**Chapter 5: Image Compression with Optimal Source Coding.**

**5.1 Introduction.**

In this chapter methods for coding the coefficients of an orthogonal transform are discussed. A method is described which optimally quantises the coefficients of any orthogonal transform to produce the minimum Mean Square Error (MSE) for a particular compression, in the quantised coefficients of the transform. It has been applied in the field of vector quantisation and wavelet image compression by several previous workers [23, 45 - 50]. This chapter introduces the technique and also looks at the practical requirements of the method and its applications.

The basic processes in an image compressor are shown in figure 5-1. The transform, quantisation and lossless compression stages have been previously described in detail in chapter 2. In the following section the relationship between the processes will be examined.



Figure 5-1 Block diagram of the stages of an image coder.

The first stage of the image compressor decorrelates the image data. The transform makes the coefficients easier to compress, but if it is lossless (which we assume here), no data is lost, so the relationship between the transform and the source coding stage is fixed and can be safely ignored.

The source coding is split into two parts: quantisation and lossless coding. These processes are usually simplified since we are only dealing with sets of transform coefficients. This data can easily be quantised and subsequently losslessly coded. This is much easier than the situation encountered in vector quantisation, where there is a more complicated problem.

**5.2 Quantisation and Lossless Coding.**

Together the quantisation and lossless coding can be considered to be the source coding stage of the process. The lossless coding is considered to be entropy coding for the purposes of this chapter, to avoid unnecessary complications. This means that applying a certain quantisation on the transform coefficients will lead to the coefficients being described by a fixed number of bits. The transform produces a set of coefficients, as shown in equation 5.1 :

 (5.1)

These coefficients are then quantised by *qi* to give a quantised set of coefficients as shown in equation 5.2 :

 (5.2)

These quantised coefficients can then be entropy coded to find out, how much information they contain. This can be thought of as applying an entropy coding function (*E*) to the quantised codes, as shown in equation 5.3 to give a set of compressions *X* :

 (5.3)

The set of compressions X, as shown in equation 5.4, is the number of bits required to quantise each individual coefficient :

 (5.4)

The process of taking the coefficients from equation 5.1 and transforming them through to equation 5.4 is a complicated system, which can be greatly influenced by any of the by any of the processes shown. The coefficients are affected by the quantisation scheme to produce a certain overall compression for the system, but the effectiveness of this depending on how well each coefficient compresses during entropy coding.

The overall compression of the system can be expressed as the average of the compressions of the individual transform coefficients. This can be simply expressed as shown in equation 5.5 :

 (5.5)

The Mean Squared Error (MSE) of the image is also the sum of the MSE in the individual transform coefficients, provided that we are using an orthogonal transform. This is demonstrated in equation 5.6. Orthogonal transforms have additive MSE, because they obey Parseaval’s theorem as shown in equation 5.7 :

 (5.6)

 (5.7)

The processes that have been discussed here can be thought of as one stage, as shown in figure 5-2.



Figure 5.2. Block Diagram of source coder

Although it can be seen that this system is quite complicated, the process can be considered in isolation. If the transform stage is considered fixed, then the source coding stage can be of three types:

* Driven by quantisation, such as JPEG.
* Driven by entropy coding, such as hierarchical coders.
* Driven by a combination of the first two methods, such as zero tree wavelet coders.

The basic requirement of an image coder is to produce the minimum distortion in the recovered image for the maximum compression and this problem will be discussed in the following sections.

**5.3 Definition of Problem.**

Although the requirement of a good image coder is to produce an optimal rate distortion performance curve, the effectiveness of the coder is optimised by having the minimum MSE at a particular compression. Since more than one coefficient is affected by the source coder (as shown in equation 5.8), to minimise the MSE this is a very complicated multivariable problem in n+1 dimensions :

 (5.8)

There is no obvious solution and so most compression techniques choose to make assumptions about the data to be compressed. These assumptions often link the multivariable problem together, in some sensible way. This is acceptable, while the source coding method is relatively simple, but becomes more difficult with complicated source coders.

Most image compression schemes overlook this relationship and effectively treat the source coding as two isolated steps. The effect of this is to produce a functionally simple system, where the user has very little control.

JPEG is one such system. The ‘quality’ of the image required is specified by the user. This quality affects the level of quantisation applied to the DCT transform coefficients and they are then entropy coded using a fixed Huffman table method. The user can therefore control the compression of the image coder, but the level of control is very low.

In an ideal system the exact compression required by the user would be entered and the appropriate quantisation and entropy coding scheme would be chosen.

This can be achieved by progressive systems like Shapiro’s zero tree implementation [36]. The progressive coders assume some type of quantisation scheme but apply it in steps so that the image can be slowly built up. This method allows any compression desired to be obtained and can be applied to most coder types. Unfortunately as the complexity of the quantisation scheme increases so does the difficulty of applying the scheme progressively. Progressive coders do not tackle the problem of source coding directly, but achieve the desired results by an abstracted method.

1. **Optimal Source Coding.**

The problem is to produce a near optimal MSE for a fixed total compression ratio. A method is discussed here, which limits the problem sufficiently so that it has a well defined solution, but does not make assumptions about the data source. The basic principles of the method are discussed in this section and the details of their application to a real source are discussed in the next section.

The method can be applied to the coefficients of any orthogonal transform, with any source coding method, provided that a rate distortion characteristic can be produced for each coefficient. This means that the source coder must be able to readily produce a rate distortion characteristic for each coefficient of the transform. Although this process may take some time, it can been shown that the rate distortion characteristics are very similar and can be adequately described by models. This modeling by templates is discussed in a later section and makes the process applicable to fast systems such as real time video.

The method assumes that the rate distortion characteristics of the coefficients have some features in common. Although these assumptions are necessary for the method to work, they do not arise from the nature of the source and are reasonably trivial. The assumptions are:

* The gradient of the rate distortion curve slowly decreases from a maximum at *x=0* to a minimum at large *x*.
* The maximum error is at *x=0,* when the coefficient is completely truncated as shown in equation 5.9 :

 (5.9)

* The gradient is asymptotic at the compression, which corresponds to no quantisation.

These assumptions are shown in figure 5-3, which shows the typical shape of a rate distortion curve for an image.



Figure 5-3. Graph demonstrating the typical shape of a rate distortion curve expected from image data. The key points being the truncation error at *x=0*, and the asymptotic but steady decrease as *x* tends to a low compression.

In practice, real curves are not always as regular as assumed here and so all the rate distortion curves are passed through a filter to make them acceptable to be used in the method. This does affect the accuracy of the results produced, but allows the method to function, when a failure would have otherwise occurred. The details of this filter are given in a later section.

If the rate distortion curves are known for all the transform coefficients, then the problem is to minimise the total MSE in *n* variables. This can be ‘solved’ by a conventional iterative method, such as the simplex method [62], but there is no fixed time for this to converge. Since the method is repeatedly used for a large number of coefficients, it needs to be simple and well terminated. To make the problem more simple to solve, the n dimensional space is limited (by the assumptions), so that there are no local minima and only one global solution. The solutions found in this way should be close to or exactly the same as the real solution, provided the assumptions were not widely different from the real situation.

When the rate distortion curves for each coefficient are calculated and the required limitations imposed on the data, the actual solution is trivial. The best MSE at a fixed compression ratio occurs, when the gradient of the MSE vs. Compression graphs are equal as illustrated in equation (5.10) :

 (5.10)

This can be demonstrated by considering a system with just two coefficient sets to be source coded. The rate distortion curves are shown in figure 5-4.



Figure 5-4. Graph demonstrating the equal gradient method.

Both rate distortion points must lie somewhere on the rate distortion curves and have a fixed compression. To improve the MSE, the number of bits per pixel of one of the coefficients must be increased at the expense of the other, to maintain constant compression. This is shown in figure 5-5.



Figure 5-5. Diagram showing the Two gradients of the points from figure 5-4.

While the gradients of the MSE vs. compression curves are different, it is always possible to produce a net improvement in MSE. The only stable position for the points on the set of rate distortion graphs is, where the gradients of all the curves are equal. This explanation applies equally well to more than two coefficient sets, because to produce the maximum MSE improvement only the highest and the lowest gradient points need to be altered (any points in between would have less of an effect than the extreme points).

To contrive a working method to find the best MSE for a fixed compression, the rate distortion graphs are first differentiated to produce a new set of differential curves as shown in figure 5-6.



Figure 5-6. Diagram of rate distortion curves after differentiation. The fixed gradient line shows a point of optimal compression.

A gradient is chosen to give an overall compression close to the desired compression ratio. This is done by averaging the gradients found, when each coefficient is passed an equal share of the bits available (). The actual gradient, which corresponds to the desired compression ratio is found by using a binary search, which is illustrated in figure 5-7. An initial guess provides the first gradient used (gradient A), but the compression produced is too high. The gradient is then ‘stepped’ in the right direction to gradient B. Again this gradient is not correct but this time it is too low, so a new gradient C is tried, which is half way between A and B (this is the essence of a binary search). This process is continued, gradually bringing the ‘test’ gradient closer to the desired gradient which results in the correct compression.



Figure 5-7. Diagram of rate distortion curves illustrating the binary search.

This is possible since:

 then the gradient is too large

 then the gradient is too small.

The binary search is terminated at the level of accuracy desired. If a curve does not have a significant gradient at *x=0*, then it is effectively truncated. This often occurs at higher compression ratios, until only the mean coefficient values are left.

If a full rate-distortion curve is required for the entire method, it would be possible to simply step through the whole set of gradients, producing an optimal rate distortion curve for the whole image.

1. **Application of Method to a Real Source.**

The previous discussion has ignored the problems caused by dealing with a real data source. In a real application the rate distortion curves are stored as a collection of points. This problem could be ignored, if the number of data samples was excessive but in a real system this is not an option. The shape of the rate distortion curves indicate that very few samples are required to obtain a good representation of the data’s statistics. This is because few fluctuations in the rate distortion are expected, and these fluctuations would have to be filtered out, if they were found. In this way a coarse sampling of the rate distortion curves can be thought of as finding the general shape of the rate distortion curve but ignoring the details.

The processes applied to the data can be kept simple, because the level of accuracy required to produce a noticeable improvement in the MSE is quite low. The reason for this is simple. The gradients are all equal at the optimal point and therefore small differences caused by unequal gradients cause very little degradation in the overall MSE. The only problem, which can be caused by the system’s accuracy is truncating a coefficient instead of applying a very small number of bits to it, gradients can vary rapidly at very low bit rates and a small error in the compressions can give a large relatively important MSE improvement.

The rate distortion curves need to be sampled about 10-20 times across their active regions. If there is more than 1% of the coefficient’s amplitude left after quantisation then the rate distortion curve is still considered active. After this level a simple linear approximation of the change in MSE with compression is more than adequate. The sampling of the curves gives a set of ‘support points’, which can be used to find the relative compressions applied to each set of transform coefficients. The processes carried out on this data are shown in figure 5-8.



Figure 5-8. Block Diagram of steps required to calculate the optimal compressions.

The first step is to convert the support points into a differential form. The two points at the opposing limits of the curve are replaced with the exact values, to ensure that the data varies from the truncation value to zero at a sufficiently large compression. These two sample points correspond almost exactly to those predicted in all of the cases examined but this stage is included for completeness.

When the limits have been ensured, it is possible to convert the rate distortion curve into differential form. This is done by fitting a cubic spline through the support points and differentiating the resulting function to produce a set of differential support points that can be used to solve the equal gradient problem.

The gradients between the differential support points are estimated using linear interpolation since their accuracy is sufficient for the gradient process and it is faster than the spline method.

The next stage is to filter the gradient vs. compression graph to remove any unwanted fluctuations. In the case of the DCT, where image coders are used to source code the coefficient sets, the fluctuations are found to be of the form shown in figure 5-9, (this is an exaggeration of the actual results, fluctuations did not regularly occur to this extent, but it illustrates the effect). In certain circumstances large fluctuations did occur and this problem had to be addressed. During these large fluctuations the method would fail during the binary search stage, because some of the gradient levels were multi-valued, which confused the coder. The effect of this was to cause the coder to wildly swing between several solutions. One of these solutions was often acceptable, but there was no way of knowing, which solution the method would choose. Filtering the data gives a predictable solution.



Figure 5-9. Rate distortion curves will often fluctuate with real data.

The best fit line is shown in figure 5-9, and the filter chosen should be able to produce a similar result. A forward looking average gradient filter was chosen to correct the shape of the rate distortion curves. The filter basically assumes that the gradient of all points after itself should be progressively shallower, as required by the method. The filter is described in equation 5.11 and is of a very arbitrary form. The filter is only applied to points, which do not conform with the required standard as chosen in equation 5.12. This means that original data points are preserved if possible.

 (5.11)

 (5.12)

Equation 5.11 is shown in expanded form to clarify how it is formed. Figure 5-10 shows a small section of the gradient curves and demonstrates how the filter works. The aim of the filter is to render the point closer to the best fit line. This is done by assuming that on average the lines connecting to the other points on the graph give an idea of where the best fit line should be. To use this idea the gradients of these connecting lines are used to predict , but they are scaled by 1,2 and 4 respectively to represent the influence that each point has on as they move further from the anchor point.



Figure 5-10. Diagram showing how the forward looking gradient filter works.

By repeatedly applying this filter the data quickly converges to the required form. Before each pass of the filter, it is checked to see, if it conforms to the limits required. This prevents the over use of the filter, which can cause the new curve to spread away from the best fit line.

The binary search is conducted as previously described, the only difference being that the accuracy of the search is entered manually. This is only done once for the coding of a particular transform type and is chosen to be a small percentage error. After the coder has chosen the individual compressions of the coefficient sets, the residual compression is distributed between every non-zero coefficient set.

1. **Templates**

In the previous sections it has been assumed that there is sufficient time to compute the rate distortion curves. This is sometimes possible, if the quantiser is simple enough, but this is not always true. If the quantiser is very computationally complex it may be necessary to find a method to approximate the shapes of these rate distortion curves without actually evaluating them. Obviously such a process will cause a further loss of accuracy, for an increase in speed, but it is a common trade off.

The shape of the rate distortion curves has already been described and they are visually very similar to each other . The form of the rate distortion curves results from the nature of the statistics of the data they describe. If the data being approximated comes from broadly the same source, such as images, then it should be expected that the curves have similar shapes. The statistic, which forms the rate distortion curve, is the probability density function (PDF) and for most orthogonal image transforms the PDFs of the coefficients are similar to that shown in figure 5-11.



Figure 5-11. Stylised graph showing a typical PDF function.

The only exception is the DC coefficient, but it is common to delta code DC components and this makes their PDF functions similar to the other coefficients.

Although broadly speaking all the PDF functions, and hence the rate distortion curves, have roughly the same shape there is a much stronger correlation between the PDFs of the same coefficients sets from different images. This is true, because there are broad classes of images, which have very similar frequency distributions, that most transforms are based on. To produce the best approximation of the rate distortion curves separate approximating functions should be used for each separate coefficient set.

The next consideration is that these approximating functions are being used to approximate the rate distortion curves that were previously found from a set of support points and not from a function. Since this system is already in place and because simple functions proved ineffective in approximating the true curve, it was decided to use templates. A template is basically a set of support points, which can be scaled to fit the rate distortion curves. Each coefficient set should be given a template, which is a good approximation of the shape of its given rate distortion curve.

The templates can be thought of as functions, which vary from 0 to 1, as shown in equation 5.13 :

 (5.13)

Each template is scaled by the truncation MSE to fit it to the data shown in equation 5.14. The only real difficulty is to actually produce effective templates.

 (5.14)

It is relatively simple to find the average templates for a subset of data but it is not always possible to do this effectively. A more interesting proposal is to make the templates ‘learn’ from experience. In this way the method always tries to be optimal within the data set it is given. This idea is useful if the coder is given data of roughly the same type, or slowly changing between types rather than rapidly switching data types. Since the coder was intended for use with image compression it is assumed that the data is usually of the same type.

The templates  are intended to learn from the data sources they are presented with but this is difficult since the data sources are no longer sampled directly. The only data produced is when the optimal combinations of compression are applied to the system. This gives a single data point on each rate distortion curve for each coefficient set. From this information it should be possible to modify the templates to better represent the shape of the incoming data source. This idea is discussed in the next section.

1. **Template Transformation**

In this section we will discuss, how templates are transformed. It is assumed that there are master templates for an average situation and these are used as a basis for the work. Since templates are changed each time they are used, coming closer to the true shape of the rate distortion curves, it is not critical that the master templates be very accurate. For this reason the master templates can be formed by hand, by taking real data from previous methods and simply scaling it.

The problem which need to be addressed is demonstrated in figure 5-12 and can be broken down into several steps.



Figure 5-12. Graph showing a template compared to a new data point B.

The new data point B must lie somewhere between support points A and C. It can be seen that simply adding the point to the data set is inappropriate. The information stored in point B must be appended to the template by transforming the other support points in the template. Obviously the method cannot rely on every new data point being typical and therefore it is necessary to weigh the method, to avoid rapid changes in the templates. This is shown in figure 5-12 by demonstrating that the best template position is somewhere between the old template and the template which passes through the new point.

The first step is to establish the value of the old template at point B. This can be done by using the cubic spline again, precalculated for the template.

It is now necessary to decide at what point between  and  (calculated from the template) should the new transformed template pass. This is illustrated in figure 5-13.



Figure 5-13. Graph showing the positions of the calculated MSE for point B.

It is therefore necessary to weight every template support point, compared to the new point. Again it is necessary to calculate the weighting of the template at B as demonstrated in equation 5.15.

 (5.15)

By having every support point on the template weighted separately it makes certain areas of the template more resistant to change, where more information has been stored from previous transformations of the template. It is assumed that the new data point is weighted as one compared to the rest and therefore the new point where the new template should pass is

 (5.16)

Initially all the weightings are set to one but as the templates are transformed the weighting increase according to how many hits occur.

From  and  it is possible to transform the support points of the template making them more representative of the real data. The process is constantly repeated so it is only necessary to ‘push’ the data in the right direction, since it is impossible to produce an accurate picture of the data. There are several considerations:

1. The truncation error must be maintained constant.
2. The templates all converge to approximately the same gradient at large *x*, denoted *x0*.
3. The point  must be on the new template.
4. The transformed template must conform with previous statements about the data. This requires the gradient to be single valued and at some stage the gradient of the old template must cross that of the new one. The gradient cross occurs because the templates converge to the same value at large *x* and therefore a larger inital gradient must be short lived and visa versa.

A simple approach was adopted to achieved these four requirements. To transform the template a simple linearly changing scalar was applied as shown in equation 5.17.

 (5.17)

This forces a contradiction between *x=x0* and *x=0* since the scaling is required to decline towards *x0* . The situation in equation 5.18 can not occur with the simple linear scaling.

 (5.18)

To achieve the desired result with a linear scaling it is necessary to pre-transform the templates before any template transformation are made as shown in equation 5.19. This then allows the calculation of *m* and *c* from equation 5.17 because there are two fixed points required by the method. This is shown in equations 5.20 to 5.23.

 (5.19)

 (5.20)

 (5.21)

 (5.22)

 (5.23)

Given this information it is possible to transform the template but it is important that the data conforms to point four. i.e. the should be single valued across its range. For this to be true equation 5.24 must be valid across the active range of the template.

 (5.24)

If it is assumed that the previous template conformed to point four then by expanding equation 5.19 it should be possible to see how the transformed template is affected as shown in equations 5.25 and 5.26.

 (5.25)

 (5.26)

For to be non zero for all x it is necessary that it is ever negative or positive over its whole range. For the linear scaling ,  is positive, *f* is positive and  is negative. The value of  depends on *R* and so the system is only correct when *R*<1 which makes  negative. When *R*>1 it is possible the a min/max/inflection point is created in the gradient of the template. Providing we know that this exists it is possible to filter it out of the data when it occurs. Using slightly more complicated functions has proved difficult since they have the same sort of problems as with the linear but this area has not been fully explored.

1. **Conclusions.**

This chapter has covered the basic equal gradient method. This will be used further in chapters 6 and 7. The process of using templates has been explored here, but while still applicable to still image systems, it is more useful in video applications, where it is not possible to fully calculate the rate distortion performance curves at every stage.