) Order 3 (d) Order 5

RMSE 16.248 RMSE 14.272 RMSE 12.635



(e) Order 6 (f) Order 8 (g) Order 10 (h) Order 12

RMSE 12.223 RMSE 11.985 RMSE 11.929 RMSE 11.576

Figure 3-14. Sections from the Goldhill test image compressed to 0.07BPP

**3.4.3 Experiment: Compare the DCT to the Polynomial Approximation.**

The rate distortion curves for the order 6 DCT and order 6 polynomial using Q=4 are compared in figure 3-15. It can be seen that the DCT has slightly better performance than the polynomial method. Although this difference is quite small, it is enough to discard using the polynomials in future work, since they are the same computational complexity as truncated DCTs. The rest of this chapter and Chapter 4 uses the DCT to approximate the image rather than constantly trying both polynomials and DCT.



Figure 3-15. Rate distortion curves comparing polynomial basis functions with the DCT basis functions for compression of the Goldhill test image.

1. **Fractal Transform**

The fractal transform works by mapping large regions of an image onto smaller ones, hence improving the approximation of the smaller area. Fractal methods work best with images that contain self-similarity at different scales, and hence are improved, when fractal mapping or transforms are applied. The basic form of a fractal transform is shown in equation 3.11. It contains two parts, a basis approximation and a fractal transform. The basis functions give an initial approximation of the image block and the fractal transform improves upon this approximation.

 (3.11)

where  is the approximated image block,

 is the basis function *k,l* (such as a polynomial or DCT),

 is the transform coefficient for ,

 is a region of the image that has been mapped onto (parent block),

and  is the fractal coefficient.

A good demonstration of a fractal transform is shown in figure 3-16(a), where an edge passes through a large region (referred to as the parent block) and a small region (known as a child block). The edge is shown in profile in figure 3-16(b), and using a simple basis function an approximated edge can be generated. The approximated edge is not perfect, but if the approximation is shrunk and mapped back onto itself, it is possible to improve the approximation (figure 3-16(c)). This demonstrates, how a fractal method introduces higher order details into the image, without the use of higher frequency components, which can have a rate distortion penalty.



Figure 3-16. Diagrams demonstrating the operation of a fractal transform.

Fractals were first exploited for image compression by Barnsley [13] , who used a simple greyscale level for the basis function. This produced poor results, since very little of the image’s detail could be encoded by a greyscale level across an image block. The fractal transform works best, when it is used to improve a reasonable approximation of an image. If the approximation of the image is too good, then the fractal finds nothing to correct and it is an ineffective overhead. If the approximation is not good enough, then the fractal is unable to correct the image sufficiently to be useable. The fractal works best, when a medium complexity approximation is used as the basis function. To this aim Monro [14 - 17] continued the work of Barnsley by increasing the complexity of the basis function. It was found that a bi-quadratic polynomial function worked best (Order 5). A further enhancement to fractals was provided by Øien [19] who orthogonalised the parent block with respect to the basis functions used, after it had been mapped onto the child block. This prevents the fractal transform repeating anything already described by the basis functions, and is therefore more efficient.

Other work has been done on increasing the complexity of the fractal transform by varying the position of the parent block. In figure 3-16 the parent block is centred on the child, and although this position does give an acceptable result, it is possible to place the parent elsewhere within the image. The amount of information used by the fractal part of the transform increases, if the best parent location in the image is searched for. It also increases the complexity of the transform, since the parent has to be searched for by the encoder. There are several methods of limiting the parameters of the search for the best parent block.

In this section an orthogonal basis of order 6 DCTs will be used from the previous section (equation 3.10) and the parent block will be orthogonalised with respect to the basis function. Several variations of the fractal methods will be assessed, including centering the child on the parent, local parent searching and a new method that reduces the searching even further.

**3.5.1 Experiment: Using a centred parent fractal to improve order 6 DCTs.**

A fixed block size order 6 DCT was used to approximate the image. A fractal parent block twice the size of the child was assumed to be centred around each child block, where the image allowed (the position was modified so that the parent block was never outside the image) as shown in figure 3-16. During the fractal transform the parent block was shrunk to map onto the child block and the shrunk parent was orthogonalised with respect to the order 6 DCT. The orthogonal parent was then fitted to the image to improve the order 6 DCT approximation.

The image was rendered by calculating the DCT basis approximation once for each block and iterating the fractal part [18] (including orthogonalisation) three times. The fractal transform used here converged to an acceptable solution after the first iteration, since the basis approximation is quite effective (and hence there was a limited amount of ‘fractal’ content), but further iterations were performed to improve the fidelity of the image.

The block size used to apply the order 6 DCT was varied to produce a rate distortion curve shown in figure 3-17, where the order 6 DCT is compared to the centred parent block fractal method. It can be seen that the fractal method produces an improvement in the DCT at higher compressions but at lower compressions it does not benefit as much from using fractals. The fractal transform are also more visually pleasing than the plain DCT method, since they enhance the edges in the image which has a more noticeable psycho-visual effect.



Figure 3-17. Rate distortion performance curve for centred parent fractal methods compared to order 6 DCTs.

1. **Experiment**: **Use an offset parent block, found by best fit searching to improve on the centred fractal parent method.**

A fixed block size order 6 DCT was used to approximate the image. A fractal parent block twice as large as the child was located around each child block. The offset of the parent block away from being centred around the child was allowed to vary child block sizes (N) in both the x and y directions, as shown in figure 3-18. The offset to the parent block that improved the child block the most was Huffman coded and stored with each child. The parents were orthogonalised [19] (using Gram-Schmidt) with respect to the order 6 DCT to prevent duplication of data. The fractal was rendered using three iterations as in the previous experiment.



Figure 3-18. Diagram of searching offsets for parent position.

The size of the child blocks was varied to produce a rate distortion curve for fractal searching. It can be seen from figure 3-19 that offset searching slightly improves on the centred parent block fractal methods. Unfortunately the offset searching method is quite slow in encoding terms and is therefore only suitable for off-line coding methods. Several workers [15] had suggested that fractal searching did not improved over centred parent blocks. The difference in this work is that the parent blocks are orthogonalised, this removes the components of the order 6 DCT that were undoubtedly influencing the older fractal encoders to use blocks more like the child (hence they tended to use the centred child block since it was the closest possible match). This lead to the method correcting the quantisation error in the image rather than the shape of the image.



Figure 3-19. Rate distortion performance curve for fractal searching compared to the centred fractal and the order 6 DCT.

1. **Limited Fractal Searching**

Experiment 3.5.2 demonstrates a simple method of fractal searching, where the search for the best parent block is conducted locally around each child block. This seems a sensible way of finding the best parent block, until the effects of the fractal are examined. The major use of fractals in image compression is edge enhancement. An image compressor usually discards high frequency components of the image and these are basically the edges. The fractal provides an excellent way to rebuild these edges at a reduced rate distortion cost [6].

There are only so many ways an edge can pass through a centred parent block and when the parent is orthogonalised, groups of parent blocks emerge that have very similar shape. This quality of fractals can be exploited to improve the rate distortion performance of the system.

It is necessary to develop a method to reduce the total number of parent blocks used to code an image. It is assumed that a pool of parent blocks has been found (possibly from centred parents around each child) and that this pool of parents needs to be pruned in such a away that the rate distortion characteristics of the compressed image are improved.

If a rate distortion curve can be generated for the basis approximation (e.g. for the order 6 DCT), then at a given compression the gradient of the curve can be calculated and used as a measure to compare with the fractal enhancement. A single fractal coefficient can be compared to the average rate distortion performance of the basis approximation, as shown in equation 3.12. This assumes that the fractal coefficients are Huffman coded with a previously generated PDF. If the fractal coefficient’s rate distortion is less than the average then it does not improve the overall rate distortion performance of the system and therefore can be discarded.

 (3.12)

These coefficients are discarded for reasons shown graphically as in figure 3-20. When all the fractal coefficient in an image are calculated the net improvement is shown in figure 3-20a, unfortunately the coefficients with the highest improvement are made less significant by other less effective ones. If the ineffective fractal coefficients are removed or pruned then the net fractal improvement increases, as shown in figure 3-20b. To decide which fractal coefficients to discard, it is necessary to set a threshold. Since the fractal method is attempting to improve the basis approximation, it makes sense to discard all fractal terms that are not as effective as the basis approximation alone.



Figure 3-20. Diagram showing the effect of fractal component pruning.

This gives a method for reducing the number of parent blocks in the parent pool. The parent blocks, which produce a rate distortion contribution less than the average for the basis approximation, can be discarded. The parent pool can then be reduced to only the parents that improve the basis approximation.

**3.5.4 Experiment:** **Demonstrate the Limited search method, using a parent pool generated form local centred parents and compare the results to other fractal methods.**

The image was broken into segments, approximately 128x128 pixels (depending on the block size and image edges). These segments were approximated using order 6 DCT blocks, with centred fractal parent blocks, to improve their quality. The initial parent pool of blocks was formed from the parents of all the child blocks in the segment. A null/flat block was added to the parent pool to cope with non fractal blocks. After this parent pool was formed, each child in the segment was fitted to the parent which best improved its fidelity from the parent pool. The probability density distribution (PDD) for the parent block was found to be of the form shown in figure 3-21. Certain parents were much more popular than others and this can be exploited to improved the rate distortion of the method.



Figure 3-21. Diagram showing the Probability Density Distrubtion for parent blocks chosen by child blocks as best fits.

The individual  was calculated for each parent block (accounting for the number of time a parent was used). All the parents, which were not used were pruned form the parent pool along with any that were below the average rate distortion performance of the order 6 DCT at a particular compression. The average rate distortion performance of the order 6 DCT at a particular compression was calculated from previous results, by linear interpolation.

The child block size was varied to produce a rate distortion curve, as shown in figure 3-22, where limited searching is compared to offset searching, centred fractals, and the order 6 DCT. It can be seen that limited searching outperforms all the previous methods, but its takes even longer to encode than offset searching. Both methods require block matching, but limited searching only works well then the original block matching pool is very large. This can be around 1000 blocks matching for every child blocks in limited searching, compared to around 81 blocks matching for the offset searching method used here (offset of 4 child block sizes).



Figure 3-22. Rate distortion performance curve for limited fractal searching compared to normal searching, centred fractal and the order 6 DCT.

**3.6 Vector Quantisation**

There has been a lot of work done in the area of vector quantisation (VQ) and major contributions were made by Linde [24], Gersho and Gray [20] in the early 80’s. VQ has been primarily concerned with compressing the whole image by using a fixed block sized vector quantiser with a predetermined library of VQ blocks. Further work improved on this fixed block size method and used hierarchical structures [40, 41] and wavelet methods [27].

In this section we will look at a very simple application of VQ, which is relevant to the fractal transform. This method uses a variant of vector quantisation first described by Gray [42] and a similar method was discussed by Bethel [1]. The VQ is not used for coding the basic details of the image but is rather used to encode the residual error generated after coding the image with a truncated DCT. The pruning criterion is similar to that used in the fractal method previously and is original to this work. The vector quantiser works in a similar way to fractals in this case and is shown in equation 3.10.

 (3.10)

where  is the approximated image block,

 is the basis function *k,l* (such as a polynomial or DCT),

 is the transform coefficient for ,

 is the *nth* vector quantisation block,

n the index of the best vector quantised block that improves the DCT approximation,

and  is the vector quantisation block scaling coefficient.

This equation can be seen to be very similar to the fractal method described in the previous sections. The primary difference is that the set of VQ blocks () are stored and transmitted with the compressed image. Transmitting the set of VQ blocks can be a large overhead, but the blocks have a much more beneficial effect than fractals and hence the overhead is warranted.

The set of VQ blocks is generated using the Lloyd-Max quantisation process explained in chapter 2, but applied to image blocks, rather than quantisation bins. The vectors are Huffman coded but this does not gain much compression, since they are produced by Lloyd-Max quantisation. In this method a set of vectors are transmitted for each 128x128 pixel section of the image, so that local similarities of the residual error image can be exploited. The vectors are generated in the following way:

* The image sections are coded by an NxN approximation (DCT)
* The approximating function (in equation 3.10) is then quantised and the residual error image is calculated.
* An error block for each NxN tile is then generated and orthogonalised with respect to the approximating basis function.
* Every NxN error block is coded by fitting another NxN error block to it (except for itself). In this way the number of times an error block was chosen and how much error was cleared up by it can be calculated. This gives each block in the image section a rate . If this rate is larger than the approximating function , then the block is maintained since it is useful, otherwise it is discarded.
* The error blocks are pruned using this Rate distortion criterion and the surviving blocks are used to partition the error blocks into new sets.
* Each partition is averaged (weighted by its scaling) to produce a new set of better VQ blocks.
* These VQ blocks are used to code the error blocks again, generating a new quantisation partition.

This process is continued until a stable partition is formed, where all the blocks have a sufficiently high rate-distortion pay off.

**3.6.1 Experiment: Generate the Rate Distortion Curves for an Order 6 NxN DCT Cleared Up Using VQ.**

The image was approximated by using an order 6 DCT with an *NxN* block size. The coefficients were quantised and losslessly coded, as in section 3.4. The residual error image was then formed from the original and approximated images. This residual error image was split into approximately 128x128 pixel sections. The *NxN* Residual Image Blocks (RIBs) within these regions were coded by using the vector quantisation processes described above. Each 128x128 section of the image had associated sets of RIBs that were transmitted with the section. Individual blocks had vectors assigned to them that produced the highest reduction in error as well as a scaling associated with the VQ block.

The results for this method which were produced by varying blocks sizes (N) are shown in figure 3-23 and compared with the limited searching fractal method. It can be seen that the vector quantisation method is superior to the limited search method at the lower compressions. Fractals do not work well, when the image basis approximation is already very good, but as the compression increases and the overhead of transmitting the VQ RIBS also increases, the fractal method once again becomes dominant.

This VQ method is clearly very effective at lower compressions and therefore might be useful if it is used at low compressions rather than the fractal.



Figure 3-23. Rate distortion performance curve for VQ RIBs compared to limited fractal searching, using order 6 DCTs.

**3.7 Summary**

This chapter has explored the low complexity transform methods. Most of these methods can be improved on, when they are placed in hierarchical structures as shown in chapter 4, although there are some difficulties. Both the limited fractal searching and the vector quantisation methods rely on sections of the image having the same block size and these do not appear in hierarchical methods. The last two methods are also very slow, compared to the other methods described in this chapter, which might not be appropriate to video systems.

In this chapter several points have been shown:

* DCTs are better for compressing images than orthogonalised polynomials.
* Fractals can be used to improve DCT basis approximations.
* Increasing the complexity of the fractals used has diminishing gains, but does improve their performance.
* The rate distortion limiting/switching is an effective method of improving a compressor.
* VQ RIB has the best performance of the fixed block size compressors at low compressions and limited search fractal coding has the best performance at high compression.

Although these have been shown to be true for fixed block size methods this does not mean that the same will be true for hierarchical methods and this will be explored in the next chapter.